NAND 1/3: Introduction to logic simulation and implementation

Kerstin Eder

September 23, 2021

In this lab, you will begin to familiarise yourself with Logisim and the NAND boards. Together, these tools will help you learn more about logic. The labs allow you to develop practical skills that deepen your understanding while building the fundamental components used in computer architecture.

Goals of this lab

This lab will give you an opportunity to get to grips with Logisim, learn how the NAND boards work, help you understand the lecture material, and prepare you for future NAND labs.

For the NAND labs, it is recommended that you work in small informal lab groups, e.g. with two to three students located close to you in the lab. Please note, however, that each of you needs to develop a good understanding of the material so that you can perform the tasks on your own once you have worked through the lab sheets.

Herein follows a series of tasks for you to complete. Please attempt them in order. If you get stuck and are waiting for help, feel free to move on to the next task, if you feel like you can do so.

1 Starting Logisim

Logisim is a Java application. It requires a Java installation to run, so you may need to install Java before you can run Logisim. (This applies in particular if you want to run Logisim on your own computer.)

Logisim can be obtained from http://sourceforge.net/projects/circuit/. Simply download the JAR file and run it from the terminal.

You may want to place the JAR file into the working directory from which you start your terminal and where you intend to keep your work. Assuming the download is called `logisim-generic-2.7.1.jar`, you can then run Logisim as follows:

```
java -jar logisim-generic-2.7.1.jar
```

Or, if your JAR file is in your `Downloads/` directory, you can run it using:

```
java -jar ~/Downloads/logisim-generic-2.7.1.jar
```

You will need to use Logisim in order to complete this lab and future NAND labs.

Tips

The screenshot in Figure 1 shows a Boolean function implemented in two ways, and highlights useful parts of the Logisim interface.

To place components, select them in the library on the left, configure them using the box in the bottom-left, then click in the drawing area to place them.

