

EECS150 – Digital Design

Lecture 1 – Introduction

January 17, 2012

John Wawrzynek

Electrical Engineering and Computer Sciences
University of California, Berkeley

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



Teaching Staff

Professor John Wawrzynek

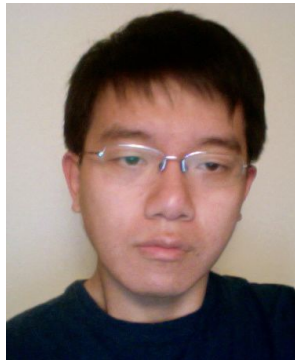
(Warznek)

631 Soda Hall

johnw@cs.berkeley.edu

Office Hours: Tu 1-2pm, & by appointment.

Shaoyi Cheng:
discussions sessions,
homework



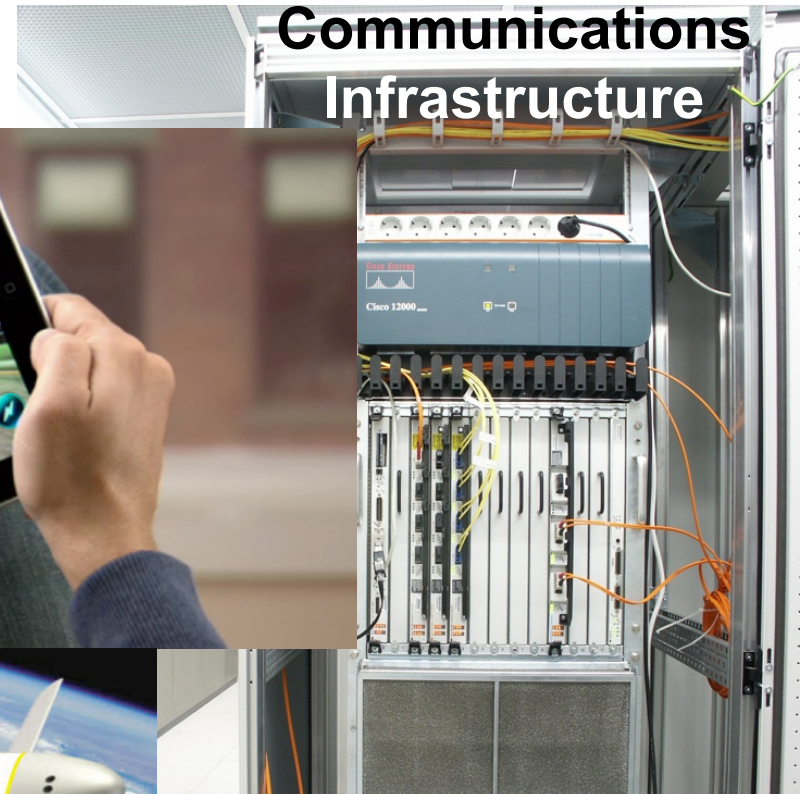
Daiwei Li:
labs, project



James Parker:
labs, project

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

