CPSC 121: Models of Computation
Lab #6: Sequential Circuits

Objectives

In this lab, you will learn about sequential circuits and get pratice implementing them in Logisim. You will also learn about a new type of memory called RAM, or random access memory. RAM is different from the other memory circuits you've seen so far in that it contains multiple pieces of data instead of just one. RAM is a very common type of memory used in real machines and you will see how it functions in a working computer in the last lab of the term.

1 Pre-lab

Recall that in the last lab, we looked at a different way of thinking about counting. The formula

$$t_n = t_{n-1} + 1$$

(where $t_0 = 0$) uses the previous value of t to get the next one. For example,

$$t_1 = t_0 + 1 = 0 + 1 = 1$$
$$t_2 = t_1 + 1 = 1 + 1 = 2$$

Take a look at the image below, which shows an incomplete sequential circuit that we want to finish implementing so that it simulates counting. That is, starting from 0, the output of the circuit should increase by 1 each time we clock the system.

TODO (pre-lab): Complete the sequential circuit by adding wires and appropriate inputs so that the system counts. Your inputs need to be **specific** values and not just variables, say, x or y. The **adder** takes in two inputs and produces the sum of their values and the **register** is a type of memory circuit which behaves like a multi-bit flip-flop. Note that the register initially contains the value 0.

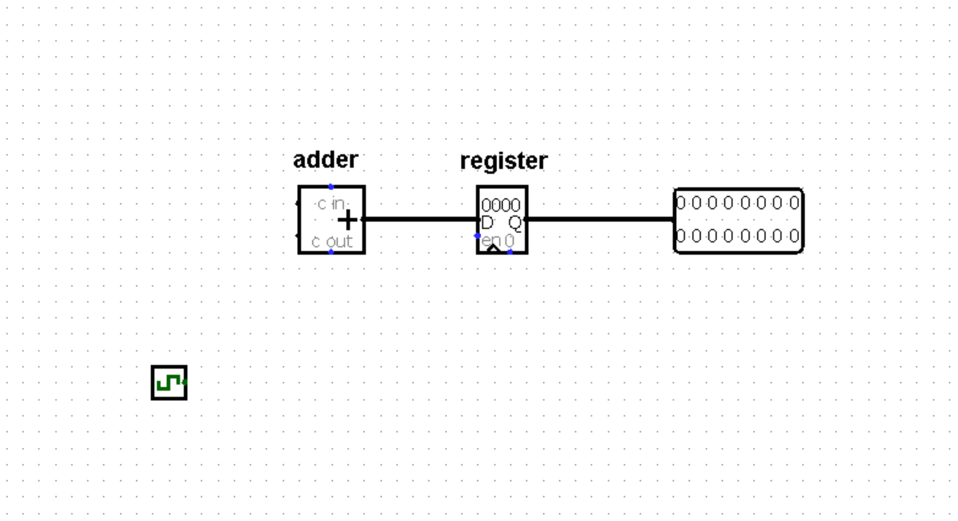
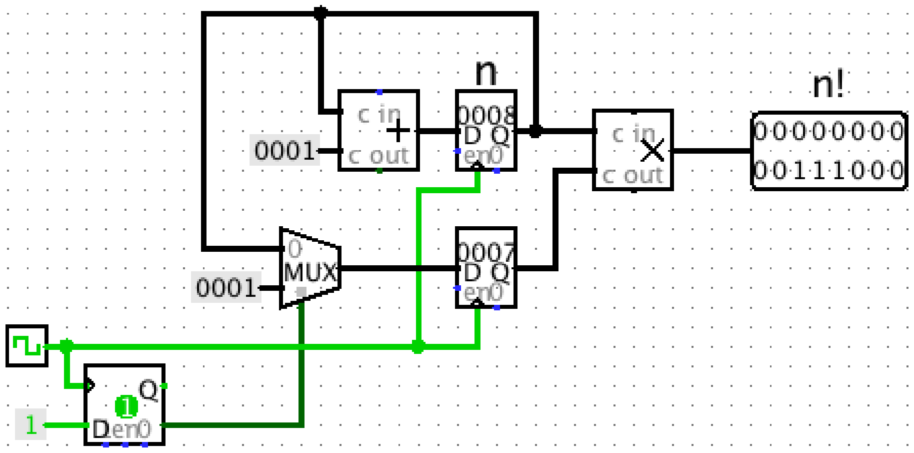


