ECE 590.03: Fundamentals of Computer Systems and Engineering
Dr. Andrew (Drew) Hilton

General Information
Professor: Andrew Hilton
Office: Hudson 211
e-mail: adhilton@ee.duke.edu
Office Hours: Monday 11am—noon, Thursday 1pm—3pm

Teaching Assistants:
Andre Van Rynbach (GTA)
Amay Jhaveri
Zach Michaelov
Mason Meier
John Cuffney

More Information
• TA office hours: TBD
  − Probably held in the lab
  − More info on the lab later
• Recitation Fridays
  Run by GTA (Andre) with help from UTAs
  Learn about tools (Quartus, SPIM)
  Work problems, etc…
  Bring laptops!

A bit about me
• Teaching is my primary job
  − Don’t be afraid to come to my office hours!
  − Don’t be afraid to ask me to setup some other office hours time!
• Please, feel free to call me “Drew”
  − If you are uncomfortable with that, Dr. Hilton or Prof. Hilton are ok

A bit about this class
• Brand new class in ECE!
• For: MS/MEng students who want Comp Eng focus..
  − …but don’t have Comp Eng undergrad
• Background for
  − ECE 522: Advanced Computer Architecture
  − ECE 554: Fault-Tolerant and Testable Computer Systems
  − ECE 556: Wireless Networking and Mobile Computing
  − ECE 559: Advanced Digital System Design
  − CS 510: Operating Systems
  − CS 512: Computer Networks/Distributed Systems

A bit about you
• Before we get too much further, I’d like you all to introduce yourselves
  − I will try really hard to learn everyone’s names quickly… generally takes me a couple weeks to get them all down though
  − Go around the room:
    • Name
    • Undergraduate major
    • What you hope to get out of this class
What we will learn: 10K feet

- Transistors -> Processor
  - Logic gates, combinational logic, sequential logic, FSMs
  - Adders, multipliers, shifters
  - Latches, Flip-flops, SRAMs, DRAMs, CAMs
  - Single-cycle datapaths, pipelining
  - Caches, memory hierarchy, virtual memory
  - Interrupts, exceptions, IO

- Hardware/software interface (ISA)
  - MIPS assembly

- Operating System basics
  - System calls, protection, multi-tasking, ...

- Networking basics
  - 7 layer OSI model, TCP/IP, routing,...

How we will learn it

- Must "learn by doing":
  - 4/6 homeworks: Implement something VHDL
    - Write VHDL, synthesize it
    - Load it on Altera DE2 board
    - Run it, demo it to TAs
  - 1/6 homework: Write MIPS assembly
    - Run it in SPIM
  - 1/6 homeworks: just pencil and paper (last one)
    - Will cover OSes and Networking portion
    - Don’t have time to learn C in this class to have a "doing" here

A word about varying backgrounds

- I expect wide variations in backgrounds for this class
  - E.g., some know VHDL, some have never seen it
- Even if you are familiar with a topic we are covering…
  - You may learn something new
  - You may refresh rusty memory
  - I may present it slightly differently than you are used to
  - You may be able to help other students learn it
- If you missing some background (feeling lost)
  - Please come talk to me or a TA sooner rather than later!

Homeworks

- Homeworks: 6 of them over semester
  - Work in groups of 2 to 3
    - Fixed groups for semester (except unusual circumstances)
  - Last ~5—10 min of recitation this week/next week to form/meet with groups
  - VHDL tasks:
    - Hwk1 Read switches, display as hex number on LEDs
    - Hwk2 Write a VGA controller ("baby video card")
    - Hwk4 Write a simple datapath ("baby processor")
    - Hwk5 Write a cache (simplified a bit from a real cache)

Homeworks cont’d

- Homeworks will be turned in and the VHDL portion demoed
  - Due date: turn electronically in to Sakai
    - Note: tar, tar.gz, and zip only acceptable archives. No Word documents for written portions!
  - Demo: sometime in the next week, setup with a TA
- During demo:
  - Show TA that it works
  - Answer questions about/discuss what you did
  - ALL group members MUST know how it all works
    - TA will ask a particular group member to explain something
      - "I don’t know, Joe worked on that part" will lose you points

