# Week 2 Lab A: Adding binary numbers

## Objectives

Develop understanding and experience of:

1. Adding binary numbers
2. Building components to add binary numbers
3. Using sub-circuits within Logisim evolution

This lab will build on last week’s lecture on binary numbers and need knowledge of logic gates from last week’s lab.

## Binary addition

1. Rules for adding binary numbers
   * 1 + 0 = 1
   * 1 + 1 = 10
   * 1 + 1 + 1 = 11
2. Work out the answers to following binary addition exercises using pen and paper where needed.

When we add up, we always start from the right-hand side and may need to carry to the next column to the left.

* + 101

1 +

======

110

======

* + 110001

1 +

==========

110010

==========

* + 1001

111 +

==========

10000

==========

1 1 1

* + 10101

111 +

==========

==========

## Binary addition using Logic gates – creating a half-adder

Consider adding two 1-bit binary numbers, all the possible outputs can be written in a truth table. Complete the truth table below for adding A and B. The output is represented by two bits, O1 and O0 with O0 as the lowest order digit (the units column) and O1 as the twos column. That means if your answer has only one digit, it should be put in O0 with O1 as zero. A circuit to do this operation is called a **half-adder.**

1. Fill in the truth table

|  |  |  |  |
| --- | --- | --- | --- |
| A | B | O1 | O0 |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 |

What gate is represented by the output O0? XOR

What gate is represented by the output O1? AND

1. Using the gates you wrote down above, **create** a half-adder using Logisim Evolution. There should be two input pins, labelled A and B and two output pins, labelled S (sum) (previously O0) and C (carry) (previously O1).
   * The inputs from A and B both need to go to **both** the gates, remember that a dot on a wire shows a signal that is being copied in two directions. Wires that cross over without a dot do not affect each other (think of them having plastic insulation).
2. Change the circuit name to half\_adder.
   * Note that spaces or hyphens cannot be used as names in Logisim Evolution.
3. Save and test the circuit by changing the inputs A and B
   * Remember that the arrow icon is for editing the circuit and the hand icon for testing it.

Paste images of your circuit here showing some testing.

