# Week 1 Lab A: Logic Gates

## Objectives

Develop understanding and experience of:

1. Expectations for the lab sessions
2. Basic logic gates and truth tables
3. Using Logisim Evolution to simulate logic circuits

## Basic Logic Gates

You should create a folder in your university OneDrive area for your computer architecture work. Complete the gaps so that you have a copy of this information.

Binary data has two states. In this unit we will use 1 (on) and 0 (off) to represent the states. Gates are used to process or combine binary inputs.

### AND gate



A and B are inputs and X is the output.

The AND gate outputs a 1 when all of its inputs are 1 and outputs 0 otherwise.

Truth table

|  |  |  |
| --- | --- | --- |
| A | B | X |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

### OR gate



The OR gate outputs a 1 when any (or all) of its inputs are 1 and outputs 0 when all inputs are 0.

|  |  |  |
| --- | --- | --- |
| A | B | X |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

### NOT gate



The NOT gate outputs 1 when the input is 0 and outputs 0 when the input is 1.

|  |  |
| --- | --- |
| A | X |
| 0 | 1 |
| 1 | 0 |

The NOT gate is also called an **inverter**.

### Getting started with Logisim Evolution

1. Open Logisim Evolution and create a circuit that has two input pins labelled A and B that feed into a 2-input AND gate. The output of the AND gate should feed into a NOT gate and to an output pin labelled X.

* On diagrams we might write a letter, but the wire needs to come from a pin or another component.
* Wires need to be connected where there is a connection point on the component.

1. Click on the hand icon to test the circuit manually by trying all possible combinations of inputs A and B.
2. Check the truth table generated by Logisim Evolution for your circuit.

* Go to Project then Analyze Circuit and click the Table tab.

1. Save your Logisim Evolution file. There isn’t an autosave option in Logisim Evolution. It is your responsibility to make sure that you have saved your work. Choose a suitable filename. Note that you cannot use spaces in names in Logisim Evolution.
2. Add an image of your circuit and the truth table generated through Logisim Evolution to this worksheet.

AND gate leading into NOT gate images: A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

## Additional Logic gates

### NAND gate

The circuit you created above of an AND gate followed by a NOT gate is equivalent to a NAND gate.

1. Complete the truth table for a NAND gate:

|  |  |  |
| --- | --- | --- |
| A | B | X |
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

The NAND gate outputs a 1 when any (or all) of the inputs are 0 and outputs 0 when the inputs are 1

1. Use Logisim Evolution to find the symbol for a NAND gate. Add a NAND gate to a circuit and add an image below. You don’t have to wire up the gate.
   * Look at the components that are available above the circuit building area or in the Gates folder on the left-hand side.
   * To **add a new circuit** in the same Logisim Evolution project, click on the green plus sign above the folders on the left-hand side. Give the new circuit a name.

NAND gate symbol:

A screenshot of a computer

Description automatically generated

### NOR gate

In a similar way to a NAND gate, a NOR gate is equivalent to an OR gate followed by a NOT gate.

1. Complete the truth table and use Logisim Evolution to find the symbol for the gate and add to this worksheet as in the previous exercise.

|  |  |  |
| --- | --- | --- |
| A | B | X |
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

The NOR gate outputs 1 when both its inputs are 0 and outputs 0 when any inputs are 1

NOR gate symbol:

A screenshot of a computer

Description automatically generated

Note that the gates have the same shape as the gate they are based on but have a small circle at the output indicating that the output has been **inverted**. That is a NOT has been applied to the output from the basic gate.

### XOR gate

1. Find the XOR gate in Logisim Evolution. Create a circuit that has two input pins labelled A and B and an output pin labelled X. Use an XOR gate to combine the inputs A and B with the output of the XOR gate going to pin X. Add an image of your circuit.
   * Use a new circuit in your existing Logisim Evolution project and it might be easier to copy the wiring from your initial circuit rather than starting again.

