EECS150 - Digital Design

Lecture 28 - Course Wrapup

April 29, 2010

John Wawrzynek

Why Study and Learn Digital Design?

- We expect that many of our graduates will eventually be employed as designers.
  - ***Digital design is not a spectator sport.*** The only way to learn it and to appreciate the issues is to do it.
  - To a large extent, it comes with practice/experience (this course is just the beginning).
  - Another way to get better is to study other designs. Not time to do this during the semester, but a good practice for later.
- However, a significant percentage of our graduates will not be digital designers. What's in it for them?
  - Better manager of designers, marketers, field engineers, etc.
  - Better researcher/scientist/designer in related areas
    - Software engineers, fabrication process development, etc.
  - To become a better user of electronic systems.

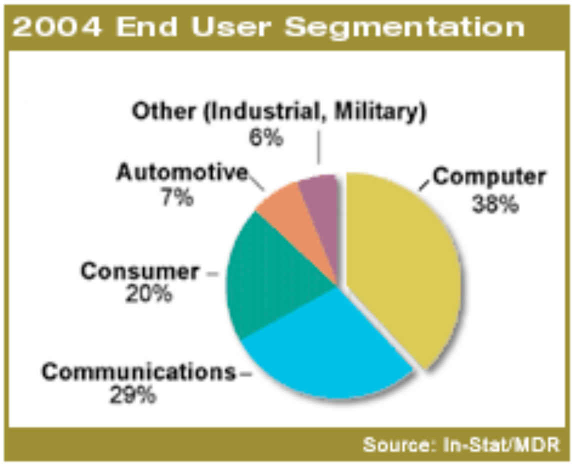
In What Context Will You be Designing?

Engineers learn so that they can build.
Scientist build so that they can learn.

- Electronic design is a critical tool for most areas of pure science:
  - Astrophysics – special electronics used for processing radio antenna signals.
  - Genomics – special processing architectures for DNA string matching.
  - In general - sensor processing, control, and number crunching. In some fields, computation has replaced experimentation – particle physics, world weather prediction (fluid dynamics).
- In computer engineering, prototypes often designed, implemented, and studied to "prove out" an idea. Common within Universities and industrial research labs. Lessons learned and proven ideas often transferred to industry through licensing, technical communications, or startup companies.
  - RISC processors where first proved out at Berkeley and IBM Research

Designs in Industry

- Of course, companies are the primary employer of designers. Provide some useful products to society or government and make a profit for the shareholders.



The Big Ideas from EECS150

1. **Modularity and Hierarchy** is an important way to describe and think about digital systems.
2. **Parallelism** is a key property of hardware systems and distinguishes them from software.
3. **Clocking** and the use of state elements (latches, flip-flops, and memories) control the flow of data.
4. **Cost/Performance/Power tradeoffs** are possible at all levels of the system design.
5. Hardware Description Languages **(HDLs) and Logic Synthesis** are a central tool for digital design.
6. **Finite State Machines** abstraction gives us a way to model any digital system – however, usually only used for controllers.
7. **Arithmetic circuits** are often based on "long-hand" arithmetic techniques.
8. **FPGAs** give us a convenient and flexible implementation technology.

The Useful Skill from Class

We hope that after have taken this class that …

Given an English language description for the function of a digital system covering any of a wide variety of domains*:

You can organize and describe a digital system, and

using Verilog and logic synthesis, generate a detailed circuit at the "logic gate level", and

map to an FPGA, and

debug it, and

optimize for cost or performance or both.

** Well, at least for processors.*

What We Didn’t Cover

- Design Verification and Testing
  - Industrial designers spend more than half their time testing and verifying correctness of their designs.
    - Some of this covered in the lab. Didn’t cover rigorous testing procedures and “formal verification”.
  - Most industrial products are designed from the start for testability. Important for design verification and later for manufacturing test.
  - Built in self test (BIST), Automatic Test Vector Generation, Scan-chain techniques.
- Other High-level Optimization Techniques
  - Automatic Retiming
- Other High-level Architectures: video processing, network routers, ...
- Power Aware Design Techniques and Tools
- DRAM design and interfacing

