EE165/CS147: Digital Design Verification
Fall 2023

Meeting time: TuTh 3:00-4:15
Format: in-person
Instructors: Joel Grodstein (Tufts), Scott Taylor (Nvidia)

Prerequisites: Either ES4, EE25/CS45, EE156/CS146, graduate status or consent of the instructor.

Course Overview:

We will learn about verification of digital hardware designs at the Register-Transfer Level (RTL) using System Verilog. Industrial chip-design groups, who live or die based on their ability to produce working designs, have evolved numerous best-in-class practices that we will learn.

In fact, entire large teams of people typically work to find bugs in a design and, eventually, declare the design "mostly" bug-free. We will learn the techniques and infrastructures that these teams use.

Verfication is a discipline that lives at the intersection of hardware and software – it's writing very large programs, using modern software techniques, whose job is to test whether hardware functions correctly. Most verification teams tend to be a roughly even mix of hardware and software engineers.

Why would I care about this course as a CompE?

In most university Computer-Engineering departments, verification is a poor stepchild of design. CompE students learn, in great detail, how circuits work and how to design them. But while we are responsible for debugging the circuits we build, we never really learn the methods to do that task well. As a result, many (most?) of the projects we build in school never fully work.

When we then transition to our first job after graduation, a bit of a rude surprise awaits us – people expect our projects to work 😊. We quickly find that by and large, we are not being paid because of our shining intellect, nor even because of our great ideas. Instead, we are paid because we are good at taking a big, complex project and getting it to work. And – doing that on a tight schedule!

Verification is the process of testing whether a chip's implementation (as given by its description in, e.g., Verilog or VHDL) matches the chip's specification (often given in English), and debugging any mismatches. This course will help you learn the techniques and tools that are used for debugging (a.k.a. verifying) large industrial designs at the register-transfer level.

Mentor (one of the large EDA companies) surveys the ASIC and gate-array industries every two years. Their 2020 survey showed that:

- The average project employs roughly 10% more people with the title "verification engineer" than "design engineer."
- People with the title "design engineer" nonetheless spend roughly 50% of their time doing verification
- The job-growth rate from 2012-2020 was 5.5%/year for verification engineers vs. 1.5%/year for design engineers

But why would I care about verification as a CS major?

If your goal in life is to stay in the world of pure software and write code to abstract specifications, then you wouldn't! But there's a big world at the intersection of software and hardware. Specifically, verification is the job of finding as many bugs as possible in a hardware design before it's released to customers. While the verification task requires knowledge of hardware (the first commandment of any job: know thy customer!), it's largely a software task – it typically takes more lines of code to verify a hardware design than to implement the design. Most verification teams are roughly evenly staffed between software and hardware engineers, and in fact few pure-hardware engineers have the software skills for the task.

How did this course come about?

This is a bit of an odd-duck course – the teaching is split between Tufts faculty and Nvidia engineering, and it's a topic that is rarely taught in universities. It came about because while verification is a vital real-world skill, only a handful of universities teach it. The deficit was important enough that Nvidia gave a senior engineer clearance to take time to teach half the course and bring Tufts students (and faculty!) up to speed on the topic.

Course Objectives:

Upon completion of the course, students will:

1. Understand what it means to verify designs at the register-transfer level.
2. Learn the various standard parts in an industrial design-verification testbench and infrastructure.
3. Learn about test writing, test coverage, and deciding when you are done testing.
4. Be competent in writing tests in SystemVerilog (which is essentially the standard language for that task).

Course format

Most of the course work will be done in small groups of 2-3 students. You will work on your assignments as a group and hand in just one assignment per group. Each group will be building up a more and more sophisticated verification infrastructure, with each assignment building on the one before.

We will have oral quizzes; a short quiz where, again, your group works as a whole and you all get the same grade. We do not plan to have any written quizzes, not tests.