Grading

- Grade breakdown
  - Class Attendance/Quizzes 10%
  - Midterm Exam 25%
  - Final Exam 35%
  - Homework 30%
- Late homework policy
  - 5 late days per group total for the semester
    - Does not change demo deadline, only submission deadline
    - Days, not classes
    - Used in entire days at a time: 10 min late = on next day.
    - After used up, no credit for late work.
    - No, really. You can’t ask for more later
- This course takes time, start assignments early.
  - Debugging VHDL can be very time consuming....
Class Attendance/Quizzes

- I expect you to attend class.
  - But understand that sometimes things come up and you need to miss.
- We will have 12 “quizzes” throughout the semester in lecture.
  - You get 50% for being there, 50% for correct answers.
  - Drop 2 lowest quiz grades (count 10 out of 12).
  - Questions mostly check if paying attention/understanding discussion.
  - Recommend asking questions if you are unclear on things.
  - Reading in advance of lecture.
  - May try out a few different quiz formats.
- Drop quizzes: account for needing to miss lecture.
  - Long term circumstances, please talk to me.

Academic Integrity

- Academic Integrity Expectations
  - I take academic integrity VERY seriously, and you should too.
  - Basic principles for Duke in general:
    - I will not lie, cheat, or steal in my academic endeavors, nor will I accept the actions of those who do.
    - I will conduct myself responsibly and honorably in all my activities as a Duke student.
  - If I suspect academic misconduct in my class...
    - Reported to the appropriate Associate Dean.
    - Due process to determine if you did commit academic misconduct.
    - If found responsible,
      - I will give you a 0 on the assignment.
      - Appropriate Associate Dean may apply additional penalties.

Academic Integrity: Homeworks

- Expectations for homeworks
  - You will work in groups of 2 or 3, so collaborate freely within group.
  - Should not discuss specifics of homework solutions with other students outside your group.
  - May freely discuss general class material, study for exams etc.
  - Do not exchange (or look at/show) code with other groups.
  - Exceptions: TAs and myself.
  - Outside resources for general information:
    - Can reference webpages/texts/etc for general information.
    - Examples: general VHDL syntax, DE2 board user’s manual.
    - Note that you may not download code and re-use it, even if you cite it.

Academic Integrity: Exams

- Exams in this class are individual effort.
  - No outside resources/help except 1 page of notes.
  - No textbook.
  - No talking to friends/group members.
  - No text messages/cell phones/laptops/calculators/smart phones.
  - Before I return your exams, I will photocopy and keep a random subset.
  - If you request a re-grade, I will compare your solutions to my photocopy. If they have been changed, I will report the incident directly.
- Related exam policies:
  - Questions? Raise hand, TA or I will come to you (don’t get up).
  - Need restroom? Raise hand, we will let you go one at a time.
  - No calculators/smart phones: too easy to use to chat.

Academic Integrity: Mini-Quizzes

- You may use your book, notes, and/or slide print outs.
  - I.e., any hard-copy written material.
- You may not discuss with other classmates.
  - Talking/text messages/note passing etc.
- You may not use electronic/interactive resources.
  - No internet, no computers, no smartphones, no calculators.

Academic Integrity: General

- Some general guidelines
  - If you don’t know if something is OK, please ask me.
  - If you think “I don’t want to ask, you will probably say no” that is a good sign its NOT acceptable.
  - If you do something wrong, and regret it, please come forward—I recognize the value and learning benefit of admitting your mistakes. (Note: this does NOT mean there will be no consequences if you come forward).
  - If you are aware of someone else’s misconduct, you should report it to me or another appropriate authority.
    - Within your homework group, this becomes even stronger: if you are aware that one of your group members has committed misconduct on a homework submission for your group, you are complicit in it if you do not report it.
Course Problems

• Can’t make midterms / final, other conflicts
  – Tell us early and we will schedule alternate time
• Irresolvable group problems
  – Come see me. Will allow group changes in extreme circumstances
  – Prefer you try to resolve issues first.
• Other problems:
  – Feel free to talk to me, I’m generally understanding and will try to work
  with you
  – Some problems may extend well beyond my course
  – Talk to your Director of Graduate Studies (probably Dr. Cummer)
  – Talk to your Academic Dean

