CPSC 121: Models of Computation
Lab #3: Scalability

Objectives

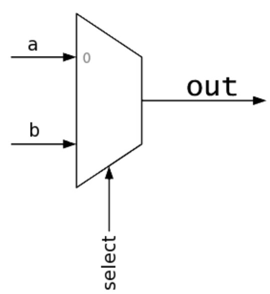
In the last lab, you were introduced to a type of circuit called a multiplexer, which is used for selection. In this lab, you will review multiplexers in more detail by implementing them on *The Magic Box* and by building a multiplexer that selects from four inputs. You will also learn about the issue of scalability and be introduced to a type of circuit called the priority chain.

1 Pre-lab

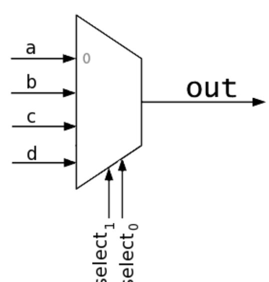
A 1-of-2 multiplexer is a circuit with three inputs: `a`, `b` and `select`:

- When `select` is 0, the multiplexer outputs the value of `a`.
- When `select` is 1, the multiplexer outputs the value of `b`.

A 1-of-2 multiplexer is usually drawn out like this:



We can have multiplexers with even more inputs. For example, a 1-of-4 multiplexer is drawn out like this:



The table below shows what a 1-of-4 multiplexer outputs for different values of `select1` and `select0`

select1	select0	output
0	0	a
0	1	b
1	0	c
1	1	d

TODO (pre-lab): A 1-of-4 multiplexer can be built using three 1-of-2 multiplexers. Show a diagram of how this can be done, explicitly labeling `select1`, `select0`, `a`, `b`, `c`, and `d`. Use the symbol above for your 1-of-2 multiplexers rather than drawing out individual AND, OR, and NOT gates. Your diagram **must** agree with the table above.

TODO (pre-lab): Using NAND gates, draw a corresponding circuit for each of the two following equations:

$$(i) \qquad \sim(\sim(\texttt{a} \wedge \texttt{b}) \wedge \sim(\texttt{c} \wedge \texttt{d}))$$

$$(ii) \qquad \sim(\texttt{a} \wedge \sim(\texttt{b} \wedge \sim(\texttt{c} \wedge \texttt{d})))$$

TODO (pre-lab): Are circuits (i) and (ii) from the preceding question (and therefore their equations) equivalent? Why or why not?

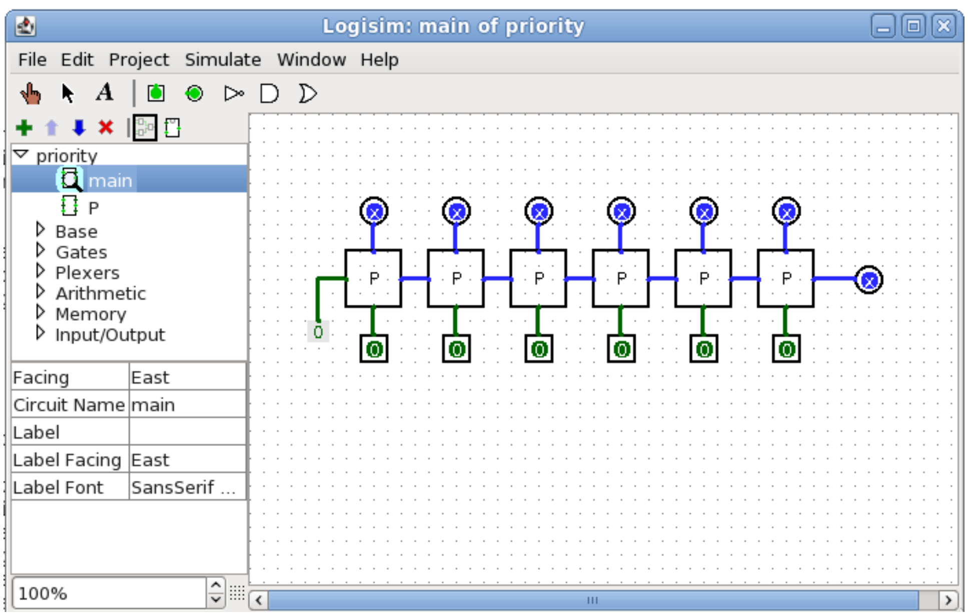
2 Working with multiplexers

Look up the LS157 chip in The Magic Box User's Manual. The LS157 chip contains four 1-of-2 multiplexers (with a single shared `select` input). Be sure that you read the documentation on the `ENABLE` input carefully! **TODO: Wire up a multiplexer in the LS157 chip on your breadboard, and verify that it has the same truth table as a 1-of-2 multiplexer.**

