NAND 3/3: Storage

Kerstin Eder

September 23, 2021

This lab continues to use Logisim and the NAND boards. In the previous lab, you made arithmetic logic, all circuits were *combinatoric*, i.e. results appear as quickly as the gate delays allow. This week, you will make latches and flip-flops to store data. Flip-flops are *sequential*, meaning they are synchronous to a *clock signal*. 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

- Continue to familiarise yourself with Logisim for logic circuit design, simulation and testing.
- Transfer designs onto NAND boards, aiming for a one-to-one match of the Logisim design to the implementation on the NAND boards. Please consult the previous lab sheets for guidelines on the NAND board kits and the note below.
- Create synchronous logic that could be combined with combinatorial logic to create finite state machines.

Lab overview and guidelines

The lab is broken down into tasks, with each task of increasing difficulty or requiring some additional understanding. It is recommended that you proceed in the order the tasks are given in this lab sheet.

To achieve a good level of understanding and to complete this lab in the allocated time, you will need to come well prepared to this lab session.

Please read through the entire lab sheet before starting the lab. Important information is contained on the last page for you to successfully complete the task on **JK flip-flops**.

In general, it is assumed that you have invested a significant amount of study time prior to attending the lab, e.g. watching the recording, engaging in the exercises, reading up on the topics covered in the lectures, practising, designing and testing your circuits in Logisim in your own time in preparation.

Remember that you can get support from the students sitting in close proximity to you, i.e. your informal lab group, and from your TA during the weekly lab sessions.

1 Preparation: NAND boards

1.1 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.

1.2 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.

1.3 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 1 illustrates the basic layout of the NAND boards. You may want to use this as a template for designing your circuits before building them.

1.4 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 1 can serve as a template for your designs.

Please note that, while the Logisim simulator offers a wide range of logic gates, the NAND boards only have `NAND` gates with *two* inputs. The fundamental idea is that the designs you build and test with the Logisim simulator are a one-to-one match with what you implement on the NAND boards. So, it is really important that your Logisim designs are directly implementable on the NAND boards, i.e. without modification of the logic. In particular, do not use 3-input `NAND` gates in your Logisim designs because the NAND board does not have these.

