CSCB58 Lab 1

Part A: Introduction to the Design Tools

1 Introduction

In this part of the lab, we introduce **Logisim-Evolution** – the hardware design tool we will use in this course. You will need to download and install it on your PC, and you should spend as much time as necessary in this lab to make sure you are comfortable with these tools. If you run into problems or have questions, ask on the discussion board (forum)!

After this part, you should be able to:

- Use Logisim-Evolution to build a circuit.
- Test and understand your circuit design.

Before lab, read through the Logisim-Evolution beginner's tutorial and download the reference document provided in the next section so that you have a high-level understanding of the steps required for the lab. The TA will be available to answer questions during lab or to help you if you get stuck, but you must be familiar with the steps to complete the lab in two hours.

2 Obtain the Software

With Logisim-Evolution, it is easy to work on the labs on almost any computer. You can either download an installer for your operating system (Windows/Mac/Linux), or you can download the Java jar file. The software can be downloaded for free at this link:

`https://github.com/logisim-evolution/logisim-evolution#download`

Some notes:

- **Make sure you are using Logisim-Evolution**, linked above. Logisim has different variants such as the original Logisim, Logisim-holycross, Logisim-iitd and others. Do not use them.
- We will use the latest released version (currently 3.7.2) but we've used older versions such as 3.4.1 before with no issues.
- Logisim-Evolution requires a Java installation (Java Runtime Environment, or JRE). You can download an installer that contains Java (.dmg for Mac, .msi for Windows, .rpm and .deb for Linux). Java version 16 and above should work. If you prefer to download Java yourself or are having issues, here are a some places to download a JRE from:
  - `https://www.oracle.com/java/technologies/javase-downloads.html`
  - `https://www.azul.com/downloads/zulu-community/?package=jre`
- If you are unable to get this working, make sure you try to search for help online, or ask in the forum.

3 Logisim-Evolution Tutorial

You should be familiar with the basics of creating a simple logic circuit in Logisim-Evolution. Logisim-Evolution has a built-in beginner's tutorial available on the menubar: `Help` → `Tutorial`.

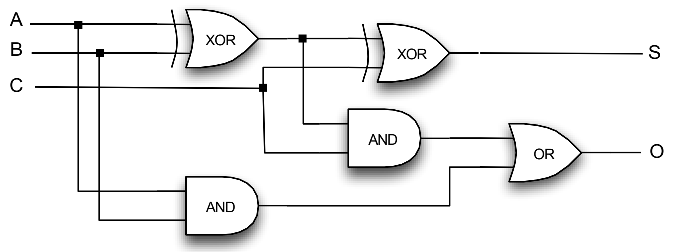
[TASK] Complete steps 0 to 4 of the Logisim Beginner's Tutorial.

In addition to a built-in tutorial in Logisim-Evolution, there is a short reference document available at:

https://q.utoronto.ca/courses/251270/files/18439709?wrap=1

This document contains a birds-eye view of the main sets of features you will find useful during the semester. You should look at the in-built tutorial for further details. Note that this reference is a little old, and some of the images (e.g., input and output pins) look a little different in newer versions.

4 Mystery Circuit



[TASK] Investigate the operation of the circuit shown in the figure above

- Create a new project, and build this circuit. Make sure to use input and output pins (A,B,C are inputs, S,O are outputs) and to label the pins.
- Make sure to save the circuit since you will need to submit it to Quercus, and also to demonstrate its live operation to the TA!
- Try to predict the operation of the circuit. What do you *think* will happen for various inputs?
- Test the circuit using Logisim poke tool to determine how it behaves.
- Complete the truth table below for all possible inputs.
- Write the truth table as a simple plain text file (.txt) or create a PDF with the table. You'll need to submit it with your circuit to Quercus.

inputs			outputs	
A	B	C	S	O
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

A Note on file types: Please make sure to **submit files of the types we have asked for** in the lab. When we ask to submit a Zip file, do not submit a RAR file. When we ask to include a PDF, a plain text file (.txt), or an image (e.g., jpeg or png) – do not submit Word files (.docx, .doc) or PowerPoint files (.pptx).

