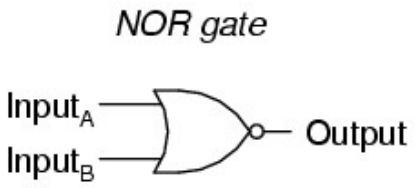
CSCE-312 Project 1

Background

A typical computer architecture is based on a set of elementary logic gates like AND, OR, MUX, etc., as well as their bit-wise versions AND16, OR16, MUX16, etc. (assuming a 16-bit machine). This project engages you in the construction of a typical set of basic logic gates. These gates form the elementary building blocks from which more complex chips will be later constructed.



A	B	Output
0	0	1
0	1	0
1	0	0
1	1	0

Objective

1. Build all the logic gates described below, yielding a basic chip-set. The only building blocks that you can use in this project are primitive **NOR gates and the composite gates that you will gradually build on top of them. Also, you must submit truth tables of all single bit logic gates built as part of this exercise**.
2. After implementing all elementary logic gates, use them to implement chips for the following logic scenarios:
    a. A student would fail an exam if he spent the previous night studying for the exam, or if he has not had breakfast before the exam.
    b. You cannot get onto the ride if you are too young and too short, or too old and have heart disease.

    Hint: Convert the sentence to Boolean algebra equation and draw the truth table of each scenario before implementation.

Deliverables

(a) HDL files for all the chips implemented and (b) one PDF containing truth tables for all single-bit logic gate functions.

Chips

Chip Name	File Name	Description
Not	Not.hdl	Not Gate
And	And.hdl	And Gate
Or	Or.hdl	Or Gate
Xor	Xor.hdl	Exclusive Or Gate
Mux	Mux.hdl	Multiplexer
DMux	DMux.hdl	Demultiplexer
Not16	Not16.hdl	16-bit Not Gate
And16	And16.hdl	16-bit And Gate
Or16	Or16.hdl	16-bit Or Gate

Credit: nand2tetris.org for original source and Yang Yang (TAMU) and Jerry Yiu (TAMU) for content additions

Mux16	Mux16.hdl	16-bit Multiplexer
Or8Way	Or8Way.hdl	8-bit input Or Gate
Mux4Way16	Mux4Way16.hdl	4-way 16-bit input Multiplexer
DMux4Way	DMux4Way.hdl	4-way 16-bit input Demultiplexer

Contract

When loaded into the supplied Hardware Simulator, your chip design (modified .hdl program), tested on the supplied .tst script, should produce the outputs listed in the supplied .cmp file. If that is not the case, the simulator will let you know.

Resources

The relevant reading for this project is Chapter 1 and Appendix A. Specifically, all the chips described in Chapter 1 should be implemented in the Hardware Description Language (HDL) specified in Appendix A. Another resource that you will find handy in this and in all subsequent hardware projects is this HDL Survival Guide, written by Mark Armbrust.

For each chip, we supply a skeletal .hdl file with a place holder for a missing implementation part. In addition, for each chip we supply a .tst script that instructs the hardware simulator how to test it, and a .cmp ("compare file") containing the correct output that this test should generate. Your job is to complete and test the supplied skeletal .hdl files.

Tips

Prerequisite: If you haven't done it yet, download the Nand2Tetris Software Suite to your computer. Read Chapter 1 and Appendix A, and go through parts I-II-III of the Hardware Simulator, before starting to work on this project.