A screenshot of a computer

Description automatically generated

1. Use Logisim evolution to generate the truth table for an XOR gate and use it to fill in the truth table below.

|  |  |  |
| --- | --- | --- |
| A | B | X |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

1. Based on the truth table express the operation of an XOR gate in words

The XOR gate outputs 1 if either of the inputs are 1 but not both and outputs 0 if both the inputs are the same (both 0 or both 1)

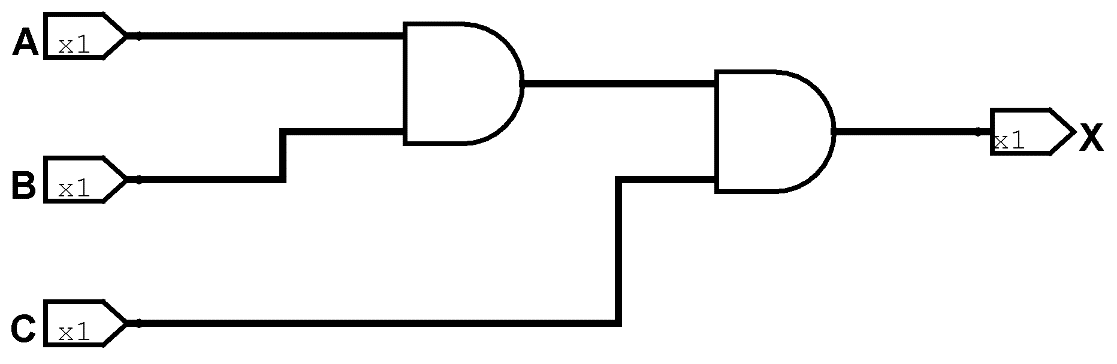
## Combining Logic Gates

We can have more than two inputs and use multiple gates to combine them. The order that we connect the components in can make a difference. There are two ways in which three inputs (A, B and C) could be combined using two AND gates. The parts shown in round brackets (parentheses) are combined first and then the output of that combined. You do **not** need to create the circuits shown here but adapt them as described below.

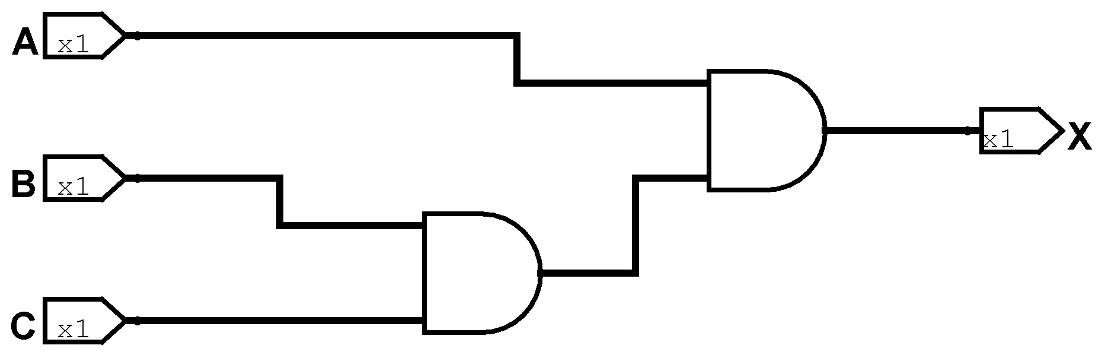
The possibilities are:

* (A AND B) AND C

In this case A and B go into an AND gate and the output from that AND gate is combined with C using an AND gate.



* A AND (B AND C)



In this case, using AND gates, the truth tables are the same for both arrangements of gates.

|  |  |  |  |
| --- | --- | --- | --- |
| A | B | C | X |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

Note that there are now 8 different possible combinations of the inputs. Each input can be 0 or 1 and there are three of them, so the number of combinations is 2 \* 2 \* 2.

1. Use the same approach to compare the three other combinations listed below. Create the circuit in Logisim Evolution, generate the truth table and determine whether the two truth tables are the same for each pair.

