2 ADDER

2.1 PURPOSE

This lab builds a full 1-bit adder, but the intent is to continue to familiarize students with *Logisim-Evolution* and how basic arithmetic functions can be completed using simple gate-level logic. Additionally, this lab develops an automated testing system that will be used to test future lab submissions.

2.2 PROCEDURE

2.2.1 *Half-Adder*

Open Logisim and start a new project. In *Logisim-Evolution* circuits can contain any number of sub-circuits. Subcircuits fill the same role in a physical circuit as a function or procedure fills in a software project. A new subcircuit can be added to a circuit by clicking PROJECT -> ADD CIRCUIT. Name the new circuit `Half_Adder`. Open the new subcircuit by double-clicking its name in the Explorer Pane.

Because this is a new subcircuit, the drawing canvas is blank. A half-adder is a circuit that will add two input bits. Because it is possible to add two bits and generate a carry, the half-adder must allow for a carry out bit. Figure 2.1 is the circuit diagram for a half-adder.

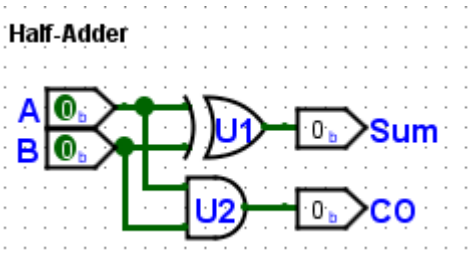

Figure 2.1: Half-Adder

This is fairly easy to build. Component `U1` is an `XOR` gate and `U2` is an `AND` gate. There are two inputs and two outputs and all of the devices should be connected as shown. To test the circuit, whenever input *A* and input *B* are different the *Sum* should be high and when input *A* and input *B* are both high the *CO* (Carry Out) bit should also be high.

2.2.2 *Full Adder*

A full adder takes two input bits and adds them, like a half-adder, but it also includes the logic necessary to input or output a carry bit so it can be cascaded with other adders. Figure 2.2 is the logic diagram for a full adder.

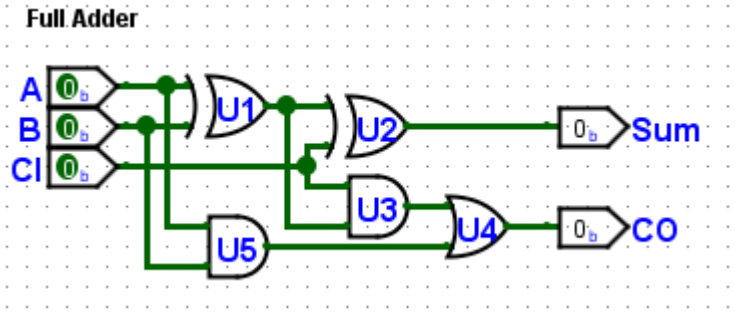

Figure 2.2: Full Adder

Create a new subcircuit named `Full_Adder` and wire all of the components as illustrated in Figure 2.2. Notice that `U1` and `U2` are `XOR` gates, `U3` and `U5` are `AND` gates, and `U4` is an `OR` gate. There are also three inputs and two outputs.

2.2.3 *Main Circuit*

In most *Logisim-Evolution* projects, the `main` circuit is used to provide a nice user interface for the project. Typically, various subcircuits are dropped onto the main circuit canvas and various inputs and outputs are wired to them. A user can then test the project without worrying about the details of the subcircuits.

For this project, open the `main` circuit by double-clicking its name in the Library panel. Click one time on the `Full_Adder` subcircuit then move the mouse over the drawing canvas. Notice that the mouse pointer has changed into a representation of the subcircuit. Drop that subcircuit anywhere on the drawing canvas and then wire the various inputs and outputs as shown in Figure 2.3.

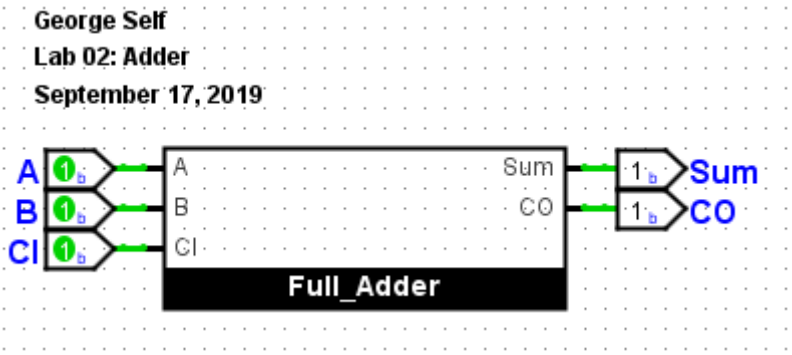

Figure 2.3: Full Adder

This circuit can be tested by using the *poke* tool and entering these values. After each line, check to see that the outputs are correct.

Inputs			Outputs	
A	B	CI	Sum	CO
0	0	0	0	0
0	1	0	1	0
1	0	0	1	0
1	1	0	0	1
0	0	1	1	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	1

Table 2.1: Test Vector For Full Adder

2.2.4 *Automated Testing*

While it is possible to use the *poke* tool and check the outputs for various input combinations, as digital logic circuits become more complex it is important to automate the testing process so no input combinations are overlooked. *Logisim-Evolution* includes a SIMULATE -> TEST VECTOR feature that is used for automating circuit testing.

The first step in using automatic testing is to create a *Test Vector* file. This is a simple *.txt* file that can be created in any text processor, like *Notepad*. The format for a test vector is fairly simple.

Do not use a word processor to create the Test Vector since that would add unneeded codes for things like fonts and margins.

- Every line is a single test of the circuit, except the first line.
- The first line defines the various inputs and outputs being tested.
- Any line that starts with a hash mark (#) is a comment and is ignored.

Following is the test vector file used to test the `main` circuit.

```
1  # Test vector for Adder
2  CI A B Sum CO
3  0 0 0 0 0
4  0 0 1 1 0
5  0 1 0 1 0
6  0 1 1 0 1
7  1 0 0 1 0
8  1 0 1 0 1
9  1 1 0 0 1
10 1 1 1 1 1
```

Following is an explanation for the *Test vector for Adder* file.

LINE 1 This is just the title of the file. Because this line starts with a hash (#) it is a comment and will be ignored by *Logisim-Evolution* .

LINE 2 This line lists all of the inputs and outputs in the circuit under test. In this case, there are three inputs, *CI*, *A*, and *B*, along with two outputs, *Sum* and *CO*. *Logisim-Evolution* is able to determine whether the pin is an input or output from its properties.

LINE 3 This line contains the first test for the circuit. This line specifies that *Logisim-Evolution* make *CI*, *A*, and *B* equal to zero and then check to be certain that *Sum* and *CO* are also zero.

OTHER LINES All other lines set the three input bits and specify the expected response in the output bits.

To start a test, click SIMULATE -> TEST VECTOR. The window illustrated in Figure 2.4 opens.

Figure 2.4: Test Vector Window

Click the *Load Vector* button at the bottom of the window and load the test vector file. The test will automatically start and *Logisim-Evolution* will report the results, like in Figure 2.5.

