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. 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 Logisim tool.

2 Marking Scheme

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

- Prelab: 3 marks
- Part I (in-lab): 1 mark
- Part II (in-lab): 1 mark
- Part III (in-lab): 1 mark

2.1 The Prelab Exercises

All of the lab exercises for this course involve ”pre-lab” tasks that you must complete in preparation for your lab demo. Unlike other courses, the lab time is meant for the demonstration of your designs, which means that your circuit design and implementation must be complete and handed in before 6pm on your lab demo day.

BEFORE each lab, you must read through all the sections of the lab handout, paying particular attention to the pre-lab preparations which are clearly marked in red. These pre-lab exercises (both design and implementation) are meant to be completed individually and submitted electronically on Quercus before the lab.

During your weekly lab demo, you will show your pre-lab designs to your TA and demonstrate your circuit implementation to confirm that it performs the tasks required by the lab handout. These demonstrations are specified as ”in-lab” requirements. The more time and attention you put into your pre-lab work, the smoother your in-lab demo will go.

2.2 Deliverables

These are general guidelines for this lab and all future labs. What to include in the pre-lab report:

- Answers to the questions
- Hand drawn version of the designs when asked
- Screenshot of the designs
- Screenshot of the simulated test vectors

What to do during the demo sessions:

- Going through the pre-lab report
- Running and simulating the designs by poking and/or test vectors

- Answer questions about the design and/or the handout

Files required to be uploaded to Quercus prior to lab time:

- Pre-lab report
- Logisim files (.circ)
- Test vectors (.txt)

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 on the output of the circuit. Your task is to design and implement all of the circuits in both Parts I and II using **only AND** , **OR** **and NOT** **gates**.

For example, consider the following function:

$$f = a * b + (c + b)$$

This notation is shorthand for the following logical expression:

f = (a AND b) OR (c OR b)

The resulting schematic will look like this in Logisim:

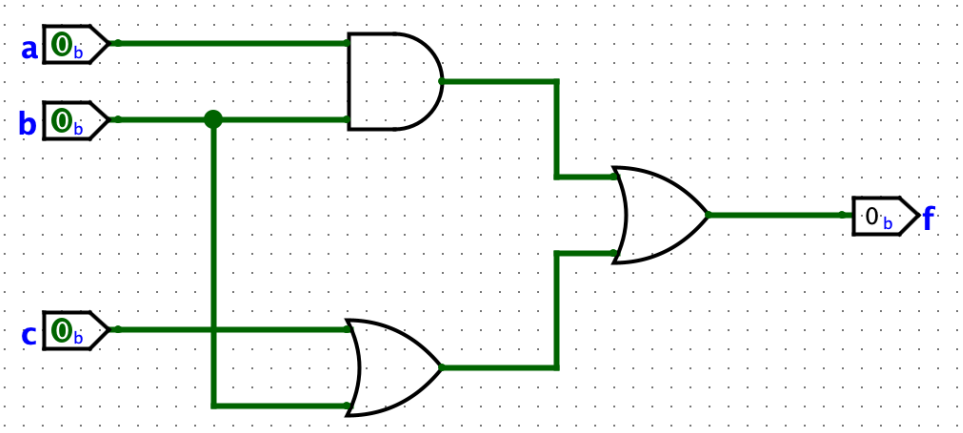


Figure 1: Schematic of the logical expression

Note that the AND gate at the top creates the *(a * b)* behaviour while the OR gate at the bottom creates the *(c + b)* behaviour. The OR gate on the right then combines these two parts into the overall expression.

In each of the parts below, show all of the steps required to go from the logical expression or circuit specification we provide to the final circuit for your demo. The will involve assigning names to input and output wires, deriving a truth table, deriving the logic expression, and then creating 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

For this part, you will implement a **multiplexer** device out of AND, OR and NOT gates. The multiplexer is a device that has multiple inputs and a single output. The device has another set of inputs (called ”select bits”) that chooses which of the inputs will be connected to the output.

The following Boolean function is the logical expression for a 2-to-1 multiplexer (two inputs, one output).

$$f = xs' + ys$$

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.

For this part of the lab, you must perform the following steps:

1. Draw the gate diagram (similar to Figure 1) that implements this 2-to-1 multiplexer design using the AND, OR and NOT gates as specified above. Include this diagram in the prelab report you submit on Quercus. Have this diagram ready to show during your demo so the TA can verify the correctness of your design. **(PRELAB)**
2. Write out the truth table for this design. You must also 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:

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

For this part of the lab, you must perform the following steps:

1. Draw the gate diagram for the function shown above using only AND, OR and NOT gates. 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 this expression. Include this truth table in your pre-lab report and have it ready to show the TA during your in-lab demo. **(PRELAB)**
3. Is there a cheaper implementation for your design, assuming you are still limited to using the same three gate types? If yes, explain by providing the boolean function and the corresponding gate diagram. 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 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.jar` from `https://github.com/reds-heig/logisim-evolution/releases/tag/v3.3.0`.
2. Make sure to read through Tutorial 1 for general instructions of Logisim.
3. Launch Logisim (you will need Java 13 installed) and complete the Logisim *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. 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.

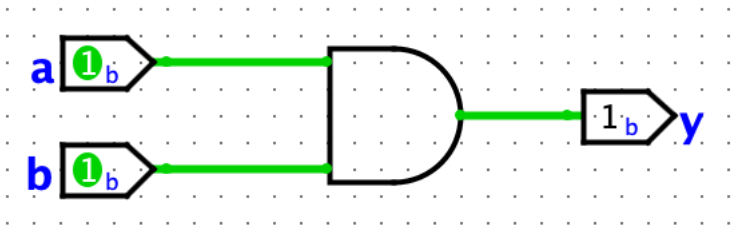
(d) Have the circuits from Parts I and II implemented in Logisim and ready to demo to your TA (PRELAB).

5. To test your circuit in Logisim

(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

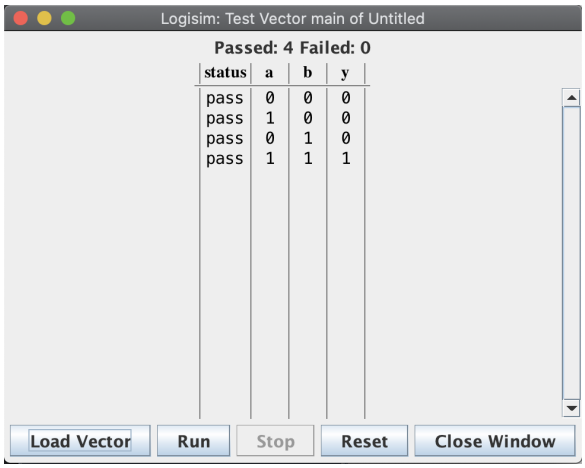
The corresponding test vector file:

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 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).