ECSE 425: Computer Organization and Architecture Project Overview

Winter 2017

This semester, the final computer architecture project is to design, implement, test, and optimize a pipelined processor. The project will be completed in two phases: (a) an introductory phase, completed individually, and (b) a group phase, completed in groups of four. The purpose of the introductory phase is to reintroduce VHDL and build basic components using principles that will be expanded upon in the group phase.

Change Log

27-Jan	Finalized plans for PD3, a Cache, a group assignment, worth 10%.
21-Mar	Finalized plans for PD4, PD5, and the final report.

Deliverables

The project is divided into a number of deliverables. The first ~~three~~ two correspond to individual work; the final four will be completed in groups of four.

1. VHDL: Finite state machine	**due January 18, 2017**
2. VHDL: Pipelining	**due January 25, 2017**
3. Cache	**due February ~~8~~ 10, 2017**

The following deliverables are still being planned; updates will be made as decisions are finalized, and will be summarized in a *Change Log* above.

4. Pipelined processor	**due March 19, 2017**
5. Optimized processor	**due April 11, 2017**
6. Final report	**due April 11, 2017**

The project will begin with a refresher of VHDL concepts necessary for the project, and culminate with the development and testing of a pipelined MIPS processor.

To the first order, all students in a given group will receive the same grade for written deliverable reports and final testing. However, an effort will be made by instructors to identify, and appropriately penalize, any students who fail to adequately participate in the project.

Project Overview

Phase 1

In the first phase, you will complete a series of exercises designed to get up to speed with VHDL. These exercises have been prepared for you with the support of funding from the Engineering Undergraduate Support Fund (EUSF). The first deliverable requires that you design a finite state machine for identifying comments in source code. In the second, you will pipeline the execution of a complex arithmetic operation. In the third, the first group assignment, ~~you will design a cache for a single-ported memory~~ you will design a write-back cache.

Phase 2

In the second phase, you will begin by implementing a pipelined processor (five stages) based on a subset of the MIPS ISA. In order to test your processor, you will also need to write assembly for conversion into machine code (we will provide an assembler). Machine code in our case will be ASCII text used to populate a model of main memory in VHDL.

Next, you will optimize the performance of this processor; the available avenues for optimization will be determined later in the semester. Previously, caching and branch prediction have each been possibilities.

Grading

The project is worth 30% of your final grade. The project is further subdivided:

1. VHDL: Finite state machine	2%
2. VHDL: Pipelining	3%
3. Cache	~~5~~ 10%
4. Pipelined processor	20%
5. Optimized processor	30%
6. Final report	~~40~~ 35%

* *These weights are subject to change.*

All deliverables are due by 11:59 PM on their due date, and must be submitted to MyCourses. Deliverables will be evaluated *as is*. To receive full credit, please provide detailed instructions and ensure that your VHDL is packaged to readily facilitate our evaluation of it using the given testbench infrastructure.

The grading of (1-3) above will be based on a combination of the completeness of the submitted testbench and the passing of tests based on the instructional staff's testbench.

The grading of (4) will be limited to functional testing using a combination of assembly language codes provided to and withheld from student groups. If all test codes execute properly, full marks will be given.

The grading of (5) will, to the first order, be similar to that of (4): given the optimized implementation, if all test codes execute properly, full marks will be given.

The grading of (6) will focus on the organization and clarity of the report, including description of the optimization strategy, resulting design, test procedure, and results of design evaluation.

Resources

A multitude of resources are available on the Internet. Some useful ones include:

1. MiSaSiM, http://users.ece.gatech.edu/~scotty/misasim/. MiSaSiM is a MIPS instruction set simulator, implemented Python. It can simulate all of the instructions that you are expected to implement, and can therefore be used as a golden model to test your functional simulator. Of course, it's a third-party tool: it may have bugs!
2. MIPS Reference Data, including opcodes, instruction formats, and other useful information for the MIPS ISA: http://www-inst.eecs.berkeley.edu/~cs61c/resources/MIPS_Green_Sheet.pdf.
3. ModelSim PE Student Edition, http://www.mentor.com/company/higher_ed/modelsim-student-edition. You can download and install ModelSim (like Quartus, but without synthesis for FPGAs) on your personal computer. Follow the link, and download the software. After installation, you'll be prompted to request a license.