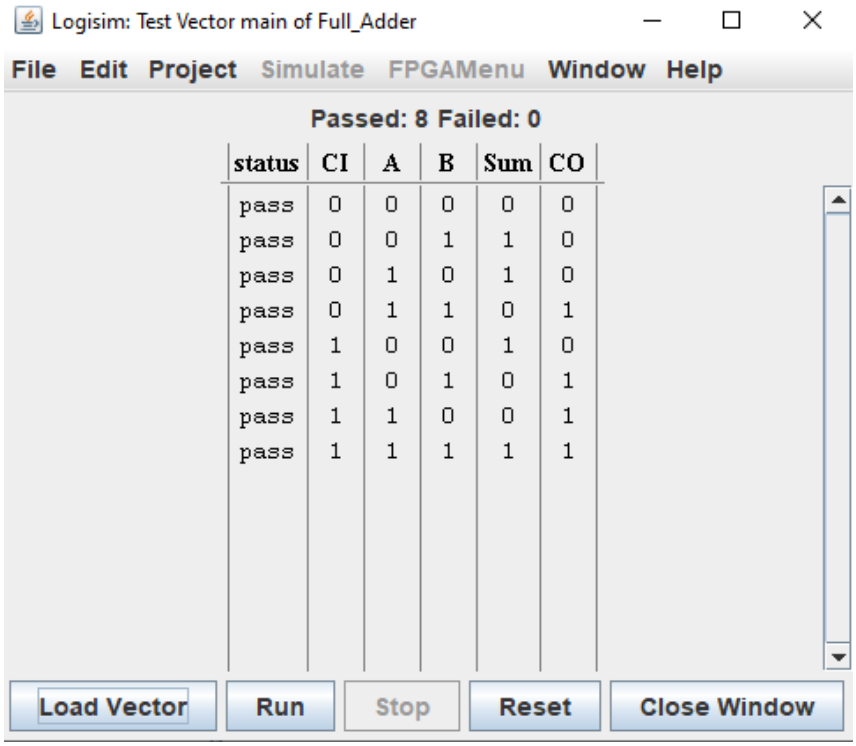

Figure 2.5: Test Completed

An error was intentionally added to the test vector file to generate a failed test.

The test indicates all 8 lines passed and zero failed so it could be reasonably concluded that the circuit is functioning properly. Figure 2.6 illustrates a failed test. The circuit designer would then need to troubleshoot to determine what went wrong with the circuit.

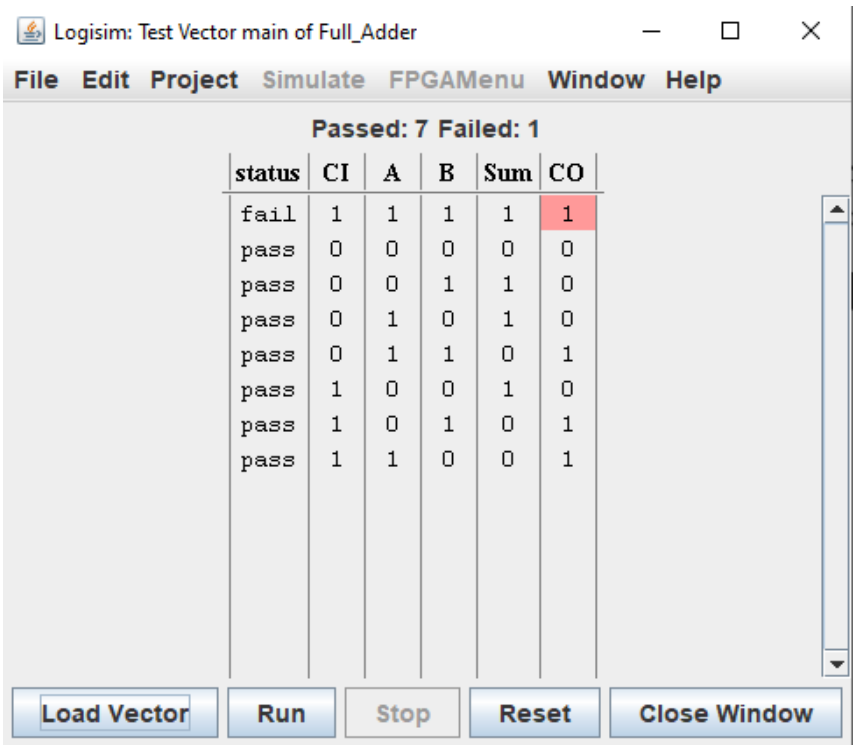

Figure 2.6: Test Failure

Note: the test vector files for all labs are made available to students so they can check their work prior to submitting them.

2.3 DELIVERABLE

To receive a grade for this lab, build both the half-adder and full adder and then add the full adder to the `main`. It is important to ensure the input and output pin names are the same as in the lab instructions since a test vector will be used to check the circuit. Be sure the standard identifying information is at the top left of the `main` circuit, similar to:

```
George Self
Lab 02: Adder
September 17, 2019
```

Save the file with this name: *`Lab02_Adder`* and submit that file for grading.