# Week 1 Reinforcement

These exercises follow from the lab exercises will be discussed in next week’s lectorial. You should try them and make sure that you have your attempts available during the lectorial. Some of the exercises are designed to make you think, so you may want to work on these in small study groups.

### Part 1 Building and testing circuits in Logisim

NAND gates are universal gates. A universal gate is a gate which can be used to create any Boolean operation without needing any other gate type. NAND gates are also cheaper to make as electronic components so are used in many chips and have a shorter delay (the time taken for an electrical signal to go through them). In industrial usage, an AND gate is typically created as a NAND gate followed by an inverter (NOT gate) rather than the other way around. A NOR gate is also a universal gate, but in this exercise, we will use NAND gates.

This exercise follows the Logic Gates levels from nandgame.com (Kjær, no date) and start by creating the NOT, AND and OR gates that form the basis of Boolean algebra. These can be quite tricky and there are multiple correct solutions and plenty of websites that show solutions.

Build the following gates using only the gates specified:

1. Create a NOT gate using a NAND gate only. Remember that a NOT gate has only one input
2. Create an AND gate using NAND and NOT only
3. Create an OR gate using NAND, NOT and AND only
4. Create an XOR gate using gates from NAND, NOT, AND and OR.

### Part 2 Binary and Hexadecimal

1. Convert these binary numbers to decimal

110100

10011

1. Put the following binary numbers in order from lowest to highest

10011

1011

101

1100

11101

1. Add the following binary numbers

1011 + 0101

1. Convert these decimal numbers to binary

16

31

1. Convert the following numbers from binary to hex

1001011111001101

1111111111111111

1. Put the following hexadecimal numbers in order from lowest to highest

1ab

cd

f4

abc

ef

d01

### Part 3 Creating components

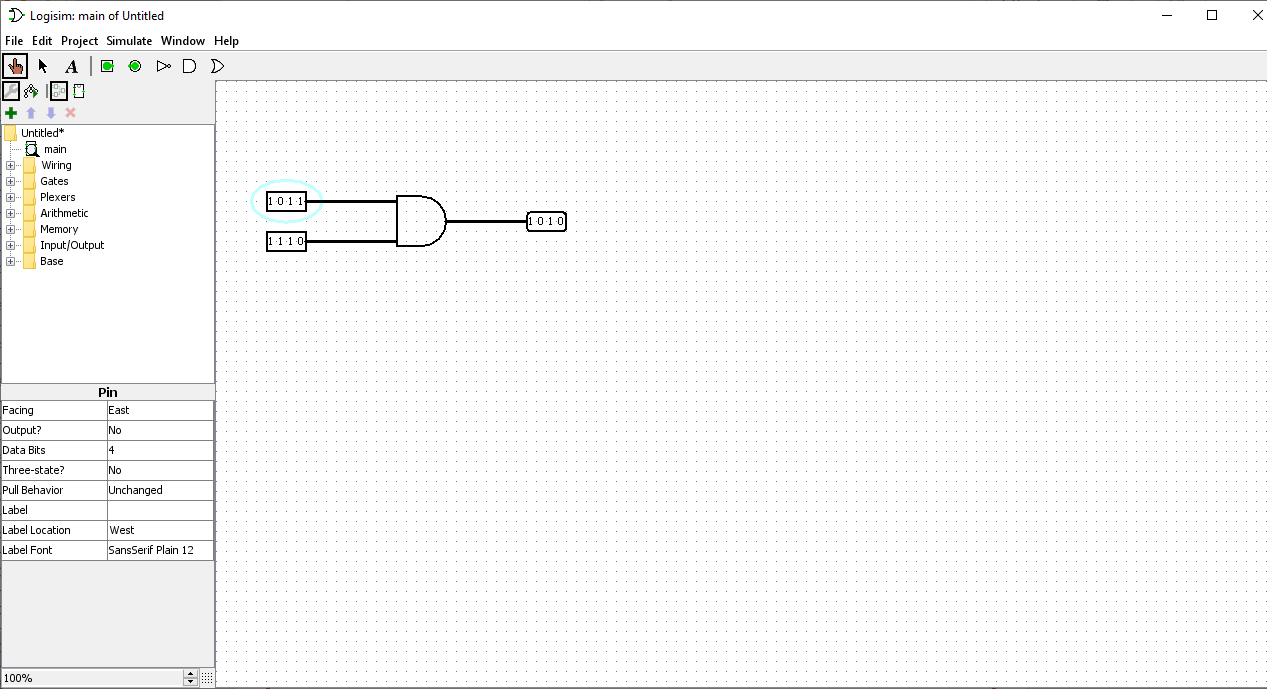
So far, we have been using Logisim with wires and components that deal with a single bit at a time. We have seen that computers are based on digital signals that consist of multiple bits. We can use place value to represent integers, meaning that we have to be careful about the order of the bits.

