ECE 2300 Digital Logic and Computer Organization, Fall 2025

Course Syllabus

School of Electrical and Computer Engineering
Cornell University

revision: 2025-09-06-11-12

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 Office Hours: Thursday, 4:30–5:30pm @ 323 Rhodes Hall
Head TA	Niklas Schmelzle
Graduate TAs	Vesal Bakhtazad, Han Mo, Nicholas Papapanou, Htoo Wai Htet, Xingze Xu
Undergraduate TAs	Mohammad Al-Labadi, Stephen Barlett, Madison Costello, Yaqi Gao Jamayne Gyimah-Danquah, Rohan Kalluraya, Rachel Lee Zephan Sanghani, Md Shad, Cynthia Shao, Anthony Song, Simeon Turner Irwin Wang, Paige Yun
Lectures	Tue/Thu, 11:40–12:55pm @ 155 Olin Hall
Lab Sections	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
Discussion Sections	Fri 1:25– 2:15pm @ 225 Upson Hall Fri 2:30– 3:20pm @ 225 Upson Hall Fri 3:35– 4:25pm @ 225 Upson Hall
Office Hours	Mon 5:30– 7:00pm @ 203 Phillips Hall Tue 5:30– 7:00pm @ 203 Phillips Hall Tue 7:30– 9:00pm @ 203 Phillips Hall Wed 5:30– 7:00pm @ 203 Phillips Hall (some exceptions, see Canvas) Thu 5:30– 7:00pm @ 203 Phillips Hall Thu 7:30– 9:00pm @ 203 Phillips Hall (some exceptions, see Canvas)
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 ($70) `https://www.amazon.com/dp/0128200642`
Website	`http://www.csl.cornell.edu/courses/ece2300`
Communication	All communication should use Ed Discussion unless a student needs to discuss a personal matter in which case they should email the instructor. Students should never directly email any teaching assistant.

2. Description

The field of computer engineering is at the interface between computer science and electrical engineering, where engineers design systems that must meet software application requirements within hardware 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, while *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 principles 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 two parts. In the first part of the course, students will learn about digital logic (e.g., digital circuits, combinational logic, Boolean algebra, combinational building blocks, number systems, sequential logic, finite-state machines, and sequential building blocks). In the second part of the course, students will then apply their understanding of digital logic to explore computer organization (e.g., instruction set architecture, single-cycle processors, multi-cycle processors, pipelined processors, caches, and integrating processors with caches).

The course includes hands-on discussion sections and a series of four 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 design, implement, test, and evaluate a two-function calculator, music player, and 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** digital logic concepts across multiple levels of abstraction including digital circuits, combinational logic, Boolean algebra, combinational building blocks, number systems, sequential logic, finite-state machines, and sequential building blocks; and
- **describe** computer organization concepts across multiple levels of abstraction including instruction set architecture, single-cycle processors, multi-cycle processors, pipelined processors, caches, and integrating processors with caches; and
- **apply** this understanding to new hardware design problems including quantitatively analyzing a design's propagation delay, setup and hold time constraints, 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 to the other approaches; and
- **demonstrate** the ability to design, implement, test, and debug hardware designs of varying complexity at both the gate- and register-transfer-levels using the Verilog hardware description language and open-source simulation tools; and

- **demonstrate** the ability to integrate, synthesize, analyze, and configure these designs using a commercial tools and an FPGA in the lab; and
- **write** concise yet comprehensive technical reports that describe designs implemented at the gate- and register-transfer-level, explain the testing strategy used to verify functionality, and evaluate the designs to determine the superior approach.

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 Python (i.e., through CS 1110 or CS 1112). Students must have a strong understanding of variables, expressions, conditionals, iteration, lists/arrays, and functions in addition to the practical aspects of developing, testing, and debugging software. No prior knowledge of the Verilog hardware description language is necessary.

