Overview of Computer Architecture, Lab 1, Introduction to logic simulation and implementation

Steve Kerrison, University of Bristol Computer Science Dept.

6th October, 2014

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, and build the fundamental components used in computer architecture.

Goals of this lab

This lab is not assessed, but will give you an opportunity to get to grips with Logisim, see how the NAND boards work, help you understand the lecture material, and prepare you for Lab 02, which will be assessed. This lab is also an opportunity for you to find a lab partner, who you will work with for the next few weeks.

Lab partner

This lab is to be completed in pairs. You will need to find somebody to work with. Once you have agreed to work together with somebody, make sure you both have the ability to contact each other (exchange e-mail addresses, for example).

You will need to submit the details of your lab partner using the submission system. See Section 2 of this document for instructions.

1 Tasks

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.

1.1 Using Logisim

Logisim is a Java application, it can be obtained from http://sourceforge.net/projects/circuit/. Simply download and run it, for example:

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

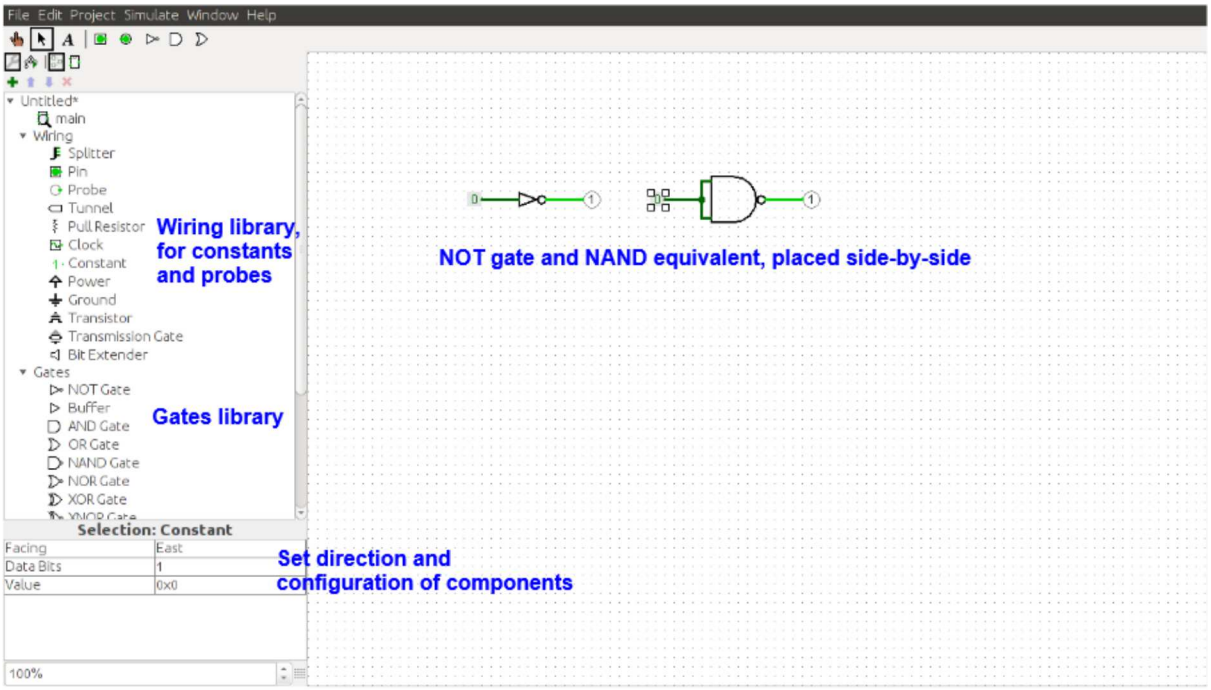

Figure 1: Logisim interface

Alternatively, a version has been placed on the network in a publicly accesible location. From a lab machine, you can access it like so:

```
java -jar /home/research/micro/logisim/logisim.jar
```

You will need to use Logisim in order to complete this lab, and future labs, including assessed ones.

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.

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.

1.2 Transistor circuits

The lecture slides in L03 contain descriptions of various logic gates, including their transistor-level implementation. You will now recreate some of these circuits in Logisim.

1.2.1 Inverter (NOT gate)

To build a CMOS inverter, as described in L03, slide 6, 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 diagram in L03, slide 6, 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.

1.2.2 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.

1.2.3 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.

1.2.4 NOT gate, revisited

You are going to make an inverter again. But this time, implement it using NAND-based logic. Compare the NAND 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 implementation are.

1.2.5 AND gate

This is the final transistor-level gate you will 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.

Extra work 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.

NAND boards

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. The silkscreen (white print on the boards) shows how everything is connected.

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 two push-buttons, which provide `logic 0` when not pressed, `logic 1` when pressed. They are connected to the second and sixth pins at the top left of the board.

Figure 2 shows a NAND board in use, with a push-button pressed whilst connected to both inputs of a NAND gate, resulting in the output LED showing `logic 0` (off) - it is an inverter! Figure 3 shows a representation of the NAND boards in Logisim. It is probably too dense to use as a basis for circuit design, but is intended to help you map the simulated circuit layout to the physical layout on the board. The circuit file is also available on the unit web page.

