Logisim Tutorial

1 Frequently Asked Questions

- **What is Logisim?**
  Logisim is a digital design tool for educational purposes designed by Carl Burch of Hendrix University. Logisim can be used for the logical design of circuits and is the tool you will be using for the ECS 154a design projects
- **Where to get Logisim?**
  You can find Logisim at `http://ozark.hendrix.edu/~burch/logisim/index.html`, or by Googling "Logisim"
- **Where can I learn more about Logisim?**
  Outside of this tutorial the main website contains a substantial amount of documentation. Here I will highlight the basics.
- **When do I need to start this tutorial?**
  It is vital to get an early start in this class, particularly with the short summer session. You should familiarize yourself with the design environment **ASAP**!

2 Getting Started

In the remainder of this document I'll give you a quick overview of the most important aspects of Logisim. Be sure to check the website for additional information.

2.1 Environment Layout

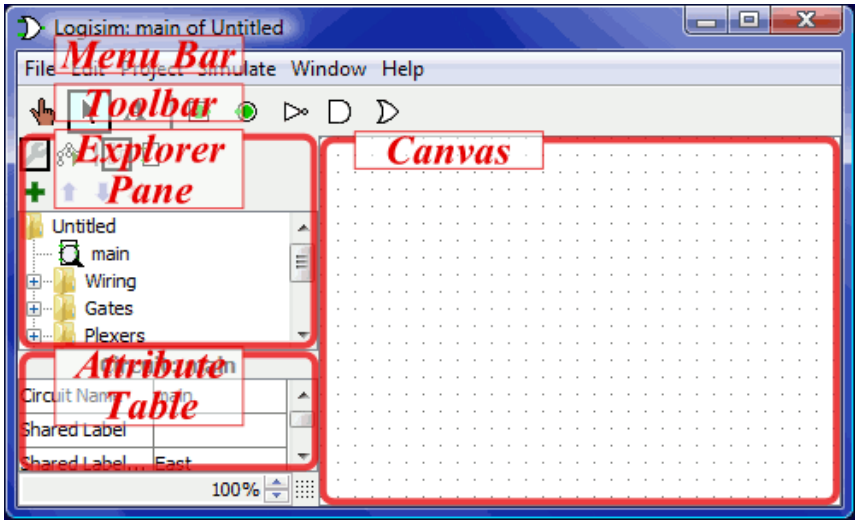


Figure 1: A labeled view of the user interface

- **Toolbar:** The toolbar contains short cuts to several commonly used items

- The *poke tool* (shaped like a hand) is used to alter input pins.
- The *input pin* (green circle surrounded by a box) is used to send a signal through a wire. When placing the input on the canvas it initializes to 1-bit. This number of bits can be increased in the Attribute Table.
- The *output pin* (green circle in a circle) is used to observe output from a gate. The output pin toggles in real time as long as the simulation is enabled from the menu bar Simulate > Simulate enabled

- **Explorer Pane:** The list of wiring, gates, multiplexers, etc... that are available for digital design in Logisim. Please note not all items are allowed to be used in every project.
- **Attribute Table:** Gives detailed attributes of digital design components (e.g., AND, OR, XOR gates). The attribute table allows you to alter the number of inputs/outputs that a digital design component.
- **Canvas:** The canvas is the area for you to create your digital circuits. In the canvas area you may simulate your circuits while designing in real time.

2.2 Quickie Design

Here we design a simple circuit that checks for equality between two 3-bit inputs. We begin by creating a circuit to determine equality between two single bits by building an XNOR gate. in the following manner:

- Begin by placing two AND gates and an OR gate
- Reduce the number of input bits to 2 for each of the gates via the *Attribute Table*.
- Place a NOT gate in front of one input of each AND gate as well as the output of the OR gate.
- Connect wires by dragging out from each input/output point as in Figure 2
- This creates an XNOR gate, which will output a 1 when both inputs are equal.

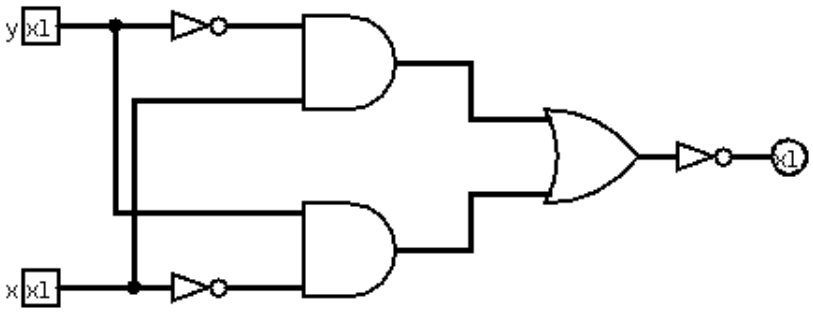


Figure 2: An XNOR gate created from AND, OR, and NOT gates.