* Make sure that you use a separate circuit in your Logisim Evolution project for each circuit othewise the truth table will get very messy.
* Take images of your circuits and the truth tables to add to this document. Move things down as needed

1. Two NAND gates comparing the truth tables for

With NAND gates the truth tables are **different**.

* 1. (A NAND B) NAND C

A and B are combined with a NAND gate and the result goes to another NAND gate with C

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

* 1. A NAND (B NAND C)

B and C are combined with a NAND gate and the result goes to another NAND gate with A

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

1. One AND and one OR gate comparing the truth tables for
   1. (A AND B) OR C

A and B are combined with an AND gate and the result goes to an OR gate with C

Diagram

Description automatically generated

A screenshot of a computer

Description automatically generated

* 1. A AND (B OR C)

B and C are combined with an OR gate and the result goes to an AND gate with A

Diagram

Description automatically generated

A screenshot of a computer

Description automatically generated

The truth tables are **not** the same.

1. Using NOT gates before or after an OR gate comparing the truth tables for
   1. (NOT A) OR (NOT B)

Diagram

Description automatically generated

A screenshot of a computer

Description automatically generated

Note that this truth table is the same as a NAND gate (which is NOT (A AND B)).

So (NOT A) **OR** (NOT B) is the same as NOT (A **AND** B)

When the NOT moves from the individual inputs to the output the OR changes to AND. This is an example of De Morgan’s law which you will see again in Maths for Computing in Semester 2.

* 1. NOT (A OR B)

Shape

Description automatically generated with low confidence

A screenshot of a computer

Description automatically generated

This is just a NOR gate (OR followed by NOT).

The truth table is not the same as example a of the pair.

## Extension exercises

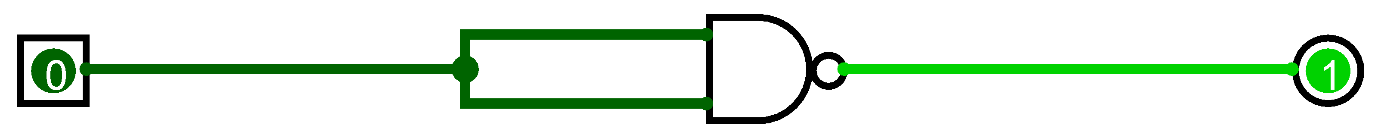
### Starting from NAND gates only

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** 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. Nandgame presents a lot of the ideas we will be covering on this unit, but in a slightly different way, so you might want to work further through the challenges there.

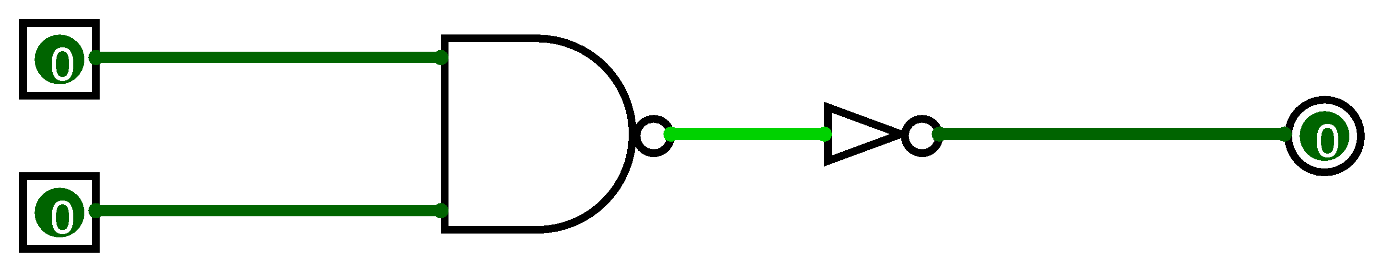
Build the following gates using only the gates specified:

