EECS150 – Digital Design
Lecture 1 – Introduction

January 22, 2013

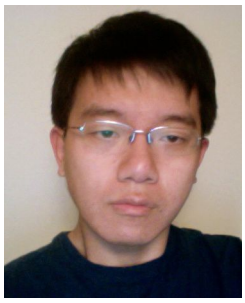
John Wawrzynek
Electrical Engineering and Computer Sciences
University of California, Berkeley

http://www-inst.eecs.berkeley.edu/~cs150

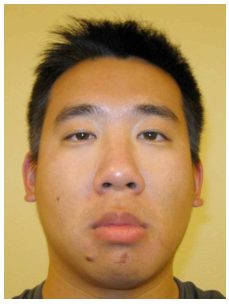
Teaching Staff

Professor John Wawrzynek
(Warznek)
631 Soda Hall
johnw@cs.berkeley.edu
Office Hours: Tu 1-2pm, & by appointment.

Shaoyi Cheng:



Vincent Lee:



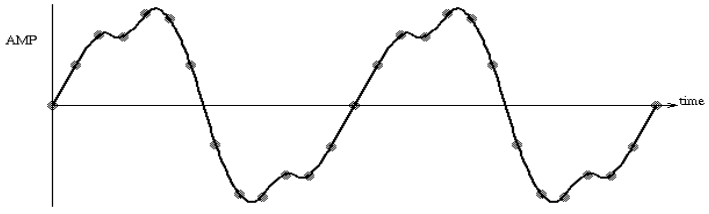
All TA office hours held in 125 Cory. Check website for days and times.

Electronics all around us

Course Content

Components and Design Techniques for **Digital Systems**
more specifically
Synchronous Digital Hardware Systems

- Synchronous: "Clocked" - all changes in the system are controlled by a global clock and happen at the same time (not asynchronous)
- Digital: All inputs/outputs and internal values (signals) take on discrete values (not analog).
  - Example digital representation: music waveform



  - **A series of numbers is used to represent the waveform, rather than a voltage or current, as in analog systems.**

Course Content – Design Layers

Not a course on computer architecture or the architecture of other systems. Although we will look at these as examples.

High-level Organization : Hardware Architectures

System Building Blocks : Arithmetic units, controllers

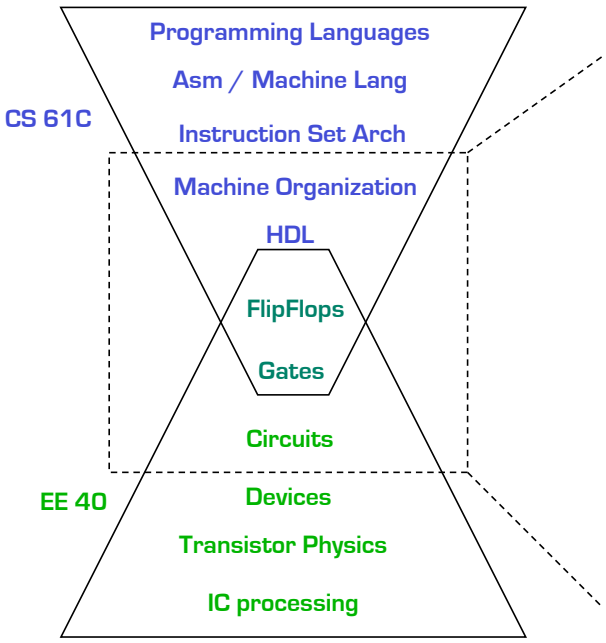
Circuit Elements : Memories, logic blocks

Transistor-level circuit implementations

Circuit primitives : Transistors, wires

Not a course on transistor physics and transistor circuits. Although, we will look at these to better understand the primitive elements for digital circuits.