Logisim allows us to work with signals made of multiple bits and also has many built in components. Before the exercises, we have some introduction to using multiple bits in Logisim.

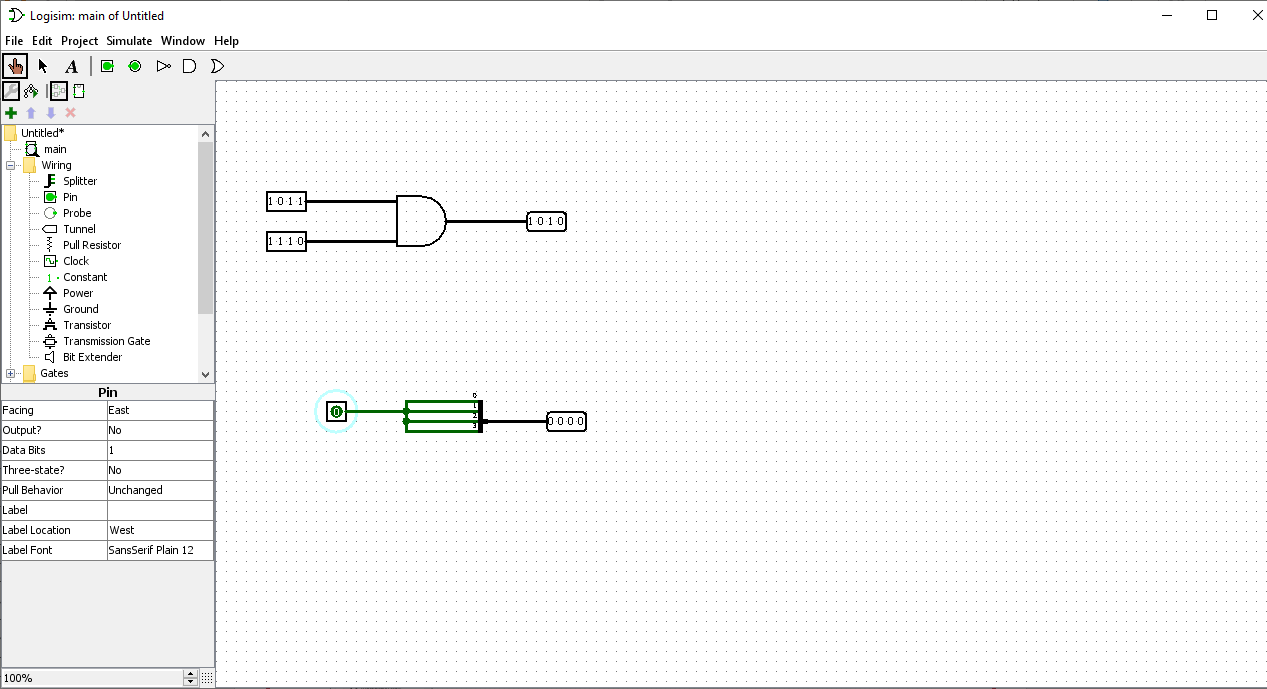
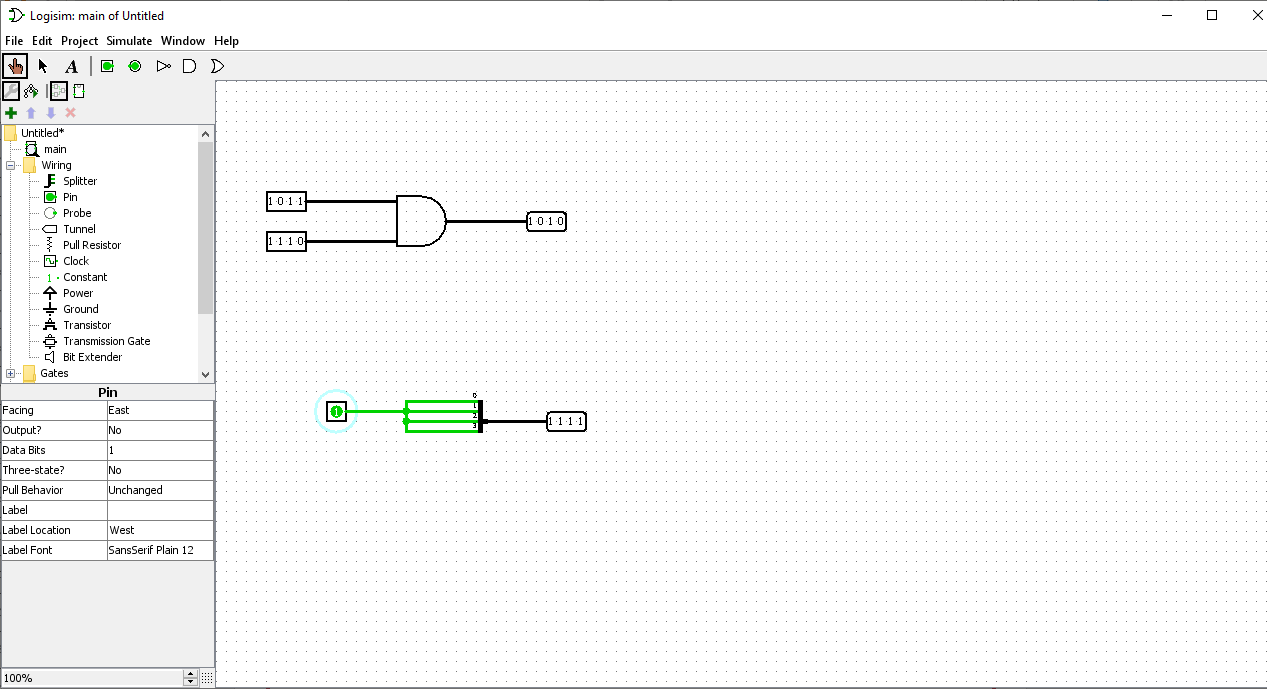
When we work with basic logic gates and multiple bits, they work *bitwise* on each bit in the signal. That is the operation is carried out on each corresponding bit in turn. In the Figure below, you can see that each component has the Data Bits attribute set to 4.

