View the original document in French: http://reds-data.heig-vd.ch/logisim-evolution/tutoLogisim.pdf
Translated to English by Moshe Berman.

# Introduction to Using Logisim

## 1. Introduction

Logisim is an open source program which allows you to design and simulate logical circuits.[1]

This document is a tutorial which describes how to establish a digital system with the help of the schematic diagram editor. We will explain the necessary steps to design, simulate, and implement a project based on the Altera EPM 25p-25p.

There are different ways to formally describe digital systems: hardware description languages (HDL,) truth tables, state diagrams, or schematics. Logisim allows us to work only with schematics. The first chapter will explain how to make our first diagram. In figure 1, you can see the Logisim user interface.

One of the features of Logisim is that it allows simulating and editing schematics at the same time. We will explain later in this document how to simulate a circuit and how to implement it on the laboratory card.

Open Logisim and you will be prompted to enter a username - see Figure 2. It will be used to tag each component you create, to prevent plagiarism.

1. Click *Change User*
2. Enter your first and last name in the *Add New User* box in the same form as your *Heig-vd* login, without special characters, separated by an underscore.
3. Click *Add*
4. Click *Close*
5. Click *Accept Conditions*

The list of users is saved on the machine. On subsequent uses, simply click *Accept Conditions* and select your active user from the list.

[1] The latest version of Logisim can be downloaded at https://reds.heig-vd.ch/share/ logisim-evolution/logisim-evolution.t.jar. Logisim contains an auto-update mechanism. As soon as a new version is available, you will receive a notification and you will have the option to update your copy.

