# CSC258 - Lab 2

**Multiplexers, Design Hierarchy, and HEX Displays**

## 1 Learning Objectives

The purpose of this lab exercise is to use gates to build simple circuits, and to learn the importance of simulations and hierarchies. This exercise is also meant to illustrate the importance of Karnaugh maps in designing circuits, and introduces important devices such as the multiplexer and seven-segment decoder.

This lab also introduces FPGA design concepts through references to the DE1-SOC board (the hardware used in the Bahen Centre's physical lab rooms). In an in-person semester, the final stage of these labs would be to upload your circuit designs to the DE1-SOC board and test them on the physical hardware. Since the CSC258 labs are taking place online, we won't be performing that final stage. You'll still model the high-level inputs and outputs of your design after those on the DE1-SOC board, such as switches $SW_{9-0}$, Light Emitting Diodes (LEDs), and 7-segment displays.

This lab is also meant to introduce new notation and terminology that is used to describe physical hardware. For instance, the lab refers to signals on the DE1-SOC's input switches as $SW_{9-0}$, *i.e.*, with the subscripts. In place of these physical switches we will approximate these inputs and outputs in Logisim instead, while still using the physical board notation.

## 2 Marking Scheme

This lab is worth 6% of your final grade, where the marks are allocated as follows:

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

## 3 Preparation Before the Lab

For this lab and all future labs, your prelab report should include schematics (which can either be drawn by hand on paper or using a software tool), and simulations/screenshots that illustrate the functionality of your circuits (like showing the results from running your test vector file).

Schematics typically illustrate the structure of your circuit, much like the schematics in Lab 1 showed how your circuit should be built. Figure 1 and Figure 3 are examples of how to draw the schematic for multiplexers.

### 3.1 Modules

Your circuit design for each part of this lab will consist of a number of **modules**, where a module is like the circuitry equivalent of a class in Java or Python, encapsulating and implementing a circuit instead of code:

- Similar to the functions you were asked to design for Lab 1, each module you create for this lab will contain the circuitry needed to perform a particular task.
- Like functions and objects in software, modules can contain other modules. Internal modules are just another component in the circuit, no different than an elaborate logic gate.
- The top-level module of your circuit will have specific labels for its inputs and outputs, specifically switches, LEDs and seven-segment displays. These labels correspond to the input and output components on the DE1-SOC board. These labels need to match, just in case we want to uploading this top-level module to the DE1-SOC board.
- Significant wires in your module need to be labeled as well. The input and output pins are particularly important to make sure that input and output values from outside modules are sent to the correct pins. Some internal wires in your schematic are worth labeling as well, especially if these wires might be useful to include in your test vector.

### 3.2 Simulating your Design

Using the Poke function is fine when testing out small circuit designs. For larger modules, or as a module nears completion, you'll want to create a test vector file or files, meant to verify the performance of your design. When we asked you to provide your simulation, we're asking you to include a screenshot of Logisim's test vector results in your prelab report.

If the simulation is very long, make sure to record (screenshot or printout) the salient parts to demonstrate that you have performed the simulations and that the key elements of your circuit are working. It is not necessary to have pages and pages of results. However, occasionally, you will be asked to demonstrate and explain part of your simulation to the TA in the lab, so be prepared for this.

As an example, if your circuit implements a logic function with three or four inputs, it is reasonable to demonstrate the functionality of all possible combinations of input values. However, if your circuit implements a logic function with ten inputs, it would be unreasonable to simulate all $2^{10}$ possible input values. Instead, use similar approaches to testing that you would use for software; find groups of inputs that are logically related and test each group independently. For instance, if you have a multiplexer module and an adder module, it is reasonable to assume that you can test each independently instead of testing all combinations of all their inputs combined.

## 4 Part I

This part of the lab expands slightly on the mux circuit from Lab 1, introducing the concept of creating a module and associating the input and outputs to the physical hardware of the physical DE1-SOC board.