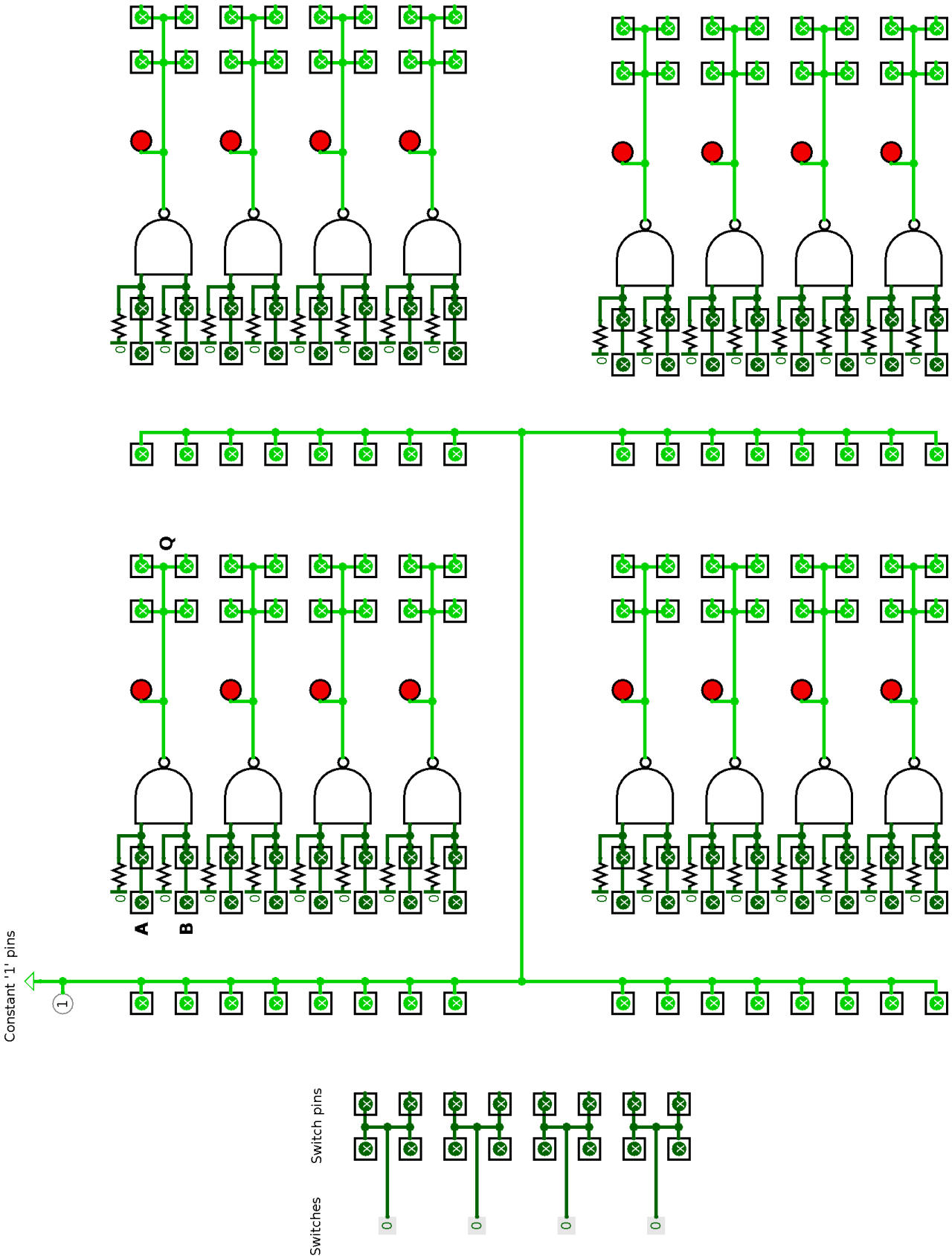

Figure 1: NAND board layout

Storage

The following tasks require that you understand the material in the lecture on "Storage". Referring to the slides will assist you in completing them successfully. Please note that you are also expected to do additional self-study in preparation of this lab.

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

2 Warm-up: SR latch

The `NAND`-gate based SR latch has two inputs, S' (or $\overline{S}$) and R' (or $\overline{R}$), and two outputs, Q and $\overline{Q}$. As per the transition tables (which you can find in the lecture slides), when $S' = 1$ and $R' = 1$, the current values on the outputs remain unchanged (HOLD). Setting $S' = 0$ will ***s**et* $Q = 1$ and $\overline{Q}$ to its complement (SET). Setting $R' = 0$ will ***r**eset* the latch so that $Q = 0$ (RESET). The condition $S = 0, R = 0$ 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`-gate based implementation, the inputs S' (or $\overline{S}$) and R' (or $\overline{R}$) are active low, so you may wish to invert the inputs in order for the circuit to be easier to understand.

Test your SR latch systematically, both in Logisim and on a NAND board.

3 Stepping stone: D-type latch

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.

Building a D-type latch is a stepping stone to building a D-type master-slave flip-flop. It is strongly recommended that you perform this task, fully test your design and explain why it works to some other students located in close proximity to you before moving on. Take turns with this, please, so that everyone can contribute to the discussion.

4 D-type master-slave flip-flop

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.

Again, think carefully about how best to test that your solution works correctly.

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 or toggling a switch.

5 Shift-register

This task requires combining several D-type master-slave flip-flops. You can do this in the lab with working D-type master-slave flip-flops of other students, or by re-using your D-type master-slave flip-flop design in Logisim.

Before connecting designs, demonstrate to the other students how your design works and show that it works correctly. Please take turns so that everyone gets a chance to explain their design.

You can then 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 signals together.

A shift register *shifts* values along its chain. For instance, at time $T = 0$ cycles, if a value $D = 1$ is input into the left-most shift register, then 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.

Again, think carefully about how best to demonstrate that your shift register works correctly.

6 JK flip-flop

This lab has, so far, focused on D-type devices, with a clock or enable input, and a data input. Your final task is to design, implement and build a device conforming to the *JK flip-flop* definition. Please note, you will need the template as described below to complete this task.

The JK flip-flop has *three* inputs: `J`, `K`, and `clock`. The characteristic table is as follows:

J	**K**	**Description**	$\mathbf{Q_{next}}$
0	0	hold	Q
0	1	reset	0
1	0	set	1
1	1	toggle	$\overline{Q}$

Table 1: JK flip-flop characteristic table.

Make a version of this that is *positive edge triggered* and uses a master-slave configuration.

You will need to do your own research to establish how to make such a circuit. Conceptually, how does a JK flip-flop differ from other types of flip-flops?

Template

Logisim is a simple tool and has its own flip-flop libraries. However, this lab asks you to build your own flip-flops from `NAND` gates. Logisim sometimes has difficulty resolving the state of wires if there are 'unknown' conditions several layers deep. To that end, simulating in Logisim is *slightly* tricky, but not impossible.

We provide a template, `jk-ff-template.circ` to get you started, and to overcome this problem. It is available from Blackboard.

Use the labelled inputs as they are intended. There are two feedback inputs that you will very likely need in your final design. Your first two `NAND` gates are provided — connect them to the rest of your design appropriately. To reset the design, simply set the Reset input to 1, then cycle the clock input (press ctrl+t twice). Then, return Reset to 0. This is known as a *synchronous reset*, because it is controlled by the clock.

Your NAND board implementation should work *without* needing the reset circuitry, therefore J, K and the feedback inputs can be *directly connected* to the first `NAND` gates.

7 Extension: 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 the lecture slides on **"Storage"**) using your NAND board(s). Here are some questions for you to think about:

- When you complete the chain, what can you see and why is this happening?
- What happens when you use an even number of inverters instead of odd?
- What information do we need in order 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? (The standard provided via USB is typically 5 V.)

Thinking about and trying to answer these questions will further your understanding of digital logic. Discuss your answers with other students on this unit.

Well done **for completing all the NAND labs.**