Rough course sequence:

- What is verification? Why is it important?
- Introduction to SystemVerilog and the FIFO
- Understanding a design
- Verification overview
- Writing a testbench
- Building a mesh router
- Generating stimulus
- Correctness checking
- Writing a test plan
- Measuring and improving test coverage

Grade Formula

The exact grading breakdown is TBD. One reasonable version could be

- Programming assignments – 65%
- Quizzes – 35%

Quizzes and exams:

There will be several quizzes assigned during class. Dates will be posted on the course calendar and communicated in class. Quizzes are typically graded and returned promptly; thus, they can only be taken late by prior arrangement (e.g., for sickness or a hard scheduling conflict). We do not plan to have any exams.

Programming Assignments:

There will be programming assignments throughout the term. You will be writing the RTL for your hardware (a mesh router) and writing the verification testbench for it. We will be starting small and building on the code over the semester until we have a reasonably sophisticated mesh router and verification testbench.

Final project or exam:

We do not plan to have a final exam. Our final assignment will be a sort of final project; you will get a routing mesh that

- has been designed by the instructor(s)
- has several features that you will not have coded before (but that are discussed in class)
- will have one or more bugs – your job is to find them
- as usual, you may work in teams.

Late Assignments:

Late assignments will be penalized by 10% per day. Any extensions due to extenuating circumstances (illness or family emergencies) must be arranged ahead of time with the instructor before the original due date.

Textbook and references

The textbook for the course is "*SystemVerilog for Verification: A Guide to Learning the Testbench Language Features*" by Chris Spear. It is available online at Tisch Library for free.

"*Verilog and SystemVerilog gotchas: 101 common coding errors and how to avoid them*" by Stuart Sutherland is another tried-and-true textbook.

There are numerous textbooks on SystemVerilog available for purchase as well.

Collaboration policy

Learning is a creative process. Individuals must understand problems and discover paths to their solutions. During this time, discussions with friends and colleagues are encouraged—you will do much better in the course, and at Tufts, if you find people with whom you regularly discuss problems. But those discussions should take place in English, not in code. If you start communicating in code, you're breaking the rules. When you reach the coding stage, therefore, group discussions are no longer appropriate. Each program, unless explicitly assigned as a pair problem, must be entirely your own work. Do not, under any circumstances, permit any other student to see any part of your program, and do not permit yourself to see any part of another student's program. In particular, you may not test or debug another student's code, nor may you have another student test or debug your code. (If you can't get code to work, consult a teaching assistant

or the instructor.) Using another's code in any form or writing code for use by another violates the University's academic regulations. Do not, under any circumstances, post a public question to Piazza that contains any part of your code. Private questions directed to the instructors are OK. Suspected violations will be reported to the University's Judicial Officer for investigation and adjudication. Be careful! As described in the handbook on academic integrity, the penalties for violation can be severe. A single bad decision made in a moment of weakness could lead to a permanent blot on your academic record. The same standards apply to all homework assignments; work you submit under your name must be entirely your own work. Always acknowledge those with whom you discuss problems! Suspected violations will be reported to the University's Judicial Officer for investigation and adjudication. Again, be careful

Academic Support at the StAAR Center:

The StAAR Center (formerly the Academic Resource Center and Student Accessibility Services) offers a variety of resources to all students (both undergraduate and graduate) in the Schools of Arts and Science, Engineering, the SMFA and Fletcher; services are free to all enrolled students. Students may make an appointment to work on any writing-related project or assignment, attend subject tutoring in a variety of disciplines, or meet with an academic coach to hone fundamental academic skills like time management or overcoming procrastination. Students can make an appointment for any of these services by visiting go.tufts.edu/TutorFinder, or by visiting our website (https://students.tufts.edu/staar-center).

Accommodations for Students with Disabilities:

Tufts University values the diversity of our students, staff, and faculty; recognizing the important contribution each student makes to our unique community. Tufts is committed to providing equal access and support to all qualified students through the provision of reasonable accommodations so that each student may fully participate in the Tufts experience. If you have a disability that requires reasonable accommodations, please contact the StAAR Center (formerly Student Accessibility Services) at StaarCenter@tufts.edu or 617-627-4539 to make an appointment with an accessibility representative to determine appropriate accommodations. Please be aware that accommodations cannot be enacted retroactively, making timeliness a critical aspect for their provision.