### Logisim screen with pins showing 4 bits. The AND gate is highlighted and data bits attribute set to 4.

When I use the hand icon to change inputs, I can click on any of the four bits in each input to change it from 0 to 1 (or vice-versa).



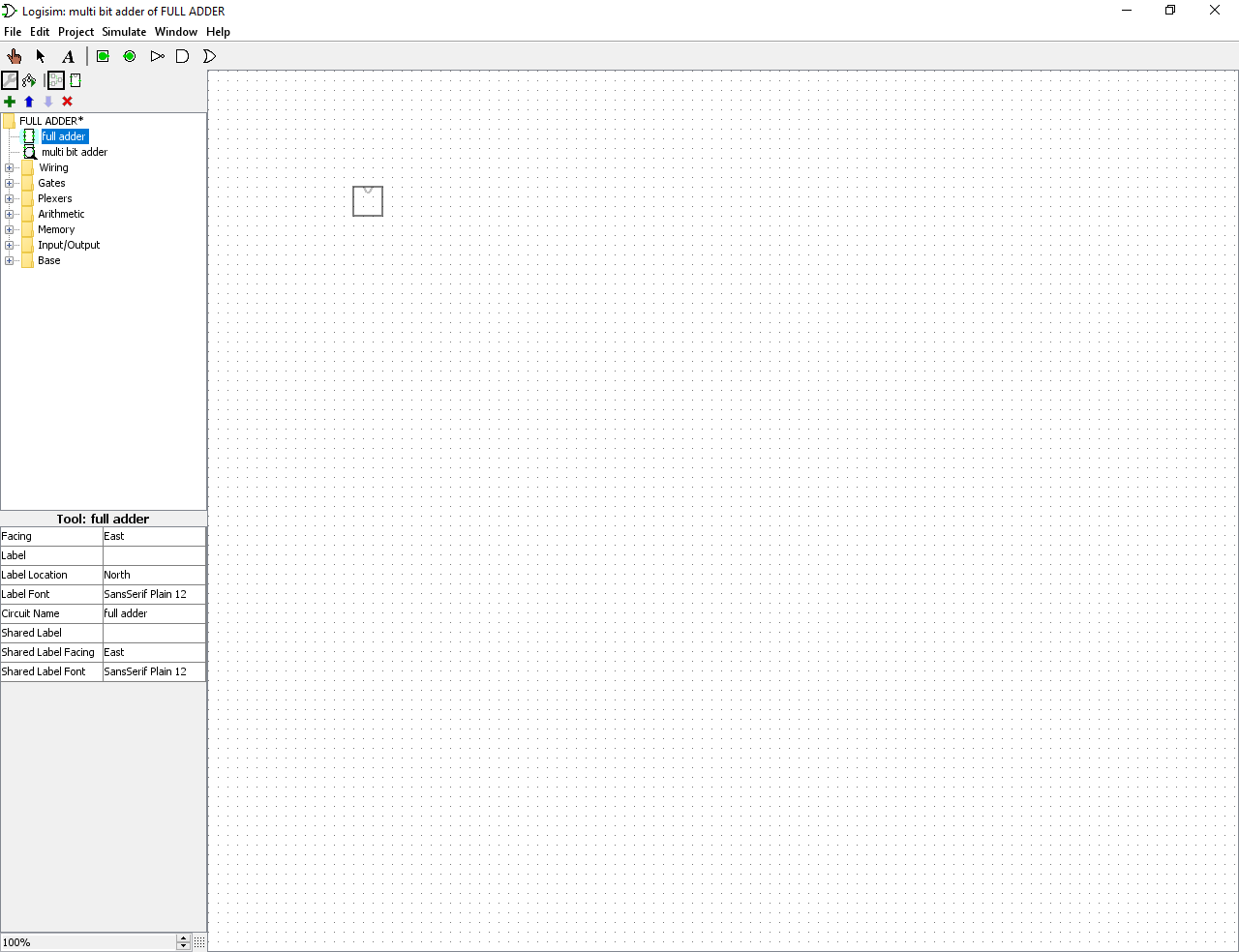
When you change a gate to have 4-bit inputs, all the inputs need to be 4-bits so that it can do bitwise logic on each bit in turn. A 1-bit signal can be duplicated to give a 4-bit signal. That is when the bit is 0, it should pass 0000 to the next gates, when it is 1, it needs to pass 1111. This can be done in Logisim using a splitter (from the wiring tools) which can also join signals together. In the example below, I set the splitter to face East and have Fan Out and Bit Width In as 4. The output pin shows that the signal has been combined correctly (I would feed this to another component rather than just a pin). Normally when using a splitter, you have to be particularly careful about the order of the bits, but here I needed them all to be the same. Note that a dot on the wires in Logisim shows that the signal is copied to multiple wires.