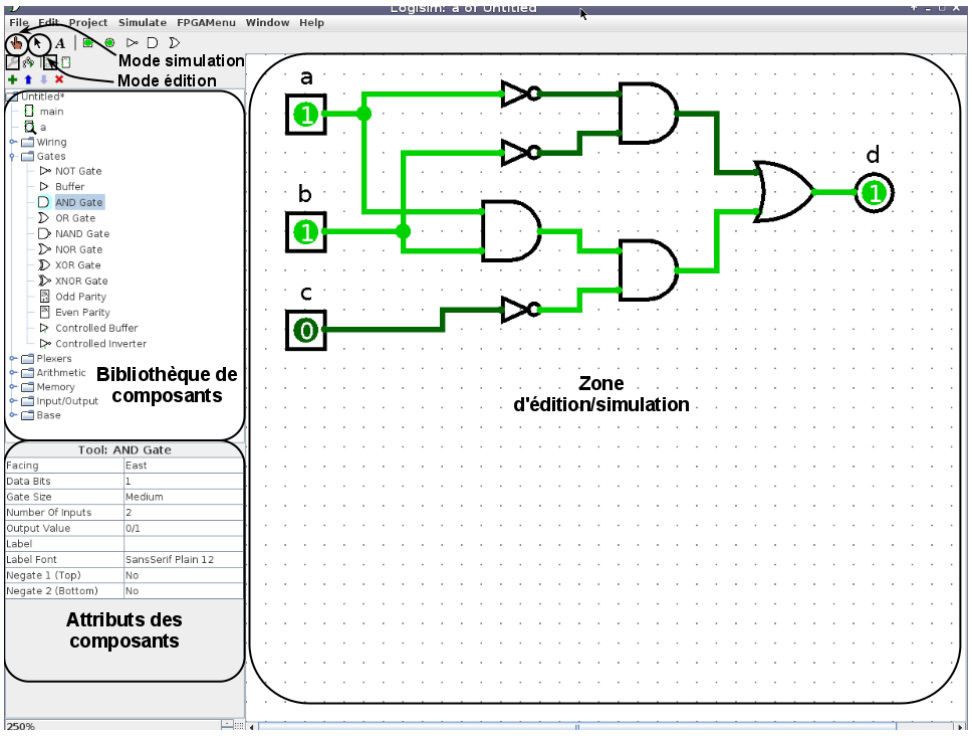


**Figure 1:** The Logisim interface


**Figure 2:** The login window

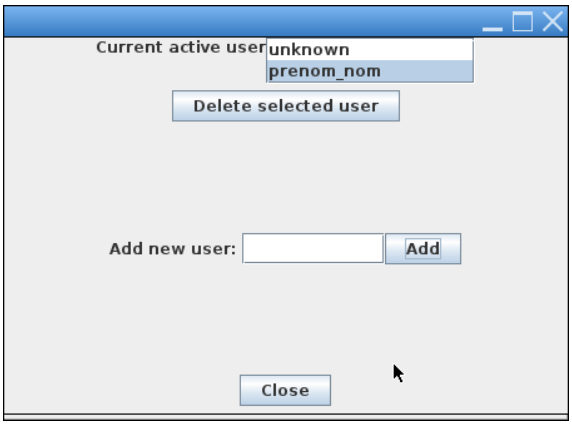


**Figure 3:** Add a user

## 2. Edit Mode

1. To use edit mode, simply select the arrow, as shown in Figure 1.
2. You can then select a component from the library on the left. To add it into the schematic, simply cluck on the desired component, then click on the diagram.
3. Every component you will use has modifiable attributes un the lower left area in Logisim. For example, if we set an AND gate, we can modify the number of signals it takes as an input, or invert one of its inputs.
4. It is also possible to copy/paste one or more components. In that case, pasted components retain all of the attributes that we previously defined.
5. Here is a list of the items you'll need for this tutorial:
   - For input, the *pin* and *wiring*.
   - For output, *pin* and *wiring* and the "output" attribute set to *yes*.
   - Logic gates are in the *Gates* library.
   - *Splitter* and *wiring.*
   - The *ground*, *power*, *and wiring.*
6. Once we lay out all of the components, we must connect them. To do this, simply place the mouse on one of the gates. Click and drag to the destination gate.

## 3. Creating our First Circuit

All circuits made in Logisim can be used in other circuits. To create a new circuit, navigate to *Project -> Add Circuit… ->* and then name the circuit. The newly created circuit will become available in the library.

### 3.1 Add1bit

Implement the schematic shown in Figure 4. Name it *Add1bit*. Notes:

- The currently edited circuit is the one that has a small magnifying class below the name of the project.
- Don't worry about the color of the wiring or the value of the input pin. The most recently added pin is blue by default.
- You can change the orientation of the components by changing the *Facing* attribute.