Electronics all around us



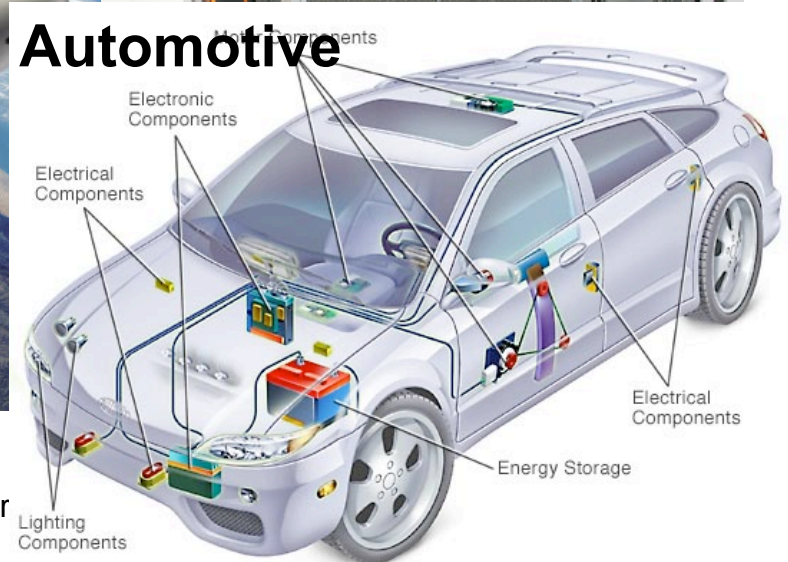
Consumer Products



Aerospace and Military



Automotive

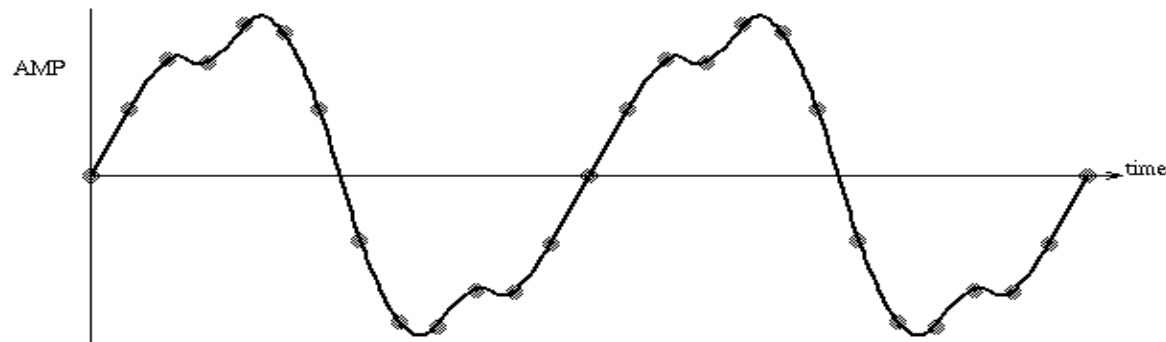


Course Content

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.**

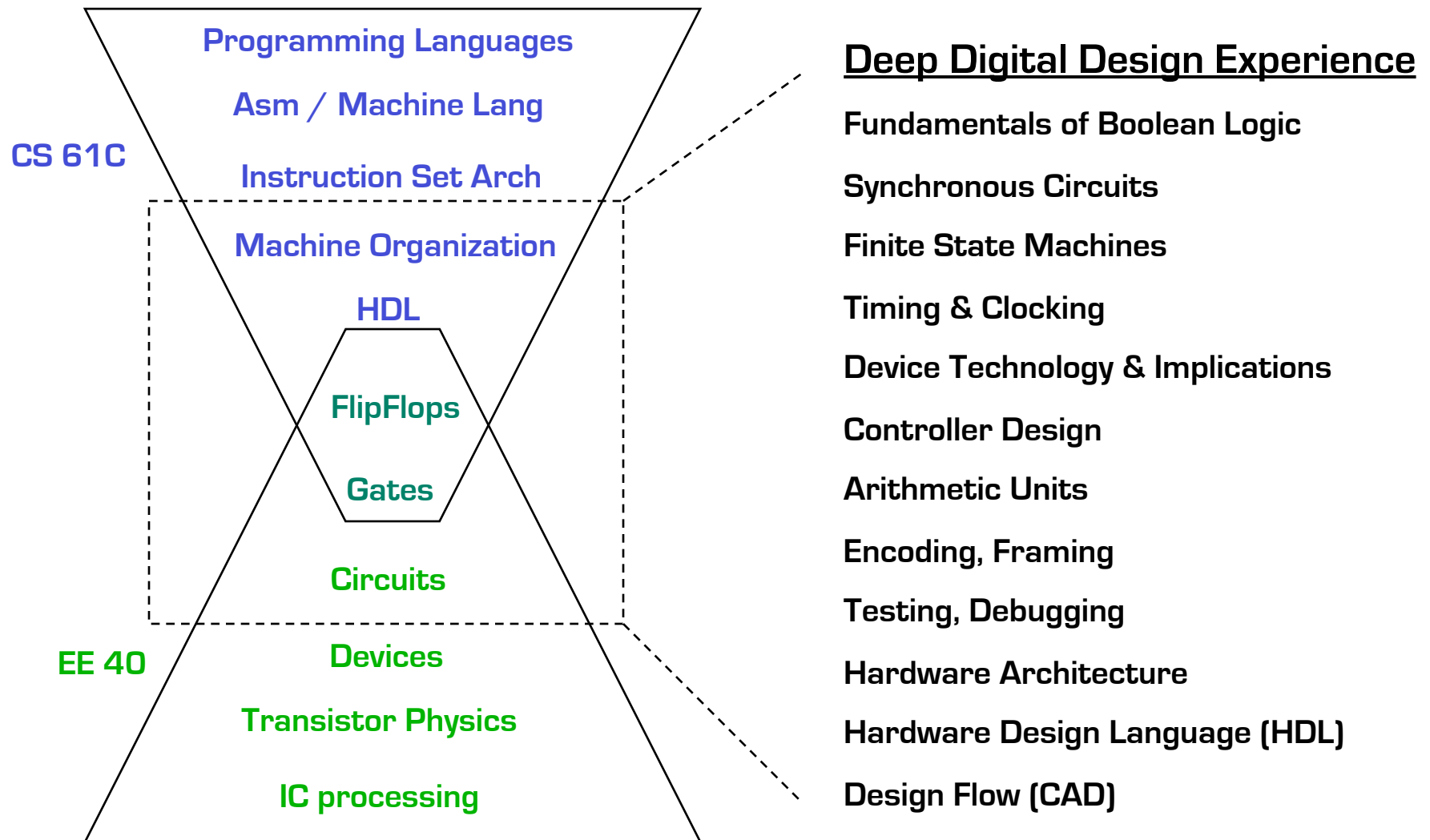
Course Content – Design Layers

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

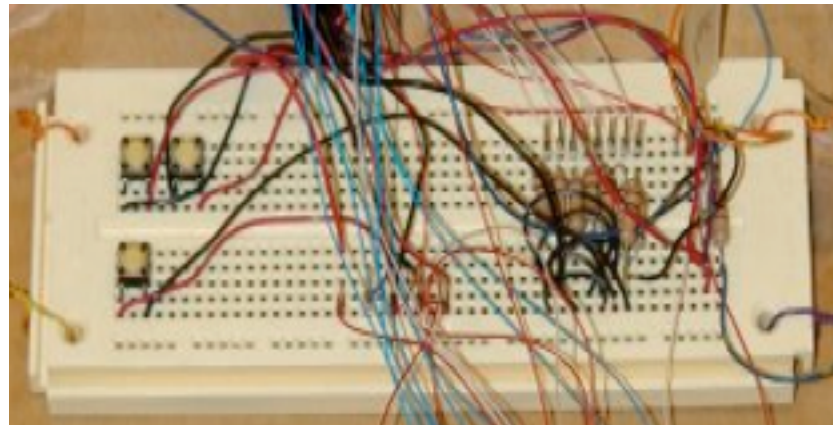
Not a course on transistor physics and transistor circuits. Although, we will look at these to better understand the primitive elements for digital circuits.

