Overview of Computer Architecture, Lab 2, Logic simulation and construction

Steve Kerrison

October 13th – October 20th, 2014

In this lab, you will build on your Logisim experience so far, and also begin to use the NAND boards. Together, these tools will help you learn more about logic, and build the fundamental components used in computer architecture.

This lab will span two weeks. It is paced to give you sufficient time to get used to the format of the labs, working with others, asking questions, and submitting assignments. However, time management is very important, and it is expected that you may want to do some work outside of the lab in order to make sure you maximise your mark.

Goals of this lab

- Continue to familiarise yourself with Logisim for logic circuit design and simulation.
- Transfer designs onto NAND boards.
- Begin to formally work with a lab partner, which will help when the lab tasks become larger and more complex.
- Combine your own work on the NAND boards with the work of others, demonstrating how simple structures can build more complex designs.
- Prepare and submit the first assessed piece of work, which will contribute to your final unit mark.

Assessment

This is an **assessed lab**. Your performance in this lab, and materials that you submit following the lab, will contribute towards your overall mark for this unit. Therefore, this lab is not only useful to help you understand the material, it is also essential in helping you reach your goals for the Unit.

The lab is broken down into tasks, with each task of increasing difficulty or requiring some additional understanding. Only certain tasks are assessed, but you will need to complete all of the tasks in order to have sufficient understanding to succeed in the assessed tasks. The rest of this lab sheet details the tasks, as well as the marks available for all of them.

This lab contributes 5% towards your Unit mark. **The deadline for submission of circuit files and in-lab marking is Monday 20th October, 17:00.** Your physical designs **must** have been observed by then in order to guarantee that you will receive marks for them. Circuit files that are submitted late will receive reduced marks as per the submission system's penalty rules.

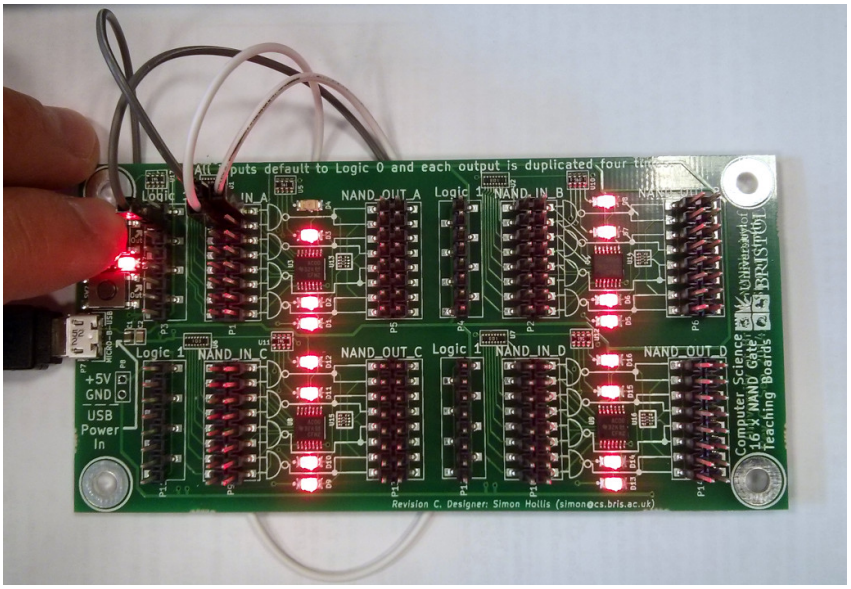

Figure 1: NAND board

Please note that marks for each task are not atomic — partial marks may be awarded at the marker's discretion. Finally, please **both partners submit your work via SAFE**; this will be considered your acknowledgement that you both worked together on the submission and both understand what you have submitted.

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 1 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 2 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 pair will be given another NAND board, so that all students have their own board to take away and keep!