We are not out to get you: our TAs simply don't always have RAR, Word, or Powerpoint. Note however that many programs can easily export to a PDF, so you can definitely use them to make truth tables and so on. Just make sure to export it to PDF and submit that.

5 Summary of Tasks

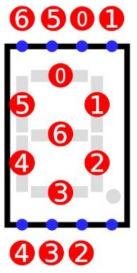
Here is the summary of the steps to be completed for this lab:

1. Install Logisim-Evolution and make sure you can run it.
2. Read and practice the Logisim Beginner's tutorial and download the reference document.
3. **Think!** Try to predict the behaviour of the mystery circuit before implementing it.
4. Implement the mystery circuit using Logisim-Evolution.
5. Test the circuit, and create a truth table for all possible input combinations.
6. **Submit solution to Quercus on time**: create a zip file with the saved circuit `.circ` file and the truth table (you can write it as a simple text file or create a PDF with the table). Upload to Quercus.
7. During your lab slot with the TA, demonstrate the operation of the circuit and answer questions.

Part B: Circuit Creation

In this part of the lab, we will practice using Karnaugh maps (K-maps) to design combinatorial circuits made of logic gates. You'll need to be familiar with the software tools we used in part A; refer back to the previous part to refresh your memory.

Driving 7-Segment Display

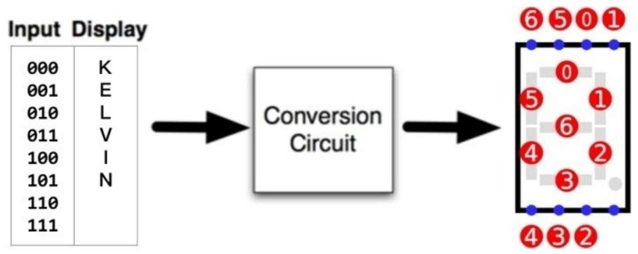


The figure on the right shows the "7-Segment Display" component in Logisim-Evolution. You can find it the in the components list (also called explorer pane), inside "Input/Output".

A 7-segment display takes 7 input bits (wires); each bit controls one of the 7 display lights: lit ("on") or dark ("off"). The numbers of the figure on the right show show the correspondence between input bits (wires) and the display segments. The 8th unmarked wire controls a decimal point segment; we will not use it in this assignment and you can ignore it.

Showing your name

Your goal in this lab is to build a circuit that displays the first 8 letters of your first name on the 7-segment display. For example, if your name is Kelvin, we want to build a circuit that can display the letters `K`, `E`, `L`, and so on.

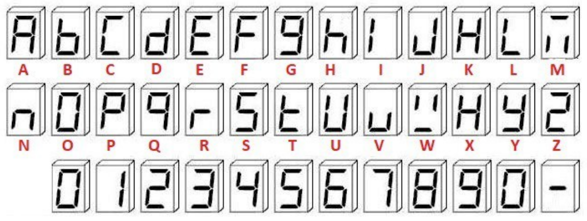


Input	Display
000	K
001	E
010	L
011	V
100	I
101	N
110	
111	

The objective here is to design a circuit that takes in 3 bits as inputs, and has 7 corresponding outputs for the 7 segment display as shown above. The input should correspond to the index of the character in the string that should be output. In the above example, this means that the input `000` should display the letter `K`, `010` should display the letter `L`, and so on. Since we are only showing the first 8 letters of your name, we only need 3 bits for the input. Since each of the strings are 8 characters, we only need 3 bits for the input. If your name is shorter than 8 characters, just show spaces for the extra letters (i.e., all display segments are off), as in the example above.

To display each of the letters, your circuit needs to output the correct 7 bits of output that need to be passed into the 7-segment display. So, for example if on input X = `001` we want to display the letter `E`, then the corresponding output S that goes into the 7 segment display should be: $S_6 = 1$, $S_5 = 1$, $S_4 = 1$, $S_3 = 1$, $S_2 = 0$, $S_1 = 0$, and $S_0 = 1$. We can read this as a 7-bit number with bits listed from higher to lower: S =`1111001` (we always list digits of binary numbers in descending order, like we do with decimal digits).

The graphic on the right shows how to display each of the English letters and numbers on the 7-segment display. You need to figure out the corresponding 7 bit codes for the ones you need. To display space or empty characters, set all segments to off.



