Computer Science 110 **Name:**
Algorithms to Machines Laboratory

From Logic to Function

Today's lab will build on the previous introduction to *Circuit Sandbox* as you will design your own circuits to have specific functional purposes. In this lab we will see how it is possible to do comparisons of values and mathematics with circuits.

To begin, launch *Circuit Sandbox*.

Expressions to circuits

1. Build the circuit described by this logical expression

    [(P+Q)'(PQ)']'

then test to see what the circuit does. Draw a picture of your working circuit.

Suggest a simpler circuit which would give the same result.

Designing function

2. EQUALITY circuit

This circuit will have an output of 1 if both of the inputs are the same, otherwise the output is 0. Draw a picture of your working circuit.

3. NON-EQUALITY circuit

This circuit will have an output of 1 if both of the inputs are the different, otherwise the output is 0. Draw a picture of your working circuit.

4. The non-equality circuit is extremely useful in many applications. One of the standard logic components provided by circuit simulator performs exactly this same function. Experiment with components until you find the one that is equivalent to the NON-EQUALITY circuit. Which component is equivalent to the NON-EQUALITY circuit?

Connect this component to a NOT gate. Now what have you made?

5. GREATER-THAN circuit

Design a logical circuit to compare two values then output 1 if the first value is greater than the second value, otherwise the output is 0. We will assume that our values are two bits in length. Complete the logic table before beginning to design the circuit

A		B		Output (A>B)
0	0	0	0	
0	0	0	1	
0	0	1	0	
0	0	1	1	

Draw a picture of your working circuit.

Arithmetic from logic
In binary math, we add zeroes and ones just like in decimal math, for example,

```
      0         0         1         1
+     0    +    1    +    0    +    1
----------- ----------- ----------- ----------
      0         1         1        10
```

Or, put another way,

0	+	0	=	0	then carry the 0
0	+	1	=	1	then carry the 0
1	+	0	=	1	then carry the 0
1	+	1	=	0	then carry the 1

6. Design a sum and carry circuit with the following properties.

Inputs		Outputs	
In-1	In-2	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Note that there will be two input switches and two output lights along with the other logic components. If you do this wisely, you will only need to use two logic components. Draw a picture of your working circuit.

7. We will save this as a custom component called HA (for Half-Adder). To do this delete everything from the sandbox except your completed circuit (and its switches and LEDs) then select the menu choice

File | Save As Custom Component…
Component Name: HA
Click "Save"
File: HA
Click "Save Component"

You can now drag the HA component from the Custom Components onto the sandbox. Note that this component will have two inputs and two outputs. It's important to know which of the outputs represents "sum" and which represents "carry". Experiment with your component.

8. Which of HA's outputs (the upper or lower output) represents the "carry" result?

9. The HA circuit is called a “half”-adder because it uses the idea of carry-out but it does not all a third bit to be carried-“in” from the previous position. Let’s construct a Full-Adder as follows.

Using two HA components, connect the “Sum” output from one of the HA’s to one of the inputs of the other HA. Place an OR gate in the sandbox, then connect the two carry outputs from the HAs to the OR gate.

You should now have a place for three switches, and two lights. If it is working properly, you have designed a full-adder. Test to be sure that it works as expected:

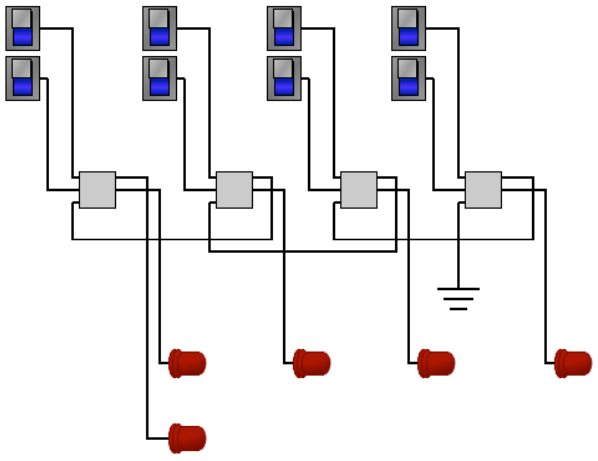
Inputs			Outputs	
In-1	In-2	Carry-In	Sum	Carry-out
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

If it works properly, save this as a custom component called FA.

Draw a picture of your working circuit.

10. We now have a new custom component that does simple adding for one-bit numbers. We can use copies of this component to do more complex addition. Let’s try doing 4-bit arithmetic.

You will need 4 FA components, 8 switches, 5 lights, and a ground terminal. Connect them as follows (be sure that the carry-out from any FA is connected to the carry-in of the FA on its left):



11. Test your 4-bit adder by trying all possible 4 bit binary math problems:

```
0      +      0      =
0      +      1      =
0      +      10     =
0      +      11     =
…
1111   +      1101   =
1111   +      1110   =
1111   +      1111   =
```

Do the lights properly show the results of these math problems?