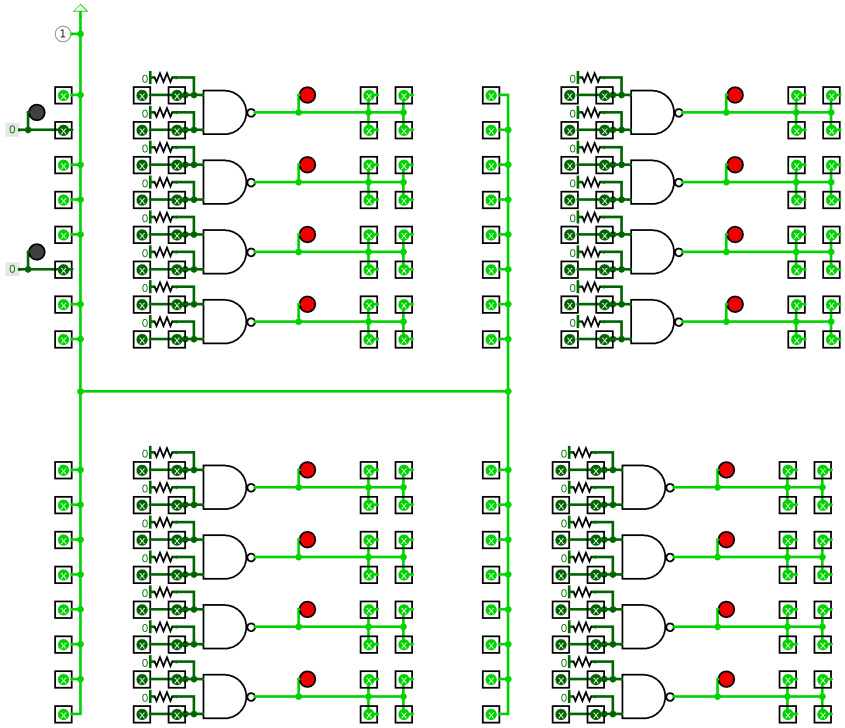

Figure 2: NAND board Logisim circuit

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.

Sensible layout of designs

To make life easy for yourselves and the markers, please try to observe the following conventions, wherever possible:

- Use the push-buttons as inputs, wherever possible. If you need more than two inputs, use a loose wire that you can connect/disconnect as required.
- Try to use the bottom-right gates as outputs, so that the LEDs are easy to see.
- When demonstrating your physical implementations, make it clear to the observer which button/wire is input A, input B, etc. Similarly, be sure to identify which outputs are Q, S, etc. This is also essential when you come to connect your design to other boards!

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
```

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
```

1 Tasks

These are the tasks you must perform in the lab. Please **read the whole lab sheet first**, so that you can manage your time on each task, and see which tasks are connected.

1.1 Basic logic structures — Logisim

This task is very similar to the previous lab. However, you no longer need to use P- and N-MOS transistors. **Just use the NAND symbol provided in the Logisim library.**

Using Logisim, construct the following Boolean logic functions, both using the library-versions of them, and a NAND-logic based version of your own construction. Lecture 3 contains material that will help you achieve this.

- NOT
- OR
- AND
- XOR
- NOR
- XNOR

Connect `Constant` sources to each of the inputs of the blocks that you build. These can be found in the "Wiring" section of the library pane. Connect a `Probe` to each output. By clicking on and changing the constant value, you should see output read by the probe change according to the truth table for that Boolean function.

Place your library and NAND versions of your functions side-by-side, so that you can easily compare them, and to verify that your NAND implementation of each function is correct.

Save your work in an apporpriately named file. You can put all of your functions in one circuit file, or split them up; you may decide.

Tips

- You can highlight gates and wiring by clicking and dragging, then move, copy, paste and so on, much like in any drawing software.
- Gates in Logisim have configurable number of inputs and other features. When you have placed a gate, click in it, and in the bottom-left you can change certain features. For example, **ensure you set your NAND gates to have two inputs, rather than the default of five.**
- The value of constants can be changed by clicking on them and typing the new value, or by using the bottom-left configuration pane, once the constant object has been selected.

The screenshot in Figure 3 shows the first Boolean function implemented, and highlights useful parts of the Logisim interface.

