Project 1: Elementary Logic Gates

A typical computer architecture is based on a set of elementary logic gates like And, Or, Mux, etc., as well as on their bitwise versions And16, Or16, Mux16, etc. (assuming a 16-bit machine). In this project you will build a typical set of basic logic gates. These gates form the elementary building blocks from which you will build the computer's CPU and RAM chips in later projects.

Below we describe the tools, resources, and implementation tips needed for completing project 1.

Objective

Build the following chips (we use the terms *chip* and *gate* interchangeably):

Nand (given)
Not
And
Or
Xor
Mux
DMux
Not16
And16
Or16
Mux16
Or8Way
Mux4Way16
Mux8Way16
DMux4Way
DMux8Way

Since Nand is considered primitive, there is no need to implement it.

Files: For each chip Xxx in the list, we provide a skeletal Xxx.hdl program, also called *stub file*, with a missing PARTS section. In addition, for each chip we provide an Xxx.tst script that tells the hardware simulator how to test the chip, along with an Xxx.cmp compare file containing the correct outputs that the supplied test is expected to generate. Your task is writing, and testing, the chip implementations (specifically: Completing the supplied Xxx.hdl files).

Contract: For each chip in the list, your chip implementation (modified Xxx.hdl file), tested by the supplied Xxx.tst file, must generate the outputs listed in the supplied Xxx.cmp file. If the actual outputs generated by your chip disagree with the desired outputs, the simulator will report error messages.

Building the chips

A new online IDE (Integrated Development Environment) was recently launched for learners of Nand to Tetris courses. Therefore, there are now two options for completing project 1:

If you are using the Nand2Tetris IDE Online (which is recommended), all the Xxx.hdl, Xxx.tst and Xxx.cmp files are available in your browser memory. To develop and test a particular chip, select the project / chip from the simulator's drop-down menus. Your edited HDL code will be saved automatically. If you wish to download the HDL files to your local PC, click the *download* button. The current version of all the project's Xxx.hdl files will be downloaded as one zip file.

Using the desktop Nand2Tetris hardware simulator is also possible. If you've downloaded the Nand to Tetris software suite from www.nand2tetris.org, and extracted it into a folder named nand2tetris on your computer, the nand2tetris/tools folder contains the desktop version of the hardware simulator, and the nand2tetris/projects/1 folder contains all the files needed for completing this project. You can write/edit the HDL code of each Xxx.hdl file using any plain text editor, and then test your code using the desktop simulator.

HDL documentation: We sometimes use abbreviated notation. For example, the comment "if (in) out = 1, else out = 0" means: "if (in = 1) then set out to 0, else set out to 1". Likewise, the condition "if (a and b) …" means "if (a = 1 and b = 1) …" , etc.

References

HDL Guide

Chips Set API

Tutorials

The tutorials below focus on using the desktop version of the hardware simulator. Tutorials for the online simulator, the preferred tool for this project, will be available soon. However, you can apply the principles from these tutorials to perform similar actions in the online simulator (a major difference is that there is no need to load any files in the online simulator).

Hardware Simulator: Intro

Building and Testing Chips

Script-Based Chip Simulation

Hardware Simulator Tutorial (click *slideshow*)

Consult each reference / tutorial as needed: There is no need to go through the entire resource.

Implementation Tips

0. Before implementing a chip, it is recommended to experiment with its builtin implementation. If you are using the online simulator, simply click the *builtin* toggle; If you are using the desktop version, load the builtin chip from the tools/builtInChips folder.

1. Each chip can be implemented in more than one way. The simpler the implementation, the better. As a general rule, strive to use as few chip-parts as possible.
2. Although each chip can be implemented directly from Nand gates only, you can use any other chip from project 1 as a chip-part, as needed (see the previous tip).
3. There is no need to build “helper chips” of your own design. Your HDL programs should use only the chips listed in this project.
4. We recommend implementing the chips in the order in which they are listed. If, for some reason, you don’t complete the HDL implementation of some chip, you can still use it as a chip-part in other HDL programs. In the online simulator, the chip evaluation process uses builtin chip implementations, so there is no further ado. If you are using the desktop version, rename the chip file that you did not implement, or remove it from the project 1 folder. This will force the desktop simulator to use the chip’s builtin implementation.
5. Each chip can be tested interactively, or using a test script. In the online simulator, simply run the test script. In the desktop simulator, you must first load the chip’s test script, and then run it.