Lecture

• You all will get more out of this if you participate than if I just talk for 75 minutes
  – Please ask questions, discuss things you are unclear on, etc.
  – I will ask you all to answer questions
  – Don’t be afraid of this, I’ll ask everyone
  – If you are wrong, it’s not the end of the world, we’ll stop and make
  sure you get it
  – Typically, I’ll work my way around the room, so nobody will be
  surprised that they are next to be called on
  – Speaking of which, this is a good time to stop and have everyone
  remind of their names… (Hey, that’s a pretty easy question right?)

Resources

• Sakai
  – Turn in assignments
• Piazza
  – Discussions, questions, etc
  – Announcements I make: required reading
  – Other discussions: strongly recommended reading
• Course Web Page
  http://people.ee.duke.edu/~adh39/courses/fall_2012/ece590.03/

Textbook, etc.

• Text: Computer Organization & Design.
  (Patterson & Hennessy)
  – You are expected to complete the assigned readings
  – Some material on the CD (e.g., Appendix)
  – Note: get revised 4th edition, but not any of the unusual variants (e.g.,
    ARM version)
• We are going to skip around a bit relative to the book
  – Digital logic earlier/more focus on it
  – Pipelining later, less focus
• Read
  – Start reading Chapter 1 and Appendix C now

Transistors => Processors

• Processors are made of **transistors**
  – Electrically controlled switches (more on this later)
  – Anyone have any idea how many transistors are in a modern
    chip?

Transistors => Processors

• Processors are made of **transistors**
  – Electrically controlled switches (more on this later)
  – Anyone have any idea how many transistors are in a modern
    chip? 2.6 BILLION

• How do you put together 2.6 Billion of anything
  – …and make sure the product works right?
  – …in every corner case
  – …and is really fast
  – …and do it within a reasonable budget/timeframe?
• More fundamentally, how do you engineer any large
  system?
**Transistors => Processors**

- Processors are made of **transistors**
  - Electrically controlled switches (more on this later)
  - Anyone have any idea how many transistors are in a modern chip? **2.6 BILLION**

- How do you put together 2.6 Billion of anything
  - ...and make sure the product works right?
  - ...in every corner case
  - ...and is really fast
  - ...and do it within a reasonable budget/timeframe?

- More fundamentally, how do you engineer any large system? **Abstraction**

**Abstraction: The Key to Computer Engineering**

- Abstraction: Divide interface from implementation
  - Interface: how it's used
  - Implementation: how it does it

  - Build larger components from smaller ones
    - Larger ones use interface of smaller ones to do tasks
    - Don't care about implementation

  - Tasks can be split between engineers:
    - You make a piece that does xyz, and I'll use it to do my job

  - Components can be re-used
    - Also good: making them generic, so they can be re-used more

**Other key to engineering: tools**

- Processors designed in Hardware Design Languages
  - Verilog
  - VHDL
  - Learn one, you can pick the other easily

- You don’t layout ever transistor by hand…
  - Instead you write a description of the hardware in an HDL
  - ...a lot like a programming language,...
  - ...then run it through synthesis tools

- We’ll use VHDL and Quartus

**Levels of Abstraction**

- Transistors: “electrical switch”
  - Can go lower (those with EE background have)
  - ...but no need for us

- Gates: a few transistors
  - Implement logical functions: And, Or, Nor, Xor

- Meaningful logic elements: a handful of gates
  - Combine into meaningful elements: muxes, 1-bit adders, flip-flops
  - May build larger items: N-bit adders from 1-bit adders

- Large elements (stages, units): combining logic elements
  - Core
  - Chip: now with multiple cores

**A Different Kind of Abstraction**

- Previous discussion: abstraction to build a processor

- Also: abstraction to **use** a processor
  - How/why?
Wrap up there for the day

• We’ll end there for the day
• Remaining time:
  – Any questions I need to take individually

• Next time:
  – Digital Logic: transistors to gates
  – Start reading Ch1 and Appendix C