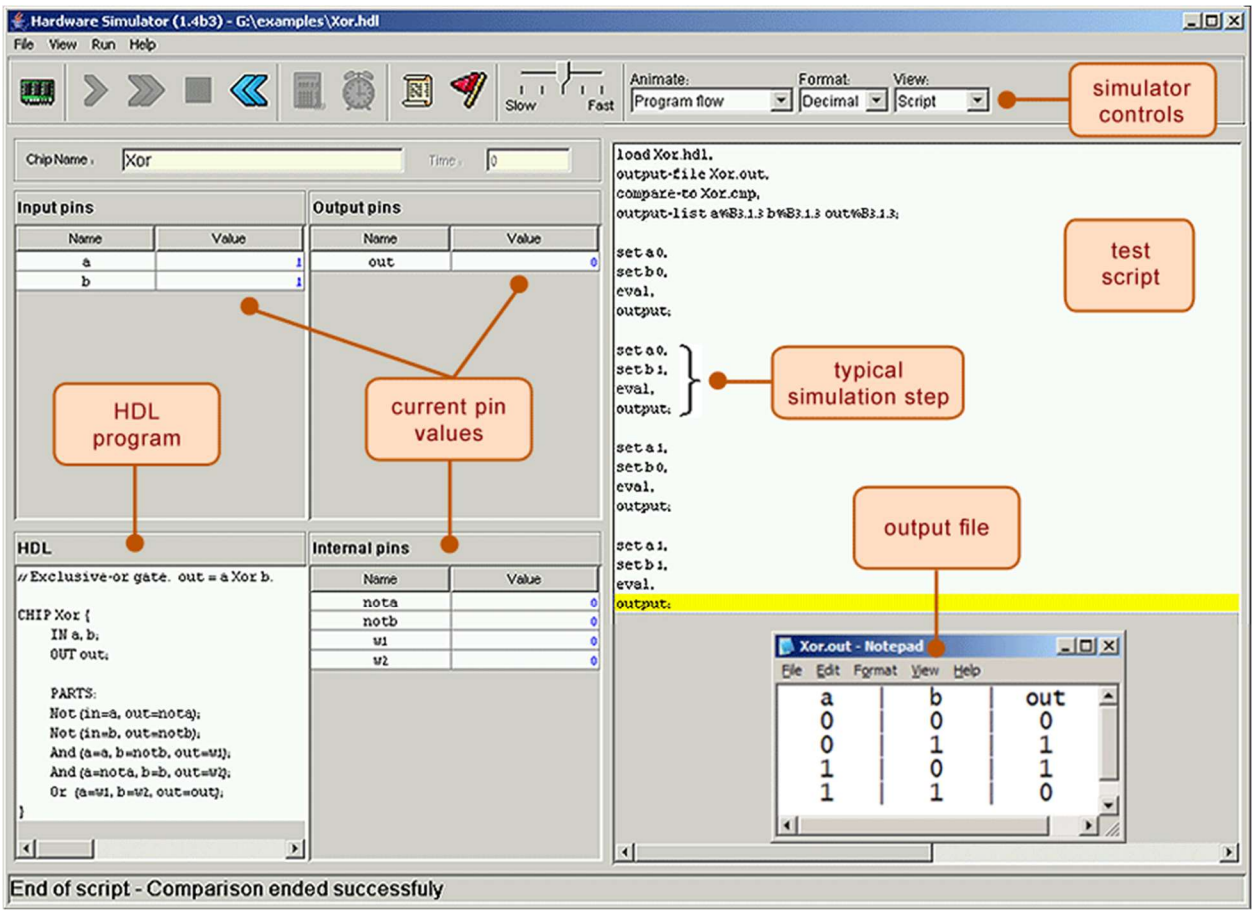
Built-in chips: The NOR gate is considered primitive and thus there is no need to implement it: whenever a NOR chip-part is encountered in your HDL code, the simulator automatically invokes the built-in tools/builtInChips/Nor.hdl implementation. We recommend implementing the other gates in this project in the order in which they appear in Chapter 1. However, note that the simulator's environment includes a library with built-in versions of all these chips. Therefore, you can use any one of these chips before implementing it: the simulator will automatically invoke their built-in versions.

For example, consider the supplied skeletal Mux.hdl program. Suppose that for one reason or another you did not complete the implementation of Mux, but you still want to use Mux chips as internal parts in other chip designs. You can easily do so, thanks to the following convention. If the simulator fails to find a Mux.hdl file in the current directory, it automatically invokes a built-in Mux implementation, which is part of the supplied simulator's environment. This built-in Mux implementation has the same interface and functionality as those of the Mux chip described in the book. Thus, if you want the simulator to ignore one or more of your chip implementations, simply rename the corresponding chiPname.hdl file, or remove it from the directory. When you are ready

to develop this chip in HDL, put the file chipName.hdl back in the directory, and proceed to edit it with your HDL code.

Tools

All the chips mentioned projects 0-5 can be implemented and tested using the supplied Hardware Simulator. Here is a screen shot of testing a Xor.hdl chip implementation on the Hardware Simulator:



Credit: nand2tetris.org for original source and Yang Yang (TAMU) and Jerry Yiu (TAMU) for content additions