Figure 1: An incomplete sequential circuit that should simulate counting

2 Factorials

The factorial of a positive integer n can be defined as $n! = n \cdot (n-1)!$ for all numbers greater than 0; if n is 0, then $n!$ is 1.

TODO: Download the file `factorial_bug.circ` (shown below) from the course website. This circuit is meant to calculate factorials: identify the problem with the circuit, and correct it. The bottom-left flip-flop in this circuit is used to load in an initial value of 1 on the first clock-tick.



You might find it helpful to use Logisim's logging function. Open "Logging" from the "Simulate" menu. The left-hand column under the "Selection" tab displays components that you can add to the right-hand column, which will allow you to track their values as you clock the circuit. Once you've done this, you can click each item in the right-hand column and press "Change Radix" to choose whether you'd like the value to display in binary, decimal, or hexadecimal. To view the values, click on the "Table" tab. As you clock the circuit, you will now see a log of the changing values.

3 Fibonacci Numbers

A Fibonacci number works like this:

- if n is 0 or 1, $f(n)$ is 1
- all subsequent numbers follow the pattern $f(n) = f(n-1) + f(n-2)$

For example, the series starts as follows:

$$f(0) = 1$$
$$f(1) = 1$$
$$\left.\begin{array}{l} f(2) = f(1) + f(0) = 2 \\ f(3) = f(2) + f(1) = 3 \\ f(4) = f(3) + f(2) = 5 \\ f(5) = f(4) + f(3) = 8 \\ f(6) = f(5) + f(4) = 13 \end{array}\right\} \rightarrow f(n) = f(n-1) + f(n-2)$$

TODO: Design and implement a sequential circuit (much like the Factorial circuit above) which calculates Fibonacci numbers. Though you are free to begin working in Logisim, it might be easier for you to figure out this circuit if you begin by designing it on paper.

When implementing the circuit, use Logisim's register and adder modules, which work with multi-bit inputs. You can change the number of bits a component works with by selecting it with the edit tool, and changing the "Data Bits" field in the lower left-hand navigation panel.

Note: Your circuit does not have to produce the value of $f(n)$ when n is 0 and 1. For example, if you build an 8-bit circuit, it can calculate the sequence 2, 3, 5, 8, 13, 21, ... , 89.

4 Random-access memory

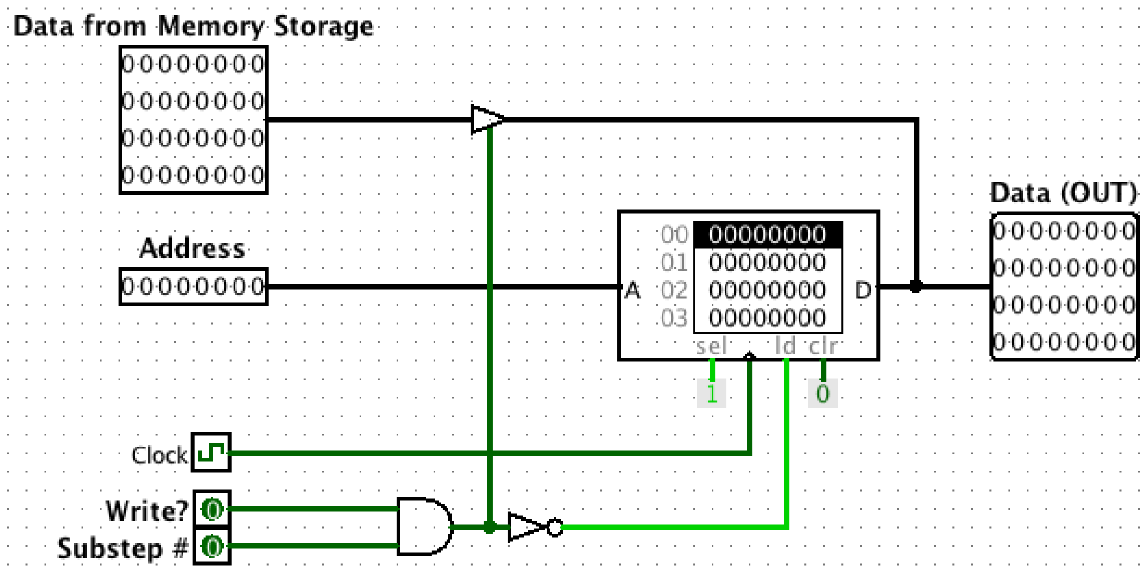
You have now seen two different memory circuits in Logisim: the flip-flop and the register. This section introduces you to RAM or random-access memory.

