Course Syllabus

ECE 5775 High-Level Digital Design Automation

Fall 2022, Tuesday and Thursday 11:25am-12:40pm, Phillips 403

---

1. Course Information

Lectures:	TuTh 11:25am-12:40pm, 403 Phillips Hall
Website:	http://www.csl.cornell.edu/courses/ece5775
CMS:	https://cmsx.cs.cornell.edu
Ed:	https://edstem.org/us/courses/24225

Instructor:	Zhiru Zhang, zhiruz@cornell.edu
Office Hours:	Thursday 4:30-5:30pm, Online
Staff Email:	ece5775-staff@csl.cornell.edu

Course Texts:

◦ Lecture slides/notes on course website
◦ R. Kastner, J. Matai, and S. Neuendorffer, Parallel Programming for FPGAs, arXiv, 2018.
◦ G. De Micheli, Synthesis and Optimization of Digital Circuits, McGraw-Hill, 1994.

Supplementary Materials:

◦ S. Dasgupta, C.H. Papadimitriou, and U.V. Vazirani, Algorithms, McGraw-Hill, 2007. [link to online draft]
◦ Additional reference papers will be posted as a course reader.

2. Course Description and Objectives

Targeted specialization of functionality in hardware has become arguably the best means to achieving improved compute performance and energy efficiency for a plethora of emerging applications. Unfortunately, it is a very unproductive practice to design and implement special-purpose accelerators using the conventional register transfer level (RTL) methodology. For this reason, both academia and industry are seeing an increasing use of high-level synthesis (HLS) to automatically generate hardware accelerators from software programs.

The course provides an introduction to the hardware accelerator design principles and the modern HLS design methodologies and tools, focusing on FPGA targets. Specific topics include C-based HLS design methods, hardware specialization, scheduling, pipelining, resource sharing, and case studies on deep learning acceleration. This course also discusses the applications of several important optimization techniques in the context of HLS, such as graph algorithms, dynamic programming, and linear programming. In addition, commercial C-to-FPGA tools will be provided to the students to implement real-life image/video processing and machine learning applications on programmable system-on-chips that tightly integrate CPU and FPGA devices.

2.1. Prerequisites

This course assumes the student has a working knowledge of C/C++ and familiarity with basic concepts of digital logic and computer architecture, such as sequential circuits, timing analysis, pipelining, etc. A knowledge of basic algorithms and data structures is preferred. Experiences with RTL design for either ASICs or FPGAs would be helpful, although not required.

2.2. Target Audience and Learning Outcome

This course is open to graduate students and senior undergraduates, who are interested in (1) learning application-/domain-specific hardware acceleration and (2) understanding the capabilities of current HLS tools and design methodologies. Upon completion of this course, students will be able to use HLS tools to design realistic hardware accelerators on FPGAs.

3. Course Organization

This course includes a combination of lectures, paper readings, homework assignments, a midterm exam, and a final project.

3.1. Lectures

The lecture sessions will cover the following topics before the final project begins. Note that these topics are tentative and may be covered in a slightly different order. Please refer to the course website for the detailed up-to-date lecture schedule.

Background
Introduction .... 1 session
Hardware specialization .... 1 session
Algorithm basics .... 2 sessions
High-Level Synthesis
FPGA .... 1 session
C-based synthesis .... 1 session
Front-end compilation .... 2 sessions
Scheduling .... 2 sessions
Resource sharing .... 1 session
Pipelining .... 2 sessions
Advanced Topics
Deep learning acceleration .... 2 sessions
Domain-specific programming .... 1 session

3.2. Participation

Students are expected to attend lectures and actively participate in various technical discussions during the lecture.

3.3. Quizzes

There will be short pop quizzes during most lectures to cover key topics discussed in the current or previous lecture. The overall quiz grade will be determined by the average of all quizzes, *excluding the two lowest scores.*

3.4. Paper Readings

There will be several paper reading sessions where all students are expected to (1) read an academic paper before the lecture, (2) participate in the in-class discussion, and (3) take quiz questions related to the paper. The paper assignment will be announced at least one week in advance.

