CSCE-312 (504-505) | Fall 2018

Project 1

Building Boolean Logic Gates

Due Date: **Tue, Sept 11, 3:00 PM**

Grading

(A) Project Demo [50%]: Offline

You will be graded for correctness of the chips (hdl) you have designed and coded. We will be running offline test of all your HDL codes using Nand2tetris software (Hardware Simulator). The same simulator you will be using to check your chips in the course of this project. So, make sure to test and verify your codes before finally submitting on eCampus.

Rubric: PriorityEncoder83.hdl is worth 5 points and the others are worth 3 points each. Each circuit needs to pass all its test cases to get the points, else you will receive 0 on that circuit.

(B) Lab Quiz [50%]: Mon, Sept 10 lab

A paper-based quiz in the lab session. This is designed to test your knowledge of the project. These questions will not be disclosed prior for studying; however, they should not be difficult for you if you have understood the core inner workings of your project.

Deliverables & Submission

You need to turn in only the HDL files for all the chips implemented. Put your full name in the introductory comment present in each HDL code. Use relevant code comments and indentation in your code. Also, include this cover sheet with your signature below. Zip all the required HDL files and the signed cover sheet into a compressed file *FirstName-LastName-UIN.zip* (TA will show the students an example for this). Submit this zip file on eCampus.

Late Submission Policy: Refer to the Syllabus

Aggie Code of Honor:

"An Aggie does not lie, cheat, or steal or tolerate those who do."

Required Academic Integrity Statement:

"On my honor, as an Aggie, I have neither given nor received unauthorized aid* on this academic work."

_____ (Print your name)

_____ (Your signature)

* sharing or copying solutions from online resources or peers or any person (does not include teaching staff)

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.

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 gate** (HDL provided) and the following composite gates that you will gradually build on top of them.

2. After implementing all elementary logic gates, use them to implement chips for the following logic scenarios:

- **Exercise 1:** 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.
- **Exercise 2:** 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.

Here are the Chips you are required to implement:

Chip Name	**File Name**	**Description**
Not	Not.hdl	Not Gate
Or	Or.hdl	Or Gate
And	And.hdl	And Gate
Xor	Xor.hdl	Exclusive Or Gate
Mux	Mux.hdl	Multiplexer Gate
DMux	DMux.hdl	Demultiplexer Gate
Not16	Not16.hdl	16-bit Not Gate
And16	And16.hdl	16-bit And Gate
Or16	Or16.hdl	16-bit Or Gate
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
Exercise1	Exercise1.hdl	Exercise 1 based circuit
Exercise2	Exercise2.hdl	Exercise 2 based circuit
8 to 3 Priority Encoder	PriorityEncoder83.hdl	**Attend Wed, Sept 5 lab**

Contract

Download and uncompress the *P1Codes.zip* file in your local nand2tetris folder. You should have already deleted the default project directory already present in nand2tetris folder, as announced on the Piazza post.

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 of the textbook.
Specifically, all the chips described in **Chapter 1**
https://docs.wixstatic.com/ugd/44046b_f2c9e41f0b204a34ab78be0ae4953128.pdf
should be implemented in the Hardware Description Language (HDL) specified in **Appendix A**
https://docs.wixstatic.com/ugd/44046b_2cc5aac034ae49f4bf1650a3d31df32c.pdf
Another resource that you will find handy in this and in all subsequent hardware projects is this **HDL Survival Guide** written by Mark Armbrust https://www.nand2tetris.org/hdl-survival-guide

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,

1. Go over Piazza post https://piazza.com/class/jkmb76dcbzw2u4?cid=12
2. Read Chapter 1 and Appendix A, and go through parts I-II-III of the Hardware Simulator Tutorial ppt (see Piazza Resources section) , before starting to work on this project.
3. 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 builtin 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 in Projects 1-3 and 6 can be implemented and tested using the supplied Hardware Simulator. Here is a screenshot of testing a Xor.hdl chip implementation on the Hardware Simulator: