ECE 2300 Digital Logic and Computer Organization, Fall 2024

Course Syllabus

School of Electrical and Computer Engineering
Cornell University

revision: 2024-10-30-13-27

1. Course Information

Cross-Listings	ENGRD 2300 Digital Logic and Computer Organization
Prereqs	CS 1110 or CS 1112
Instructor	Prof. Christopher Batten, 323 Rhodes Hall, `cbatten@cornell.edu` Office Hours: Friday, 4:30–5:30pm @ 323 Rhodes Hall
Head TAs	Derin Ozturk, Aidan McNay
Graduate TAs	Mahathi Andavolu, Vesal Bakhtazad, Ali Taha, Bolong Tan Klora Wang, Zichao Yue
Undergraduate TAs	Mohammad Al-Labadi, Anjelica Bian, Dyllan Hofflich, Zach Jessup Zarif Karim, Nita Kattimani, Amy Le, Nicole Li, Kevin Rodriguez Paige Shelton, Max Trager, Justin Wong, Steven Yu, Wei Zheng
Lectures	Tue/Thu, 11:40–12:55pm @ 155 Olin Hall
Lab Sessions	Mon, 11:15– 2:15pm @ 238 Phillips Hall Mon, 7:30–10:30pm @ 238 Phillips Hall Tue, 1:25– 4:25pm @ 238 Phillips Hall Wed, 7:30–10:30pm @ 238 Phillips Hall
Optional Discussion Section	Fri, 1:25–2:15pm @ 225 Upson Hall Fri, 2:30–3:20pm @ 225 Upson Hall Fri, 3:35–4:25pm @ 225 Upson Hall
Required Textbook	D. M. Harris and S. L. Harris "Digital Design and Computer Architecture: RISC-V Edition" 1st edition, Morgan Kaufmann, 2021 Available through Canvas via the Cornell Academic Materials Program Available in paperback from Amazon ($95)
Website	`http://www.csl.cornell.edu/courses/ece2300`
Staff Email	`ece2300-staff-l@cornell.edu`

2. Description

The field of computer engineering is at the interface between electrical engineering and computer science and seeks to balance the tension between application requirements and technology constraints. Digital logic and computer organization form the heart of computer engineering: *digital logic* transforms low-level circuits into hardware blocks dedicated to processing, storing, and moving digital data; and *computer organization* transforms these hardware blocks into programmable computing systems capable of executing high-level software. This course aims to provide a comprehensive introduction to the principle and practice of digital logic and computer organization. The course will demystify how computer hardware works and at the same time serve as a foundation for more advanced courses in embedded systems and computer architecture.

The course lectures are structured into four parts. In the first part, students will learn about combinational digital logic (e.g., digital circuits, logic gates, Boolean equations, decoders, priority encoders, multiplexors, demultiplexors, comparators, adders, multipliers, number systems). In the second part, students will learn about sequential digital logic (e.g., latches, flip-flops, finite-state machines, counters, shift registers, memory arrays). In the third part, students will leverage their understanding of digital logic to explore computer processor organization (e.g., instruction set architecture, single-cycle processors, multi-cycle processors, pipelined processors). In the fourth part, students will focus on computer memory organization (e.g., main memory, virtual memory, caches).

The course includes hands-on discussion sections and a series of five laboratory assignments for students to put the principles they have learned into practice. These laboratory assignments will make extensive use of the Verilog hardware description language. Students will use open-source tools to model and simulate hardware using Verilog, before using commercial tools to synthesize their Verilog designs to field-programmable gate arrays (FPGAs) in the lab. Throughout the semester, students will gradually design, implement, test, and evaluate a simple calculator, a music player, and a programmable processor capable of running simple RISC-V assembly programs.

3. Objectives

This course is meant to be a foundational course in computer engineering. The course will prepare students for more advanced coursework in computer engineering (e.g., embedded systems, computer architecture) as well as provide context for more advanced coursework that focuses lower in the computer systems stack (e.g., micro-electronics, digital VLSI) or higher in the computer systems stack (e.g., compilers, operating systems). By the end of this course, students should be able to:

- **describe** a variety of concept and mechanisms in digital logic and computer organization and explain how these concepts and mechanisms interact;
- **apply** this understanding to new hardware design problems by quantitatively assessing a design's propagation delay, cycle time, execution time, and/or area;
- **evaluate** various hardware design alternatives and make a compelling qualitative and/or quantitative argument for why one design is superior; and
- **demonstrate** the ability to implement and verify hardware designs of varying complexity at both the gate- and register-transfer-levels using the Verilog hardware description language.

4. Prerequisites

This course is targeted towards sophomore-level undergraduate students, although it is also appropriate for advanced freshman students and upperclassman. An introductory course on computing is required. Students need to be comfortable using at least one programming language (e.g., Python through CS 1110 or MATLAB through CS 1112). No prior knowledge of the Verilog hardware description language is necessary.

5. Topics

The course includes four parts. The first two parts cover digital logic, while the second two parts cover computer organization. A list of topics for each part is included below. The exact topics covered in the course are subject to change based on student progress and interest.

- **Part 1: Combinational Digital Logic**
  - Topic 1: Digital Circuits
  - Topic 2: CombinationalLogic Gates
  - Topic 3: Boolean Equations
  - Topic 4: Combinational Building Blocks
  - Topic 5: Number Systems
- **Part 2: Sequential Digital Logic**
  - Topic 6: Sequential Logic Gates
  - Topic 7: Finite-State Machines
  - Topic 8: Sequential Building Blocks
- **Part 3: Computer Processor Organization**
  - Topic 9: Instruction Set Architecture
  - Topic 10: Single-Cycle Processor
  - Topic 11: Multi-Cycle Processor
  - Topic 12: Pipelined Processor
- **Part 4: Computer Memory Organization**
  - Topic 13: Main Memory
  - Topic 14: Virtual Memory
  - Topic 15: Caches

6. Required Textbook

The required textbook for the course is *"Digital Design and Computer Architecture, RISC-V Edition,"* by D. M. Harris and S. L. Harris (Morgan Kaufmann, 2021). There are three different editions of this book for three different instruction set architectures: MIPS, ARM, RISC-V. It is critical that students use the RISC-V edition. All students will have access to the textbook online through Canvas via the Cornell Academic Materials Program.

7. Format and Procedures

This course includes a combination of lectures, quizzes, discussion sections, readings, practice problems, lab assignments, and exams.

- **Lectures –** Lectures will be from 11:40am to 12:55pm every Tuesday and Thursday in 155 Olin Hall excluding the following academic holidays: Indigenous People's Day (10/15) and Thanksgiving (11/28). We will start promptly at 11:40am so please arrive on time. Students are expected to attend all lectures, be attentive during lecture, and participate in class discussion. Please turn off all cellular phones during class. Use of cellular phones and laptops during lecture is not allowed (see Section 10.B). Tablets are allowed.
- **Quizzes –** There will be a short quiz at the beginning of some lectures. The quiz should take about five minutes, and will cover some of the key topics discussed in the previous lecture. Quizzes may or may not be announced ahead of time, and there are no make-up quizzes. The lowest quiz score is dropped. Students should prepare for a potential quiz by simply reviewing the material from the previous lecture before coming to class. Solutions to quizzes will be available online soon after the quiz is given for formative self-assessment.
- **Discussion Section –** Optional discussion sections will be on Fridays at 1:15–2:15pm, 2:15–3:15pm, and 3:15–4:15pm in 225 Upson Hall. These discussion sections will be relatively informal, with the primary focus being on facilitating student's ability to complete the lab assignments and on reviewing material from lecture using problem-based learning. Students that wish to attend a discussion section must reserve a seat through Canvas.
- **Readings –** Students are expected to complete all of the required reading according to the schedule on the course website. Students are responsible for all material covered in lecture and in the assigned readings.
- **Practice Problems –** The course will include practice problems distributed throughout the semester to help students put the concepts learned in lecture and reading into practice. Solutions will not be provided for the practice problems. Students should work either individually or collaboratively on the problem sets and then discuss their solutions with course instructors during office/lab hours.
- **Lab Assignments –** The course includes five lab assignments. Students are expected to work with a partner on all lab assignments. The first four lab assignments have two parts: a *simulation* part and an *FPGA* part. The simulation part involves students writing and testing their designs in Verilog using open-source simulators and is meant to be completed using the `ecelinux` servers. During simulation weeks, students are expected to attend their assigned lab session to work on their code. However, students are expected to spend significant time outside of their lab session to complete the simulation part. The Verilog code must be submitted via GitHub on Thursdays at 11:59pm. The FPGA part involves students synthesizing their designs using commercial tools and is meant to be completed during their assigned lab session. Students will complete a variety of tasks including collecting evaluation data assessed via a check-off sheet during their lab session. **Students must attend their assigned lab session, and their check-off sheet must be completed and turned in by the end of their lab session.** Students will not be allowed to complete check-off tasks after the end of the lab session, at a session other than their assigned lab session, nor during office hours. Students should not schedule make-up exams for other classes during their lab session since this means they will not be able to attend their lab session. Students will then prepare a short lab report which must be submitted in PDF format via the online Canvas assignment submission system on Thursdays 11:59pm. The final lab will only consist of an FPGA part.
- **Exams –** The course includes two prelim exams and a cumulative final exam. The exams assess student understanding of the material presented in lecture and assigned readings. If students have a scheduling conflict with the exam, they must let the instructor know as soon as possi-

ble, but no later than one week before the prelim or final exam. Usually the only acceptable scheduling conflict is due to another exam at the same time. Exceptions can only be made for a family or medical emergency. Graded exams and the exam solutions are only available for review under the supervision of a course instructor. You may not remove your graded exam, nor may you remove the exam solutions.

8. Assignment and Exam Schedule

The current schedule is on the course website. All lab assignments are due on Thursdays at 11:59pm. Changes to this schedule will be posted as announcements via Ed.

Thu	Sep	12	Lab 1.1: Display (simulation part)
Thu	Sep	19	Lab 1.2: Display (FPGA part)
Thu	Sep	26	Lab 2.1: Calculator (simulation part)
Thu	Oct	3	Lab 2.2: Calculator (FPGA part)
Tue	Oct	8	Prelim from 7:30-9:00pm (101 & 219 Phillips Hall)
Thu	Oct	17	Lab 3.1: Music Player (simulation part)
Thu	Oct	24	Lab 3.2: Music Player (FPGA part)
Thu	Oct	32	Lab 4 Milestone #1
Tue	Nov	5	Prelim from 7:30-9:00pm (155 Olin Hall)
Thu	Nov	7	Lab 4 Milestone #2
Thu	Nov	14	Lab 4.1: Processor (simulation part)
Thu	Nov	21	Lab 4.2: Processor (FPGA part)
Thu	Dec	5	Lab 5: Assembly (FPGA part)
	TBD		Final Exam (location TBD)

9. Grading Scheme

This course will be adopting a philosophy of "grading for equity" where grading is exclusively used to assess mastery of the material covered in the course as opposed to rewarding effort and/or incentivizing specific behaviors. To this end, each part or criteria of every assignment is graded on a five-point scale without any curve according to the following rubric.

- **5 (Mastery):** Submitted work demonstrates no misunderstanding (there may be small mistakes which do not indicate a misunderstanding) or there may be a very small misunderstanding that is vastly outweighed by the demonstrated understanding. Student has mastered learning objectives; can independently apply course material in later courses and/or career.
- **4 (Accomplished):** Submitted work demonstrates more understanding than misunderstanding. Student has accomplished learning objectives; would probably need some additional learning/help to apply course material in later courses and/or career.
- **3 (Progressing):** Submitted work demonstrates more misunderstanding than understanding. Student is still progressing towards learning objectives; would need additional study and practice to apply course material in later courses and/or career.
- **2 (Beginning):** Submitted work is significantly lacking in some way. Student is just beginning towards learning objectives; would need significant additional study and practice before being able to apply course material in later courses and/or career.

- **1 (Minimal Understanding)**

A score of 5 corresponds to an A, 4 corresponds to a B, 3 corresponds to a C, and so on. A score of 5.25 is reserved for when the submitted work is perfect with absolutely no mistakes or is exceptional in some other way.

Total scores for an assignment are a weighted average of the scores for each part or criteria. Parts or criteria are usually structured to assess a student's understanding according to four kinds of knowledge: basic recall of previously seen concepts, applying concepts in new situations, qualitatively and quantitatively evaluating alternatives, and creatively implementing new designs; these are ordered in increasing sophistication and thus increasing weight. In almost all cases, scores are awarded for demonstrating understanding and not for effort. Detailed rubrics for all quizzes, lab assignments, and exams are provided once the assignment has been graded to enable students to easily see how the score was awarded.

The final grade is calculated using a weighted average of all assignments. All quiz grades are averaged to form a single total. Students can drop their lowest quiz score.

Quizzes	5%	(students can drop lowest score)
Lab 1	4%	(part 1 and 2 weighted equally)
Lab 2	8%	(part 1 and 2 weighted equally)
Lab 3	8%	(part 1 and 2 weighted equally)
Lab 4	12%	(part 1 and 2 weighted equally)
Lab 5	4%	
Prelim 1	17%	
Prelim 2	17%	
Final Exam	25%	

We will drop a student's lowest quiz grade. This means if a student has to miss one quiz for any reason (including for a family or medical emergency) there is no need to notify the course instructor. If a student has to miss more than one quiz due to a family or medical emergency then the student should notify the instructor in advance to discuss an alternative.

Note that the exams account for over half of a student's final grade. The exams in this course are very challenging. Successful students begin preparing for the exams far in advance by carefully reviewing the assigned readings, independently developing study problems, and participating in critical study groups.

To pass the course, a student must at a bare minimum satisfy the following requirements: (1) submit six out of the nine laboratory assignment parts; (2) take both prelim exams; and (3) take the final exam. **If a student does not satisfy these criteria then that student may fail the course regardless of the student's numerical grade.** The instructor reserves the right to award a D letter grade for students who barely satisfy this criteria but are clearly making no real effort to engage in the course and their own learning.

10. Policies

This section outlines various policies concerning auditors, usage of cellular phones and laptops in lecture, turning in assignments late, regrading assignments, collaboration, copyright, and accommodations for students with disabilities.

10.A Auditor Policy

Casual listeners that attend lecture but do not enroll as auditors are not allowed; you must enroll officially as an auditor. *If you would like to audit the course please talk to the instructor first!* Usually we wait until the second week of classes before allowing auditors to enroll, to ensure there is sufficient capacity in the lecture room. The requirements for auditors are: (1) attend most of the lectures; (2) complete most of the in-class quizzes; and (3) perform reasonably well on these quizzes. If you do not plan on attending the lectures, then please do not audit the course. Please note that students are not allowed to audit the course and then take it for credit in a later year unless there is some kind of truly exceptional circumstance.

10.B Cellular Phones and Laptops in Lecture Policy

Students are prohibited from using cellular phones and laptops in lecture unless they receive explicit permission from the instructor. It is not practical to take notes with a laptop for this course. Students will need to write on the handouts, quickly draw timing diagrams, and sketch gate- or block-level diagrams during lecture. The distraction caused by a few students using (or misusing) laptops during lecture far outweighs any benefit. Tablets are allowed as long as they are kept flat and used exclusively for note taking. If you feel that you have a strong case for using a laptop during lecture then please speak with the instructor.

10.C Late Assignment Policy

Lab assignment code must be submitted electronically via GitHub. Lab reports must be submitted electronically in PDF format. **No other formats will be accepted!** Assignments must be submitted by 11:59pm on the due date unless otherwise specified. No late submissions will be accepted and no extensions will be granted except for a family or medical emergency. The instructors must be notified of this emergency in advance if at all possible. You can continue to push your code to GitHub and resubmit your report to Canvas as many times as you would like up until the deadline, so please feel free to upload early and often. **If you submit an assignment even one minute past the deadline, then the assignment will be marked as late and not graded.** We simply cannot accept late work given the tight timeline of the course. If you do not finish the code for the simulation part of an assignment, push what you have to GitHub and you can continue working on it after the deadline to: (1) ensure you have working code for the FPGA part; and (2) earn some points back for the simulation part through the revision process.

10.D Regrade Policy

Addition errors in the total score are always applicable for regrades. Regrades concerning the actual solution should be rare and are only permitted when there is a significant error. Please only make regrade requests when the case is strong and a significant number of points are at stake. Regrade requests should be submitted online via a private post on Ed within one week of when an assignment is returned to the student or within one week of when an exam is reviewed with the class. You must provide a justification for the regrade request.

10.E Collaboration and Artificial Intelligence Policy

The work you submit for the lab assignments is expected to accurately demonstrate your understanding of the material. The use of a computer in no way modifies the standards of academic integrity expected under the University Code. You are encouraged to discuss information and concepts cov-

ered in lecture and relevant to the lab assignments with other students. You can give "consulting" help to or receive "consulting" help from other students about the lab assignments. Students can also freely discuss basic computing skills or the course infrastructure. **However, this permissible cooperation should never involve one student (or group) having possession of or observing in detail a copy of all or part of work done by someone else, in the form of an email, an email attachment file, a flash drive, or on a computer screen.** Students are not allowed to seek consulting help from online forums outside of Cornell University. **If a student receives consulting help from anyone outside of the course staff, then the student must acknowledge this help on the submitted assignment.**

As an exception to the outside consulting policy described above, **students *are* allowed to use artificial intelligence (AI) systems (e.g., OpenAI ChatGPT, Anthropic Claude, Microsoft Copilot) in absolutely any way they want in the course as long as all submitted material still represents the students' understanding.** Students are free to use AI to explain lecture concepts, create practice problems, explain problem solutions, write Verilog code, debug Verilog code, analyze Verilog compile-time errors, brainstorm test cases, and/or edit lab reports. **The only condition is that students *must* acknowledge how they used AI in any submitted work.** Students must include an AI acknowledgment as a comment at the top of any source code for which AI was used in any way. Students must include an AI acknowledgment at the end of lab report for which AI was used in any way. The AI acknowledgment should clearly specify which AI was used and how it was used. **Using AI without acknowledgment will be considered and academic integrity violation.** Students are responsible for all of their submitted work and that work must represent their understanding even with an AI acknowledgment. The instructor reserves the right to use a short oral inquiry with students to verify that they understand anything they submit as their own work.

During in-class paper quizzes and examinations, you must do your own work. Talking or discussion is not permitted during the in-class paper quizzes and examinations, nor may you compare papers, copy from others, or collaborate in any way. Students must not discuss a quiz/exam's contents with other students who have not taken the quiz/exam. If prior to taking it, you are inadvertently exposed to material in an quiz/exam (by whatever means) you must immediately inform an instructor.

Should a violation of the code of academic integrity occur, then a primary hearing will be held. See `https://deanoffaculty.cornell.edu/faculty-and-academic-affairs/academic-integrity` for more information about academic integrity proceedings.

Examples of acceptable collaboration:

- Ben is struggling to complete a lab assignment which requires implementing a carry select adder. He talks with Alice and Cathy and learns that all three students are really struggling. So the three students get together for a brainstorming session. They review the lecture and reading materials and then sketch on a whiteboard some ideas on how to implement a carry select adder. They might also sketch out some code snippets to try and understand the best way to implement the adder. Then each student independently writes the Verilog code for the assignment and includes an acknowledgment of the help they received from the other students. At no time do the students actually share code.
- Alice and Amy are having trouble figuring out difficult test cases for their single-cycle processor. They post on Ed to see if anyone has some general ideas for tricky corner cases. Ben and Bob figured out an interesting test case that ensures their processor correctly handles a complicated sequence of instructions, so Ben and Bob post a qualitative description of this test case. Alice and Amy independently write the code for this test case and then include an acknowledgment of the help they received from the other group. At no time do the groups actually share test code.

- Chad always has Microsoft Copilot enabled in VS Code to help him write code. When he is implementing a one-bit full adder in Verilog, Microsoft Copilot fills in the entire implementation for him. Chad checks the result to confirm it looks right and is basically what he would have written by hand. He adds an AI acknowledgment at the top of the corresponding source code file. He then submits this full adder as part of the code for his lab assignment. There is no issue because because: (1) the submitted work represents Chad's understanding; and (2) Chad included an appropriate AI acknowledgment.
- Doug and Ellen have written a first draft of his lab report but the writing is unpolished. Doug asks ChatGPT for suggestions on how to improve the writing quality, and he incorporates this feedback in his final lab report submission. Doug adds an AI acknowledgment at the end of his lab report. There is no issue because: (1) the submitted work represents Doug and Ellen's understanding; and (2) Doug and Ellen included an appropriate AI acknowledgment.