3.5. Assignments

A total of six assignments will be released in sequence on CMS. Two of them are problem sets designed to help students solidify the understanding of the important concepts covered in lectures. In addition, there are four lab assignments that involve using either C++ or Python programming languages. These labs will help students acquire hands-on experience with the HLS methodology and algorithms.

3.6. Exams

There will be an in-class midterm exam in the third week of October. The exam will be open notes and open books. There is no sit-down final exam.

3.7. Final Project

Students will have approximately five weeks to work in small groups (2-4 students per group, depending on class size) to carry out a class project based on their own interests. Students will propose, refine, and iterate the project plan with the instructor before implementing their ideas. The students will be meeting with the instructor on a weekly basis during regular class hours to track the project progress. Students are expected to present their results before the final week and submit the project report and source code by the final exam date.

4. Course Policy

This section outlines various policies regarding enrollment, grading, assignment submission, academic integrity, and accommodations for students with disabilities.

4.1. Enrollment

This course is open to graduate students and senior undergraduates. Students are expected to take this course for credit. Hence auditors are not advised to attend lectures without the intent to enroll.

4.2. Grading

We will use the following weighting scheme to calculate the final grade:

Class participation – 3%
Quizzes – 6%
Paper Readings – 6%
Midterm exam – 20%
Assignments – 30% = 22% labs + 8% problem sets
Final project – 35% = 25% actual work + 7% report + 3% demo

Please note that a student must at least satisfy the following minimum requirements in order to pass the course: (1) submit at least four assignments; (2) take the midterm exam; (3) complete the final project. If a student fails to meet any of these criteria, then that student will automatically fail the course regardless of the actual numerical grade.

4.3. Assignment Submission

For each assignment, students must submit the solution source code and report to the CMS system by 11:59pm EST on the due date. The report file must be in PDF format.

Late policy – 4% off per assignment per day (e.g., late by three days means a 12% penalty); cannot be late by more than six days.

4.4. Regrade Policy

All regrade requests must be submitted via email to the course staff. The request must state exactly what should be regraded and why. The regrade request has to be received within one week from when the grade in question has been posted.

4.5. Academic Integrity

Each student in this course is expected to abide by the Cornell University Code of Academic Integrity. Any work submitted by a student in this course for academic credit will be the student's own work.

The term "group" in this section refers to yourself if you work alone or to you and your partner in case of a group (team of two) project. The work your group submits is expected to be the result of your group's effort only. The use of a computer in no way modifies the standards of academic integrity expected under the Cornell University Code of Academic Integrity. You are encouraged to study together and to discuss information and concepts covered in lecture with other students. You can give "consulting" help to or receive "consulting" help from such students. However, this cooperation should never involve one group having possession of a copy of all or part of the work done by some other group, including work from previous years. Should copying occur, both the student(s) who copied work from another student and the student(s) who gave material to be copied will automatically receive a zero on the work, and an extra penalty will be assessed, ranging anywhere from a deduction on the final grade to failure of the course and university disciplinary action. Please notice that this implies that at no time are you allowed to grant anyone but your group partner access to your computer files. Be sure to master the use of chmod and umask before starting to work on your projects. During exams, you must do your own work. Communication among students is not permitted during the exams, nor may you compare or borrow notes, copy from others, or collaborate in any way. You are strongly encouraged to read Cornell University's Code of Academic Integrity, available at https://blogs.cornell.edu/provost/files/2015/04/AcademicIntegrityPamphlet2015-p7hd3n.pdf.

4.6. Accommodations for Students with Disabilities

In compliance with the Cornell University policy and equal access laws, the instructor is available to discuss appropriate academic accommodations that may be required for students with disabilities. Requests for academic accommodations are to be made during the first three weeks of the semester, except for unusual circumstances, so arrangements can be made. Students are encouraged to register with Student Disability Services to verify their eligibility for appropriate accommodations.