Course Content



Deep Digital Design Experience

Fundamentals of Boolean Logic

Synchronous Circuits

Finite State Machines

Timing & Clocking

Device Technology & Implications

Controller Design

Arithmetic Units

Encoding, Framing

Testing, Debugging

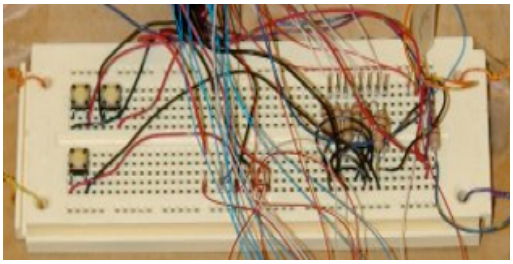
Hardware Architecture

Hardware Design Language (HDL)

Design Flow (CAD)

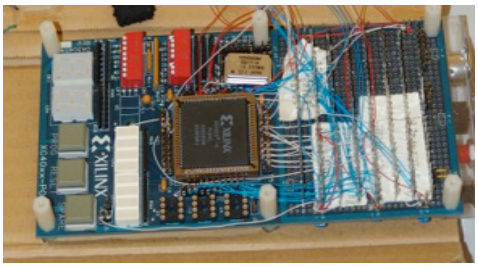
Course Evolution

- Final project circa 1980:
  - Example project: pong game with buttons for paddle and LEDs for output.
  - Few 10's of logic gates
  - Gates hand-wired together on "bread-board" (protoboard).
  - No computer-aided design tools
  - Debugged with oscilloscope and logic analyzer

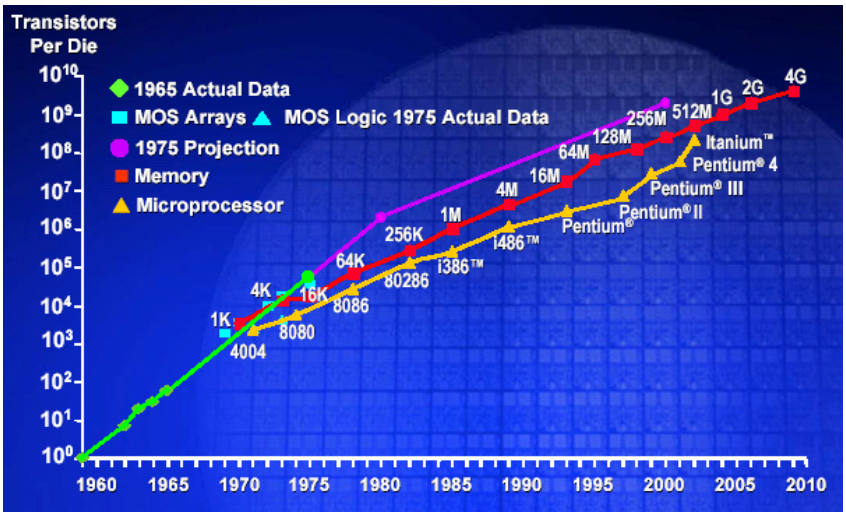


Course Evolution

- Final project circa 1995:
  - Example project: MIDI music synthesizer
  - Few 1000's of logic gates
  - Gates wired together internally on field programmable gate array (FPGA) development board with some external components.
  - Circuit designed "by-hand", computer-aided design tools to help map the design to the hardware.
  - Debugged with circuit simulation, oscilloscope and logic analyzer