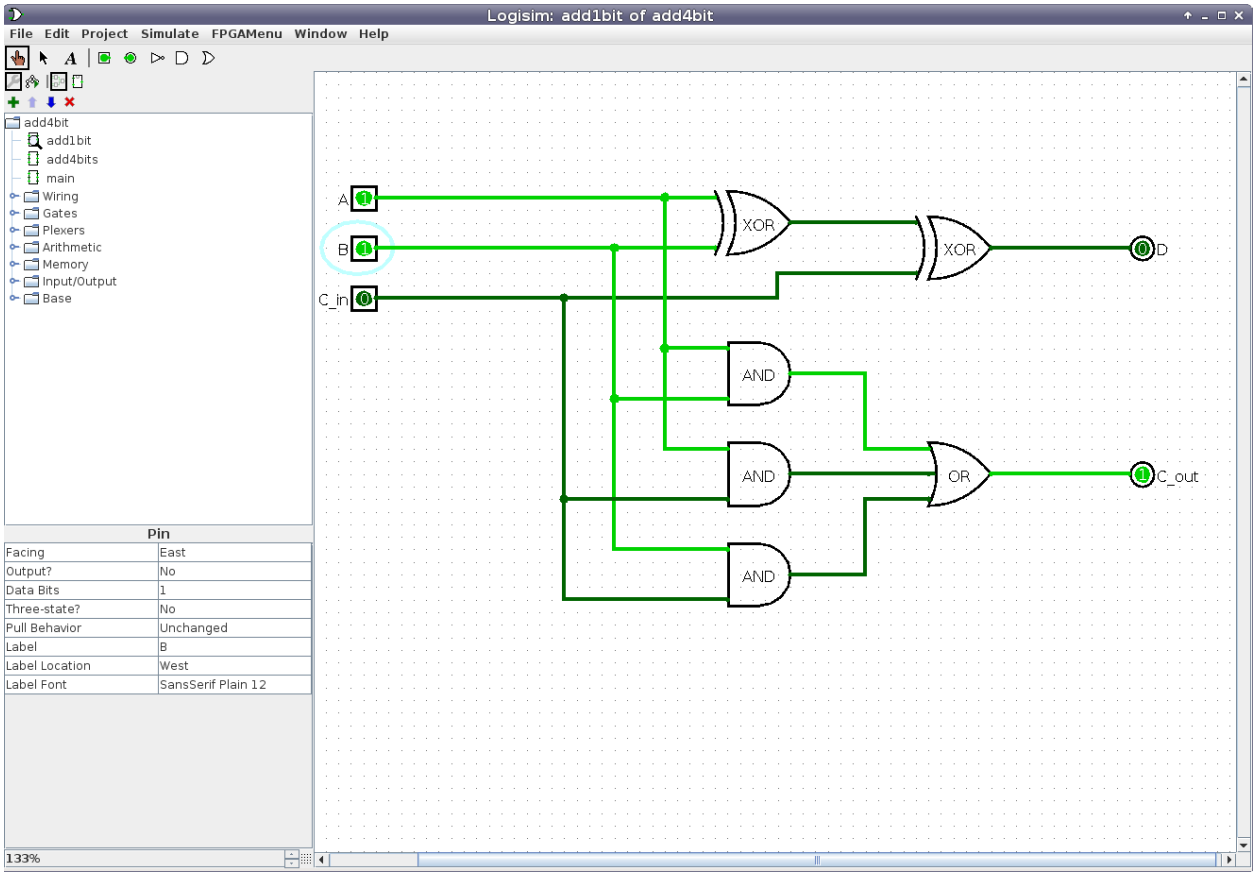


**Figure 4:** A 1-bit adder

## 4. Simulate Mode

Logisim is capable of simulating the circuit and showing the values of the signal directly in the diagram. The user can then define the input bit values and observe the reaction of the design.

1. To use simulate mode, select the hand in top left corner of Logisim.
2. Then, you are able to control the state of the different inputs by clicking directly on them.The blue X on the input pin represents high impedance. In the laboratory, we work only with *high* and *low* states. To remove the high impedance state, you must change the input pin attribute so that the three-state attribute is set to *no*.
3. By clicking on an input, its value will switch between *0* or *1*.
4. Here is a description of the colors used for the signals in simulate mode:

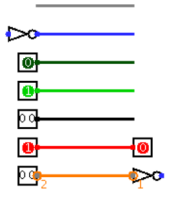

**Figure 5:** The colors of the wires in simulate mode.

- **Gray:** The wire's bit width is unknown. The wire is connected to any input or output.
- **Blue:** The wire contains a value, but we don't know what it is.
- **Dark Green:** The wire contains the value *0*.
- **Light Green:** The wire contains the value *1*.
- **Black:** The wire contains multiple bits. (A *Bus* for example.)

- **Red:** The wire contains an error, or undefined value.
- **Orange:** The components that are connected to the wire do not contain the correct bit width.

5. Test the functionality of your 1-bit adder.

### 5. Hierarchical Design

The design methodology that you just used is valid for designing rather simple digital systems, those with a low number of logic gates. When systems become more complicated, the number of gates and connections expands quickly. In those cases, the risk of introducing errors becomes substantial.

The key to properly managing greater design complexity is to user hierarchical design, because it can work at various levels of abstraction. The first level is using basic blocks, logic gates, and then using those blocks as parts of a larger system. In the case of our 1-bit adder, it can be used it to build a 4-bit adder. This new block can then also be used as part of a larger system.