Examples of unacceptable collaboration:

- Anna cannot make today's lecture since she has an extracurricular commitment. Anna asks Bart to complete two in-class paper quizzes and to put Anna's name on the second quiz. This misrepresents Anna's understanding and is not allowed.
- Ben is struggling to complete a lab assignment which requires implementing a carry-select adder. He talks with Alice and Cathy and learns that all three students are really struggling. So the three students get together for a joint coding session. They work at the same workstation to write the code together. *The three students share and copy each other's code often in order to finish the assignment.* Each student submits the final code independently. Each student acknowledges the help he or she received from the other students, but it doesn't matter since they explicitly shared code.
- Alice and Amy are having difficulty figuring out difficult test cases for their single-cycle processor. They post on Ed to see if anyone has some general ideas for tricky corner cases. Ben and Bob figured out an interesting test case that ensures their processor correctly handles a complicated sequence of instructions, so *Alice and Amy send their test code to Ben and Bob via email.* Alice and Amy modify this test code and then include it in their submission. Alice and Amy include an acknowledgment of the help they received from the other group, but it doesn't matter since they explicitly shared code.
- Chad always has Microsoft Copilot enabled in VS Code to help him write code. When he is implementing a one-bit full adder in Verilog, Microsoft Copilot fills in the entire implementation for him. Chad checks the result to confirm it looks right and is basically what he would have written by hand. He then submits this full adder as part of the code for his lab assignment. This is an academic integrity violation because Chat did not include an AI acknowledgment.
- Doug and Ellen waited until the last minute to write their lab report. They create a quick bulleted list of what they want their report to say and then ask ChatGPT to flesh out a two-page report. They then quickly skim over the report. They add an AI acknowledgment at the end of the lab report, but it doesn't matter since the lab report does not represent Doug and Ellen's understanding of the material.

Notice that **the key is that students should not share the actual code with each other unless expressly permitted by the course instructors; and that all submitted work must represent a student's understanding.**

10.F Copyright Policy

All course materials produced by the course instructor (including all handouts, tutorials, quizzes, exams, videos, scripts, and code) are copyright of the course instructor unless otherwise noted. Download and use of these materials are permitted for individual educational non-commercial purposes only. Redistribution either in part or in whole via both commercial (e.g., Course Hero) or non-commercial (e.g., public website) requires written permission of the copyright holder.

10.G 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. All communications concerning accommodations should be done privately with the instructor either via email or in-person. 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 must register with Student Disability Services to verify their eligibility for appropriate accommodations. Note that students cannot simply rely on Student Disability Services to email the instructor without actually speaking to the instructor in person. Students with a disability must speak with the instructor in person during the first three weeks to discuss their accommodations.

11. Online and Computing Resources

We will be making use of a variety of online websites and computing resources.

- **Public Course Website –** `http://www.csl.cornell.edu/courses/ece2300` is the main public course website which will also have course details, updated schedule, reading assignments, and most handouts. We intend for all course content to always be available on Canvas. The public course website is just for public access to some of this content.
- **Canvas Course Site –** We will be using Canvas to manage course content, assignment submission, and grade distribution.
- **Ed –** We will be using Ed for all announcements and discussion on course content and lab assignments. The course staff is notified whenever anyone posts on the forum and will respond quickly. Using the forum allows other students to contribute to the discussion and to see the answers. Use common sense when posting questions such that you do not reveal solutions. Please prefer posting to Ed as opposed to directly emailing the course staff unless you need to discuss a personal issue.
- `ecelinux` **Servers –** The ECE department has a cluster of Linux-based servers which we will be using for the lab assignments. You can access the `ecelinux` servers remotely using PowerShell, Mac Terminal, or VS Code. More information about accessing the ECE computing resources is available on the Canvas course site.
- **GitHub –** GitHub is an online Git repository hosting service. We will be using the commercial GitHub service to distribute code and as a mechanism for student collaboration on the lab assignments. Students will also use GitHub for submitting the code for their lab assignments. Students are expected to become familiar with the Git version control system. Note that we are not using the Cornell hosted version of GitHub as in some other courses; we are using `github.com`.