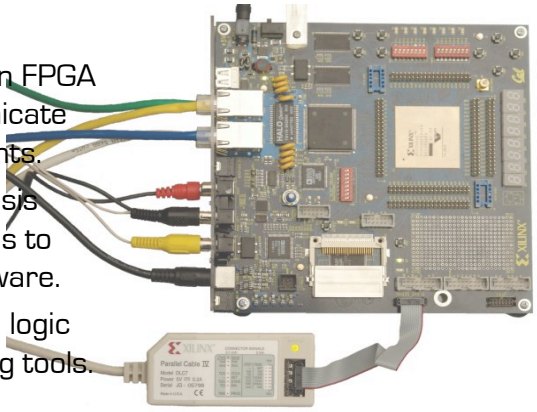
Moore's Law – 2x stuff per 1-2 yr



Course Evolution

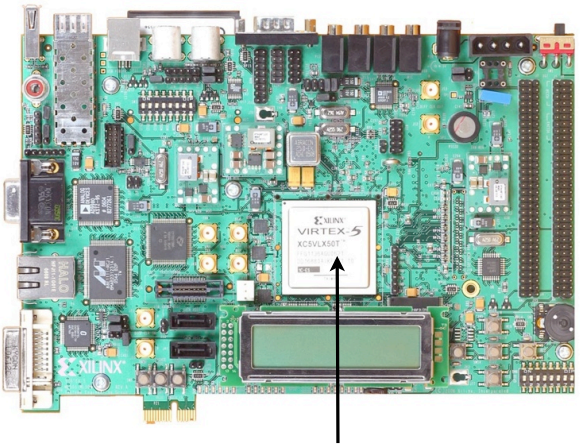
- Final project circa 2000–2008:
  - Example project: eTV – streaming video broadcast over Ethernet, student project decodes and displays video
  - Few 10,000's of logic gates
  - Gates wired together internally on FPGA development board and communicate with standard external components.
  - Circuit designed with logic-synthesis tools, computer-aided design tools to help map the design to the hardware.
  - Debugged with circuit simulation, logic analyzer, and in-system debugging tools.

Calinx Board



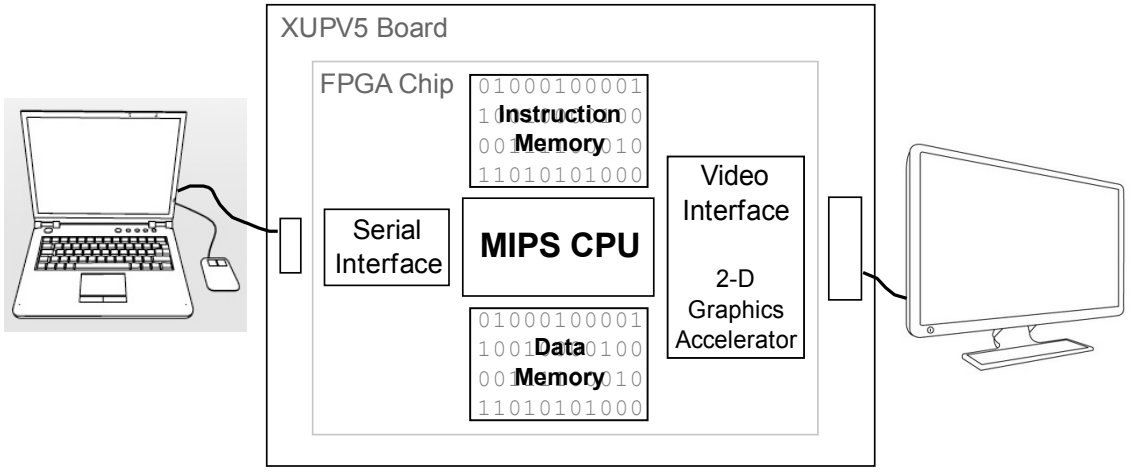
Course Evolution

- Beginning 2009:
  - Xilinx XUPV5 development board (a.k.a ML505)
  - Could enable very aggressive final projects.
  - But, modest use of resources this semester.
  - Project debugging with simulation tools and with in-system hardware debugging tools.



- LX110T FPGA: ~1M logic gates.
  - Interfaces: Audio in/out, digital video, ethernet, on-board DRAM, PCIe, USB, ...

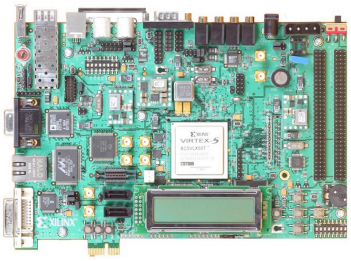
Final Project: Spring 2012



- Executes most commonly used MIPS instructions.
- Pipelined (high performance) implementation.
- Serial console interface for shell interaction, debugging, data-transfer.
- Instruction and data caches
- Video interface for display with 2-D vector graphics acceleration.
- Supported by a C language compiler.

Final Project: Spring 2013 (part 1)

- **You choose!**
- Required to use the Xilinx FPGA board, but free to add more hardware interfaces if necessary.