To create a hierarchical design using our 1-bit adder, we will take the following steps:

1. Create a new circuit as we explained in section 3.1. To switch from editing one circuit to another, double click on the name of the desired one in the lefthand menu.
2. It is then possible to add a sub-circuit. Add the 1-bit adder in the same way that you use any component. Click on the menu shown in Figure 6, and then place it by clicking on circuit diagram.
3. If the circuit *Add1bit* was created correctly, then it should be represented by a small block with three blue dots on the left corresponding to the inputs, and two red dots on the right corresponding to outputs.

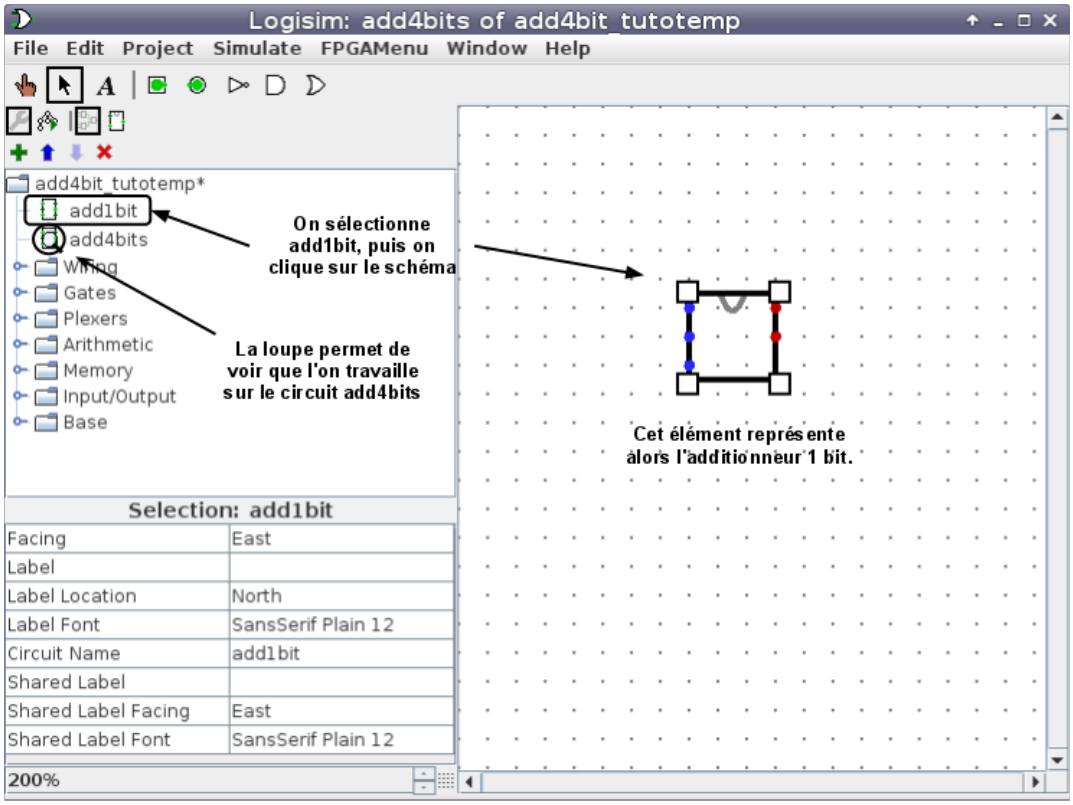


**Figure 6:** A subcircuit

4. If the outputs are blue instead of red on the diagram, check that you have set the *output* attribute to *yes* on the output pins.
5. To implement a 4-bit adder, you need four 1-bit adders. So, complete the diagram by connecting the input and output ports. One of the differences between the 1-bit and the 4-bit adder circuits is that in the case of the 1-bit adder the inputs and outputs were all independent, but with the 4-bit adder we use data buses for the input and output. For example, to set input A as a 4-bit bus, add a pin element and set its size to 4 using the *data bits* attribute.
6. When you pull a wire from one of these inputs, it's not a simple signal, but a 4-bit bus. To connect the inputs to the inputs of the 1-bit adders, we will have to separate the different outputs of the bus in order to process them one by one. The *splitter* wiring element allows such conversions in both directions: a 4-bit bus to a 4 wires, and 4 wires to a 4-bit bus. See Figure 7.

