ES 4: Introduction to Digital Logic Circuits

Steven Bell
Fall 2024
T/Th 1:30pm, Barnum LL08

Welcome to ES 4!

Any sufficiently advanced technology is indistinguishible from magic. – Arthur C. Clarke

There are few technologies for which Clarke's statement is more true than microprocessors. Today you can find a microprocessor in almost anything electronic, and yet even those of us who can program them often have very little understanding of their inner workings. In ES 4, we will study microprocessors — and digital electronics more generally — from the ground up. You may have learned in other courses how computers work at the physical level (voltages and currents) and at the application level (C++ and other languages); the objective of this course is to connect the dots between the two.

To do this, we begin with the basics of manipulating zeros and ones with circuitry, and start building *combinational* circuits, which produce outputs based on some mathematical combination of their inputs. Then we make a major turn and explore *sequential* circuits, which can store information or step through a sequence of states. With these building blocks in hand, we will take a brief look at the ARM instruction set (which powers your cell phone and a few billion other devices), and learn how to build circuits which can actually interpret and execute software instructions.

When we're done, you'll have the skills necessary to design complex digital systems to search through genomes, analyze and manipulate network traffic, mine bitcoin, or play retro video games. Additionally, you'll be able to actually implement and run these system on reconfigurable circuits known as FPGAs. In short, you'll be able to work magic.

After successfully completing this course you will be able to:

- Begin with a specification for a digital system and implement it in hardware, by working through the following steps:
    1. Decompose an imperfect or incomplete specification into a set of well-defined digital modules.
    2. Describe the modules as collections of state machines, memories, arithmetic circuits, and other digital building blocks.
    3. Implement these building blocks with combinatorial and sequential logic constructs.
    4. Construct combinatorial and sequential logic elements using using logic gates, multiplexers, and flip-flops.
- Construct digital circuits using a breadboard and discrete logic ICs.
- Use modern digital development tools including HDL synthesis tools and logic analyzers to implement and debug FPGA designs.
- Describe how a microprocessor works in terms of intermediate digital building blocks. That is, you should be able to explain to someone halfway through the course how a pile of logic gates can read a sequence of instructions and operate on a block of data to produce results.

How will this course help you succeed?

With the skills from this course, you'll be able to design computer systems which can acheive a hundred or thousand times more performance than running code on a traditional CPU. Seriously.

But even if you never design digital hardware in your career, a solid understanding of how hardware works will enable you to write better code — sometimes 10× faster on the same hardware. Moreover, specialized computer hardware is everywhere, from tensor processors in datacenters to neural compute engines in cell phones. These special processors are especially quirky, and only engineers who understand the hardware will be able to wring out every last drop of performance.

And "performance" isn't just speed: power efficiency is incredibly important in wearable and some IoT applications, and this too requires a solid knowledge of digital hardware, even if you're "just" writing code.

Where should you look for information?

The **course schedule and materials** will be posted on the course website: http://www.ece.tufts.edu/es/4.

Course announcements and Q&A will be on Piazza. You should receive an email invitation just before the course starts.

Quizzes will be on Canvas.

The **textbook** is "Sarah L. Harris & David Money Harris, *Digital Design and Computer Architecture, ARM Edition*." ISBN: 978-0128000564

The book is available online through Tisch Library, with no limit on the number of concurrent users. The paper version is available through the usual channels for about $80.

There are also some (free) online resources for the book, including an electronic-only chapter 9: https://booksite.elsevier.com/9780128000564/

All **assignment submissions** and feedback will be done through Gradescope. We'll provide the signup link once the course gets started.

There are several computer games that cover large portions of the course content. The best one I've found so far is **Turing Complete** (https://turingcomplete.game). At $19.99, it's way cheaper than a textbook and a lot more engaging!

There are also a number of great websites and videos about digital design. Some of these are posted in the "possibly helpful" and "just for fun" sections on the course website, but there are many more. If you discover other resources which are particularly helpful, please share them so we can all benefit!