- Encouraged to use existing interfaces (video, audio, network, etc.), but free to add more.
- We provide design blocks: frame buffer, simple CPU, memory interface, etc.
- Ideally, your project is structured by adding accelerator hardware to simple processor
  - lets you start from software solution and migrate intensive parts to hardware, helps in debugging, demonstration

Final Project: Spring 2013 (part 2)

- Avoid tasks that are best solved in software on processors
  - Functions go into hardware for higher performance or energy efficiency
- Examples:
  - graphics acceleration, video processing, computational biology acceleration (i.e. DNA sequencing), RF and radio processing, music synthesizer, ..., your idea here!
- Start thinking NOW about what you want to do for your project
  - Preliminary proposal due in 2 weeks!
  - We will help you refine your idea, develop a schedule with milestones, development strategy, and backup position
  - Experience shows that lots of time is spent refining the functional specification and figuring out the technical approach
- Groups of 2 people - start meeting potential partners now.

Administrivia

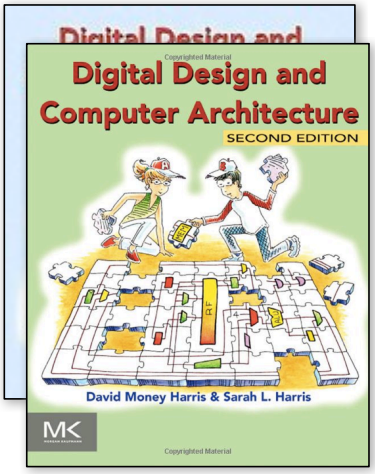
Enrollment

- If you are on the waiting list and have taken 61c or equivalent, you will be added.
- If you are enrolled and plan to take the course you must attend your lab section this week and next.
- Lab sections this week (meet TAs, pick up accounts, simple "warm-up" lab exercise)
- No discussion sections this week.

Attendance

- Attend regular lectures and ask questions, offer comments, etc.
- Attend your lab section. You must stick with the same lab section all semester.
  - Lab exercises will be done individually; project with a partner.
  - We will put together a lab section exchange in a few weeks to help you move to a different section.
- Attend any discussion section. You may attend any discussion section that you want regardless of which one you are enrolled in.
- The entire teaching staff hold regular office hours (see class webpage). Take advantage of this opportunity! Come early (and often). Don't wait until the night before an assignment is due!

Course Materials



Textbook: Harris & Harris

Publisher: Morgan Kaufmann

Class notes, homework & lab assignments, solutions, and other documentation will be available on the class webpage linked to the calendar:

http://www-inst.eecs.berkeley.edu/~cs150

- Check the class webpage and newsgroup often!
- Updated posts will occur.

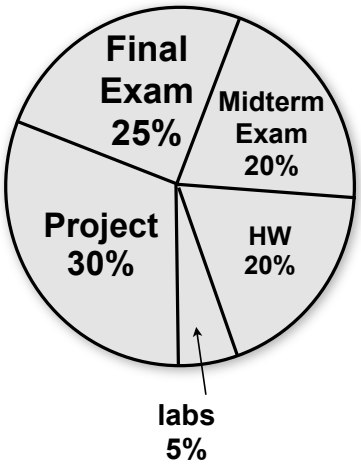
piazza For online Q/A.

http://www.piazzza.com/

More info later.

Course Grading

- Comprehensive Exam held during Finals week: Tuesday May 7 11:30-2:30.
- For those with EE120 conflict, alternative exam at 8:30AM.



- Project critical part of the course - graded on timeliness, completeness and optimality. Lots more on this later.
- Evening midterm exam, Wed March 20, 6-9pm.
- Weekly homework based on reading and lectures.
  - out before the end of each week, due before Th lecture of following week.
- Lab exercises for weeks 1-6, followed by project checkpoints and final checkoff.
- Labs due at the beginning of your next lab session.
- Checkpoints TBD.

Tips on How to Get a Good Grade

The lecture material is not the most challenging part of the course.

- You should be able to understand everything as we go along.
- Do not fall behind in lecture and tell yourself you "will figure it out later from the notes or book".
- Notes will be online before the lecture (usually the night before). Look at them before class. Do assigned reading (only the required sections).
- Ask questions in class and stay involved in the class - that will help you understand. **Come to office hours to check your understanding or to ask qestions.**
- Complete all the homework problems - even the difficult ones.
- The exams will test your depth of knowledge. You need to understand the material well enough to apply it in new situations.

You need to do well on the project to get a good course grade.

