

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 tool Logisim.

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:

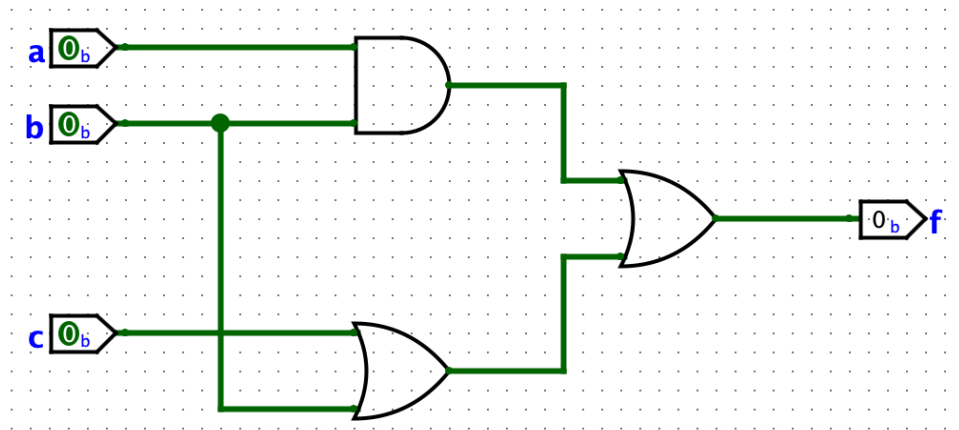


Figure 1: Schematic of the logical expression

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)**

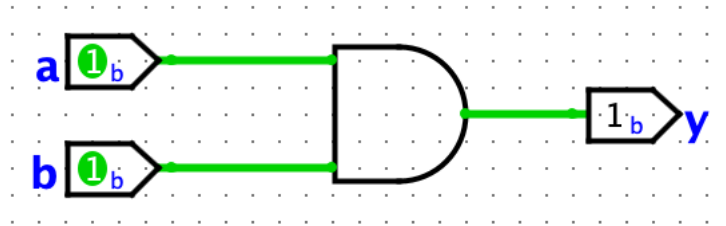
6 Part III

In this section, you will start on your first Logisim 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 *Beginners Tutorial* step 0-4. The tutorials can be found in *Help > Tutorial* when you launch the tool and *Beginners Tutorial* can be found on the left of the *Tutorial* window.
4. Change the output file to Verilog in Logisim: *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 and ready to demo to your TA **(PRELAB)**.
6. 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:

Logisim: Test Vector main of Untitled

Passed: 4 Failed: 0

status	a	b	y
pass	0	0	0
pass	1	0	0
pass	0	1	0
pass	1	1	1

Load Vector Run Stop Reset Close Window

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

Appendix: Physical Lab Instructions

A Access to Logisim and Quartus outside of Labs

In addition to Logisim, there is another tool called **Quartus** that will be used in this course to complete some of your prelabs. You don't need it for Lab 1 but you can read more below to know what to expect.

While Logisim is the simulator you use to build and test the functionality of your hardware design and debug your circuit before downloading on the board, Quartus is the software tool that is used to compile these circuit designs to a FPGA (Field Programmable Gate Array).

As outlined in the tutorial, Logisim can be downloaded from this link: <https://github.com/reds-heig/logisim-evolution/releases/tag/v3.3.0>. Remember that you will need Java to run the jar file.

Here are the ways that you can have access to Quartus:

1. Quartus Prime Lite version 18.1 is available in the Computer Science Teaching Laboratories (formerly known as CDF). You can launch it by using the `quartus` command from a terminal.
2. If you are connecting to the CDF server remotely (via ssh) you need to have X forwarding enabled as both these tools have a graphical interface. You can do this as follows:

```
ssh -Y your_username@teach.cs.utoronto.ca
```

Note that the student usernames on CDF have recently changed to be your UTORids. For more remote access options, please see:

http://www.cdf.toronto.edu/using_cdf/remote_access_server.html

3. You can also (and highly encouraged to) install the free version of Quartus, *Quartus Prime Lite Edition*, on your own laptop/desktop. Before you start, please check the system/memory requirements at: <http://fpgasoftware.intel.com/requirements/18.0/> and read the caveats below.

