Project 2: Boolean Arithmetic

The centerpiece of the computer's architecture is the CPU, or *Central Processing Unit*, and the computational centerpiece of the CPU is the ALU, or *Arithmetic-Logic Unit*. In this project you will gradually build a set of chips that carry out arithmetic addition, culminating in the construction of the ALU chip of the Hack computer.

Objective

Build the following chips:

HalfAdder

FullAdder

Add16

Inc16

ALU

We note in passing that all the chips listed above are standard, except for the ALU, which varies from one computer architecture to another.

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 2:

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. 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/2 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.

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

All the *Implementation Tips* of project 1 apply to this project also, so read them now.

There is only one modification: When implementing the chips of project 2, you can use, as chip-parts, any of the chips listed in project 1 and in project 2. If you are using the desktop simulator, don't add any files to the projects/2 folder: the desktop simulator will use builtin chip implementations, as needed.

The Hack ALU computes a 16-bit value named out. In addition, it computes two 1-bit outputs named zr and ng. We recommend building the ALU in two stages. First, implement a basic ALU that computes out, ignoring the zr and ng outputs. Then implement a final version that computes out as well as zr and ng. We provide two sets of files for testing each stage: ALU-basic.tst and ALU-basic.cmp for testing the basic version; ALU.tst and ALU.cmp for testing the final version.

If you are using the online simulator, which is the preferred option, you can select the ALU test script from the Test drop-down menu; If you are using the desktop simulator, you can load the ALU test script from the projects/2 folder.