- Take the labs very seriously. They are an integral part of the course.
- Choose your partner carefully. Your best friend may not be the best choice!
- Most important (this comes from 30+ years of hardware design experience):
  - **Be well organized and neat with homework, labs, project.**
  - **In lab, add complexity a little bit at a time - always have a working design.**

Cheating

- We have posted the details of my cheating policy on the class web site. Please read it and ask questions.
- If you turn in someone else's work as if it were your own, you are guilty of cheating. This includes homework sets, answers on exams, verilog code, block diagrams, etc.
- Also, if you knowingly aid in cheating, you are guilty.
- We have software that automatically compares your submitted work to others.
- However, it is okay to discuss with others lab exercises and the project. Okay to work together on homework. But everyone must turn in their own work.
- If we catch you cheating, I will give you an F on the assignment. If it is a midterm exam, final exam, or final project, I will give you an F in the class. In either case, will be reported to the office of student conduct.

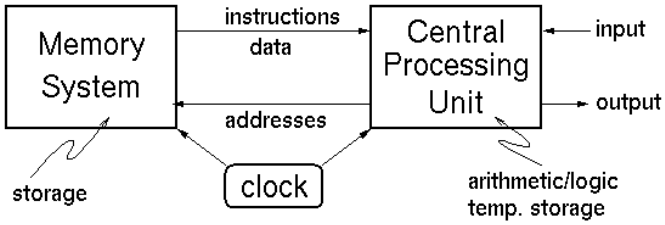
Lectures

What are they good for?

A few basic concepts

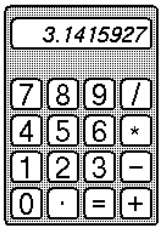
Example Digital Systems

- General Purpose Desktop/Server Digital Computer



  - Often designed to maximize performance. "Optimized for speed"

- Handheld Calculator



- Usually designed to minimize cost. "Optimized for low cost"

- Of course, low cost comes at the expense of speed.

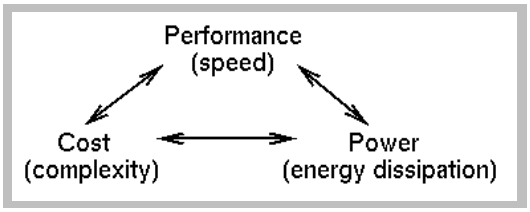
Example Digital Systems

- Digital Watch

Designed to minimize power.
Single battery must last for years.

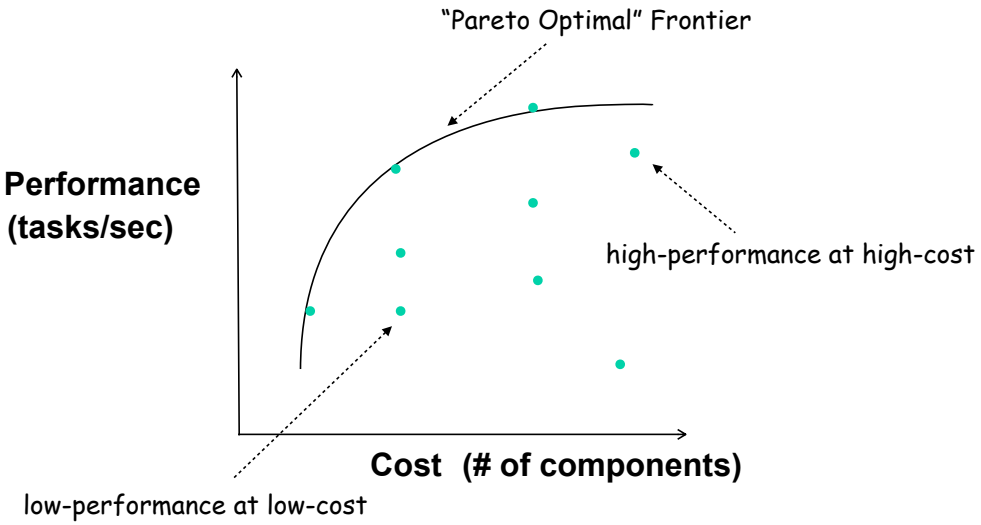
  - Low power operation comes at the expense of:
    - lower speed
    - higher cost