Here are the steps to install Quartus:

- (a) You can download **Quartus Prime Lite Edition 18.0** via the following link: <http://fpgasoftware.intel.com/?edition=lite>. This download also includes **Modelsim-Altera**.

Downloading the combined version of the files is easier, but if you want to go the “Individual Files” route (*e.g.* to save some disk space), you can uncheck all the device families **except** Cyclone V (see Figure 2).

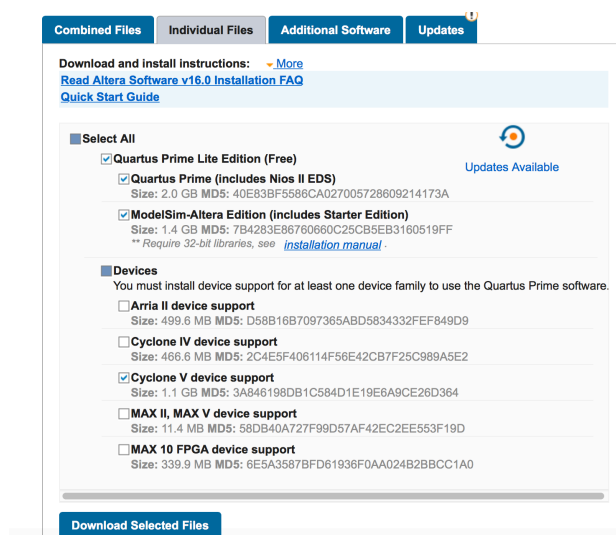


Figure 2: Minimum subset of Quartus that you need to install

A.1 Caveats

Quartus is currently supported on Windows and Linux. If you have a Mac, you would need to install and run Quartus on a virtual machine. Here are some pointers, if you want to do that:

- Before you start, make sure you have enough disk space in your virtual machine to install Quartus. The Quartus installation requires ~13GB of storage, so you'd probably need your virtual disk to have 30-35GB of storage or so.
- Install VirtualBox from here: <https://www.virtualbox.org>. It is a well-known powerful virtualization software that is completely free.
- If you are considering running Windows on a virtual machine on your Mac, you need a Windows license. You can get one *free* Windows license as a student in Computer Science, Engineering and iSchool, while attending classes. For more information, visit the Microsoft Imagine webstore at <http://e5.onthehub.com/WebStore/ProductsByMajorVersionList.aspx?ws=30b88fca-b08b-e011-969d-0030487d8897&vsro=8>. The "How do I get access?" link explains what you need to do to get an account. I will provide a CD key and an ISO file to install windows.
- If you only need to have Windows to run Quartus, then you can also use Microsoft Edge program virtual machines that you can download for free from here <https://developer.microsoft.com/en-us/microsoft-edge/tools/vms/>. Just note that those VMs expire after 90 days, which is almost enough for this semester.
- Another possible option is to install a Linux distribution (such as Ubuntu <https://www.ubuntu.com/> or Mint <https://www.linuxmint.com/>), which are completely free!
- Now you can install Quartus on your virtual machine.

A.2 Purchasing a DE1 Board

Lastly, if any of you are thinking of buying a DE1-SoC board (which is neither necessary nor required for this course, and is therefore completely optional), then you could get one from Terasic with a UofT discount. You can find more details here: http://www-ug.eecg.toronto.edu/desl/handouts/buying_DE1.html. Just be sure to check with them how long it will take for them to send it to you and you receive it.

