EE94/194: Intro to VLSI CAD Algorithms
Comp50/150 (*crosslist*)
Spring 2017

Instructor: Joel Grodstein (joel.grodstein@tufts.edu)
Meeting time: Tue/Th 4:30-5:45pm

Course description: *Introduction to VLSI CAD Algorithms* will be a class to learn the challenges and algorithms of CAD for VLSI – i.e., learn to use algorithms from computer science to automatically create and validate VLSI circuits. We will use techniques such as DFS and BFS search and dynamic programming to solve problems in VLSI timing, logic synthesis and post-silicon test. Depending on student interest, we may also apply the same algorithms to other fields. This class will not use commercial CAD tools (though we will have guest lectures from people in the local CAD industry).

Expected course outcomes:

- students will learn enough about VLSI to appreciate the challenges of designing chips that, each year, encompass more and more complexity and become more and more ubiquitous in our lives.
- students will learn how to use various fundamental algorithms from computer science to help automate the process of designing VLSI chips.
- students will, per their own interest, learn applications of these algorithms to other fields that they're interested in.

Textbook: none. The material will be covered mainly by lectures, and partially by reading research papers.

Prerequisites: COMP160, or graduate standing, or consent of the instructor. The class will deal with algorithms and with digital design, and so it would be helpful to have a bit of background in both. Our algorithms work will be self-contained: any algorithms we use will be fully explained in class, and so COMP160 is really more for background than a necessity. We will assume you know the basics of digital logic (e.g., what a NAND and NOR gate is, what a latch is, and what gate delay is); but if not, then you can hopefully catch up fairly quickly. Note that EE103 is not a prerequisite, and could be taken concurrently if desired.

Guest lecturers: we will have two guest lecturers from the local CAD industry. One will be from Synopsys in Marlboro, talking about timing verification and about the CAD industry. The other will be from Intel in Hudson, MA, talking about post-silicon debug. We will cover both these topics in the class as well.

Course structure: the course will cover half a dozen or so topics in CAD. Each topic will have 2 or so lectures. Most of the topics will then have a research paper assigned that we will discuss in class. There will then be an assignment. For each topic, students will have their choice of either a paper-and-pencil homework, or a small software project (typically in C or C++), or a short in-person oral quiz. After each homework is due, we'll go over it; there are learnings to be had from both the paper-and-pencil and the programming homework, and we want to ensure that all of the students profit from all of the assignments. Graduate students will have the same assignments and choices, but in addition may have to occasionally do more than one of the three choices or may lead some of the research-paper discussions.

***Tests, projects, grading and logistics*:** there will not be any tests. Depending on how fast the class moves and on student interest, we may allocate the last few weeks for an independent research project where students pick a research paper and report to the class on it. The class grade will be an equally-weighted average of the various assignments. If we do the final

research project, it will count for two assignments. Homework will be available on the class web page, and should be turned in via *provide*. Grades will be returned on Trunk.

Late assignments: each student will be allocated five late days per semester that can be used for homework assignments and course projects. Please make a note on top of the assignment if you would like to use a late day. After the allocated late days have been exhausted, the grade on a late assignment will be penalized by 10% per day.

Tentative schedule of course material:
The potential topics are listed below. We will cover some subset of them, depending on the pace of the class and on student interest.