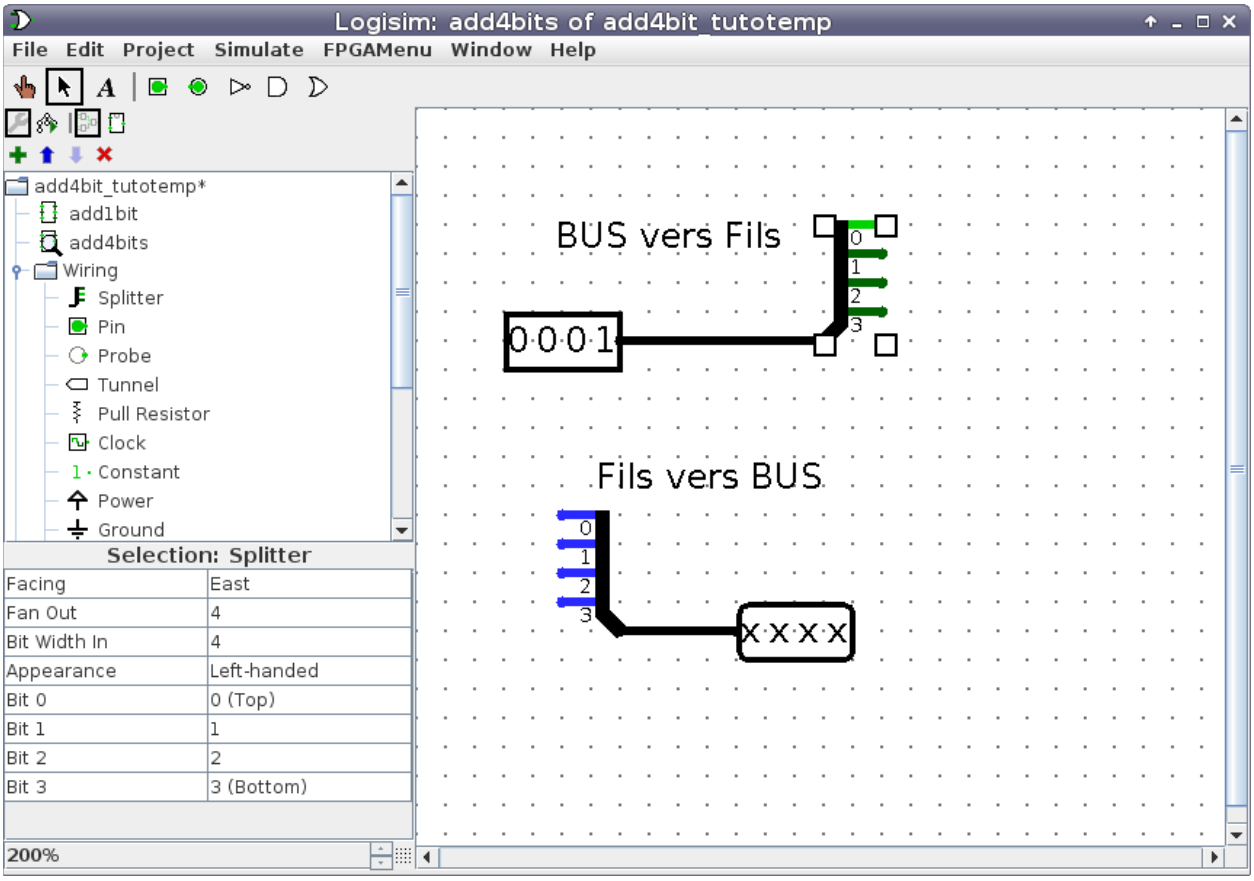


**Figure 7:** Example splitters

We define input and output sizes of the splitter via *fan out* and *bit width in* properties. In our case they are both *4*. Note: The least significant bit is indexed 0 to the output of the splitter.

