**Name:** Nick Petro **CoE 0147**

# Lab 6, Spring 2016

Introduction to Logisim

**Important:**

- While collaborations are allowed, names of all the collaborators must be listed in the assignment. Each student should submit individual assignments for grading, despite collaboration.
- Solution files (***.circ files**) for this assignment should be submitted to *Courseweb/Documents/ Recitations/Lab9* before the next recitation. Please zip the files together and submit it.

Logisim is a tool for designing and simulating logic circuits. It can be downloaded from:
*http://www.cburch.com/logisim/index.html*

**Note: For this lab, you are allowed to use only 2-input gates in your circuit designs.**

## 1 1-bit Adder

A 1-bit adder has 3 inputs— two input bits and a carry-in bit. It produces 2 output bits— the sum bit and the carry-out bit. Several 1-bit adders can be joined together to build an 'n-bit' adder. Fig. 1 shows the circuit of a 1-bit full adder.

With respect to Fig. 1:

- Square boxes are **input pins**. Their values are used to compute the result.
- Circular boxes are **output pins**.
- Triangles represent **NOT** gates. Output of a NOT gate is complement of the input.
- Green lines depict wires. Wires with bright green color are **ON** (i.e., true/1). Wires with dull green color are **OFF** (i.e., false/0). Blue wires are unconnected wires and are ignored by the simulator. Red wires (if present) indicate error.

Components required to build a circuit are available in the 'main' tool bar (below the 'File' menu).
The poke tool (hand icon) allows you to change the values of the input pins, so that different inputs can be tested. The arrow tool allows you to add, select, and manipulate wires. You can "un-draw" wires to shorten them, or use the delete key to remove a wire. Refer to 'help' menu for detailed information on using Logisim.

**Design a 1-bit adder as shown in Fig. 1 and save this file as lab6part1.circ.**

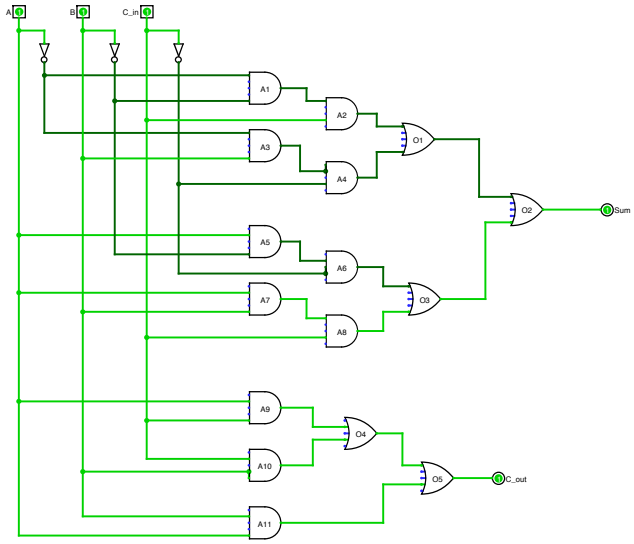


Figure 1: 1-bit Full Adder

## 2 Adder Analysis

The circuit that you built in Sec. 1 implements the following boolean equations:

$$S = ABC_{in} + \overline{A}\,\overline{B}C_{in} + \overline{A}B\overline{C}_{in} + A\overline{B}\,\overline{C}_{in} \tag{1}$$

$$C_{out} = BC_{in} + AC_{in} + AB \tag{2}$$

By changing the values of the input pins, complete the following truth table. A truth table describes the behavior of a circuit given all possible inputs. Feel free to type up the table in a word processor, spreadsheet editor, etc. Save this file as lab6part2.`ext` where `ext` is an appropriate file type extension (e.g., txt, xlsx, docx, csv). **Do not submit a hard copy of this table.**

| A | B | $C_{in}$ | S | $C_{out}$ | A1 | A2 | A3 | A4 | A5 | A6 | A7 | A8 | A9 | A10 | A11 | O1 | O2 | O3 | O4 | O5 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | | | | | | | | | | | | | | | | | | |
| 0 | 0 | 1 | | | | | | | | | | | | | | | | | | |
| 0 | 1 | 0 | | | | | | | | | | | | | | | | | | |
| 0 | 1 | 1 | | | | | | | | | | | | | | | | | | |
| 1 | 0 | 0 | | | | | | | | | | | | | | | | | | |
| 1 | 0 | 1 | | | | | | | | | | | | | | | | | | |
| 1 | 1 | 0 | | | | | | | | | | | | | | | | | | |
| 1 | 1 | 1 | | | | | | | | | | | | | | | | | | |

## 3 Timer Alert

In this section, you will create a 'TimerAlert' circuit in Logisim, as shown in Fig. 2

1. Start a new Logisim circuit.
2. Click "Project" then "Add Circuit" to add a subcircuit. A subcircuit can be designed once and instantiated multiple times. Name this subcircuit as "TimerAlert".
3. Add 2 input pins (square) and assign appropriate labels to them.