Basic Design Tradeoffs

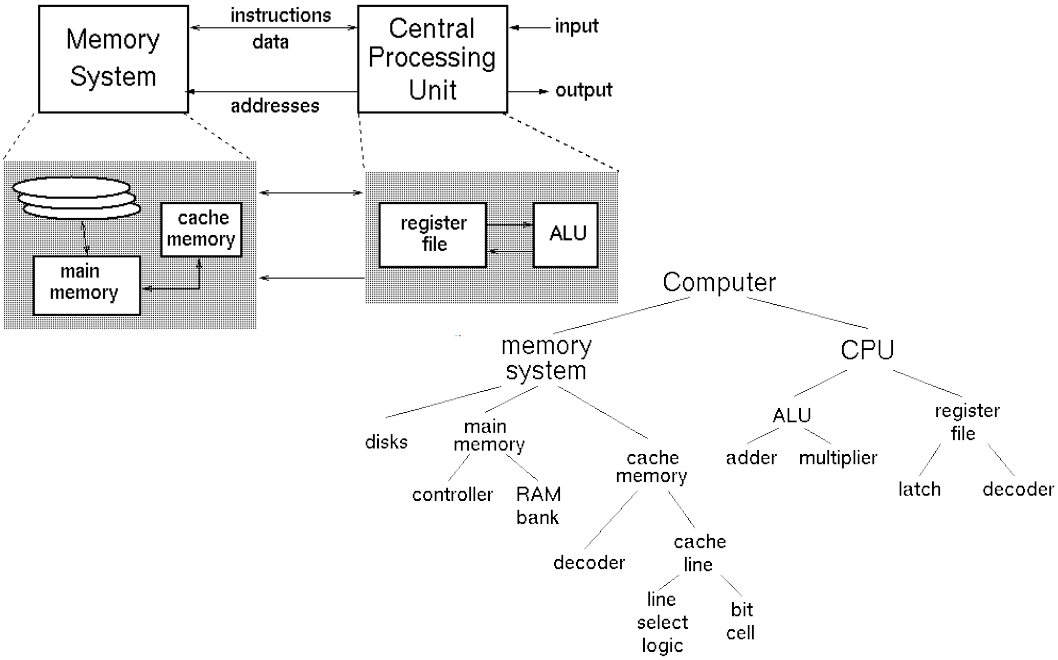


- You can improve on one at the expense of worsening one or both of the others.
- These tradeoffs exist at every level in the system design - every sub-piece and component.
- Design Specification -
  - Functional Description.
  - Performance, cost, power constraints.
- As a designer you must make the tradeoffs necessary to achieve the function within the constraints.

Design Space & Optimality

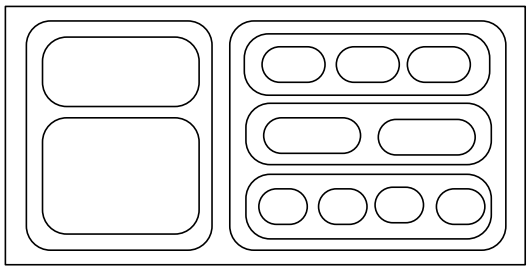


Hierarchy & Design Representation



Hierarchy in Designs

- Helps control complexity -
  - by hiding details and reducing the total number of things to handle at any time.
- Modulalizes the design -
  - divide and conquer
  - simplifies implementation and debugging



- Top-Down Design
  - Starts at the top (root) and works down by successive refinement.
- Bottom-up Design
  - Starts at the leaves & puts pieces together to build up the design.
- Which is better?
  - In practice both are needed & used.
    - Need top-down divide and conquer to handle the complexity.
    - Need bottom-up because in a well designed system, the structure is influence by what primitives are available.

Digital Design: What's it all about?

Given a functional description and performance, cost, & power constraints, come up with an implementation using a set of primitives.

- How do we learn how to do this?
  1. Learn about the primitives and how to use them.
  2. Learn about design representations.
  3. Learn formal methods and tools to manipulate the representations.
  4. Look at design examples.
  5. Use trial and error - CAD tools and prototyping. Practice!
- Digital design is in some ways more an art than a science. The creative spirit is critical in combining primitive elements & other components in new ways to achieve a desired function.
- However, unlike art, we have objective measures of a design:

Performance Cost Power