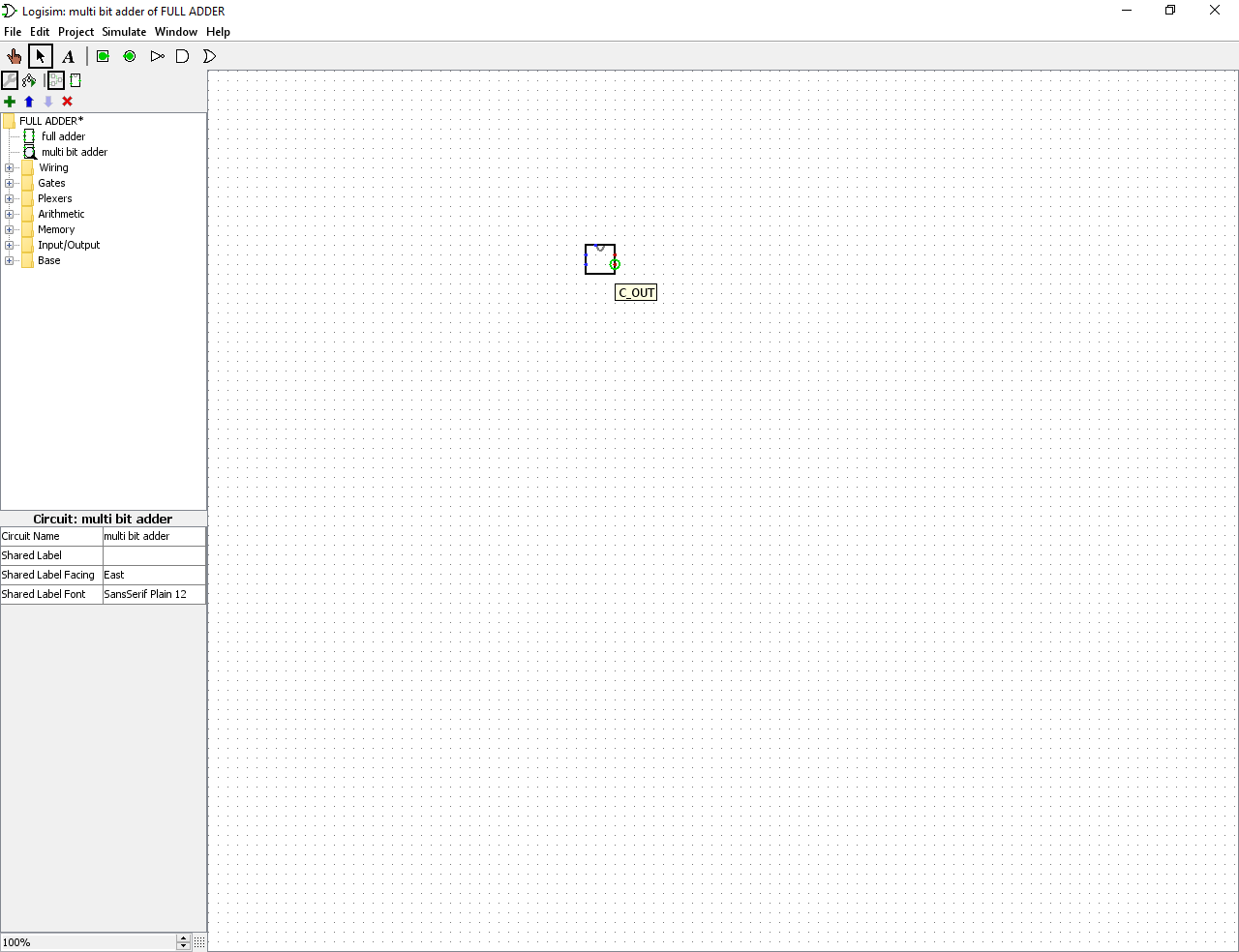
1. In the lab, you created a multiplexer which had 1-bit data inputs. Using that version as a starting point and the multiple bit ideas presented here, create a multiplexer that has two 4-bit data inputs and a 1-bit select input.
2. Using a multiple bit input pin, create a component that checks whether a signal is equal to zero. That is, have a 4-bit input, a splitter and a 1-bit output pin with gates and wires to do the following: The output should be 1 when all the bits of the input signal are zero, and the output should be 0 otherwise. You may use gates with more than two inputs.
3. Create a *comparator,* that is a component which checks whether two 4-bit signals are equal to each other. It should output a 1 if both inputs are the same and output a 0 otherwise. Note that signals will be equal when each corresponding bit is the same in both inputs.
4. In the lab, 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. Try to use pins with data bits set to 4 for the inputs and use splitters to split those to feed individual bits into the adders. 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 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, you will need to set it to have a value of hex zero. There is a reminder on sub-circuits below.

In Logisim, you can use an existing circuit as a sub-circuit. In the Figure below, I have a circuit with my full adder that has all inputs and outputs labelled. I have created a new circuit for my multi-bit adder, and I can highlight the full adder to place it in my circuit as if it was any of the components in Logisim.



The figure below shows that the labels that I used for the pins in the full adder can be seen when I highlight an input/output when adding the tool to a new circuit.



## References

Kjær, O. J. (no date) *NandGame – Build a computer from scratch* [Online] [Accessed on 10th September 2021] <https://nandgame.com>