How will you succeed in this course?

Mind the details. There are a lot of "moving parts" in this course: readings, quizzes, labs, homework, etc., so make sure you have a working system to track all these deadlines and complete each on time.

Keep up with the readings. I know that when the semester gets busy, readings are the first thing to go. And the consequences seem minor, even nonexistant. But three weeks later you discover you have no idea what's going on, and now you're behind and just as busy as ever. Don't let this happen to you!

Don't go it alone. The course will be difficult at times, and may require you to develop new ways of thinking. The whole point of learning this material in a university is that you're surrounded by other people to support you!

- Find a study group! You can utilize Piazza for this, or make a move and introduce yourself to that person who looks like they're sitting along in class.
- The teaching staff are here for you! We're all here because we want to see you grow and learn and succeed in this course. We have regular office hours (posted on the course website) and are happy to talk through anything related to the course, even when you're so confused you don't even know what question to ask.

- I have an open door policy in addition to my posted office hours — if my door is open, come on in! I'm also available by appointment if you can't make my regular office hours.
- Know when to get help! As a general rule of thumb, you should struggle with a problem for at least half an hour, but not more than an hour. If you haven't made any progress on a problem after an hour, then take a break and do something else, and reach out for help on Piazza or in person.

  No hero points will be awarded for spending eight hours debugging a circuit or staying up all night trying to finish a VHDLweb problem. I will not impressed by your determination; I will be horrified by your foolishness!

Analyze your learning. You may be used to evaluating your progress entirely based on external feedback, i.e., the grades a teacher gives you on assignments and exams. While there are plenty of graded assessments in this course, I expect that you will move toward evaluating and monitoring your own learning. Once you graduate, external evaluations will be few and far between. The ability to analyze your own thinking and identify gaps in your understanding is a crucial skill, far more valuable than the technical content of this course.

How will you and I evaluate your progress?

The upside to having so many components of the course is that you have many opportunities to check your understanding. Solutions to the odd-numbered problems in the textbook are posted on the publisher's website, and we will share solutions to the homework. VHDLweb (an online coding platform created for this course) gives you immediate feedback as to whether your designs work or not.

Reading checks (10 %) Before each class, you will complete a short multiple-choice quiz based on the textbook reading. The purpose of the reading check is to force you to read the textbook; it is *not* intended to be a summary of the most important material in the chapter. The questions are taken almost verbatim from the text, so if you're not sure about an answer, go back and read the text more carefully. Canvas will give you 3 attempts at the quiz, but may give you different questions each time.

Labs, prelabs, and lab reports (30 %) We will have lab every week. Most labs have a pre-lab assignment due 24 hours before your regularly-scheduled lab section, which allows your TAs to give you early feedback and avoid wasting precious face-to-face time in lab. Prelabs are graded on a very rough credit/partial-credit/no-credit scale.

Lab projects themselves are graded for completion. Part of each lab is to write up a plan for proving that your design works, so you'll know when you've got it working!

You will write a lab report for each major lab project (roughly every other week). Technical writing is a tough skill to learn, so we will provide abundant feedback on your early reports to help you improve.

Homework and VHDLweb exercises (15 %) Most weeks will have a homework assignment which is intended to help you practice essential skills that may not be emphasized by the lab. The homework should not take more than 2–3 hours; if it's taking longer, please reach out for help.

In-class exams (25 %) There will be two in-class exams, one after the first major section of the course, and the second just before Thanksgiving break.

Final project (20 %) The final project will allow you to showcase everything you've learned in the course by working in a team to build a digital system that is both complex and fun. Past projects have included arcade games, audio synthesizers, and more. One of the TAs will meet with you weekly to help you set realistic goals and to consult on the technical aspects of your project.

What else do you need to know?

Instructor contact:

If you have a general question about the course content or course logistics, please post on Piazza rather than emailing the teaching staff. You'll usually get a faster response, and everyone benefits from the answer.