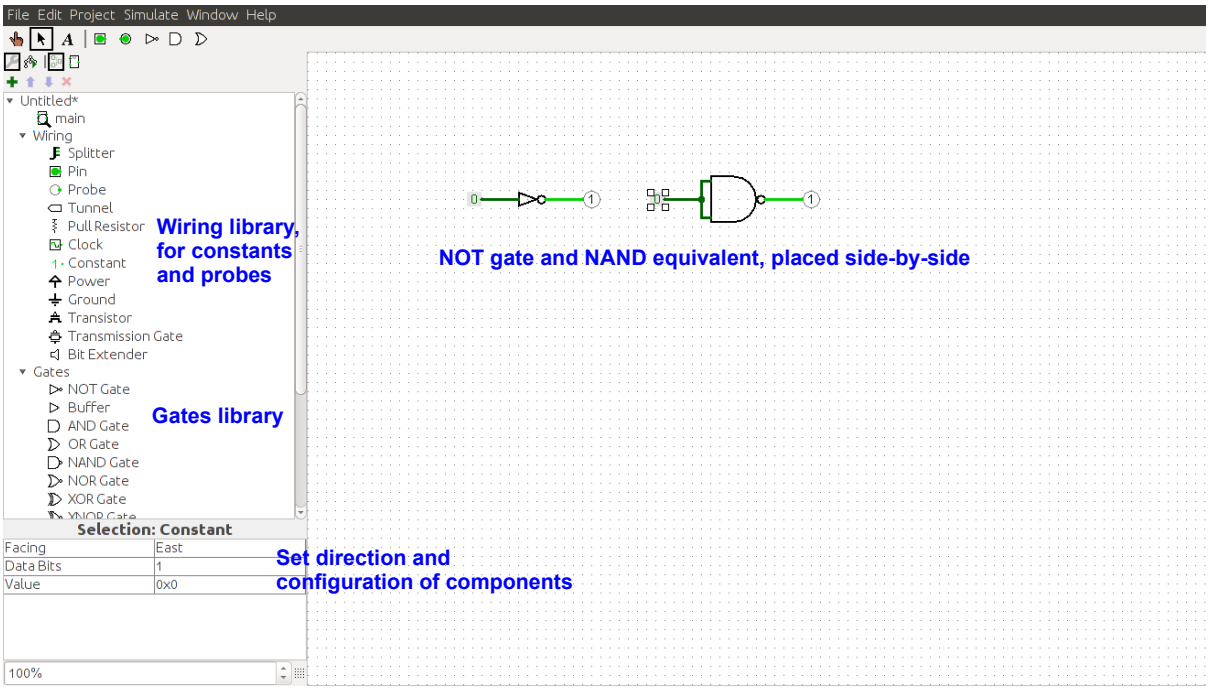

Figure 3: Logisim interface

1.2 Basic logic structures — NAND boards

Now choose two or three of the functions that you designed in Logisim, and implement them on the NAND boards.

Keep in mind that the next task is assessed and requires you to implement a specific function, so you may wish to select two or three *different* functions to get used to the NAND boards first.

1.3 XOR implemented in Logisim and on NAND boards — 10%

Take the XOR that you implemented in NAND-logic in Logisim, and re-create it on the NAND board. You should be used to this process, having practiced a few other functions in the previous task.

This task is assessed. To achieve full marks in this task, please ensure you do the following:

1. **In the lab:** Show a lab demonstrator your Logisim design *and* your NAND-board implementation. Allow them to verify that both of them implement XOR correctly.
2. **For submission:** Submit a Logisim circuit design file, called **xor.circ** via the submission system.

If you demonstrate a working design, but fail to submit your Logisim file, you may not receive full marks.

1.4 Half adder — 15%

Create a new Logisim file and implement a half-adder. Do the same with the NAND board. Lecture 4 contains material that may help you with this.

This task is assessed. To achieve full marks in this task, please ensure you do the following:

1. **In the lab:** Show a lab demonstrator your Logisim design *and* your NAND-board implementation. Allow them to verify that both of them implement a half-adder correctly.
2. **For submission:** Submit a Logisim circuit design file, called **half-adder.circ** via the submission system.