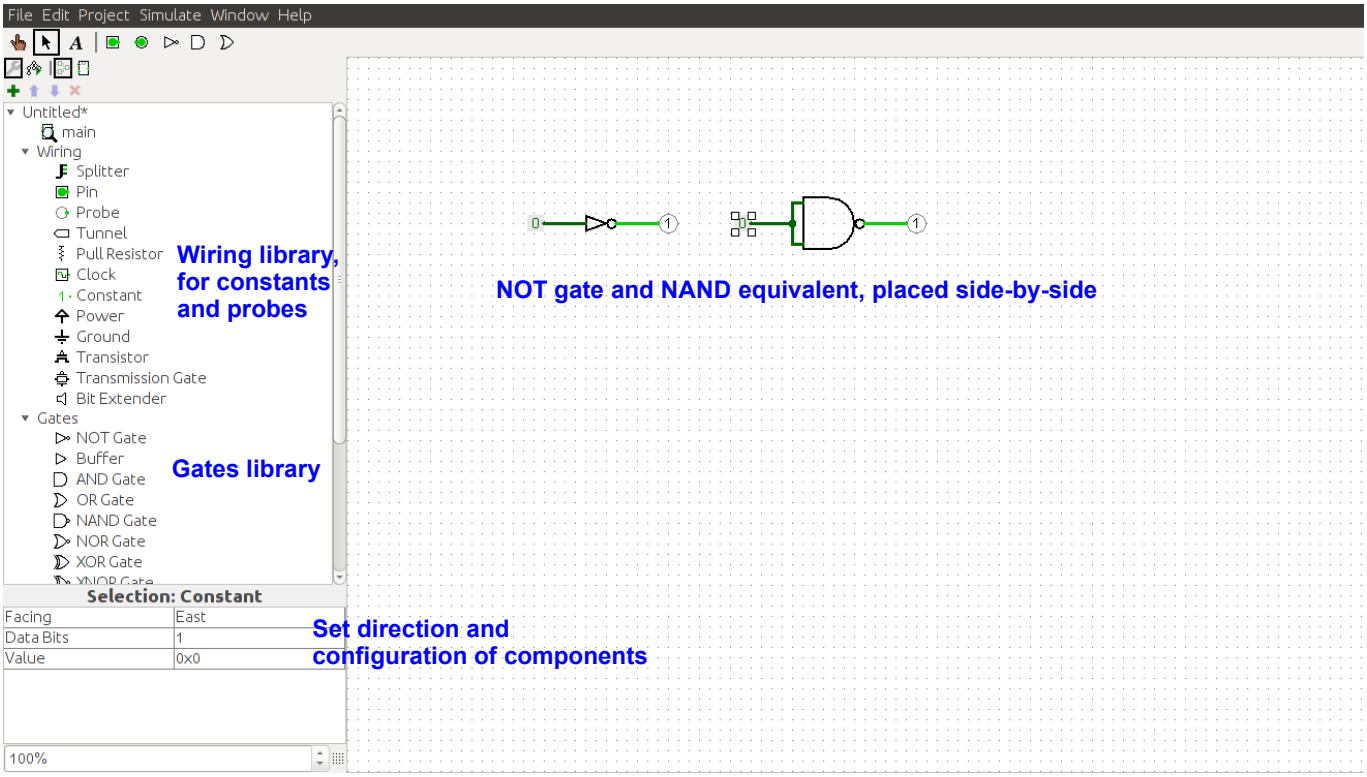

Figure 1: Logisim interface

Connections between components can be made by clicking and dragging wires. Components can be moved by clicking on them and dragging as well. Components and wires can be deleted by clicking on them and pressing the delete key on the keyboard. As such, the interface is very similar to a lot of diagram drawing software.

The configuration of a component can be changed by clicking on it and then editing its configuration in the bottom-left area. However, if the component is connected to other things, changing properties such as the direction it is facing may result in wires getting disconnected, or arranged in confusing ways. It may be better to delete that component and replace it with a new, correctly configured one, reconnecting it to the surrounding wiring.

2 Implementing logic gates using transistors

The **lecture slides on "Transistor logic and CMOS"** contain descriptions of various logic gates, including their transistor-level implementation. You will now learn how to design some of these circuits in Logisim.

2.1 Inverter (NOT gate)

To build a CMOS inverter, as described in the lecture slides, you will need the following Logisim components:

- A power source.
- A ground.
- A P-type transistor, facing south.
- An N-type transistor, facing north.
- A constant generator, to provide an input value.
- A probe to read the output value.

If you follow the steps below and build a circuit that resembles the `NOT` gate diagram in the lecture slides, you should end up with a working inverter.

1. Place a **P-type** transistor, facing **south**, on the drawing area.
2. Place an **N-type** transistor, facing **north**, below the P-type. You will notice the P-type has a small round circle at its gate and a directional arrow pointing down, whilst the N-type has no circle, and a directional arrow pointing up. What do the arrows represent?
3. Now place a power component above the top transistor.
4. Next, place a ground component below the bottom transistor.
5. Place a constant component to the left of the transistors, this is equivalent to the signal `A` in the lecture slides.
6. Place a probe on the right of the circuit. This is equivalent to the output `Q` in the lecture slides.
7. Connect the components with wires as shown in the lecture slides.

Now, by clicking on the constant component and pressing either the 1 or 0 key on your keyboard, you can change the logic value that it generates. At the same time, you should see the value at the probe change, in accordance with the logic function `NOT`.

If your circuit does not work as expected, make sure the components are wired correctly, and that the type and orientation of your transistors are correct. If they are not correct, you may see `X` at the output for a particular input value.

Save your work before continuing for future reference and revision.

2.2 NOR gate

Using the same methods as for the `NOT` gate, and using the diagrams in the lecture material for guidance, implement a `NOR` gate using four transistors. Remember to be careful in choosing the correct transistor type and orientation, or the circuit will behave improperly.

You can place the `NOR` implementation next to your previous circuit. Once you have built a working implementation, remember to save your work again.

2.3 NAND gate

Again, using the same methods as before, implement a `NAND` gate from four transistors. Place it in the same design as the previous two gates, and make sure it behaves as defined in the lecture slides.

2.4 NOT gate, revisited

You are now going to make an inverter again. But this time, implement it using NAND-based logic, i.e. based on the `NAND` gate you just made. Compare the NAND-based implementation to the pure CMOS implementation you started with - they should behave in exactly the same way. Consider what the benefits and drawbacks of the NAND-based implementation are.