Course Content



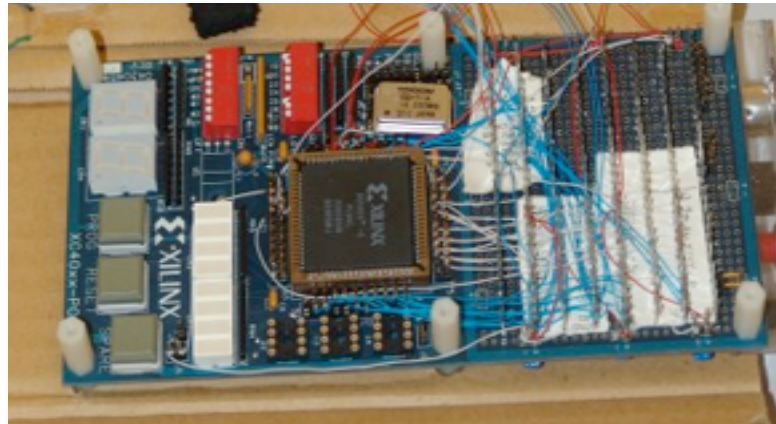
Course Evolution

- 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

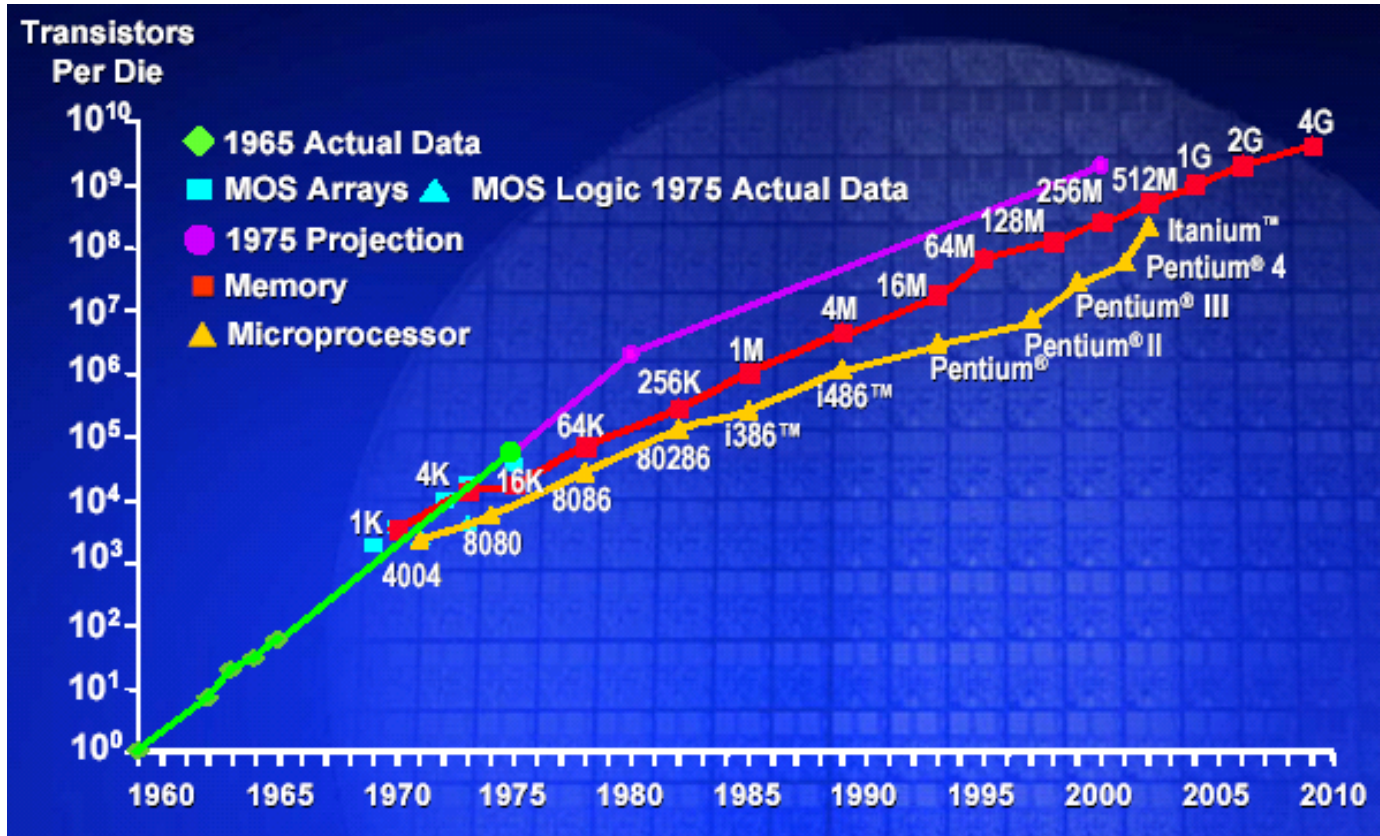


Course Evolution

- 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

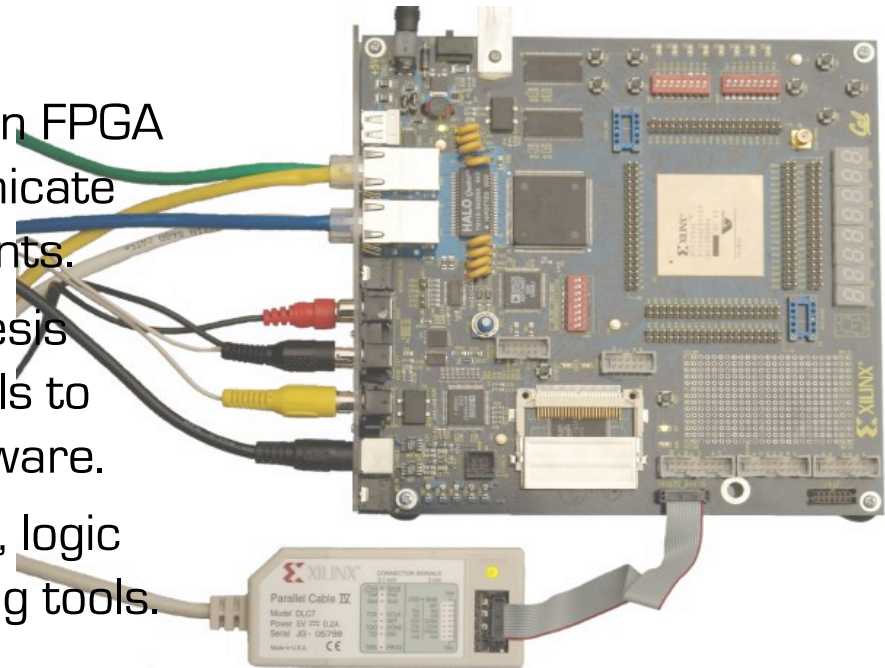


Course Evolution

- 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.

