Simulation Lab 0: Simulator Tutorial – Using Logisim

Prerequisites: Before beginning this laboratory experiment you must:

- Know how to count using the binary number system.
- Have some familiarity with hexadecimal numbers. (Helpful but not necessary.)
- Create a truth table for a three-variable logic function.
- Have watched the Logisim tutorial videos on Blackboard.

Equipment: Personal computer with Logisim installed.

Objectives: When you have completed this tutorial, you will be able to:

- Use parts in the Logisim Library; place those in the schematic window and connect those using wires.
- Apply names to signals and devices.
- Create, and modify the appearance of a subcircuit.
- Use the self-created subcircuit in Logisim library.
- Create bus lines and breakouts.
- Use the multi-bit keyboard and hex display and connect to the outputs or inputs respecting the correct bit order.

Introduction

The purpose of this lab is to get you acquainted as quickly as possible with Logisim. For a more thorough tutorial and reference information, refer to "The Guide to Being a Logisim User" at http://ozark.hendrix.edu/~burch/logisim/docs/2.7/en/html/guide/index.html.

Task A: Creating a Subcircuit

First, create the following circuit using Logisim:

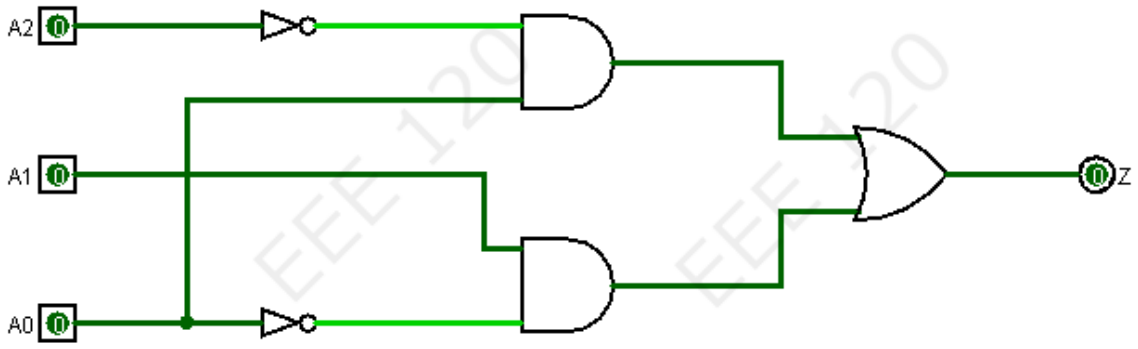

Figure 1. SUB_TRY circuit (SUB_TRY.circ)

Next, you will test the circuit to verify if it works correctly. Create a truth table for the output function. Toggle the input pins and observe the output pin. Methodically try every input combination and compare the results with that of your truth table. If your circuit does not perform correctly, use the hand tool to probe the wires and it will show the logic value of the wires. This will help you pinpoint the error. Once you are convinced that the circuit works properly, save the circuit as **SUB_TRY.circ**.

In later labs, you will build a simple microprocessor using a hierarchical design philosophy. This means that you will build simple blocks and combine these blocks to build more and more complex blocks. Now you will create a **subcircuit** from the SUB_TRY circuit you just built.

First create the following subcircuit symbol:

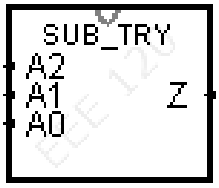

Figure 2. SUB_TRY subcircuit symbol

Next create a new circuit called **subdemo.circ**. In this new circuit, load your SUB_TRY subcircuit from Logisim library and add input and output pins as shown below:

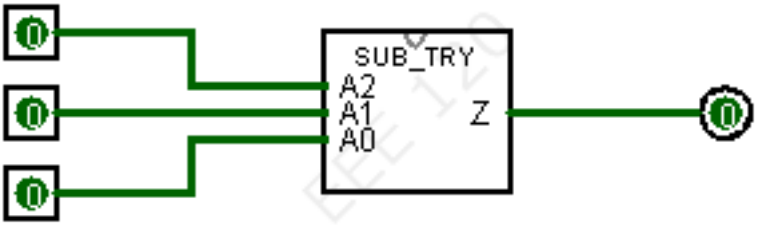

Figure 3. SUB_TRY subcircuit testing set up (subdemo.circ)

Now you can test and verify the truth table for your subcircuit is the same as the one you did earlier. You can double click on the subcircuit part to view the subcircuit internals.