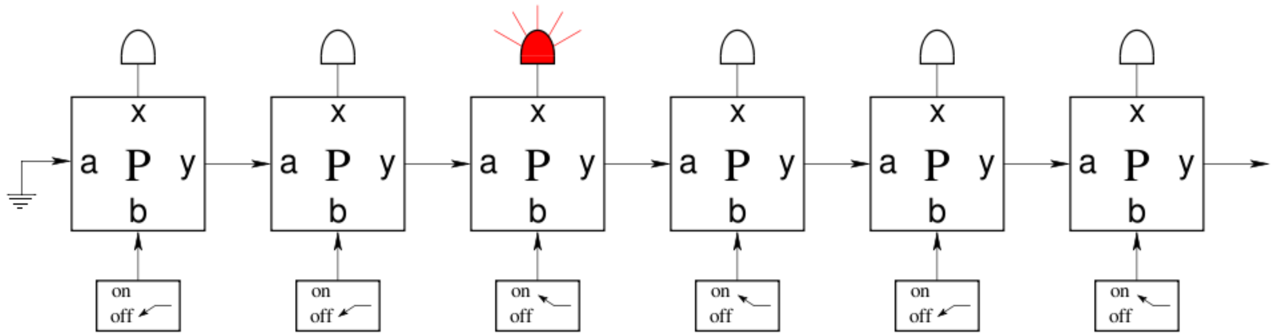
2.1 A Priority Chain

Circuit design is done much like software design. In code, we break our programs up into functions or methods: these can then act as self-contained modules, which can be reused easily. This modularity is also a principle of good circuit design.

Download the file priority.circ from the course website, and open it in Logisim (File→Open). Modules in Logisim are represented by rectangles. You will see a program that has six identical modules wired up in a chain:



Our goal with this activity is to edit the P module to implement what is called a *priority chain*. In this priority chain, the only light that will be turned on is at the first module where b is on. This module of the chain is said to have the *priority*. (We say that stages to the left have higher *priority* than stages to the right.) The chain lights the LED for the leftmost switch that is in the on position, while all other LEDs remain off. Here is an example:

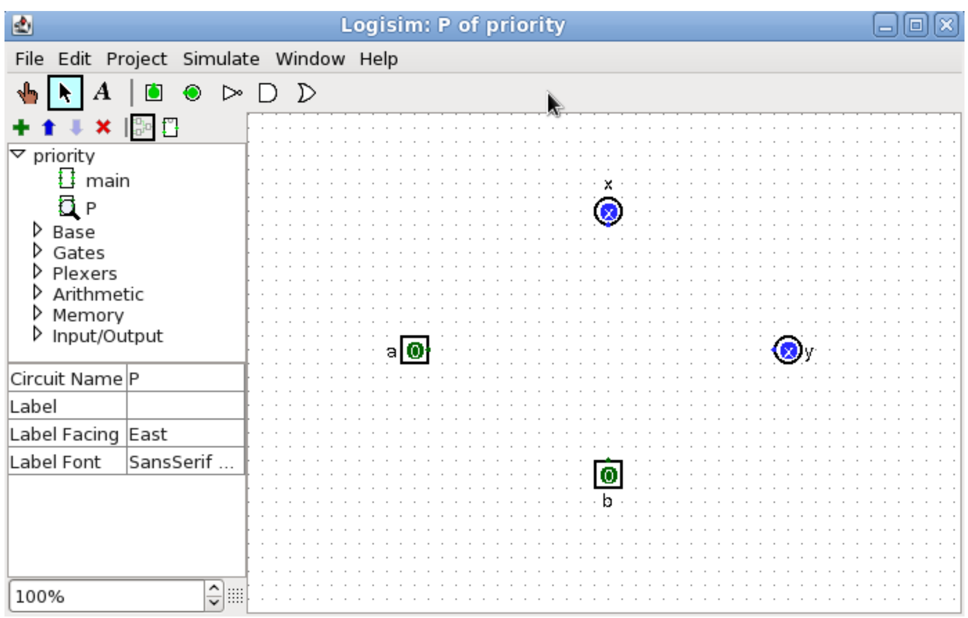


We see in this image that the first module with b = TRUE has x illuminated, but later modules with b on do not. Also, note that the first a of the circuit is connected to ground (FALSE). Therefore, we want a circuit which has the properties:

- x is only on if a is off and b is on.
- y is only on if a is on or x is on.

TODO: Fill in the P module in our Logisim circuit so that main has a working priority chain like in the example described above. Test whether the circuit works as expected.