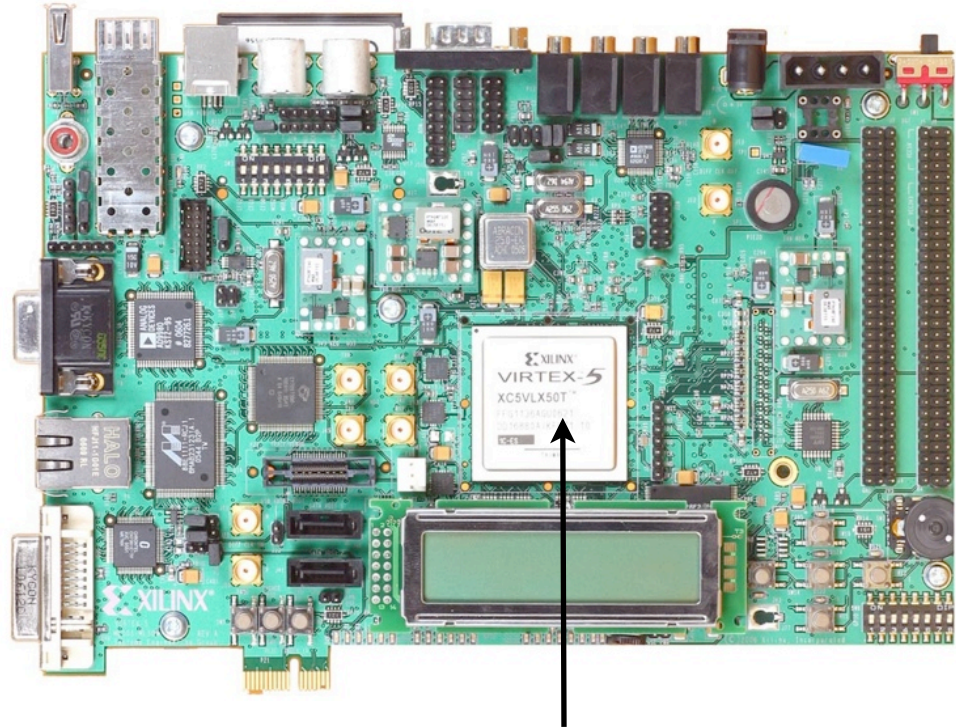


Calinx Board



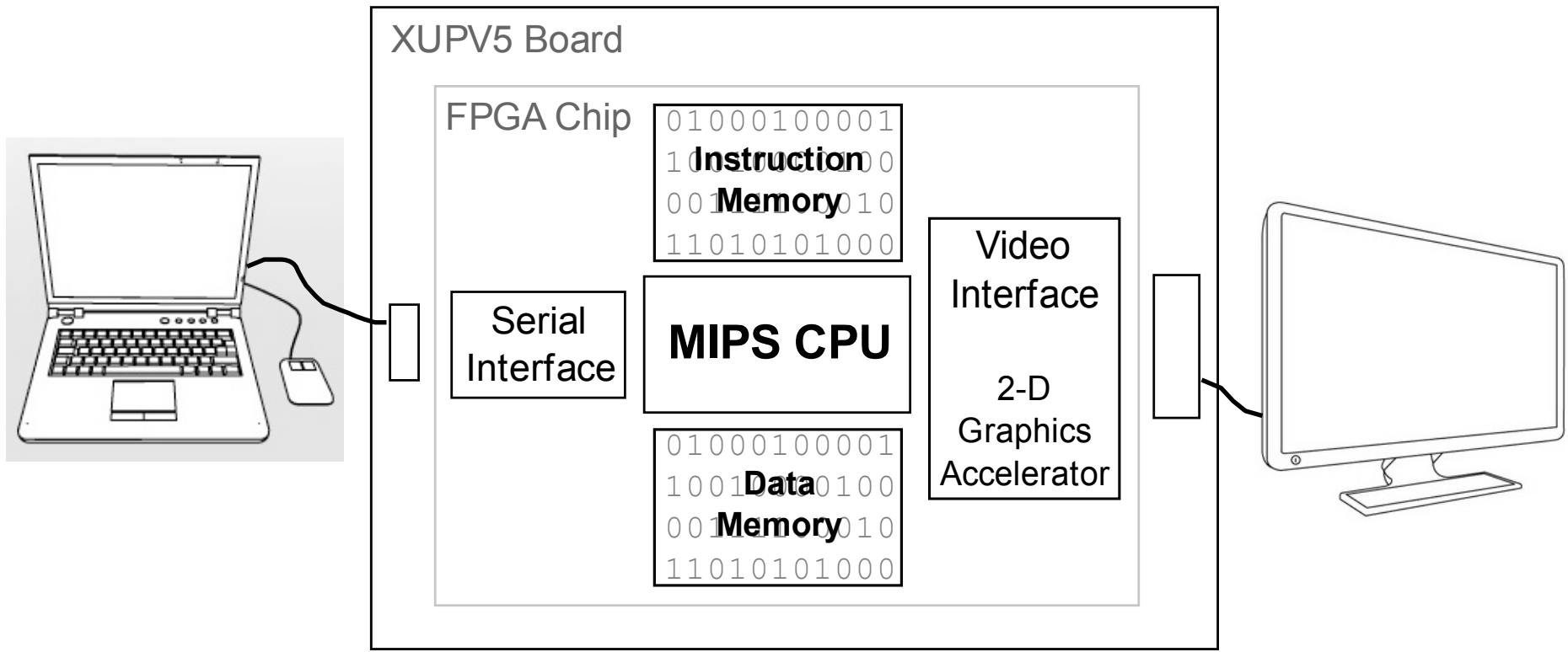
Course Evolution

- 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.



- State-of-the-art 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.

Administrivia

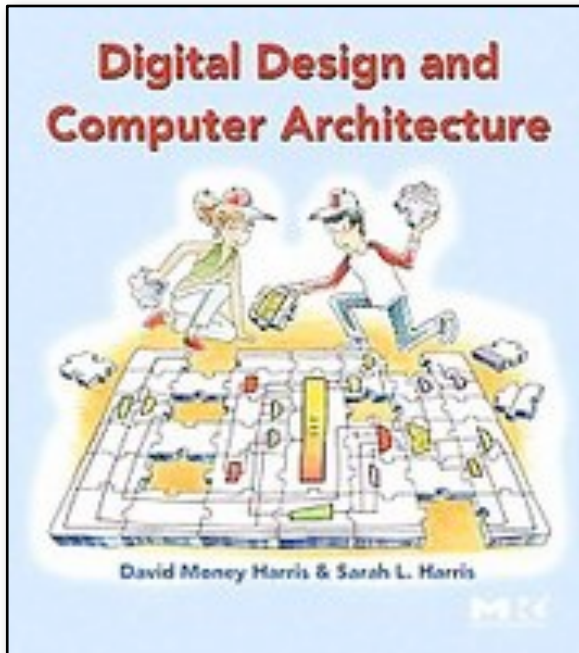
Enrollment

- 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>

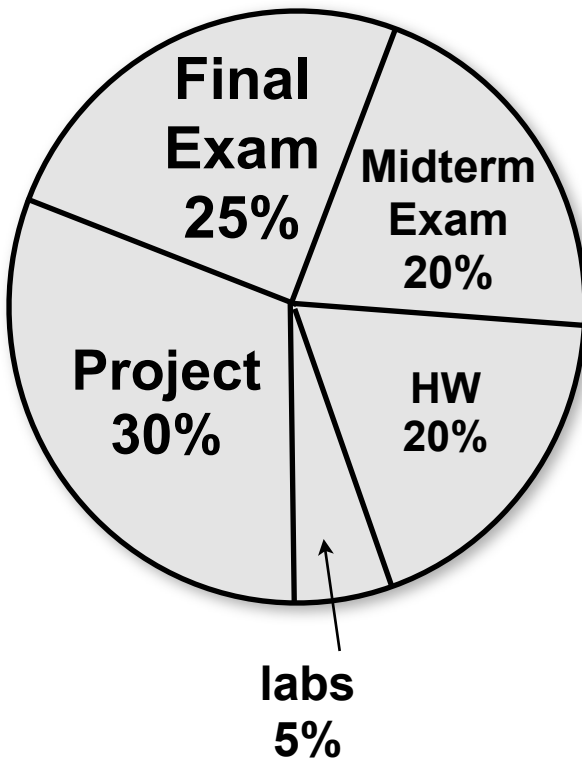
- Check the class webpage and newsgroup often!
- Updated posts will occur.

piazza For online Q/A.

<http://www.piazza.com/>

More info later.

Course Grading



- Comprehensive Exam held during Finals week: Monday May 7 11:30-2:30.
- Project critical part of the course - graded on timeliness, completeness and optimality. Lots more on this later.
- Evening midterm exam, Wed March 21, 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 due date in handouts.

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 questions.**
- 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.**