1. Create a NOT gate using a NAND gate only. Note a NOT gate has only one input.



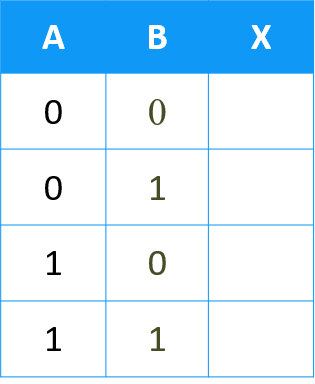
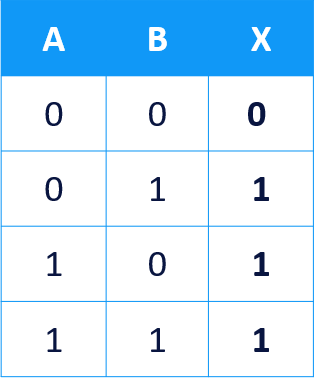
1. Create an AND gate using NAND and NOT only.

Note that if we apply a NOT to the output of a NAND gate, that is just an AND gate. The NOT gate could have been built from a NAND gate as above.



1. Create an OR gate using NAND, NOT and AND only.

When one input is 1 and the other input is 0, a NAND gate outputs 1 as we need. The NAND gate outputs the inverted value of what we need when the two inputs are both 0 or both 1. If we invert both inputs, we will get what we need.



**1**

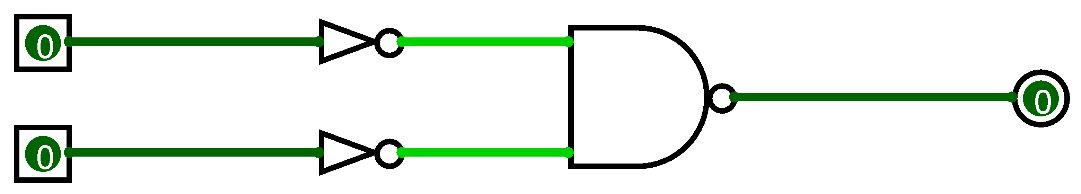
**1**

**1**

**0**

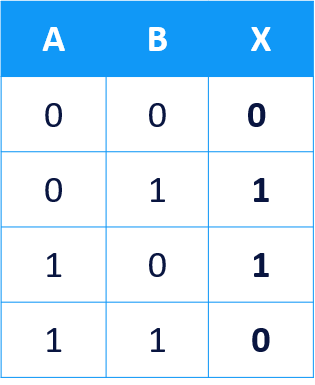
NAND

OR

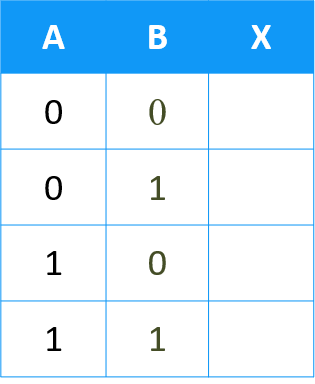


1. Create an XOR gate using gates from NAND, NOT, AND and OR.

Looking at the truth tables, I saw that the XOR outputs 1 when both NAND **and** OR output 1 and outputs zero otherwise. I used the version of OR from the previous part but used a NOT and AND gates directly but they could have also been created from NAND gates as in the earlier parts.



XOR



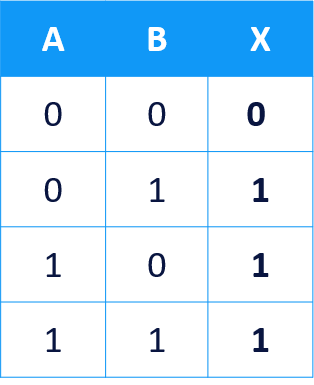
**1**

**1**

**1**

**0**

NAND



OR

Diagram

Description automatically generated

A NOT gate could be made from 1 NAND gate, and an AND gate from 2 NAND gates. That means that my implementation would need 6 NAND gates altogether.

There are more efficient ways to implement an XOR from NAND gates using fewer NAND gates  
This version uses 4 NAND gates.   
Image from https://en.wikipedia.org/wiki/XOR\_gate#/media/File:XOR\_from\_NAND.svg

Shape

Description automatically generated with medium confidence