5. Topics

The course includes two parts. The first part covers digital logic, while the second part covers 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: Digital Logic**
  - Topic 1: Digital Circuits
  - Topic 2: Combinational Logic
  - Topic 3: Boolean Algebra
  - Topic 4: Combinational Building Blocks
  - Topic 5: Number Systems
  - Topic 6: Sequential Logic
  - Topic 7: Finite-State Machines
  - Topic 8: Sequential Building Blocks
- **Part 2: Computer Organization**
  - Topic 9: Instruction Set Architecture
  - Topic 10: Single-Cycle Processor
  - Topic 11: Multi-Cycle Processor
  - Topic 12: Pipelined Processor
  - Topic 13: Caches
  - Topic 14: Integrating Processors and 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 (10/14) and Thanksgiving (11/27). 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. 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. The lowest two quiz scores will be dropped. This means students can miss up to two quizzes without penalty due to official student disability service accommodations, official varsity athletic accomodations, religious observance accomodations, serious illness, medical emergencies, family emergencies, and/or any other reason. Students do not need to notify the instructor of these excused or unexcused absences. No additional quiz scores will be dropped except in extremely unusual circumstances.
- **Discussion Sections –** Discussion sections will be on Fridays at 1:25–2:15pm, 2:30–3:20pm, and 3:35–4:25pm 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 are strongly encouraged to attend all discussion sections. However, if student has an unavoidable class conflict with the discussion section they should submit a time conflict permission form so they can still enroll in the course.
- **Readings –** Students are expected to complete all of the required reading according to the schedule on the course website. Although students are responsible for all material covered in lecture and in the assigned readings, they should prioritize the material covered in lecture when there is any discrepancy. Students can complete the readings either before or after the topics are covered in lecture.
- **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 majority of the practice problems. However, videos illustrating how to solve a subset of the practice problems will be released. Students should work either individually or collaboratively on the practice problems and then discuss their solutions with course instructors during office hours. Note that the practice problems are not practice exams, and indeed no practice exams will be provided. Some practice problems are simpler than what might be included on a exam, while other practice problems are more complex than what might be included on a exam. The goal is not to practice for an exam, but instead to use practice problems to deeply understand the course material (which is then assessed on the exams).
- **Lab Assignments –** The course includes four lab assignments. For the first lab assignment, students will work individually on the code and with a randomly assigned partner for the in-

lab portion and the lab report. Students will work with a self-selected partner for the remaining three lab assignments. Lab partners must be in the same lab section. Students should use the first three weeks of the semester to choose their partner carefully based on background expertise, work style, and course goals. Students can switch lab partners but only at the start of a new lab assignment. The lab assignments have many parts. *Simulation parts* involve students writing and testing their designs in Verilog using open-source simulators and is meant to be completed using the `ecelinux` servers. During simulation lab weeks, students are expected to attend their assigned lab section to work on their code. However, students are also expected to spend significant time outside of their lab section to complete simulation parts. The Verilog code must be submitted via GitHub on Thursdays at 11:59pm. *FPGA parts* involve students synthesizing their designs using commercial tools and then configuring an FPGA with their design. The FPGA parts must be completed during their assigned lab section. Students will complete a variety of tasks (including collecting evaluation data) assessed via a check-off sheet during their lab section. **Students must attend their assigned lab section, and their check-off sheet must be completed and turned in by the end of their lab section.** Students will not be allowed to complete check-off tasks after the end of the lab section, at a section other than their assigned lab section, nor during office hours. Students should not schedule make-up exams for other classes during their lab section since this means they will not be able to attend their lab section. *Report parts* involve students writing a short lab report which must be submitted in PDF format via the online Canvas assignment submission system three days after their assigned lab section.

