Project 1

A typical computer architecture is based on a set of elementary logic gates like And, Or, Mux, etc., as well as 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 logic gates:

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

The only building blocks that you can use are primitive Nand gates and the composite gates that you will gradually build on top of them. Since Nand is considered primitive, there is no need to implement it.

Files

We assume that you've already downloaded the Nand to Tetris software suite (from the Software page of www.nand2tetris.org), and that you've extracted it into a folder named nand2tetris on your computer. If that is the case, then the nand2tetris/tools folder on your computer contains the supplied hardware simulator, and the nand2tetris/projects/1 folder contains all the files needed for completing this project.

The gates should be implemented in HDL. For each chip Xxx, (we use the terms *chip* and *gate* interchangeably) we provide a skeletal Xxx.hdl program, sometimes called a *stub file*, with a missing implementation part. In addition, for each chip we provide an Xxx.tst script that tells the

hardware simulator how to test it, along with an Xxx.cmp compare file that lists the correct outputs that the supplied test is expected to generate. All these files are available in your nand2tetris/projects/1 folder. Your job is to complete and test all the Xxx.hdl files in this folder. These files can be written and edited using any plain text editor.

HDL documentation: In some HDL files we describe boolean conditions and boolean assignments using abbreviated notation. For example, suppose that a gate has an input pin named "in" and an output pin named "out". The meaning of the documentation "if (in) out = 1, else out = 0" is: "if (in=1) then set out to 0, else set out to 1".

HDL documentation: In some HDL files we describe boolean conditions and boolean assignments using abbreviated notation. For example, suppose that a gate has an input pin named "in" and an output pin named "out". The meaning of the documentation "if (in) out = 1, else out = 0" is: "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.

Contract: When loaded into the hardware simulator, your chip design (modified .hdl program), tested on the supplied .tst file, should produce the outputs listed in the supplied .cmp file. If the actual outputs generated by the simulator disagree with the desired outputs, the simulator will report error messages.

References

HDL Guide

Chips Set API

Hardware Simulator: Intro

Building and Testing Chips

Script-Based Chip Simulation

Hardware Simulator Tutorial (click *slideshow*)

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

Implementation Tips

Each gate 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.

Although each chip can be implemented directly from Nand gates only, use composite gates that were already implemented. See the previous tip.

There is no need to build "helper chips" of your own design. Your HDL programs should use only the chips listed above.

Implement the chips in the order in which they are listed above. 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. Simply rename the chip file, or remove it from the folder. This will force the simulator to use the chip's builtin implementation.

www.nand2tetris.org / Copyright © Noam Nisan and Shimon Schocken