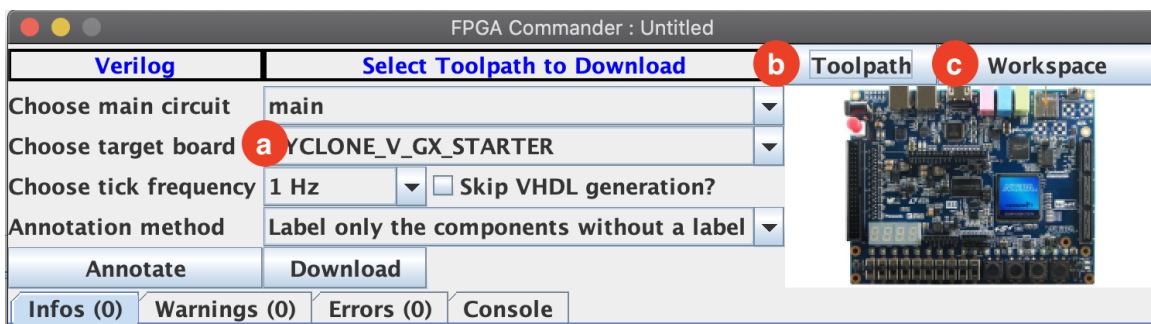
B Uploading to the DE1-SOC Board

The following steps are needed to upload the Logisim design onto the physical hardware (the DE1-SOC board). Since COVID-19 makes it impossible to access the physical labs, these steps are mainly useful for those of you who have purchased a DE1-SOC board of your own, either in preparation for the course or for your own interest. If you don't have access to the physical hardware, the follow steps are mostly of general interest and not needed to complete the lab. If you do have access to the DE1-SOC board, these are the steps you'll want to follow.

1. Download your Logisim circuit onto the DE1-SoC board. Use switches SW_{2-0} on the DE1-SoC board as the data inputs (labeled as a, b, c in Figure 1). Connect the output to $LEDR_0$. Do not forget that you will need the `DE1_SOC.xml` file to define how the switches and LEDs connect to the pins.

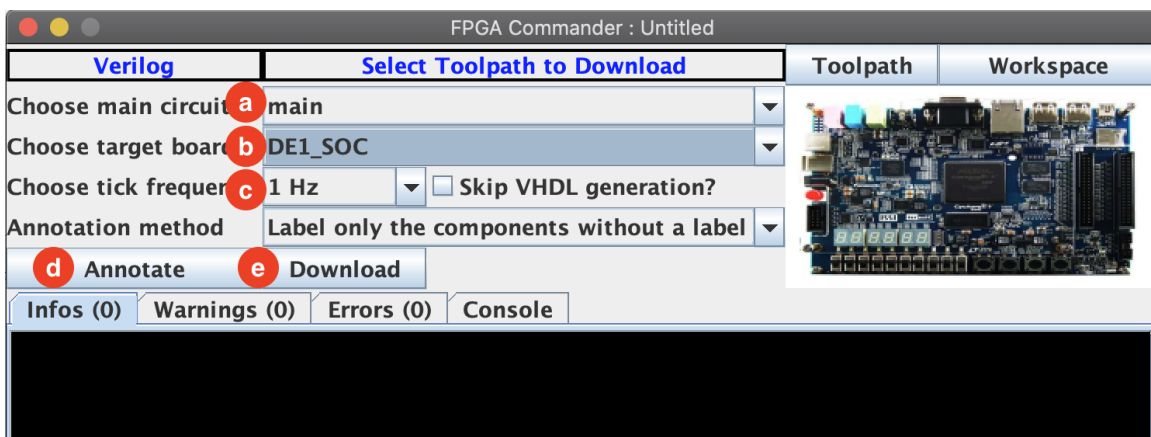
To download your circuit to the DE1-SOC board, Logisim first needs to have the DE-1 configuration file loaded, a directory to put project files configured, and the path to the compiler.