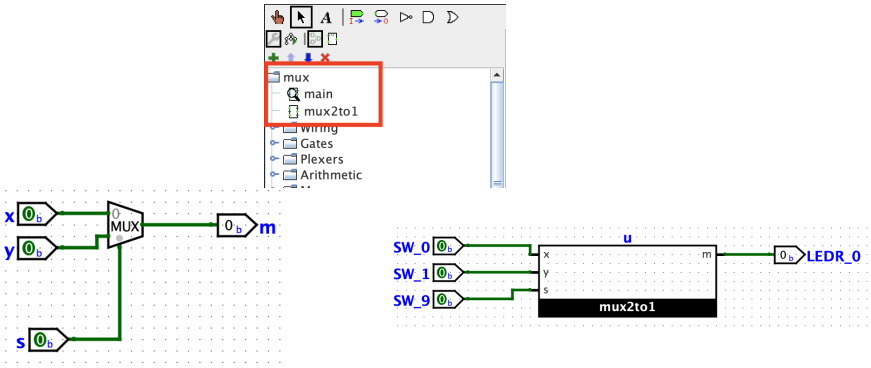
### Logisim File (.circ)

The DE1-SoC board in the on-campus labs provide 10 toggle switches, called $SW_{9-0}$ that can be used as inputs to a circuit, and 10 red lights, called $LEDR_{9-0}$, that can be set high or low to display output values. These subscript ranges indicate the names of the individual inputs and outputs. For instead, your circuit could take signal SW[0] as an input and send a result to LEDR[0] and LEDR[1]. As far as your Logisim design is concenred, we will use the same input and output pins from the tool bar that you used in Lab 1.

If we were using the DE1-SOC boards in the on-campus lab rooms, we would need to map the inputs and outputs for the top-level Logisim module to the $SW_{9-0}$ and $LEDR_{9-0}$ pins on the DE1-SOC board respectively before uploading your design to the board. Unfortunately, at this point we do not have access to the boards in the on-campus labs, but we will still use the DE1-SOC ideas and terminology

to approximate the way your design would map to the physical board. A Logisim file for a 2-to-1 multiplexer, named *mux.circ*, has already been provided to you on Quercus.

After you open this mux file in Logisim, the main is a very trivial example of a design hierarchy, as it only instantiates a single *mux2to1* module. In the more general case, a single module can instantiate a number of interconnected modules. However, in any circuit you build, there must be only one *top-level* module (called `main` in Logisim). The names of the left side of the rectangular module symbol match the input ports of the *mux2to1* module and on the right side, it is the output port of the *mux2to1* module. Note that you can have multiple modules of the same type in a single circuit, similar to how several objects of the same type can be used together in the implementation of a larger object. And similar to object-oriented programming, each use of a module is called an *instance* and every instance in a single circuit must have a unique label. In our example below we used only one instance of the mux2to1 module, which we named *u*. There are two modules, main and mux2to1 in mux.circ, you can check this in *Components*. Double-clicking on either of them to view the module on canvas:




Currently *main* has 3 inputs, corresponding to the three multiplexer inputs. The input labeled 0 on the multiplexer would be connected to an input called $SW_0$ in Logisim (which maps this to $SW[0]$ on DE1-SOC); the input labeled 1 on the multiplexer would be connected to the input $SW_1$ in Logisim (which maps to $SW[1]$ on DE1-SOC) and the input signal labeled `select` would be connected to the input $SW_9$ in Logisim (map this to $SW[9]$ on DE1-SOC). The output is the output $LEDR_0$ in Logisim, which would map to the $LEDR_0$ on the DE1-SOC board.

Figure 1 shows the symbol for a 2-to-1 multiplexer. As mentioned in Lab 1, a multiplexer is a device that uses a select signal to select which one of multiple inputs should appear on the output of the device. In Figure 1, input $s$ will control which of the inputs $x$ and $y$ will appear on the output $m$. If $s$ is 0, $x$ will appear on the output, while if $s$ is 1, $y$ will appear on the output.