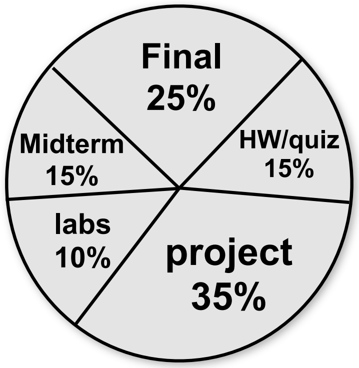
Most Closely Related Courses

- CS152 Computer Architecture and Engineering
  - Design and Analysis of Microprocessors
  - Applies basic design concepts from EECS150
- EE141 Digital Integrated Circuits
  - Transistor-level design of ICs
  - Understand how our EECS150 circuits are mapped to silicon
- CS194-6 Digital Systems Project Laboratory
  - More intensive design experience in the EECS150 style
  - Not regularly offered (someday will be a regular course?)
- CS250 VLSI Systems Design
  - Learn how to design cell-based ASICs
  - Advanced-undergrad/grad course
  - “New” format, now design-based

Future Design Issues

- Automatic High-level synthesis and optimization (with micro-architecture synthesis) and hardware/software co-design.
- Current trend is towards “system on a chip” (SOC) design methodology:
  - Pre-designed subsystems (processor cores, bus controllers, memory systems, network interfaces, etc.) connected with standard on-chip interconnect or bus.
- Increasing NREs will favor post-fabrication customization.
- A number of alternatives to silicon VLSI have been proposed, including techniques based on:
  - molecular electronics, quantum mechanics, and biological processes. “Nano-devices”
  - How will these change the way we design systems?

Course Grading & Final Exam



Review session:
- Friday May 7th
- 5-...pm, 306 Soda

- **Exam** held in scheduled final exam slot: Monday May 10, 11:30-2:30.
- Room: **2 LeConte**
- “Comprehensive” Final Exam: covers material from the entire semester with emphasis on second half
- ~2/3 of final will cover new material since Midterm exam - Lecture 19 (Combinational Logic), and on.
- ~1/3 of final will cover semester-long topics.

Important Topics from Second Half

- Definitions of the three representations for combinational logic:
  - truth tables, logic gate networks, algebraic equations
- Strengths, weaknesses, and uses for each CL representation.
- Conversion of simple logic functions among all CL representations.
- Definition and axioms of Boolean algebra
- Laws (theorems) of Boolean algebra
- Proving theorems via axioms
- DeMorgan's law and relationship to NAND/NOR gates
- Forming SOP and POS canonical forms of Boolean expressions
- Using axioms and laws of Boolean algebra for simplification
- Using K-maps for deriving reduced POS and SOP forms (up to 6 variables).

Important Topics from Second Half

- Exploiting function "don't care" values in logic simplication.
- Factoring and multi-level combinational logic.
- "Bubble pushing" and translating between AND/OR and NAND and NORs.
- Synchronizer failure and design.
- Design and operation of finite state machines (FSM) as synchronous logic circuits.
- "By hand" design procedure from state-transision-diagram (STD) to FSM circuit implementation.
- Design procedure for STD to "one-hot" encoded FSM circuit.
- Moore versus Mealy machine STDs and implementations and timing behavior.
- Moore versus Mealy in Verilog specifications.
-

Important Topics from Second Half

- Counters in controller design.
- Binary up/down counter design.
- Carry-select adder design principle, cost/performance analysis, optimization.
- Carry look-ahead adder design principle, cost/performance analysis, optimization.
- Bit-serial adders design and operation.
- Virtex-5 adder carry-chain.
- Binary multiplication principle.
- Shift-and-add multiplier design and operation.
- Extending multiplication techniques for signed multiplication.
- Bit-serial multiplier structure and operation.
-

Important Topics from Second Half

- Cost performance analysis of alternative multiplication schemes.
- Combinational (array) multiplier structure and operation.
- Carry-save addition technique and application to multiplier design.
- Multiplication by a constant, including canonic signed digit representation (CSD), and KCM factoring technique. Shifter Circuits
- Operation and implementation of FIFOs.
- Using register transfer language descriptions to specify high-level designs.
- Datapath and controller extraction from register transfer language descriptions.
- Datapath/controller optimizations for improving performance and cost.
-

Important Topics from Second Half

- Using resource utilization charts for optimization, including modulo scheduling technique.
- Design tradeoffs (optimization) through parallelism versus hardware-multiplexing.
- Ideal versus actual pipelines. Limits on pipelining.
- Pipelining with dependencies.
- SIMD parallelism.
-

The End.

- Special thanks to the Chris, Ilia, Brandon, and Kyle.
- Good luck finishing up your project and on the final.
- Thanks for a great semester!