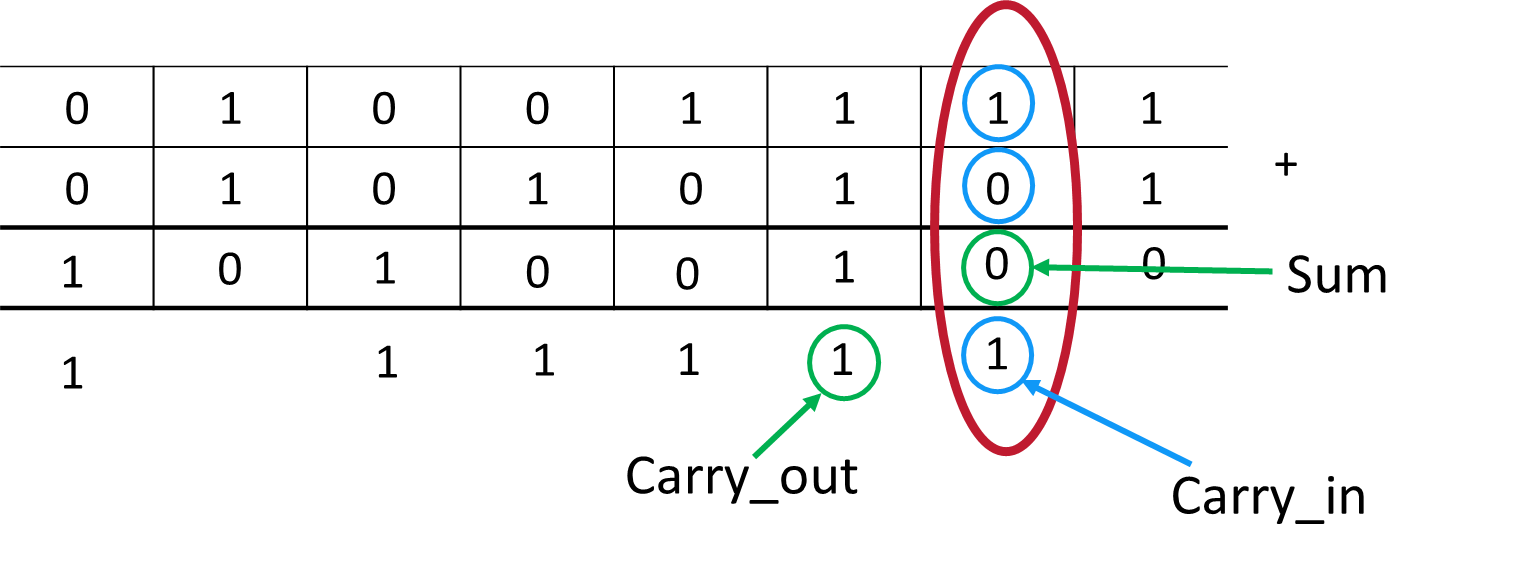
Diagram

Description automatically generated

1. Check that the truth table generated by Logisim evolution is the same as the one you have above.

## Creating a full-adder

To add longer numbers, we need to be able to include the carried bit into the calculation, so we need an adder that can take three bits of input, known as a **full-adder**. There will now be three inputs, A, B and C\_In and two outputs, C\_Out and Sum.

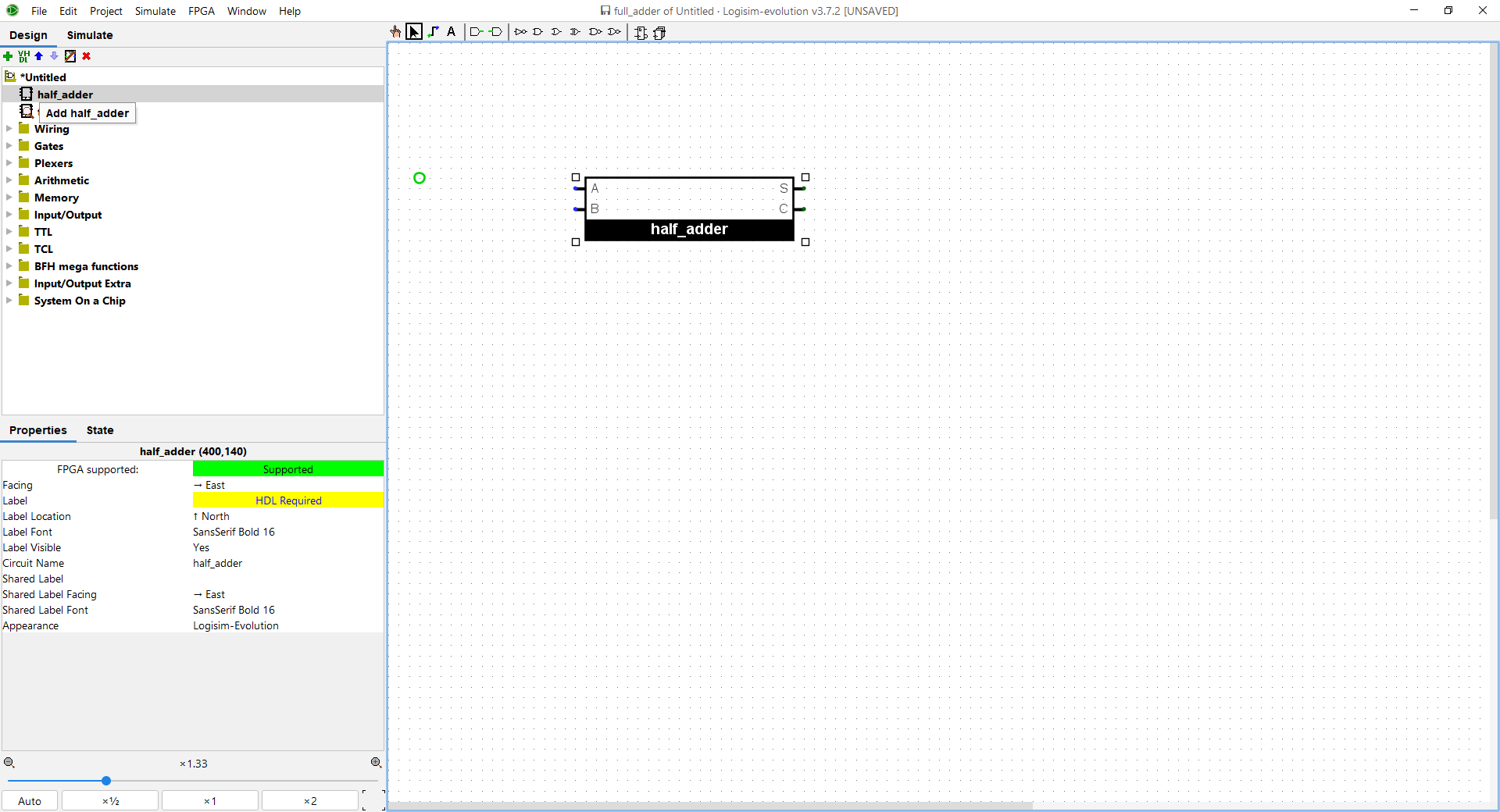


1. Complete the truth table showing all the possible outputs of a full-adder. Make sure that the inputs are given in a systematic order in the table (corresponding to the decimal numbers 0 to 7).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A | B | C\_in | C\_out | Sum |
| 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 |

We will use the half-adder to make the full-adder by using the half-adder as a sub-circuit. Sub-circuits will be used a lot in future work on this unit. To use a sub-circuit:

1. Create a new circuit in the same Logisim Evolution project as you created the half-adder. To do this you can right-click on the highest level folder in the project view on the left, or click the green “+” just above that. Name your new circuit full\_adder.
2. We can now add a half\_adder to the full\_adder circuit as a sub-circuit. I can highlight the half\_adder and place it in my circuit as if it was any of the components in Logisim Evolution. The labels that I used in the half\_adder now show in the component.



1. The challenge is to try to create a full-adder. The full adder can be created by combining **two** **half-adders** and an additional **OR gate**.
2. Make sure that you test your circuit and add some images showing your testing.

Possible approach

* Set up your three input pins and two output pins (described above).
* Two half-adders and an OR gate can be used to create a full-adder
  + Two of the inputs need to go to the first half-adder
  + The third input needs to be added to the result of the first half-adder using another half-adder
  + You need to work out what to do for the final sum and carry

Diagram

Description automatically generated

## Extension exercise

In the lab exercises above, you looked at a half and full adder. To add numbers consisting of multiple bits together, we need to connect multiple full adders. The simplest way to connect multiple full adders is called a **ripple-carry adder**.

Using several of the full adder that you created in the lab, connect them together to add two 4-bit inputs with a separate 1-bit pin for each bit. You might want to use one of the recommended textbooks, or search for ripple-carry adder to see how to connect your adders. Make sure that you consider how to thoroughly test your circuit. You should use your previous full-adder as a sub-circuit in your multiple bit adder. You will need to have a 1-bit **constant** with a value of 0 for the carry into the adder for the least significant bit. A tool to add a constant can be found in the wiring section on Logisim Evolution, you will need to set it to have a value of 0.

You might want to research other ways to create multi-bit adders and what advantages can be gained over the ripple-carry adder.

**Diagram

Description automatically generated**

Alternatives – included in Useful Links on Moodle

* + <https://www.youtube.com/playlist?list=PLCNL4SrVOxUWO79B_FL-RjXxblkTFyY2f>