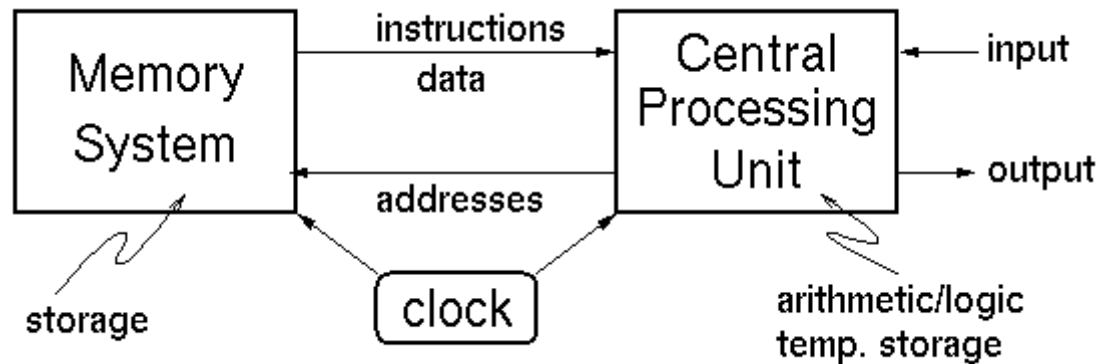
Cheating

- 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.**

A few basic concepts

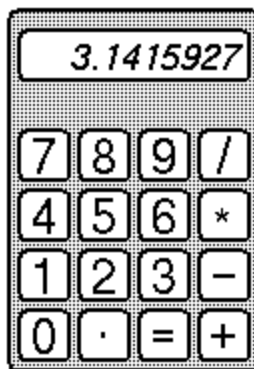
Example Digital Systems

- 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

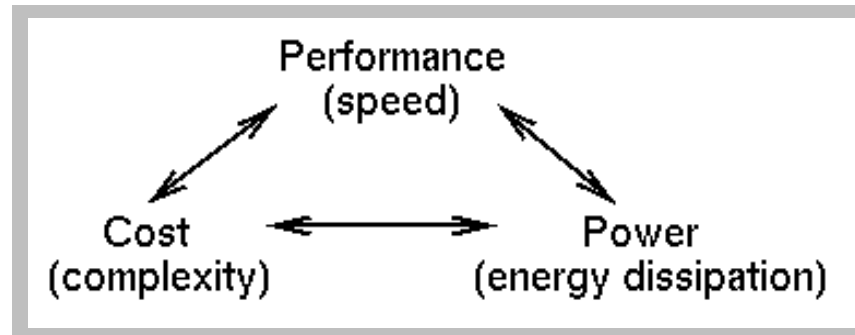
- Digital Watch



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

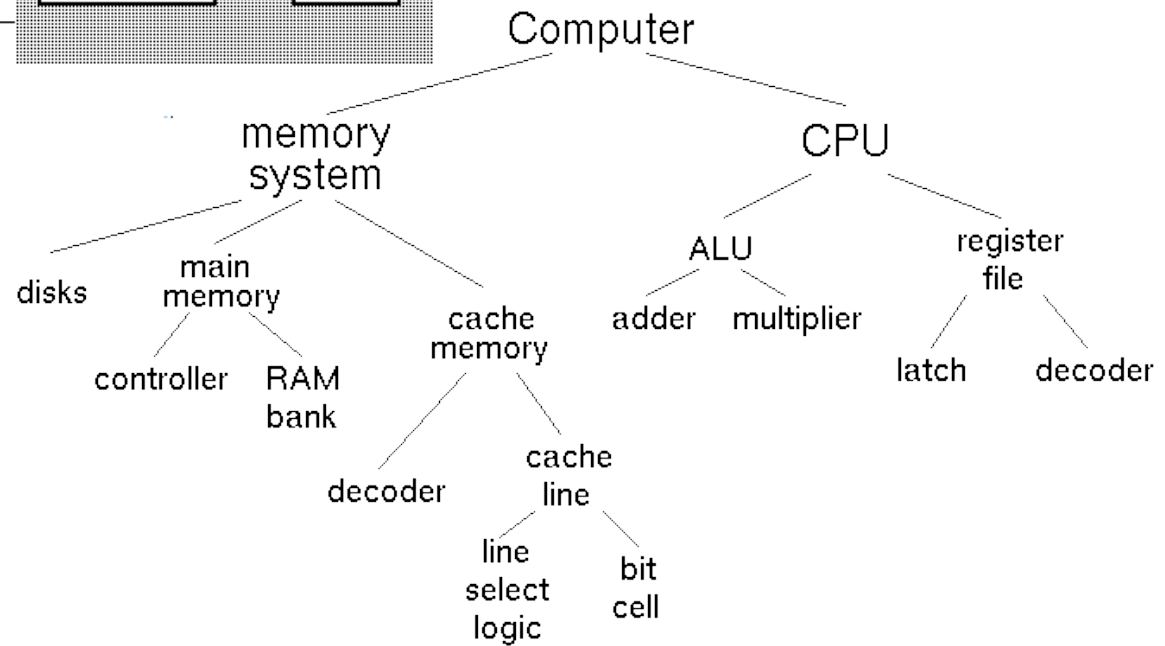
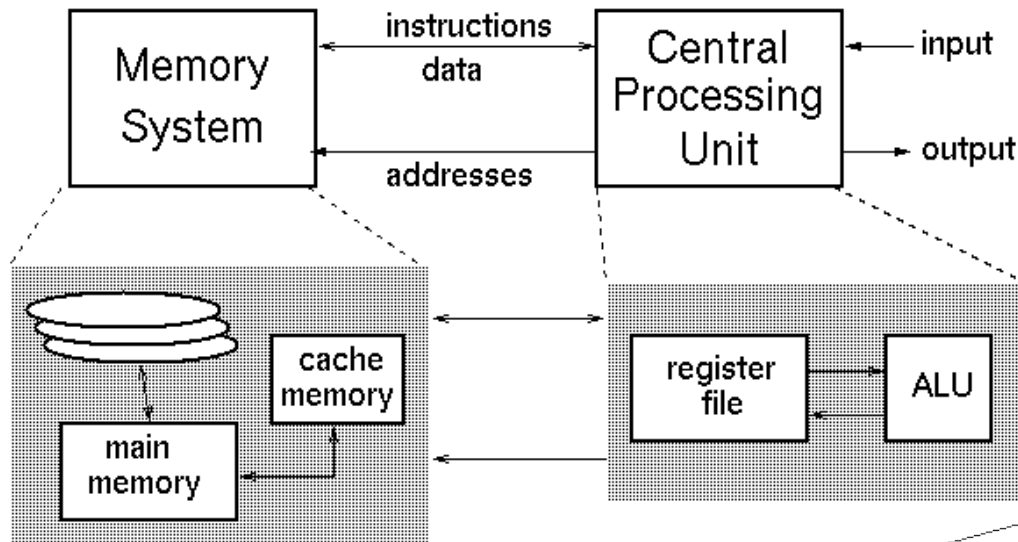
- Low power operation comes at the expense of:
 - lower speed
 - higher cost

Basic Design Tradeoffs



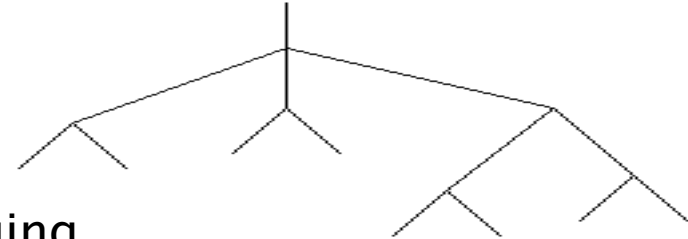
- You can improve on one at the expense of worsening one or both of the others.
- These tradeoffs exist at every level in the system design - every sub-piece and component.
- Design Specification -
 - Functional Description.
 - Performance, cost, power constraints.
- As a designer you must make the tradeoffs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

Hierarchy & Design Representation



Hierarchy in Designs

- Helps control complexity -
 - by hiding details and reducing the total number of things to handle at any time.
- Modularizes 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?