Download the file `Y86-RAM.circ` from the course website and load it in Logisim. It will look like the image below. This circuit simulates a component of a fully-functional computer processor, like the one in your computer. By the last lab of this term, you will be able to explore and understand a Y86 processor simulation. The component below is similar to the 16 MB RAM module that you will see in this Y86 simulation.

Each row in this circuit's RAM module (click on the components to identify which module this is) stores one 8-digit hexadecimal number. The address of each row in the memory is shown by a grey number to the immediate left of the row. The Y86 16 MB RAM module will hold a series of hexadecimal numbers that constitute a set of instructions used to program the hardware, much like the code you will load into the RAM below.

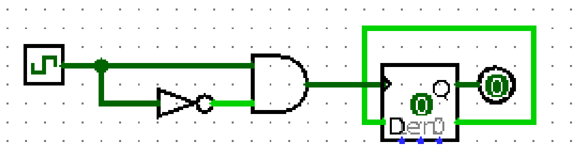
TODO: Figure out how to load the following set of hexadecimal instructions into your RAM module, and then show your TA the loaded values.

30820000 00013181 00000005 30800000



Set `Write?` and `Substep #` to 1 and ignore them: they are part of the larger circuit and not important for this exercise. Note that the instructions provided are represented in hexadecimal and this is how they will appear in the RAM module. However, they are represented in binary in both `Data from Memory Storage` and `Data (OUT)`

5 Further Analysis



TODO: Download the file `and_bug.circ` from the course website. This circuit has a `race condition`: a bug that can occur in both an electronic or software system. The Therac-25 radiation therapy machine, and the 2003 Northeast Blackout are both examples of a race condition with disastrous consquences. **Look up and explain what a race condition is, and how the circuit you downloaded illustrates this. What unexpected behaviour is occurring because of this bug? Think of potential implications that a race condition might have.**

6 End of Lab Survey

TODO: To help us improve these labs both this term and for future offerings, complete the survey at http://www.tinyurl.com/cs121labs.

7 Note about Lab 7

In the next lab, you will be completing lab 7. **For this lab, you and your partner will submit a prelab together, rather than individually**.

8 Challenge problem

Design and implement in Logisim a circuit to find the Hamming distance between two binary numbers of the same length. That is given 2 binary bit patterns, design a circuit that computes the number of substitutions necessary to transform one pattern into the other.

For example:

A = “01110001”
B = “10110000”

Hamming Distance = “0011” or 3, since 3 substitutions are necessary to transform A to B.

9 Marking scheme

All labs are out of ten marks, with two marks for pre-labs, and eight marks for in-lab work. In more detail:

- 2 marks - Pre-lab questions
- 5 marks - In-lab questions – In this lab, it is 2 marks for factorials, 2 marks for Fibonacci numbers, and 1 mark for RAM.
- 2 marks - Further analysis questions.
- 1 mark - End of lab survey.

TAs may at their discretion award one bonus mark, such as for completing a challenge problem. It is expected that most students will achieve a 6-8. If you feel like you’re heading for 0-5, **get immediate help from the TAs!**