- **Prelim and Final Exams –** The course includes two prelim exams (90 minutes each) and a cumulative final exam (180 minutes). The exams assess student understanding of the material presented in lecture and assigned readings. This includes implementing hardware using syntactically and semantically correct Verilog. Students should plan on bringing a four-function or scientific calculator for use on the exams. These are closed book, closed notes exams. No cheat sheets will be allowed. If students have a scheduling conflict with the exam, they must let the instructor know as soon as possible, 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 serious illness, medical emergency, or family 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.

- **Verilog Exam –** The course includes one in-person 50-minute Verilog coding exam in 225 Upson Hall during each students' regularly scheduled discussion section near the end of the semester. Students must attend their assigned discussion section to take the exam. Students will only be allowed to use the workstations. Students will not be allowed to use their own computers. This is a closed book, closed notes exam. No cheat sheets will be allowed. Students will be provided information about one or more simple designs they must implement and test in Verilog using VS Code and Linux similar in spirit to the lab assignments. Students will not be allowed to access the internet or use any other program other than VS Code.

8. Assignment and Exam Schedule

The current schedule is on the course website. The code for most lab assignments is due on Thursdays. Lab reports are due three days after the corresponding lab FPGA part. Assignments must be submitted by 11:59pm on the due date unless otherwise specified. Changes to this schedule will be posted as announcements via Ed.

Thu	Sep	11	Lab 1, Part A
Thu	Sep	18	Lab 1, Part B
Thu	Sep	25	Lab 2, Part A
Thu	Oct	2	Lab 2, Part B
Tue	Oct	7	Prelim Exam 1 from 7:30-9:00pm (110 & B14 Hollister Hall)
Thu	Oct	16	Lab 3, Part A
Thu	Oct	23	Lab 3, Part B
Tue	Nov	4	Prelim Exam 2 from 7:30-9:00pm (101 & 219 Phillips Hall)
Thu	Nov	6	Lab 4, Part A
Thu	Nov	13	Lab 4, Part B
Fri	Nov	21	Verilog Exam (during discussion section)
Tue	Nov	25	Lab 4, Part C
Thu	Dec	4	Lab 4, Part D
	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.

Quizzes	5%
Lab 1	4%
Lab 2	6%
Lab 3	9%
Lab 4	15%
Prelim Exam 1	16%
Prelim Exam 2	16%
Verilog Exam	5%
Final Exam	24%

All quiz grades are averaged to form a single total. We will drop a student's two lowest quiz grades. This means students can miss up to two quizzes without penalty due to official student disability service accommodations, official varsity athletic accomodations, religious observance accomodations, serious illness, medical emergencies, family emergencies, and/or any other reason. Students do not need to notify the instructor of these excused or unexcused absences. No additional quiz scores will be dropped except in extremely unusual circumstances.

Note that the exams account for 61% 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, working through practice problems, independently developing study problems, participating in critical study groups, and writing their own Verilog code from scratch.

To pass the course, a student must at a bare minimum take all four exams. **If a student does not take all four exams 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, artificial intelligence, 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.

10.B Cellular Phones and Laptops in Lecture Policy

Students are prohibited from using cellular phones and/or 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) cell phones and/or 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 via Canvas. **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 serious illness, family emergency, 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. Please plan your time accordingly!** If you do not finish the code for the simulation part of an assignment, push what you have to GitHub by the deadline. You can continue working on it after the deadline to: (1) ensure you have working code for the FPGA parts; and (2) earn some points back through the revision process. If you do not finish the lab report, then upload what you have finished by the deadline.

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 either through Gradescope or 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 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 covered 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 sources 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 using the post-lab survey.**

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 the instructor.

Violating the collaboration policy will be considered a violation of the code of academic integrity, and 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.

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 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 send their test code to Alice and Amy 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.