### 6. 4-bit Adder

Make the 4-bit adder in Figure 8, and verify proper operation in simulation. In order to facilitate the reading of input / output values, you may have to to change the radix (binary/octal/decimal) in the pin options.

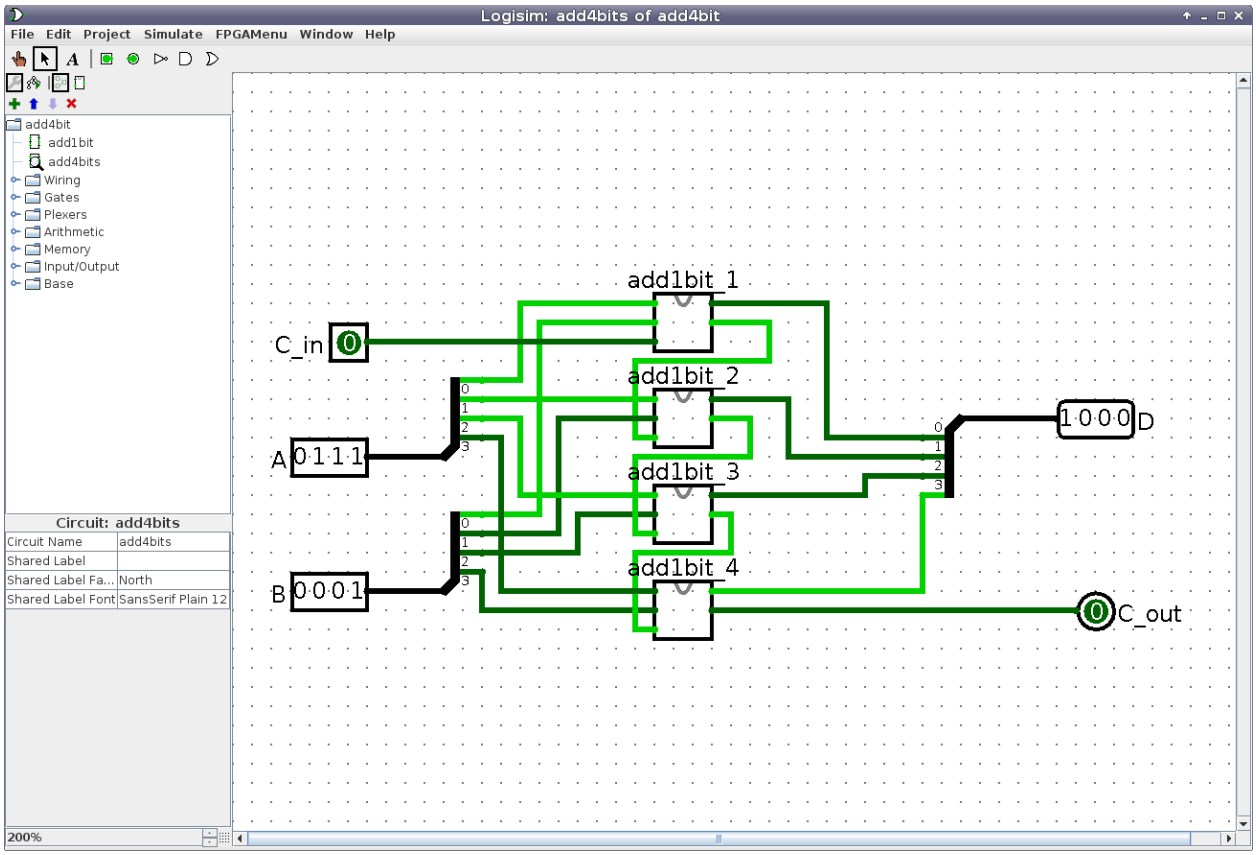


**Figure 8:** 4-bit adder