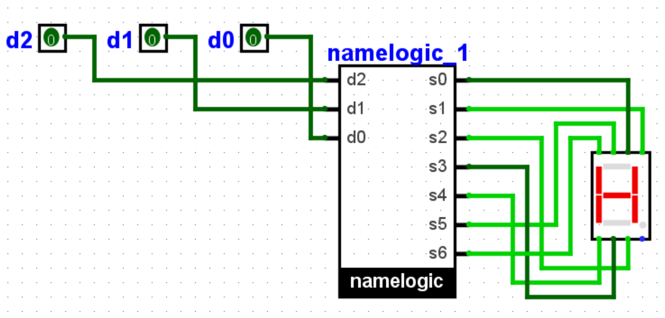
Guidance

First, generate the expressions for the seven optimized circuits – one for each output (segment on the display). To do so, you'll need to create a truth-table with 3 input bits and 7 output bits. Then, you should use seven Karnaugh-maps (K-maps) – one for each output – to generate the optimal expression that expresses it.

Warning: There are many tools that can automate building Karnaugh maps and simplifying expressions. **Do not use them at this stage.** You will need to be able to K-maps and simplify expression from scratch for this course, including in quizzes, labs, and the exam. This is your chance to get practice. Once you feel you have mastered K-maps, you can start using such tools.

Once you know what circuit you are building, you can then implement it in Logisim. Create a sub-circuit called "namelogic" (or some other name) and implement all the conversion circuit logic there. To create a sub-circuit, select `Project` → `Add Circuit...` from the menu. See the "Subcircuits" documentation for Logisim, and Section 2.5 of the Logisim Reference document.

Finally, instantiate your logic sub-circuit inside the top-level main circuit, add input pins and a 7-segment display, and wire everything together. Your main circuit should look something like the figure below. You can then test your circuit and fix any design errors. Do not forget to submit a **zip** file to Quercus with the K-maps and optimized expression (as **PDF**, or jpeg if you scanned a paper) and your Logisim circuit (**circ** file).



A Note on naming: some people see d2, d1, and d0 and think "let's name my inputs A, B, and C". Don't do it. First, because in our case d is a 3-bit number; ABC does not carry the notion of order (is the binary number ABC or CBA?). Second, because these are the lab instructions so you should follow them. Following specification will be even more important when you leave the university. Third, standard names help us understand your circuits better and faster. (To be clear, we are not going to be strict about it, and we generally don't care if you use `D 2` instead of `d2` and so on.)

Tips on submitting K-maps: If drawing the K-maps on paper, make sure you make the K-map big, which will help you draw boxes. You can use different colors and/or vary the sizes of the boxes to make things clear. You can also use PowerPoint or Google slides (or anything like that) to draw K-maps, and **export to PDF** – these can give you lots of colors and clarity and makes it easy to edit the K-map. It's up to you, we just need to see a clear K-map.

Summary of tasks

1. Create truth table for the logic circuit.
2. Create K-map for the logic circuit.
3. Derive optimal expression for the logic circuit.
4. Implement and test the circuit in Logisim. Go back to previous steps if you've made a mistake.
5. Submit zip file to Quercus with the K-map, expressions, and circuit.
6. Demonstrate the use of your circuit to the TA and explain why it is working correctly.

Overall Evaluation (Part A & B)

As always, marks are based not only on submitted work but also on oral examination by TA. You need to be able to make reasonable explanations about any details to the TA.

	Submitting everything on time	1 mark
	Part A:	
	Correct implementation of circuit	1 mark
Solution	Correct truth table	2 marks
	Part B:	
	K-maps and expressions	2 marks
	Demonstrating correct operation	2 marks
Understanding	TA oral score	1 to 4

Final lab mark (up to 8) is determined by multiplying your solution subtotal by the oral score (1–4) and dividing by 4:

$$\text{total} = \text{solution marks} \times \frac{\text{oral score}}{4}$$

NOTE: The interview with the TAs for this lab will occur during week 4 (Jan. 30 – Feb. 3) in your assigned practical time slot **in-person**. Failure to show up will result in receiving a ZERO on the oral score, and by extension, the lab.