Task B: 4-Bit Keyboard

Bus lines are a collection of wires (connections) grouped together. Using buses simplifies large circuit diagrams by reducing the number of signal wires cluttering the design window. The simpler your circuit diagram is, the less likely you are to make mistakes. Bus lines are not essential or even desirable in many design layouts; however, using bus lines will greatly simplify the circuit schematics you will build and simulate when you complete the microprocessor simulation exercises in future labs.

Create the following circuit **kbddemo.circ**:

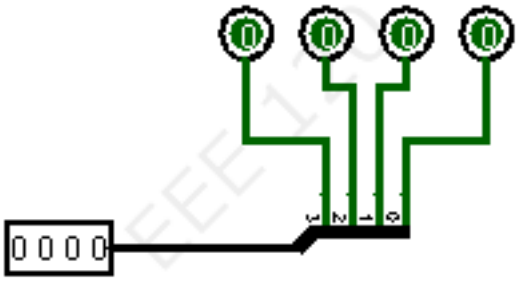

Figure 4. 4-bit keyboard (kbddemo.circ)

The circuit uses a multi-bit input pin (4-bit, in this case). The 4-bit input is connected to a 4-bit bus. A splitter is used to break out the individual bit 0, 1, 2 and 3 of the bus to four wires. The bit wires are then connected to four output pins. You can toggle each individual bit to either zero or one and you will see the bit value change at the corresponding output pin.

Task C: Hex Display

In the design of digital systems, we will often use buses containing many signal lines. One way to display the values on each line is to connect each to an output pin. This can be a tedious process. The tedium can be reduced, if we use devices that can display user-specified groups of signals. Devices that can display signals in groups of four are known as hexadecimal displays. To use these devices appropriately, you must be able to translate between

hexadecimal and binary number representations. If you already know how to translate between these two number systems, you can go directly to the building circuit part. If you don't, the following paragraphs will give you enough background to be able to complete this lab tutorial.

There are three number systems that we will be using in the labs: binary, decimal, and hexadecimal. We are all familiar with the decimal or base-ten number system; it uses the symbols 0 through 9. In the binary number system, the symbols 0 and 1 are used. The correspondence between the representations of these number systems is shown in Table 1.

The hexadecimal number system is a base-16 number system and is particularly useful when dealing with binary systems that use many bits of precision. This system uses the 16 symbols, 0 through 9 and A through F. Table 1 shows the 16 hexadecimal numbers with their binary and decimal equivalents. For example, the hex (short for hexadecimal) number '4' is equivalent to binary '0100' and hex number 'B' is equivalent to binary '1011'.

Table 1: Hexadecimal (Hex) Numbers and their Binary Equivalents.

Decimal	Hex (H)	Binary (B)
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

Now create the following circuit **dispdemo.circ**:

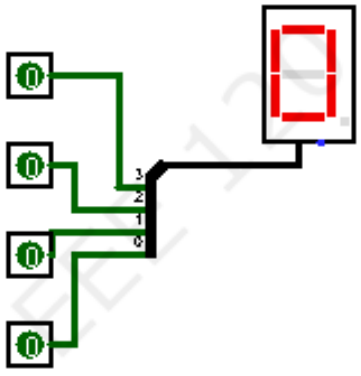

Figure 5. Hex digit display (dispdemo.circ)

In the circuit, four input pins are connected to a splitter. The four individual bits are combined to a 4-bit bus. And the bus is connected to a hex digit display. Test the circuit by toggle the input pins to show all 16 numbers in Table 1.