MAT 3670: Lab 4
Digital Logic Structures — Part I

Background

This lab will give you a chance to design and simulate some digital logic circuits, using a digital design tool known as **Logisim**[1]. This software allows you to draw diagrams representing logic circuits and, more importantly, verify the designs operate as intended. It's also fun to use!

Pre-Lab Exercises

Read all of the laboratory exercises to understand what you will be asked to do in the lab. It is to your advantage to produce "paper designs" of all circuits before arriving at the lab[2].

Laboratory Exercises

1. Create a **lab4** directory to store the files (circuits) you will create for this lab.

2. Launch Logisim by clicking on the OR gate icon, found on the OS X dock.

3. Familiarize yourself with the icons on the Logisim toolbar. If you move your mouse over each icon you will see a brief hint as to its purpose. These tools will allow you to create gates (NOT, AND, OR), add input and output pins, add wires, edit circuit components, and change logic values. You can also use the text tool to annotate your diagrams. There are many other possibilities, but these are the most basic ones.

4. Open the **Help** menu. Read and follow the steps described in the tutorial. By doing this, you will create an XOR gate made up of two AND gates, two NOT gates, and an OR gate. The mathematical basis for this circuit is the following:

$$x \text{ XOR } y = \overline{x}y + x\overline{y}$$

5. After you have completed and tested your design, save your work in **lab4-xor.circ**. Your design will look similar to the one shown in Figure 1.

6. Knowing these basics of Logisim, you can create and test many digital logic circuits. However, if you want to build larger circuits, knowing about subcircuits is very helpful. Subcircuits allow us to construct basic "building blocks" which can simplify the design task. To find out how to design and use subcircuits, read and follow the steps in the "Subcircuits" section of the tutorial. You will end up with a 4-to-1 multiplexor, which should be saved in **lab4-mux.circ**.

7. Test the MUX you just created as follows. Set exactly one of the input lines to 1. Then cycle through all four bit patterns for the two "select" lines, ensuring that the output is correct in each case. Repeat this process for a total of four times, each time allowing exactly one input line to be 1. By doing this, you will have tested 16 different possibilities. An exhaustive test would require 2^6 or 64 cases.

---

[1] Logisim is open-source software. If you wish to download a copy for your own computer, see `http://ozark.hendrix.edu/~burch/logisim/`.

[2] Hmmm; think before coding? Think before designing circuits! In addition, some artistic skill is needed to obtain pleasing diagrams.

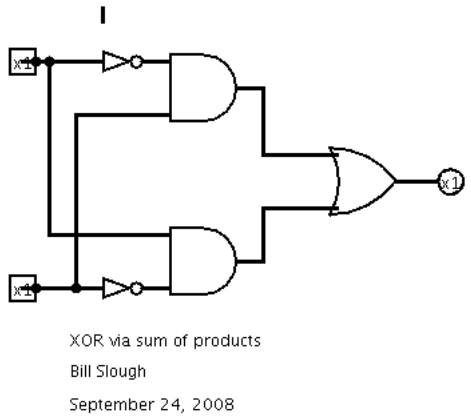
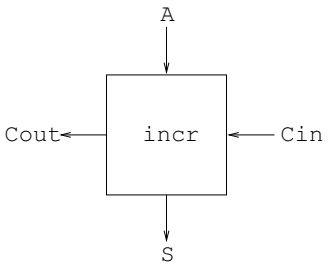


Figure 1: Implementation of XOR with fundamental gates.



A	Cin	S	Cout
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Figure 2: One-bit slice incrementer. On the left is a "black box" view; to the right is its definition via a truth table.

8. Create a new sub-circuit which will perform 1-bit incrementation. Viewed as a "black box," there are two inputs: A, and Cin, and two outputs, S and $Cout$. Our intention is to perform one slice of a multi-bit incrementation. Figure 2 provides both a "black box" view and a truth table definition.

   Determine a **sum of products** form for each of the two outputs, S and $Cout$, then implement this as a sub-circuit. Test your sub-circuit to ensure it matches the definining truth table, then construct a circuit with four of these slices which will increment a 4-bit quantity. Test your design by trying all possible inputs.

9. Using Figure 3 as a guide, show how a circuit can be constructed to increment a 4-bit quantity. Test your design by trying all possible inputs. Save your circuit in `lab4-incr.circ`.

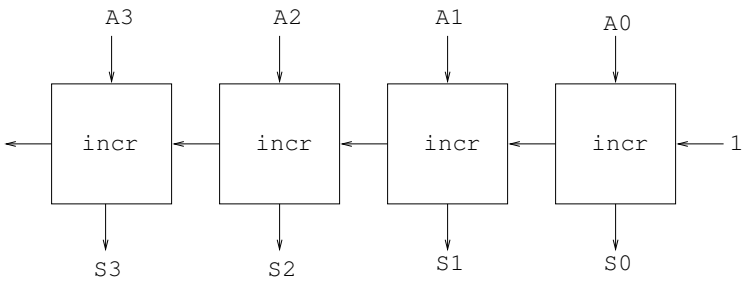


Figure 3: Four slices of a one-bit incrementer. The leftmost carry-out can be ignored.

10. A full adder is a logic circuit that adds three 1-bit binary values, X, Y, and Cin, to form a 2-bit result consisting of a sum S and a carry $Cout$. The truth table corresponding to a full adder is as follows:

Inputs			**Outputs**	
X	Y	Cin	$Cout$	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

    Write a sum of products form expression for $Cout$ and S. Using these Boolean expressions, design a sub-circuit which will perform one-bit addition. Using the truth table as a guide, test your design.

11. Using the one-bit adder sub-circuit, design a 4-bit adder. Test some representative inputs. If you wanted to perform an exhaustive test, how many different inputs would you need to verify? Save your circuit in **lab4-add4.circ**.

12. Your final task is to implement both subtraction and addition in a single circuit, following the hints in Exercise 3.24. Your circuit will be similar to the 4-bit adder circuit, but you will now need an additional input line M which will serve as a *mode* bit: when $M = 0$, your new circuit should behave like an adder, producing $A + B$; when $M = 1$, the circuit should perform subtraction, yielding $A - B$. Assume both A and B use a two's complement system. Test your design. Save your circuit in **lab4-addsub4.circ**.

Submissions

When you have completed the lab, submit your `lab4` folder by dragging it onto the EIU submit icon.