If you demonstrate a working design, but fail to submit your Logisim file, you may not receive full marks.

1.5 Full adder — 15%

The half-adder has no carry capabilities. To make a more useful adder, you should build a full-adder, which provides carry-in and carry-out. Again in Logisim and on the NAND-boards, implement a full-adder.

This task is assessed. To achieve full marks in this task, please ensure you do the following:

1. **In the lab:** Show a lab demonstrator your Logisim design *and* your NAND-board implementation. Allow them to verify that both of them implement a full-adder correctly.
2. **For submission:** Submit a Logisim circuit design file, called **full-adder.circ** via the submission system.

If you demonstrate a working design, but fail to submit your Logisim file, you may not receive full marks.

1.6 Chained full-adder — 5%

This task requires that we combine the work of multiple groups. Take your full-adder and form a carry chain with working full-adders of other groups.

At the end of the first lab session, we aim to have an n-bit adder made from full-adders that everybody has built together.

To receive marks for this task, your adder must form part of the n-bit adder, and the n-bit adder must work correctly. It is, therefore, your job to make sure that:

1. Your full-adder works correctly (which should be the case if you succeeded in the previous task).
2. You clearly indicate the inputs to your full-adder, so the previous adder in the chain is correctly connected to it, and;
3. You clearly indicate the outputs to your full-adder, so that you can chain to the subsequent full-adder correctly.
4. You maintain a good pace through the lab tasks, to get to this point in time to combine your design with others and receive the mark for this task.

Although this is the most complex part of the first section of this lab, it should be the most fun!

This is roughly the half-way point in the lab. You should be able to complete the rest of the lab work in the second session. If the first session is not yet over, feel free to continue. Regardless of your progress, it is recommended that you continue to work on the tasks outside of the lab.

1.7 2-to-1 multiplexer — 10%

So far we have built a full-adder circuit. To make this an adder-subtractor, we need some additional logic. Before assembling the adder-subtractor, we shall make a 2-to-1 multiplexer.

Design the multiplexer in Logisim, and then implement it on a NAND board. For neatness and ease of implementation, you may wish to use a separate NAND board from your full-adder.

This task is assessed. To achieve full marks in this task, please ensure you do the following:

1. **In the lab:** Show a lab demonstrator your Logisim design *and* your NAND-board implementation. Allow them to verify that both of them implement a 2-to-1 mux correctly.
2. **For submission:** Submit a Logisim circuit design file, called **mux.circ** via the submission system.

If you demonstrate a working design, but fail to submit your Logisim file, you may not receive full marks.

1.8 Adder-subtractor — 15%

Now the multiplexer and full-adder should be combined to produce an adder-subtractor. Lecture 4 may assist you in the top-level implementation details. You will need to provide the inputs B and $\overline{B}$ to your multiplexer, but this can be achieved with a single NAND gate.

Design this first in Logisim and satisfy yourself that both addition and subtraction are working correctly, then modify and combine your NAND board implementations of the full-adder and 2-to-1 mux to implement the adder-subtractor.

This task is assessed. To achieve full marks in this task, please ensure you do the following:

1. **In the lab:** Show a lab demonstrator your Logisim design *and* your NAND-board implementation. Allow them to verify that both of them implement a full-adder correctly.
2. **For submission:** Submit a Logisim circuit design file, called **full-adder.circ** via the submission system.

If you demonstrate a working design, but fail to submit your Logisim file, you may not receive full marks.

Storage

The next few tasks require that you understand the material in Lecture 5. Referring to the slides will assist you in completing them successfully, although you are also free to do additional research elsewhere.

1.9 Simple latch - The SR Latch

The tasks move now towards *storage* logic, starting with latches.

The SR latch has two inputs, S and R, and two outputs, Q and $\overline{Q}$. As per the transition tables (which you can find in Lecture 5, amongst other places), when $S = 0$ and $R = 0$, the current values on the outputs remain unchained. Asserting $S = 1$ will ***set*** $Q = 1$ and $\overline{Q}$ to its complement. Asserting $R = 1$ will ***reset*** the latch so that $Q = 0$. The condition $S = 1, R = 1$ is invalid and should be avoided.