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 Artificial Intelligence (AI) Policy

Students are allowed to use artificial intelligence (AI) systems (e.g., OpenAI ChatGPT, Anthropic Claude, Google Gemini, Microsoft Copilot) in this course as long as this usage adheres to an *AI as TA* policy. This means students can ask AI anything they would ask a TA. Students can ask AI to explain concepts from lecture, to brainstorm design or test case ideas, to provide help debugging their code, to provide a qualitative assessment of their code quality, and to provide advice on how to improve their lab report. **If a student does use AI, then the student must acknowledge this help on the submitted assignment using the post-lab survey.** They may not ask AI anything they would not ask a TA. Students cannot ask AI to write code, write test cases, fix their code, write comments, or write portions of the lab report.

Students must not enable AI coding assistants within VS Code (e.g., Microsoft Copilot) nor use AI-enabled integrated development environments (e.g., Cursor) since these tools will automatically write code for the student which violates the AI as TA policy.

Violating the AI policy will be considered a violation of the code of academic integrity, and 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 AI usage:

- Ajay is struggling to understand the connection between Karnaugh maps and Boolean algebra, so he asks Google Gemini to first explain each of these concepts in isolation. Then he asks Gemini to give him a few practice problems on each concept. He prepares solutions and asks Gemini to assess his answers. Finally, he asks Gemini to discuss the connections between these two concepts. There is no issue because it is always fine to work with TAs to understand lecture concepts.
- Frank is working through the practice problems. He is planning to go to office hours to discuss his solutions with the TAs, but he is just so excited he cannot wait. So he uploads the problem and a scanned copy of his solutions into OpenAI ChatGPT and asks ChatGPT for feedback. ChatGPT identifies a small mistake, and Frank asks follow up questions to understand where he went wrong. There is no issue because it is always fine to work with TAs to assess student solutions to practice problems.
- Chad implemented a multiplier in Verilog from scratch, but he is struggling to figure out what kind of test cases to write to verify his design. Chad asks Anthropic Claude for some ideas on different kinds of tests to use to verify corner cases in his design. Claude recommends testing small numbers, large numbers, and specifically mentions Chad might want to consider testing overflow. Chad goes ahead and creates three different test cases for each of these ideas and also adds a fourth test case to test multiplying by zero. Chad includes an AI acknowledgment. There is no issue because it is always fine to brainstorm a testing strategy at a high level with a TA.
- Doug and Ellen have written a first draft of his lab report but the writing is unpolished. Doug asks OpenAI ChatGPT for suggestions on how to improve the writing quality, and he incorpo-

rates this feedback in his final lab report submission. Doug includes an AI acknowledgment. There is no issue because it is always fine to have a TA provide feedback on a draft of your lab report.

Examples of unacceptable AI usage:

- Chad always has Microsoft Copilot enabled in VS Code to help him write code. When he is implementing a multiplier 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 multiplier as part of the code for his lab assignment. Chad includes an AI acknowledgment which specifies he used AI for coding help, but it doesn't matter. This is an academic integrity violation because Chad would never ask a TA to write the code for him.
- 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 OpenAI ChatGPT to flesh out a two-page report. They then quickly skim over the report. They add an AI acknowledgment, but it doesn't matter. This is an academic integrity violation because Doug and Ellen would never give a TA a bulleted list and ask the TA to write their lab report for them.

Notice that **the key is that students should only ask AI something they would ask a TA.** Following the AI as TA policy in the context of the lab assignments will ensure students are prepared to demonstrate their understanding of Verilog coding on both prelim exams, the final exam, and the Verilog exam.

10.G 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.

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

10.H 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 Discussion –** We will be using Ed Discussion for all announcements and discussion on course content and lab assignments. The course staff is notified whenever anyone posts on Ed Discussion and will respond quickly. Students should almost always prefer a public post. Use common sense when posting questions such that you do not reveal solutions. Public posts on Ed Discussion allows other students to contribute to the discussion and to see the answers. Please post on Ed Discussion as opposed to directly emailing the course staff unless you need to discuss a personal issue. Students should never have a need to directly email the TAs.
- `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, VS Code, and MS Remote Desktop. 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`.