2.5 AND gate

This is the final transistor-level gate you will need to implement. Once again, using NAND-based logic, and with the help of the lecture material, build an `AND` gate and verify that it works in accordance with the definition of the `AND` Boolean function.

2.6 Extra task

This task requires you to read and research beyond the content delivered in this unit. Please only attempt this task when you have done all other sections.

Optionally, you may wish to try to implement an `AND` gate using fewer transistors, by researching other ways of creating this type of gate using P- and N-type transistors. Verify that your new design works correctly, and consider how many transistors you have saved by using a function-specific implementation of `AND`, rather than one that uses *functionally complete* building blocks.

3 Implementing logic gates using NAND gates

The **lecture slides on "Transistor logic and CMOS"** contain descriptions of how various logic gates can be implemented using only `NAND` gates. You will now learn how to design some of these circuits in Logisim.

3.1 NOT, AND and OR using NAND gates

You have already implemented these functions in transistor logic using Logisim. This time, use the `gates` library in Logisim and build `NOT`, `AND` and `OR` functions using **only `NAND` gates**.

3.2 Extra work

This task requires you to read and research beyond the content delivered in this unit. Please only attempt this task when you have done all other sections.

Optionally, you may wish to implement other logic functions, e.g. `NOR` and `XOR`, using only `NAND` gates.

4 NAND boards

4.1 Introduction

The NAND boards are a set of four chips, each containing four `NAND` gates. Each gate has a set of input pins and output pins. In addition to the pins for the gate input and output, there are pins providing `logic 1`. You may notice that there are no `logic 0` pins. This is because an unconnected input will default to 0 (there is a *pull-down* circuit that keeps the input at 0 if it is unconnected). LEDs on the boards signify the output level of each `NAND` gate. The boards are powered by a USB cable.

Current versions of the boards also feature four push-buttons, which provide `logic 0` when not pressed, `logic 1` when pressed. They are each connected to a block of four pins at the left-hand side of the board.

These boards are *open source*, and were created by Drs Simon Hollis and Dan Page. There is a GitHub repository containing the designs, photos, and more information on the boards: https://github.com/danpage/nandboard. You may want to use this for further reference.

4.2 Access to NAND boards

NAND board kits have either been sent out to you or you have been asked to pick a NAND board kit up. Please be patient as some may still be in the post or awaiting collection as we attempt this lab. If you do not yet have your NAND board kit you can still complete the Logisim part of this lab. For the practical work in this section, however, you need a real NAND board. **Please inform your TA as attendance is taken whether or not you have received your NAND board kit.**

If a piece of hardware does not work, or you accidentally break something, you must report this to the TAs in the lab as soon as possible. The boards are quite robust and rarely break. However, they are not expected to last forever and we will not blame you if one reaches the end of its natural life while in your hands. We will do our best to provide a replacement where possible.

4.3 Familiarisation with the NAND boards

To familiarise yourself with the NAND boards, please watch the introduction video at:

https://www.youtube.com/watch?v=DJDXp7yXp-w&feature=youtu.be

Figure 2 illustrates the basic layout of the NAND boards. You may want to use this as a template for designing your circuits before building them.