To edit the module, right-click P in the upper left-hand menu (underneath main) and select "Edit Circuit Layout". You will get a circuit like this (**Note: Don't move the pins in the layout!**):

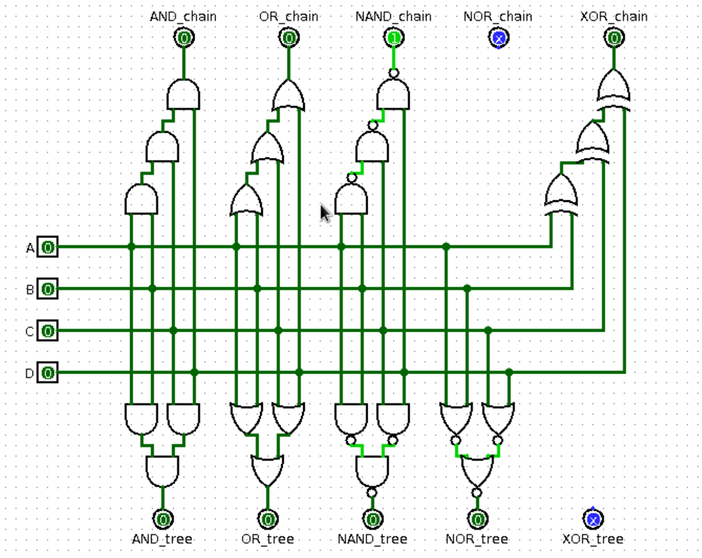


Once you've filled in this circuit, switch back to the main module by right-clicking it in the upper left-hand menu and selecting "Edit Circuit Layout". Verify the priority chain works as expected, and then show it to your TA.

Note: On the *bottom* left-hand of your Logisim screen is the Attributes Panel. This allows you to change certain attributes of various components. For example, you can change the number of inputs to a gate, or the number of data bits to each input or output. You can also change the size of a gate, or the direction it faces.

3 Chains and Trees

Download the file `chains.circ` from the course website and open it in Logisim. It will look like the image below. **Add the gates and connections for the NOR_chain and XOR_tree following the pattern of the file.**



TODO: Answer the following questions:

1. Which of the chains and/or trees compute the function that is true *if and only if* an odd number of the inputs A, B, C, and D are true?

2. Which chains and/or trees compute the function $(A \wedge B) \vee (C \wedge D)$?

TODO: How did you determine these? How many different strategies can you think of to do these? What are the advantages/disadvantages of these strategies and of the one(s) you used?

3.1 Scalability

Scalability is a critical issue in computer science: Will a system slow “too much” as we increase the number of inputs that it handles? For this week’s prelab, you drew two circuits using NAND gates. One of these was a NAND tree and the other was a NAND chain. Although they have the same number of gates, one of these circuits takes a longer time to return its output. In general, for a particular gate, the tree and chain implementations will differ in how well they scale as the number of inputs increases.

TODO: Assuming each gate takes 20 picoseconds to produce its result, how long will it take for the 3-gate AND chain to produce its result? How long for the 3-gate AND tree?

How much speed difference would there be in determining whether the 128 individual Boolean values that form a 128-bit number are 1s by ANDing them together with an AND chain vs. an AND tree? (Here, each bit corresponds to a switch at the input of the circuit.)

Calculate how long it would take for the 128-bit AND chain, and for the AND tree.

4 Further Analysis Questions

TODO (further analysis):

1. Consider the amount of time a circuit takes to produce an output for a given a number of inputs.
   Which scales better as the number of inputs increases: trees or chains? Why?
2. Look back at your pre-lab on how to make a 1-of-4 multiplexer. This 1-of-4 multiplexer has a delay of 2 time units (1 time unit for each 1-of-2 multiplexer) before the selected signal is propagated to the output. How many time units of delay would a 1-of-8 multiplexer have? **Now consider a 1-of-n multiplexer: how does the propagation delay of this multiplexer scale as n increases? We are looking for a formula that relates the number of time units in the delay to the size of the multiplexer (n)**

5 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.

6 Magic Box Cleanup

TODO: Before leaving the lab, show your Magic Box to your TA.

7 Challenge Problem

Open a new file in Logisim. Under the "Plexers" menu, find the Priority Encoder and the Decoder. **TODO (challenge): Experiment in Logisim and get these two circuits to work together: what do they do?** Hint: Try wiring the output of the Priority Encoder to the input of the Decoder.

8 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:

- Two marks - Pre-lab questions
- Five marks - In-lab questions. Two marks are for Section 2. Three are for Section 3.
- One mark - Further analysis questions.
- One mark - End of lab survey.
- One mark - Magic Box cleanup.

TAs may at their discretion award one bonus mark, such as for completing the challenge problem.