Implement an SR latch in both Logisim and on a NAND board. Take care with the fact that in the simplest NAND implementation, the inputs are treated as $\overline{S}$ and $\overline{R}$, so you may wish to invert the inputs in order for the circuit to be easier to understand.

1.10 D-type latch — 10%

Now implement a D-type latch in Logisim and on a NAND board. A D-type latch has an input D, which is propagated to the output only when the enable input, E is set to logic-1. Changes to D when $E = 0$ should not change the outputs.

This task is assessed. To achieve full marks in this task, please ensure you do the following:

1. **In the lab:** Show a lab demonstrator your Logisim design *and* your NAND-board implementation. Allow them to verify that both of them implement a D-type latch correctly.
2. **For submission:** Submit a Logisim circuit design file, called **d-latch.circ** via the submission system.

If you demonstrate a working design, but fail to submit your Logisim file, you may not receive full marks.

1.11 D-type master-slave flip-flop — 15%

Using your D-type latch implementation, extend your design to create a D-type master-slave flip-flop. This will make your storage component *edge sensitive* to the E (or clock) input, rather than *level sensitive* and transparent. **Ensure you make a master-slave implementation, as there is more than one variety of D-type flip-flop.**

You should implement this, as usual, in both Logisim and on the NAND boards. When testing with the NAND boards notice how the *master* and *slave* parts of the circuit operate on different levels of the clock input.

This task is assessed. To achieve full marks in this task, please ensure you do the following:

1. **In the lab:** Show a lab demonstrator your Logisim design *and* your NAND-board implementation. Allow them to verify that both of them implement a D-type flip-flop correctly.
2. **For submission:** Submit a Logisim circuit design file, called **d-ff.circ** via the submission system.

If you demonstrate a working design, but fail to submit your Logisim file, you may not receive full marks.

Tips

When using Logisim, you may wish to use a `Clock` component from the `Wiring` library for your clock input, instead of a `Constant`. You can then make the clock "tick" by pressing `ctrl+T`, or have a free-running simulation by pressing `ctrl+K`. Of course, with the NAND boards, you will have to clock the device yourself by repeatedly connecting and disconnecting an input wire.

1.12 Shift-register — 5%

The final assessed part of this lab is another combined effort. With other groups, build a shift register, by chaining together the Q output of your master-slave flip-flop to the D input of the next device, and so on. You will also have to chain the clock signal together.

A shift register *shifts* values along its chain. At time $T = 0$ cycles, if a value $D = 1$ is input into the left-most shift register, than at $T = 5$ cycles, that value will be present on the Q output of the *fifth* flip-flop in the chain. A history of the input sequence will therefore be visible across the shift register.

To receive the allocated marks for this task, your flip-flop design must form part of the cohort's shift register, and the whole shift register must work correctly. It is, therefore, in everybody's interest to ensure that the individual components work correctly, and that they are connected together correctly.

1.13 *For fun:* A Ring Oscillator

If you find playing with logic is fun... good for you! As a reward for getting this far, implement a Ring Oscillator (see Lecture 4) using your NAND board(s). Here are some questions for you to think about:

- When you complete the chain, what can you see?
- What happens when you use an even number of inverters instead of odd?
- What information do we need to work out what frequency the Ring Oscillator is running at?
- What would happen if we reduced the supply voltage to the NAND board, for example, to 3 V?

Although this is just for fun, thinking about and trying to answer these questions will further your understanding of digital logic.

Submission

Each assessed task has included instructions on what to submit and what to show during the lab sessions. However, to reiterate, please remember to ensure the following.

- During the lab:
  - Ensure the lecturer or lab demonstrator has seen any designs that you have physically constructed. The lab demonstrators **will be marking you** for successful designs.
- By the submission deadline, please submit the following files:
  - All Logisim `.circ` files associated with each of the tasks. Remember, submitting these is just as important as demonstrating your working designs in the lab!

Although you will receive some feedback on your designs in the lab from the markers, your final mark is subject to change.

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.