Now is the time to ask for a NAND board! Your pair will be given one board - please make sure you bring it to future labs. After the NAND-board labs have been completed, each

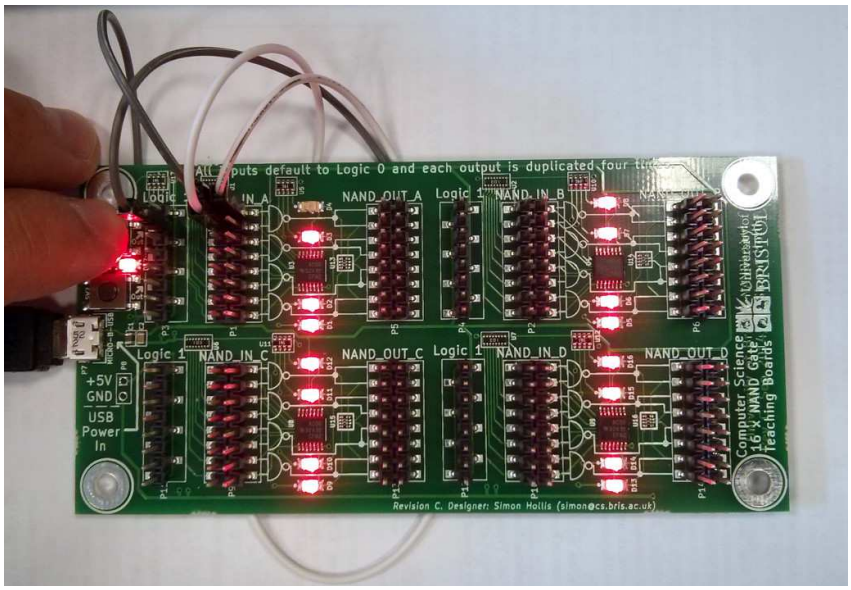

Figure 2: NAND board

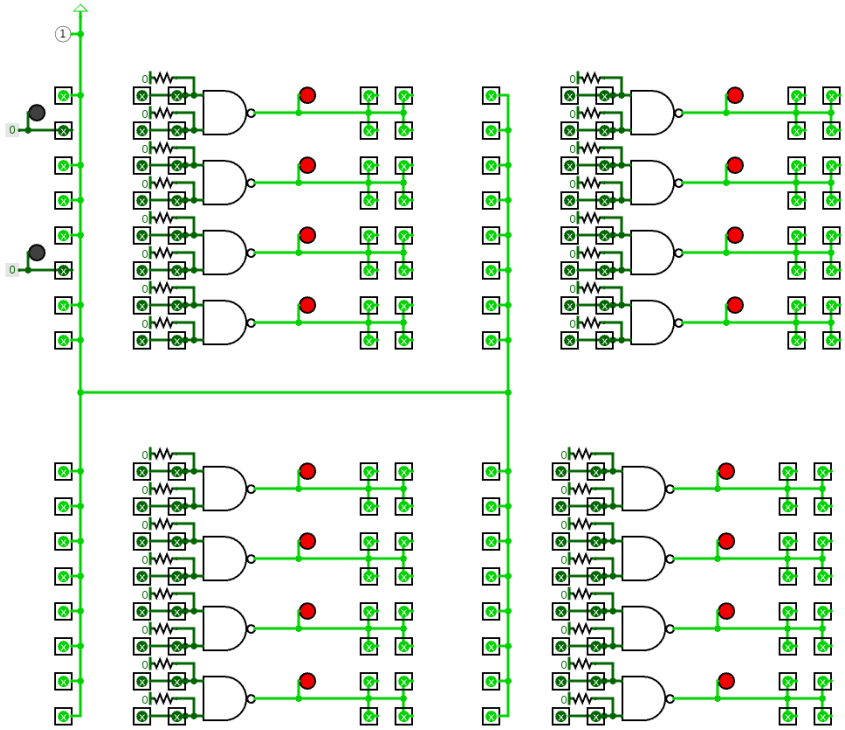

Figure 3: NAND board Logisim circuit

pair will be given another NAND board, so that all students have their own board to take away and keep!

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 recommended that you print or draw a design, choose where each NAND gate will go on the NAND board, then connect wires in a sensible order, marking down when each one is completed. That way, you are less likely to miss a wire, or get confused.

1.3 NOT, NOR and AND in NAND logic

You have already implemented these functions in transistor logic. This time, use the `gates` library in Logisim and build `NOT`, `NOR` and `AND` functions using only `NAND` gates.

Then, implement those same designs on the NAND hardware that you have been given.

2 Submission

There is only one submission requirement for this lab:

1. Submit a text file containing the name and username of the individual you are working with for this lab and subsequent NAND-board labs. Both partners will need to submit this!

Optionally, you can also:

- Submit your `.circ` file containing your various transistor-level implementations. You will be making submissions like this in future labs, so it is recommended that you get into the habit of submitting now.

Please use the unit's SAFE page to make your submission.

Questions

If you have any questions, please ask! Make use of the lab demonstrators, but do not expect them to give you complete designs, they are there to guide you, not do the work for you.