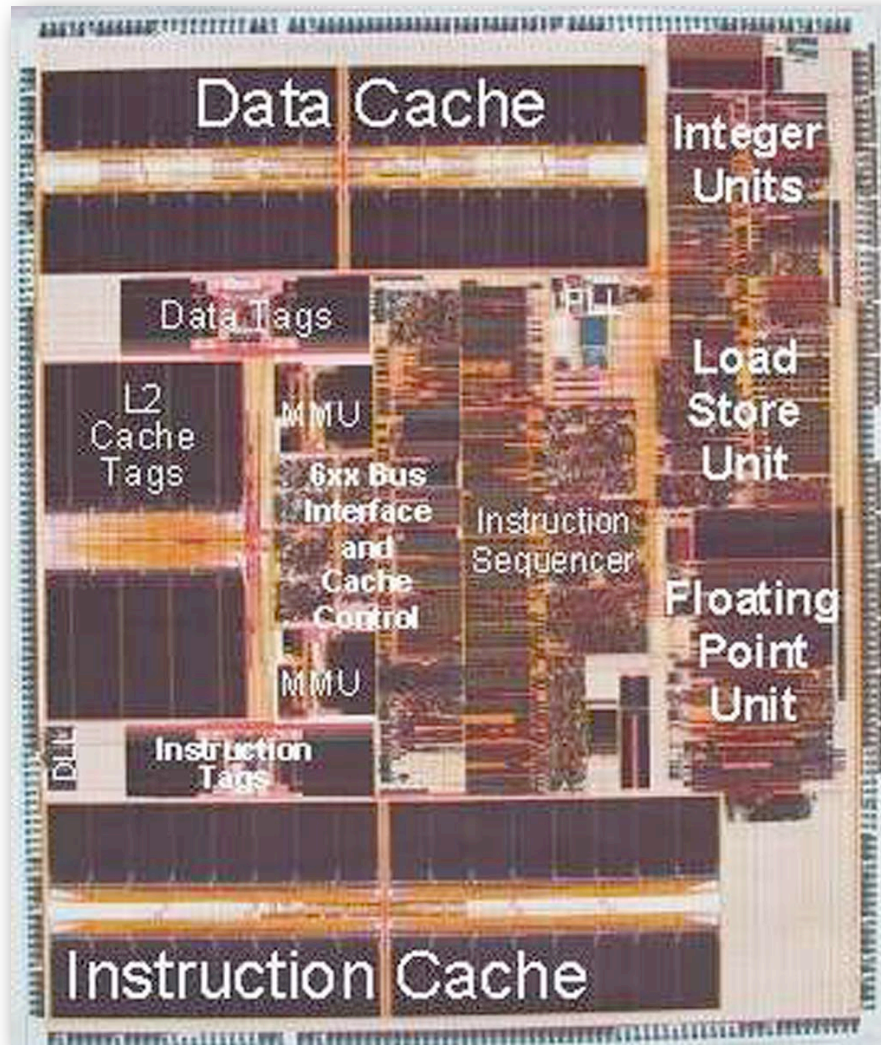
Given a functional description and performance, cost, & power constraints, come up with an implementation using a set of primitives.

- 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

Synchronous System Review from CS61C

Integrated Circuit Example



- PowerPC microprocessor micro-photograph
 - Superscalar (3 instructions/cycle)
 - 6 execution units (2 integer and 1 double precision IEEE floating point)
 - 32 KByte Instruction and Data L1 caches
 - Dual Memory Management Units (MMU)
 - External L2 Cache interface with integrated controller and cache tags.

Comprises only transistors and wires.

- ← Connections to outside world (ex. motherboard)
- Memory interface
 - Power (Vdd, GND)
 - Clock input

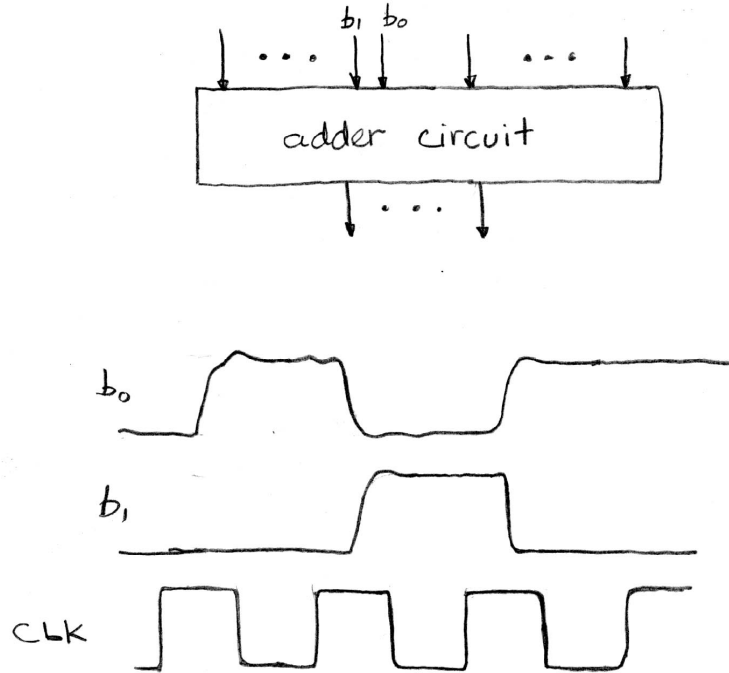
Clock Signal



A source of regularly occurring pulses used to measure the passage of time.

- Waveform diagram shows evolution of signal value (in voltage) over time.
- Usually comes from an off-chip crystal-controlled oscillator.
- One main clock per chip/system.
- Distributed throughout the chip/system.
- "Heartbeat" of the system. Controls the rate of computation by directly controlling all data transfers.

Data Signals



Random adder circuit at a random point in time:

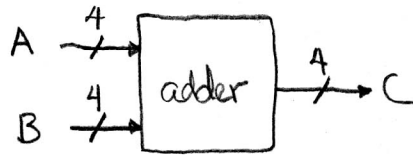
Observations:

- voltage ↑
time →
1. Most of the time, signals are in either low- or high-voltage position.
 2. When the signals are at the high- or low-voltage positions, they are not all the way to the voltage extremes (or they are past).
 3. Changes in the signals correspond to changes in clock signal (but don't change every cycle).

The facts:

1. Low-voltage represents binary 0 and high-voltage, binary 1.
2. Circuits are designed and built to be "restoring" and deviations from ideal voltages are ignored. Outputs close to ideal.
3. In synchronous systems, all changes follow clock edges.

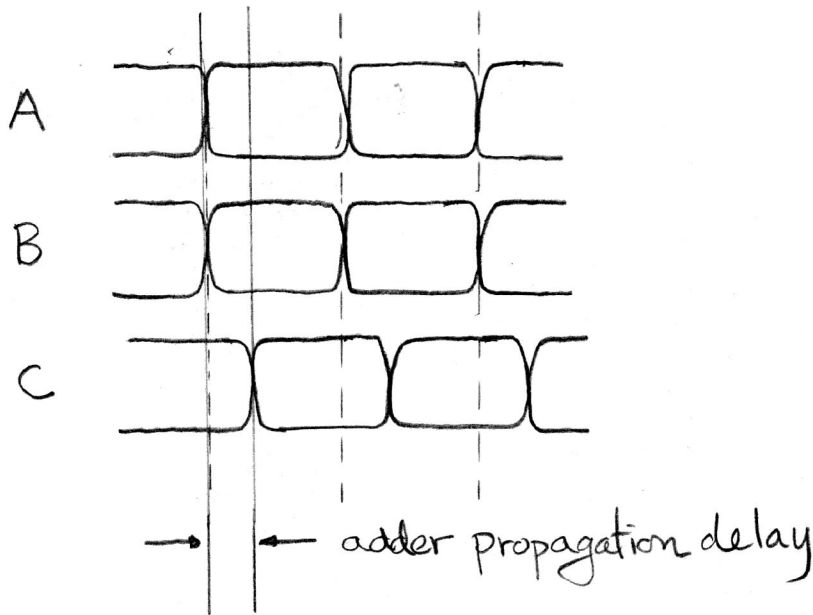
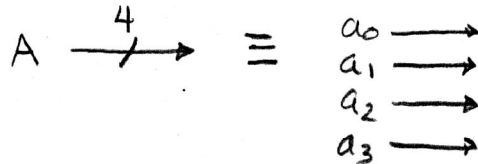
Circuit Delay