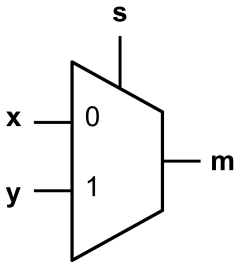



Figure 1: Symbol for a 2-to-1 multiplexer

The Boolean expression for a 2-to-1 multiplexer is `m = xs'+ ys`.

### Test Vector File (.txt)

After building the circuit in Logisim, and to verify the circuit's functionality properly, we can perform testing using a test vector file like in Lab1. This file is also provided to you (on Quercus, and in the appendices of this handout).

This test file includes the truth table for the 2to1 mux. And you should be able to run the tests through *Simulate > Test Vector...*

```
# Test Vector for mux2to1
SW_0 SW_1 SW_9 LEDR_0
0  0  0  0
0  1  0  0
1  0  0  1
1  1  0  1
0  0  1  0
0  1  1  1
1  0  1  0
1  1  1  1
```

## 5 Part II

Start with the design given in Part I and modify the design to make it a 4-to-1 multiplexer. You must use multiple instantiations of the *mux2to1* module given to you in Part I. This is known as *hierarchical design* and is a good practice especially for larger designs where the circuits you build can become more difficult to debug. Smaller submodules are generally easier to test thoroughly and debug.

To instantiate a model, create a new module by clicking on the green plus sign just above the components and then simply click on the module name in *Components* and drop it on the canvas. You can then connect the input and output of the instantiated model with input and output of your current module with wires. **Note: All inputs and outputs of the module being instantiated have to be the default type from the tool bar.** But you can do some experiments yourselves to see what could happen when we instantiate a module with other types of input output in *Input/Output*.

Figure 2 shows a schematic of two modules connected with wires.

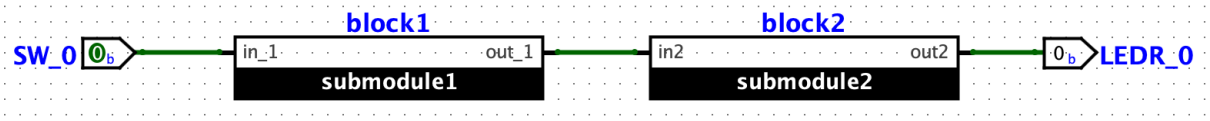



Figure 2: Using wires to make a connection between two modules

Now construct a module for the 4-to-1 multiplexer shown in Figure 3 with the truth table shown in Table 1 using wires and multiple instances of the mux2to1 module. Note that the truth table in Table 1 is given in a short-hand form. A real truth table would consist of rows enumerating all possible combinations of values of inputs $u$, $v$, $w$, $x$, in addition to $s_0$ and $s_1$, and show the value (0 or 1) of the output $m$ for each row of the truth table. Since this would result in a truth table with a large number of rows, it is written in short-hand form instead.

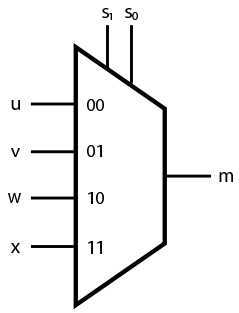



Figure 3: Symbol for a 4-to-1 multiplexer

| $s_1 s_0$ | m |
|---|---|
| 00 | u |
| 01 | v |
| 10 | w |
| 11 | x |

Table 1: Truth table for a 4-to-1 multiplexer

Perform the following steps:

1. Answer the following question: if the truth table in Table 1 was given in full, how many rows would it have? **(PRELAB)**
2. Draw a schematic (not in Logisim) showing how you will connect the *mux2to1* modules to build the 4-to-1 multiplexer. Be prepared to explain it to the TA as part of your prelab. The schematic should reflect how you are going to create your Logisim circuit. **(PRELAB)**
3. Build your circuit in Logisim **(PRELAB)**
4. Test your circuit with different values of $s$, $u$, $v$, $w$ and $x$. Do enough testing to convince yourself that the circuit is working. You must show these to the TA as part of your prelab. **(PRELAB)**
5. Map your Logisim design to the DE1-SoC board inputs and outputs. Use switches $SW_{9-8}$ as the 2-bits input, and switches $SW_{0-3}$ as the data inputs (labeled as u,v,w,x in Figure 3). Connect the output m to $LEDR_0$. **(PRELAB)**

Answers to questions and hand-drawn schematics should be included in your prelab report and submitted online before the lab

# 6 Part III

In this part of the lab, you will design a decoder for the 7-segment HEX display as shown in Figure 4. The output of the HEX display is determined by the value at the input of the decoder as shown in Table 2. We call this a HEX display because it can display all hexadecimal digits.

When you build your circuit in Logisim, use default inputs and *7-Segment Display* as output. Note that if we were uploading our design to display on the DE1-SOC's 7-seg display, those segments would be active low, meaning that the segments would light up when the signal is 0. So if you plan on uploading your design to an actual DE1-SOC board, you'll need to adjust your *7-Segment Display* in Logisim by going to the *Properties* of *7-Segment Display* and changing *Active On High?* to No.

HINT: In order to solve this part you need to first identify which segment needs to be illuminated for every input and then write a Boolean function for each one of the seven segments of the HEX display so they are turned on when needed. You must use Karnaugh maps to optimize those Boolean expressions before you build the corresponding circuits in Logisim.

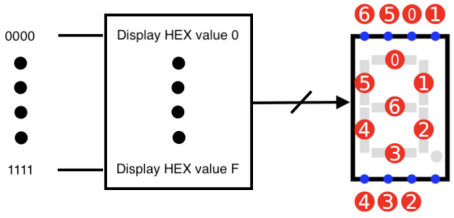



Figure 4: HEX decoder

| $c_3c_2c_1c_0$ | Character |
|---|---|
| 0000 | 0 |
| 0001 | 1 |
| 0010 | 2 |
| 0011 | 3 |
| 0100 | 4 |
| 0101 | 5 |
| 0110 | 6 |
| 0111 | 7 |
| 1000 | 8 |
| 1001 | 9 |
| 1010 | A |
| 1011 | b |
| 1100 | C |
| 1101 | d |
| 1110 | E |
| 1111 | F |

Table 2: Desired behaviour of HEX decoder

Perform the following steps:

1. Write the expressions for seven Boolean functions, one for each segment of the 7-segment decoder. You must use Karnaugh maps for optimization. You should be able to explain to the TA how you generated these expressions by showing them the truth-tables you wrote, and Karnaugh maps you used to optimize your circuits. **(PRELAB)**
2. Build the circuit for the 7-segment decoder in Logisim taking advantage of the aforementioned expressions. HINT: you can change the number of inputs for each gate in by selecting it and go to *Properties*. Also it might be helpful to separate the circuits for each bit into different modules. **(PRELAB)**
3. Check whether your implementation is correct. To verify the correctness of each segment, you need to create a test vector file for each segment's truth table. To test whether your circuits for each segment connect well with the *7-Segment Display* in Logisim, use 'Poke' on the tool bar. Show the screenshots of your simulation to your TA as part of the prelab. **(PRELAB)**
4. Map your Logisim design to the inputs of the DE1-SoC board. Connect the $c_3c_2c_1c_0$ inputs to switches $SW_{3-0}$ and test that changing the values on these switches creates the appropriate change in the seven segment display.

# 7 Resources

You can find many resources about the DE1-SoC board here `http://cd-DE1-soc.terasic.com/`. The User Manual for the DE1-SoC board can be downloaded from here: `http://www-ug.eecg.toronto.edu/desl/manuals/DE1-SoC_User_manual.pdf`