Now that we can check for equality between 1-bit we will make our XNOR gate a sub-circuit. We will then use the XNOR gate to check for equality between the two 3-bit inputs.

- Begin by selecting Project > Add Circuit
- Name your new circuit 'equality'

- Right click on equality, and click 'Set as Main Circuit'
- You can now add your XNOR circuit into the equality circuit by clicking and dropping it as with any other gate.

We can now finish our equality circuit in a few simple steps.

- Add 3 of your XNOR circuits to the equality circuit.
- Add two inputs we will check for equality.
- Select each input and in the Attribute Table change the number of data bits to 3, the outgoing wire will now be a BUS (multiple wires that appear as one wire).
- Place two splitters to check individual bits. The splitter is found in Explorer Plane > Wiring > Splitter
- Select the splitters and change the *Bit Width In* and the *Fan Out* to 3-bits (because our inputs are 3-bits wide).
- From the splitter match each of the corresponding bits to an XNOR.
- Drop an AND gate onto the circuit
- Connect all three XNOR outputs to the AND gate
- Drop an OUTPUT into the circuit, and connect to the AND gate.

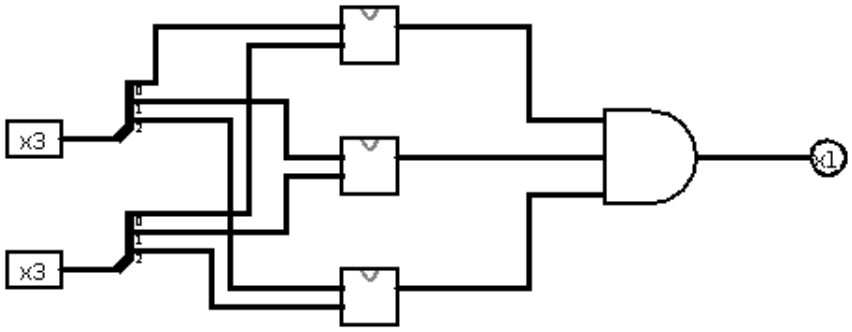


Figure 3: The complete equality circuit.

This circuit can now check for equality between the two 3-bit inputs. You can alter the inputs to test your circuit by selecting the poke tool (hand shaped tool) and clicking on the individual bits of the inputs. As you change the bit values you will notice the output changing, 1 indicating equality and 0 indicating not-equal.

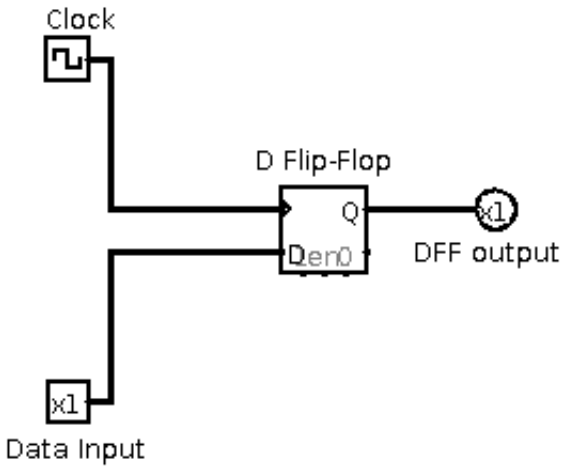
3 Simulation

In the previous example simulation is not tricky, we can auto simulate as if it were a live circuit and quickly see the result. However there are several other types of simulation which are important in different situations. The most common case will be when using a clock to enable a stateful circuit (i.e., a circuit with some type of memory). The types of simulation supported by Logisim are:

- **Simulate Enabled:** Let the circuit run based on it's inputs. must be enabled for all tick simulations to work as well.
- **Step Simulation:** Allows the user to simulate a single step at a time. If an input changes in step simulation you must advance the signal through each gate by stepping (ctrl-i).
- **Tick Simulation** Used to tick a clock (found in Explorer Plane Wiring > Clock). This is vital for stateful circuits (e.g., RAM, flip-flops, etc...)
  - **Tick Once:** Tick the clock once (go from high to low or vice versa)
  - **Ticks Enabled:** Tick automatically at the rate of tick frequency
  - **Tick Frequency:** How often to tick the clock (measured in Hz).

4 Flip-Flops

Flip-Flops are used to enable stateful circuits. Flip-Flops (FF) are found in the Explorer Plane > Memory > <type of FF>. In this example we select a typical D Flip-Flop (DFF), and show how it is used in concert with a clock. The DFF absorbs the input bit on the rising edge of the clock, that means when the clock transistions from $0 \rightarrow 1$.



There are several inputs on the DFF

- **D:** The value to input into the DFF on the next rising edge.
- **Clock:** The clock that controls the DFF
- **Q:** The output of the DFF.
- **Asynchronous reset:** pins DFF value to 0 when input is 1, found on the bottom left.
- **Enable:** When set to 0 the clock input is ignored and the DFF will mantain it's current value.
- **Asynchronous Set:** pins DFF vavlue to 1, while initialized to 1.