$$A = [a_3, a_2, a_1, a_0]$$

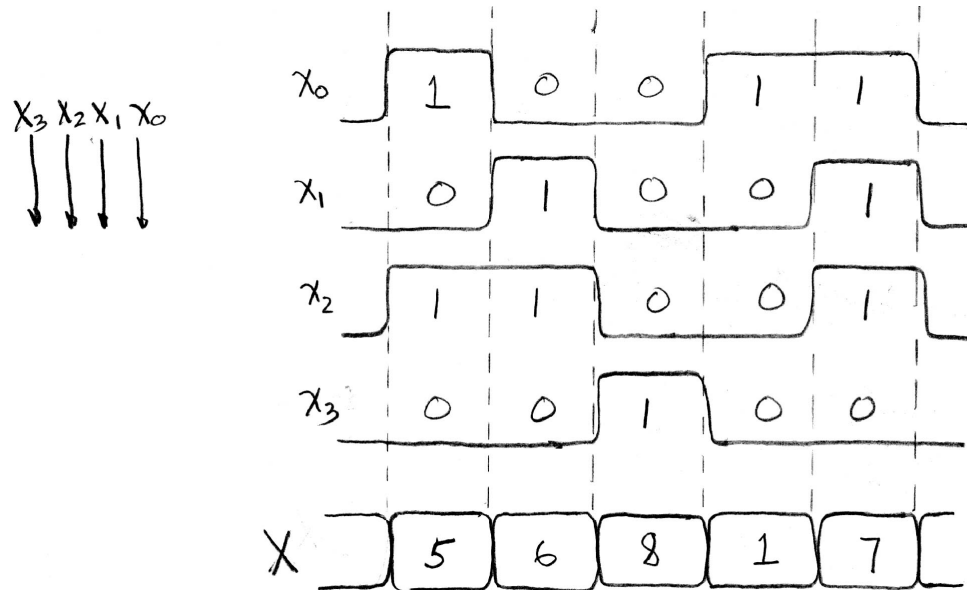
$$B = [b_3, b_2, b_1, b_0]$$

Digital circuits cannot produce outputs instantaneously.



- In general, the delay through a circuit is called the propagation delay. It measures the time from when inputs arrive until the outputs change.
- The delay amount is a function of many things. Some out of the control of the circuit designer:
 - Processing technology, the particular input values.
- And others under her control:
 - Circuit structure, physical layout parameters.

Bus Signals

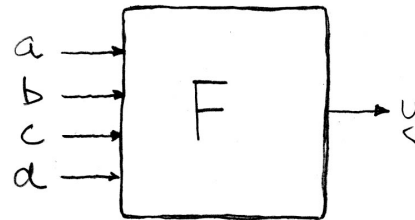


Signal wires grouped together often called a bus.

- X_0 is called the least significant bit (LSB)
- X_3 is called the most significant bit (MSB)
- Capital X represents the entire bus.
 - Here, hexadecimal digits are used to represent the values of all four wires.
 - The waveform for the bus depicts it as being simultaneously high and low. (The hex digits give the bit values). The waveform just shows the timing.

Combinational Logic Blocks

- Example four-input function:

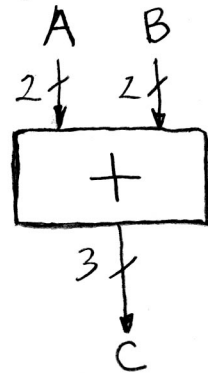


a	b	c	d	y
0	0	0	0	F(0,0,0,0)
0	0	0	1	F(0,0,0,1)
0	0	1	0	F(0,0,1,0)
0	0	1	1	F(0,0,1,1)
0	1	0	0	F(0,1,0,0)
0	1	0	1	F(0,1,0,1)
0	1	1	0	F(0,1,1,0)
1	1	1	1	F(0,1,1,1)
1	0	0	0	F(1,0,0,0)
1	0	0	1	F(1,0,0,1)
1	0	1	0	F(1,0,1,0)
1	0	1	1	F(1,0,1,1)
1	1	0	0	F(1,1,0,0)
1	1	0	1	F(1,1,0,1)
1	1	1	0	F(1,1,1,0)
1	1	1	1	F(1,1,1,1)

- True-table representation of function. Output is explicitly specified for each input combination.
- In general, CL blocks have more than one output signal, in which case, the truth-table will have multiple output columns.

Example CL Block

- 2-bit adder. Takes two 2-bit integers and produces 3-bit result.

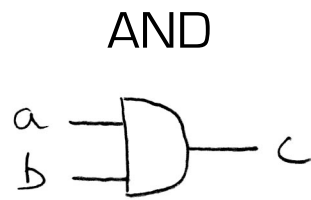


- Think about truth table for 32-bit adder. It's possible to write out, but it might take a while!

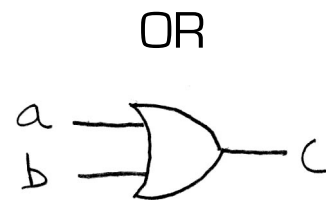
a1	a0	b1	b0	c2	c1	c0
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	0	0	1	0
0	0	1	1	0	1	1
0	1	0	0	0	0	1
0	1	0	1	0	1	0
0	1	1	0	0	1	1
0	1	1	1	1	0	0
1	0	0	0	0	1	0
1	0	0	1	0	1	1
1	0	1	0	1	0	0
1	0	1	1	1	0	1
1	1	0	0	0	1	1
1	1	0	1	1	0	0
1	1	1	0	1	0	1
1	1	1	1	1	1	0

Theorem: *Any* combinational logic function can be implemented as a networks of logic gates.

