Project 0: Getting Started

Background

Hardware engineers don't build chips in their bare hands. They design chips in a *Hardware Description Language* and test the resulting HDL programs using *hardware simulation* software. When the simulated chip passes a set of carefully planned tests, the tested HDL code can be committed to silicon, yielding a physical chip.

Objective

Get acquainted with the operational concepts of *chip* and *hardware simulation*; Simulate the operation of some sample chips using the supplied *Hardware Simulator*.

Chips

We will play with an arbitrary selection of some of the chips that appear later in the computer's construction. At this stage, what these chips actually do is not important, so don't worry about it.

Chip Name	Description	Test script	Compare file
And	And gate	And.tst	And.cmp
Mux8Way16	16-bit 8-way multiplexor	Mux8Way16.tst	Mux8Way16.cmp
Register	16-bit register	Register.tst	Register.cmp
RAM8	8-register RAM	RAM8.tst	RAM8.cmp

The chipName.tst files are test scripts designed to test the respective chips. Normally, a test script contains commands that generate output lines that are accumulated in a chipName.out file (not shown here). The chipName.cmp file contains the *desired* output of a successful test. Thus, chipName.out == chipName.cmp implies a successful test.

The first two chips in the table above are *combinational*; the latter two chips are *sequential*, i.e. clock-dependent. This distinction will become clear after you go through lecture 3 and chapter 3 ("Sequential Logic"). Until then, we recommend playing with the And and Mux8Way16 chips only. You may want to revisit this project and play with the other two chips in preparation for Project 3.

How to experiment with the chips

Prerequisite: If you haven't done it yet, download and install the Nand2Tetris Software Suite on your computer. Then Read the Introduction Chapter and go through parts I-II-III of the supplied *Hardware Simulator Tutorial*: ( )

Built-in chips: Later in the course we will learn how to implement chips using a *Hardware Description Language*, or *HDL*. For the purpose of behavioral simulation though, which is the focus of this exercise, we don't care how chips are implemented, as long as they deliver their expected functionality. If you've downloaded the *Nand2tetris Software Suite* to your computer, you now have a directory called `tools/builtInChips` on your computer. This directory contains Java implementations of various chips. Specifically, each chip has a chipName.hdl file that serves two purposes. First, it documents the expected behavior of the chip; second, it contains a command that invokes Java code that delivers the chip's functionality using Java rather than HDL. The Java code is stored in a class file named chipName.java.

Interactive simulation: To load a built-in chip into the supplied *Hardware Simulator*, invoke the simulator software, click the "load chip" icon, navigate to the `tools/builtInChips` directory, and select the desired chipName.hdl file. Once the chip's logic has been loaded into the simulator, you may enter different values into the chip's input pins, click the "Eval" icon, and inspect the results. (If you are simulating a sequential chip rather than a combinational chip, you have to use "TickTock" instead of "Eval"). You will notice that some built-in chips, e.g. Register and RAM8, have graphical side-effects. This Java-based GUI is not part of the chip's functionality; it is a simulation artifact that helps test the chip using a visual representation. This GUI is provided by the Java code stored in chipName.java.

Scripted simulation: Every chip in our chip set comes with a predefined chipName.tst file, written by us. This script includes testing commands that load the chip logic (chipName.hdl file) into the hardware simulator and then walk it through a predefined series of tests. To test a chip using a test script, load the supplied script file into the simulator, and proceed to execute it. Note that one of the first things that the test script executes is the "load chipNname.hdl" command. What happens if the user's directory contains no chipName.hdl file? In such cases, the simulator

automatically tries to load the built-in chipName.hdl file from the `tools/builtInChips` directory. This default enables chip designers to play with the expected behavior of chips before they set out to actually build them in HDL. Needless to say, in order for this practice to work, someone has to write the chipName.java code, yet this is normally easier and faster than implementing the chip's logic in HDL.

What's next?

You are now ready to do the hardware construction projects, beginning with Project 1. Most of the chips in our chip set have a built-in implementation, stored in corresponding `tools/builtInChips/chipName.hdl` files. Thus, before you set out to implement any chip in HDL, you are welcome to experiment with the chip's behavior in the hardware simulator: simply load and execute the respective built-in chip. This behavioral simulation is quite a blessing: it lets you play with a chip before you actually build it.