- *Introduction to VLSI and CAD*.
- *Event-oriented simulation*. For many products (including VLSI chips, airplanes and medical instruments), designing them correctly is far less expensive than having bugs and trying to fix them after the fact. Simulation is one of the most common means of finding bugs before manufacturing. We will learn about event-oriented simulation, and ways of making it faster so as to simulate larger systems. If desired, we may learn some more advanced topics; time-warp parallel simulation (where processors simulate the future and then, if they've simulated wrongly, send anti-messages to cancel the any incorrect messages they've sent), or simulation for biological systems.
- *Static-timing analysis*. Faster is (usually) better. We will learn about why we want VLSI chips to run faster (not just the obvious reason!), and about how we can predict the speed of a chip, and the critical paths, before building it.
- *Post-silicon speed debug*. Despite our best efforts, chips have bugs. We must find critical paths on real silicon, an environment where the bugs are in circuits far too small to see most of what is going on, mistakes flash by in picoseconds, and volume manufacturing is waiting, day by day, for you to find the "last" bug. We will learn about the ways that people debug critical paths in these difficult circumstances, with the aid of CAD algorithms and intelligent design.
- *Bounded model checking*. Simulating a product to flesh out the bugs is wonderful – except that there is usually not enough time or computers in the universe to fully simulate most real-life products. Bounded model checking is one of the simpler methods of *formal validation* – using formal techniques to prove that your design obeys various properties. Formal techniques can be used to validate many large, complex systems, including software systems.
- *Automatic Test-Pattern Generation*. It's not enough to design a chip; we must also be able to manufacture it in volume, and that means testing to weed out defective chips. We'll learn about modeling what can go wrong in manufacturing, and how to use this model so as to automatically generate tests for a product.
- *Boolean satisfiability*. Algorithms for both bounded model checking and test-pattern generation are often built on a fundamental technique call *SAT* (or Boolean satisfiability). On the one hand, SAT was the first problem shown to be NP complete; there is unlikely to ever be an algorithm that solves all large SAT problems in a reasonable amount of time. On the other hand, algorithm improvements by the CAD community have made SAT very practical for "most" problems. We will learn how this is done, based on a combination of clever algorithms and good software engineering. SAT has extremely wide-ranging applications in numerous fields.

- *Technology mapping*. The best way to design complex chips quickly is to design at a high level of abstraction – but eventually your high-level design must get turned into actual NAND and NOR gates. Technology mapping is one of the techniques involved in this process. We will learn some of the algorithms involved. Not surprisingly, they have uses in compiler design. In the future, they may be used to design synthetic biological organisms. If interested, we may also discuss the Synopsys-Magma lawsuit; one of the largest lawsuits in CAD history was fought over this technology, including $100M in claims against an individual engineer, $12M in damages, and an eventual $500M acquisition (your instructor happened to be involved in the issue).

Potential research papers to read:

Discrete-event simulation:

- *Unlocking ordered parallelism with the Swarm architecture*, IEEE Micro
- *A Scalable Architecture for Ordered Parallelism*, Micro-48, 2015
- *Fast concurrent simulation using the time warp mechanism: part I, local control*, Jefferson, David, 1982. (Time Warp was the first serious attempt at parallel event-driven simulation and is a very cool paradigm).

Static-timing analysis

- *New Game, New Goal Posts: A Recent History of Timing Closure*, Andrew Kahng, DAC '15 (a summary of recent practical issues in timing closure)
- *Analysis and Design of Latch-Controlled Synchronous Circuits*, Karem Sakallah, IEEE T.CAD 1992 (the classic paper on analyzing level-sensitive latches).

Post-silicon debug

- *AutoRex: An automated post-silicon clock tuning tool*, Desta Tadesse, ITC 2009. (Using an SMT solver to pick LCP settings for Intel CPUs).

Technology mapping:

- *Applying logic synthesis for speeding up SAT***,** Nicholas Een and Alan Mischenko (another way to speed up SAT)
- *Reducing structural bias in technology mapping*, Alan Mischenko (A competing approach to the choice gates that we discussed in class).

ATPG?

BMC

- *Checking safety properties using induction and a SAT solver*, Mary Sheeran, FMCAD 2000 (Introduces unbounded model checking).

SAT

- *Satisfiability Modulo Theories: Introduction and Applications*, Leonardo de Moura and Nikolaj Bjørner, CACM 2011
- *Combining strengths of Circuit-based and CNF-based Algorithms for a High-Performance SAT Solver*, Malay Ganai, DAC 2002 (a hybrid solver that claims to be faster than ZChaff for SAT problems that come from unrolled circuits).
- *A SAT Based Approach for Solving Formulas over Boolean and Linear Mathematical Propositions,* Gilles Audemard, CADE-18 (2002). (a.k.a., MathSAT).

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!

***Additional resources*:**
Tufts University values the diversity of our students, staff, and faculty, and recognizes 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 Student Accessibility Services office at Accessibility@tufts.edu or 617-627-4539 to make an appointment with an SAS representative to determine appropriate accommodations. Please be aware that accommodations cannot be enacted retroactively, making timeliness a critical aspect for their provision. Tufts and the teaching staff strive to create a learning environment that is welcoming to students of all backgrounds. If you feel unwelcome for any reason, please let us know so we can work to make things better. You can let us know by talking to anyone on the teaching staff. If you feel uncomfortable talking to members of the teaching staff, consider reaching out to your academic advisor, the department chair, or your dean.