`FPGA > Synthesize & Download...` to open this:



- (a) First download the DE1-SOC board file (`DE1_SOC.xml`) from the course website. *Choose target board > Other > browse to `DE1_SOC.xml` that you downloaded > hit `Open`.*
- (b) *Toolpath > type `C:\DESL\Quartus18\quartus\bin64` if you are connecting to a machine in BA3135, BA3145, BA3155, BA3165 or `/s/appspace/quartus/quartus/bin/quartus` if you are connecting to teach.cs machines. Then hit `Open`. This path may vary on your own computers. **NOTE:** Since now we won't be able to access the DE1-SoC boards, compiling is not necessary for all the labs. So if you don't have Quartus installed on your computer, it is okay, just skip this part.*
- (c) *Workspace > browse to folder where Logisim will put Quartus project folders when compiling > hit `Open`. You will need to submit the files Logisim generates, so don't just choose your home folder, make a new one.*

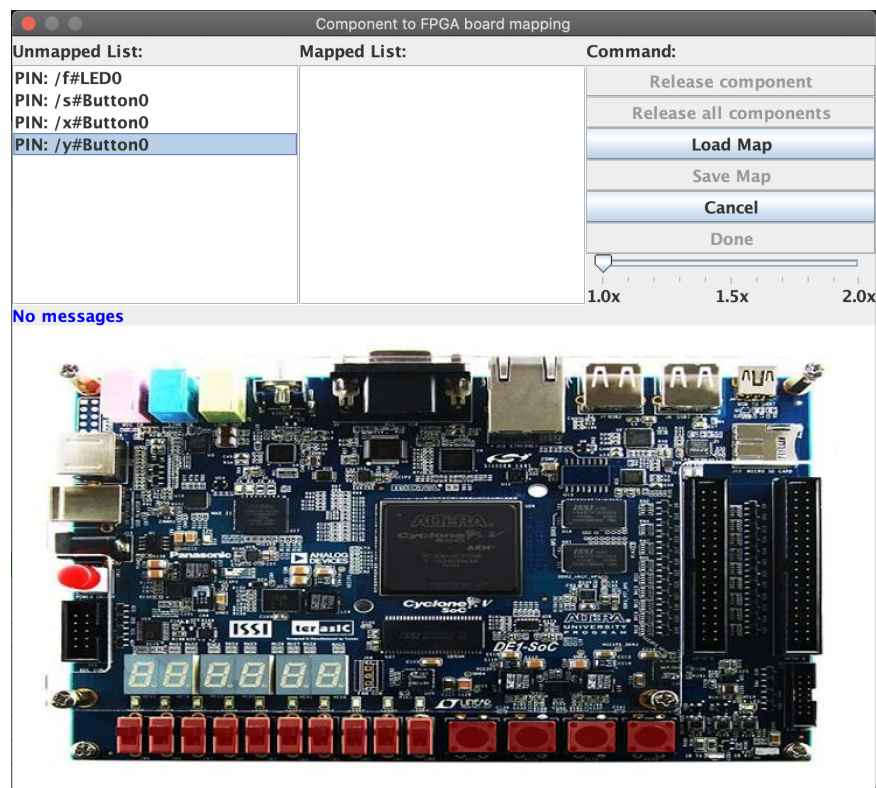
This is what you should see before proceeding to the next step:



Starting the downloading your circuit to the DE1_SOC board usually just needs a quick press of the *Download* button, but heres what everything here means:

- (a) Choose main circuit should be the top level circuit you want to program to the board. (Defaults to last active circuit)
- (b) Choose target board should be DE1_SOC
- (c) Choose tick Frequency sets how fast the Clock component toggles. This will be useful later on in the course.
- (d) Annotate will label components (by default, only components without a name).
- (e) Download will start the board programming process.

After pressing the *Download* button, you will see the pin mapping page:



Inputs can be mapped to the buttons and switches on the DE1_SOC board. Outputs can be mapped to the 7-segment display and the LEDs.

We need to map our input and output devices to things on the real DE1_SOC board. To do this, click on a input/output in the *Unmapped List*. Components that this can be mapped to are highlighted in translucent red. Click on one of these to map the device. The next device is automatically selected.

Save Map and Load Map lets us save these mappings to load them the next time we want to use the same ones. *Done* starts the compiling process.

Once its done compiling, we get this:



Verify that your board is connected and you are ready to download.

Yes, download

No, abort

Note that now we won't be able to actually download this on the board since we don't have access to the labs, but you should be able to do everything up until this point and understand the process and how the design can be mapped to the DE1-SoC board.