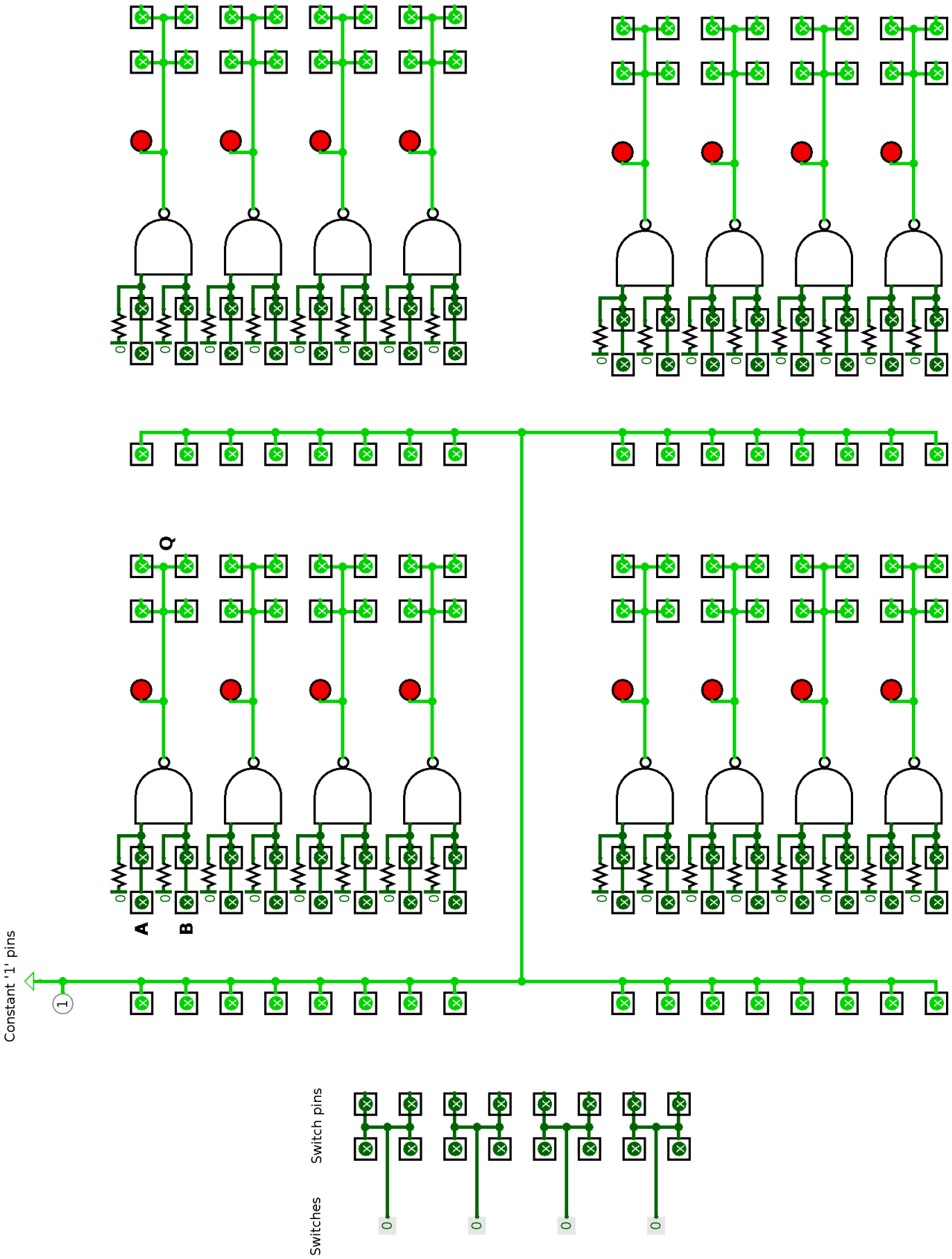

Figure 2: NAND board layout

4.4 Transferring Logisim designs onto the NAND boards

Although the NAND boards are very simple, they can quickly become a confusing mess of wires. It is important to follow a good process for implementing designs on them.

It is most strongly recommended that you design and test using Logisim before building. This means that you first print or draw a Logisim tested design. You then choose where each `NAND` gate will go on the NAND board, before you connect wires in a sensible order, *marking down* when each part of your design is completed. That way, you are less likely to miss a wire, or get confused. The NAND board layout in Figure 2 can serve as a template for your designs.

4.5 NOT, AND and OR in NAND logic

You have already implemented these functions using only `NAND` gates in Logisim. Now, implement the same NAND-gate based circuit designs on the NAND boards.

Questions

If you have any questions, please ask. Make use of the TAs, but do not expect them to give you complete designs, they are there to guide you, not do the work for you. Please be patient. You may need several attempts to get something working.

When requesting help, TAs will expect you to show them what you have done so far (no matter how sketchy) and they will ask you to clearly explain your reasoning. This makes it easier for the TAs to help you. For this reason, priority will be given to students who can show and explain how far they have got.

Well done **for completing the first NAND lab.**