Steven Bell `sbell@ece.tufts.edu`

Halligan 112 (click here for directions)

Office hours:

- Mondays 3pm-4:30pm
- Thursdays 10am-11:30pm
- I'm also available other times by appointment (generally mornings Monday-Thursday, as all of my classes are scheduled for the afternoons).

To minimize distraction, I generally only check email 2-3 times a day. However, I will make a strong effort to answer all messages within 24 hours on weekdays.

Prerequisites:

There are no prerequisites for this course. You will get more out of the course if you have taken (or are concurrently taking) CS 11 and EE 20, but I intend for this course to be acccessible to everyone, including students in the school of Arts & Sciences.

Late policy:

Life gets crazy during the semester, so you may use up to 8 late days for homework and lab reports. You do not need to do anything special to use a late day; just submit your assignment on Gradescope as normal, and it will be marked late. For simplicity of bookkeeping, late days must be used in whole days; i.e., an assignment 5 minutes late counts for one late day, as does an assignment 23 hours late.

You may not use more than 3 late days on any one assignment, since too much slack makes it easy to fall behind. We will set the late due dates on Gradescope to reflect this.

If you have a major illness, family emergency, or other extenuating circumstance such that you need more time, please get in touch before the deadline and let me know what's going on.

Academic integrity:

One question guides decisions about academic integrity in this course:

> Does this *help* me learn, or is it a way to *avoid* learning?

Collaboration: I encourage you to collaborate with your classmates on the homework, labs, and (of course) the final project. Talking through problems is one of the best ways to expose knowledge gaps and misundersandings and to build deep intuition. But just like physical exercise, those who put in the work get the gains. Thus, **the work that you turn in must be your own**.

- For written work, you're welcome to collaborate on a whiteboard or other shared thinking space, but each student needs to write up their own solution. When you're done, you should be able to work a similar problem on your own.
- For code, you may work on the problem together, discussing approaches or looking at each other's code to debug errors. However, you should not be copying code, either by copy-pasting or by typing from another screen.

- For labs, every student needs to get their own implementation of the lab project working, whether it is a circuit or FPGA design. You're welcome to sit side-by-side and build the circuits together, but you cannot borrow someone else's circuit or copy another student's code.

Internet resources: You can (and should!) look up related resources on the internet, but again they should be to supplement your learning, not replace your thinking. For example, looking up resources on how to do Boolean logic simplification is excellent, but using a simplification tool to solve the homework problems is not. Searching a VHDL error message for possible solutions is great; searching for the complete solution to a coding problem is not.

Generative AI: While generative AI systems such as ChatGPT and Github Copilot have enormous promise to amplify a skilled developer's ability to write code, the emphasis in this course is on building your core digital design skills. *It is not acceptable to use an AI tool to generate code for any assignment.* These systems also have a tendency to generate utter nonsense in a very confident tone, which makes it difficult to discern whether a generated solution is even useful as a reference.

Likewise, the goal of the lab reports is for you to practice writing technical reports clearly and concisely. ChatGPT is truly awful at writing lab reports (I tried!) and should not be used to generate content for a lab report or any other assignment.

AI writing tools like ChatGPT and Grammarly are good at generating nice-sounding sentences, and it's fine if you want to use them to reword your own writing for clarity and conciseness. But remember that the hard work of a lab report is figuring out what to say (and what to leave out), and these tools are useless for that.

Any code you submit may be run through plagarism-checking software, and I may use AI-checking software on code or text. By the terms of my employment with Tufts, I am required to report any suspicion of academic misconduct to the student affairs office.

ADA accomodations:

If you need special accommodations (extra time on exams, larger type on handouts, etc.), please initiate the process with the Tufts StAAR center (https://students.tufts.edu/staar-center/accessibility-services). Testing accomodations generally require at least a couple day's notice, so please do this sooner rather than later.