4. Add a counter (available under "Memory" in the explorer pane). Change its "Action on Overflow" property to "Stay on Value". This will ensure that when the counter reaches its maximum value, it does not wraparound to 0. Change its "Data Bits" to 10.

5. Connect the input pins to the counter as shown in Fig. 2.

6. Draw a short wire coming out of "Q" output of the counter. This is the counter's value. The output is, by default, a bundle. A bundle refers to several (in this case 10) wires connected together. Using a bundle saves space when working with multiple bits at once. A Bundle is always black in color, and it is not possible to see the values of the individual wires within a bundle.

7. Connect a splitter (from the "Wiring" category) to the end of the bundle. Change the splitter's "Bit Width In" to 10 (because the counter's output is 10-bits). Change the splitter's "Fan Out" to be 10. This will split the 10-bit input into 10 parts resulting in 10 1-bit outputs.

8. Connect the 10 outputs from the splitter to AND gates as shown in Fig. 2. Connect the AND gates together.

9. Add 2 output pins and name them as shown in Fig. 2.

This completes the TimerAlert circuit. Now we can connect multiple TimerAlert circuits into a bigger circuit, capable of generating alert signal for multiple systems.

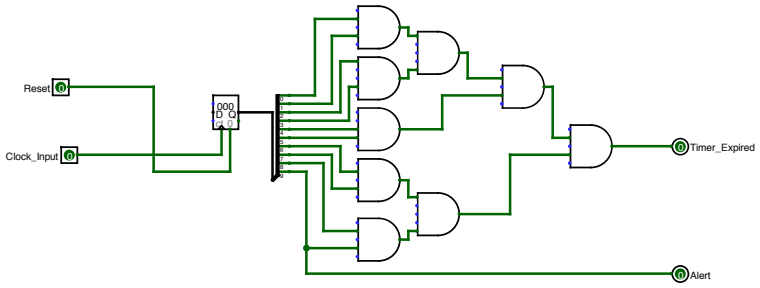


Figure 2: TimerAlert

1. Switch to the "main" circuit, which should currently be blank.

2. Add a clock component (available in the "Wiring" category).

3. Add 6 buttons (available from the "Input/Output" category).

4. Add 6 subcircuits. To insert a subcircuit, single-click on the "TimerAlert" in the explorer pane, and then click inside the canvas to place it there. A subcircuit will appear as a small box. Repeat this 6 times.

5. With the arrow tool active, hover over the four dots on a "TimerAlert" circuit. Notice that the left hand ones are blue, indicating that they are unconnected and expect an input. The right hand ones are dull green, meaning that they are producing a 0. These dots are referred to as pins. Hovering over a pin tells you the purpose of the pin, assuming that you gave the subcircuit's pin a label. For

example, hovering over the top pin on the right hand side of a TimerAlert should show that the pin is a Timer_Expired pin.

6. Add 12 LEDs (under "Input/Output"). Connect them as shown in Fig. 3.

The circuit is complete now. To test it, we will force the clock to tick, thus updating the circuit's state. We can poke the clock to switch from low (0) to high (1) and vice versa. This allows us to step up through time, one cycle at a time.

The clock can also be forced to update automatically at some fixed frequency. Under the "Simulate" menu, go to "Ticks Frequency" and select "128 Hz".Now, under "Simulate", select "Ticks Enabled". The clock is connected to each TimerAlert, and each TimerAlert circuit will eventually turn on the lights. At 128Hz, the bottom light should turn on in about 4 seconds. The turning ON of the bottom light indicates that all the six systems are in the alert state. To reset a TimerAlert circuit, press its "Reset_Button".

If the alert warnings are ignored, each TimerAlert will expire after some time.

Make sure that your circuit is working as designed. You should understand what each of the components are doing and why they are doing what they do.
**Save your Logisim circuit file as lab6part3.circ.**

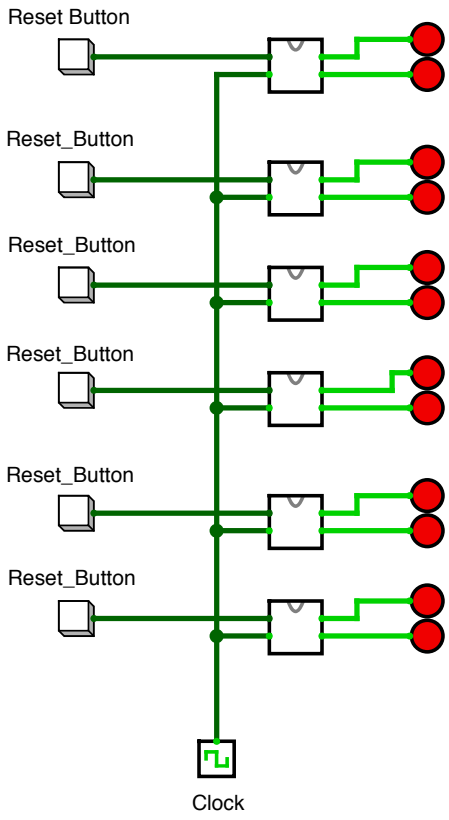


Figure 3: Six TimerAlerts