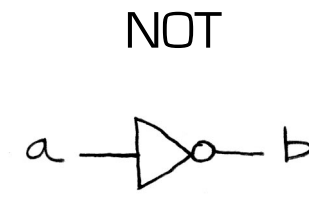
Logic "Gates"



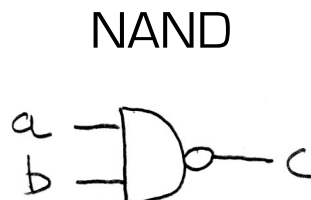
ab	c
00	0
01	0
10	0
11	1



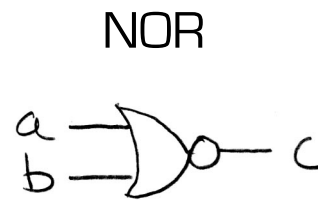
ab	c
00	0
01	1
10	1
11	1



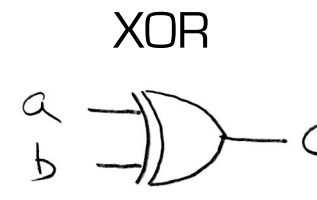
a	b
0	1
1	0



ab	c
00	1
01	1
10	1
11	0



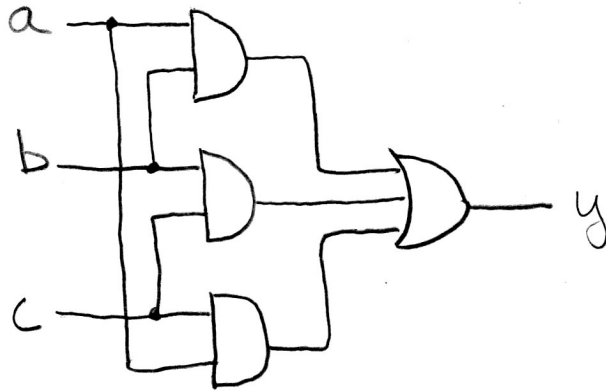
ab	c
00	1
01	0
10	0
11	0



ab	c
00	0
01	1
10	1
11	0

- Logic gates are often the primitive elements out of which combinational logic circuits are constructed.
 - In some technologies, there is a one-to-one correspondence between logic gate representations and actual circuits.
 - Other times, we use them just as another abstraction layer (FPGAs have no real logic gates).
- How about these gates with more than 2 inputs?
- Do we need all these types?

Example Logic Circuit

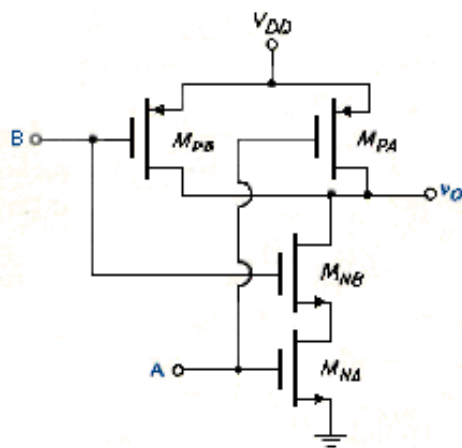


a	b	c	y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

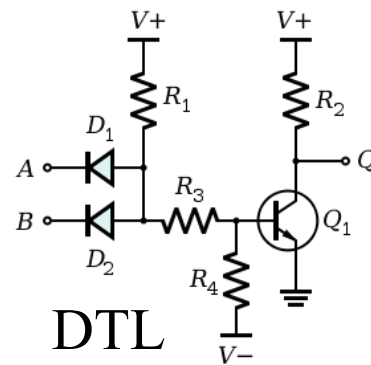
- How do we know that these two representations are equivalent?

Logic Gate Implementation

- Logic circuits have been built out of many different technologies. As we know, as long as we have a basic logic gate (AND or OR) and inversion we can build a complete logic family.



CMOS Gate



DTL

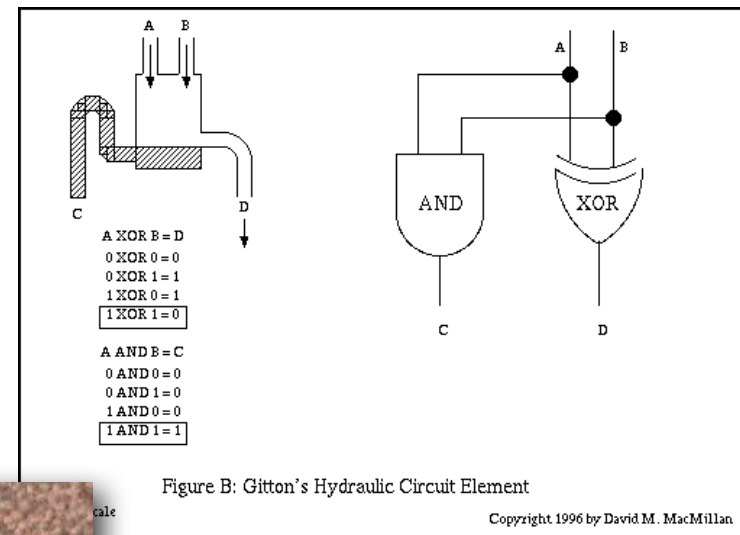
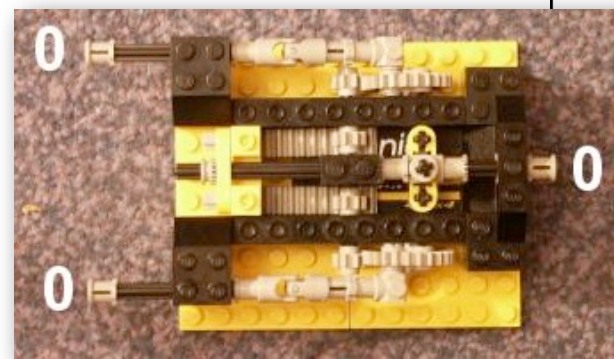


Figure B: Gitton's Hydraulic Circuit Element

Copyright 1996 by David M. MacMillan

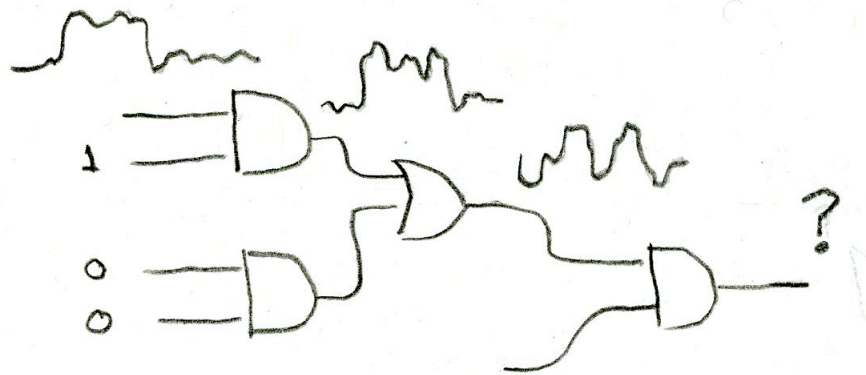
Hydraulic

Mechanical LEGO logic gates. A clockwise rotation represents a binary "one" while a counter-clockwise rotation represents a binary "zero."



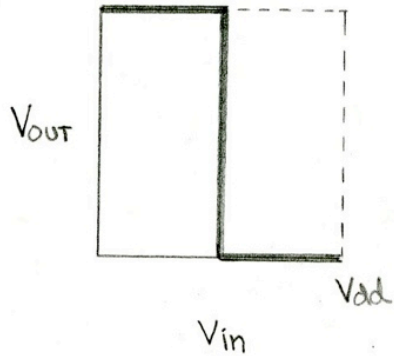
Restoration

- A necessary property of any successful technology for logic circuits is "Restoration".
- Circuits need:
 - to ignore noise and other non-idealities at their inputs, and
 - generate "cleaned-up" signals at their output.
- Otherwise, each stage would propagate input noise to their output and eventually noise and other non-idealities would accumulate and signal content would be lost.

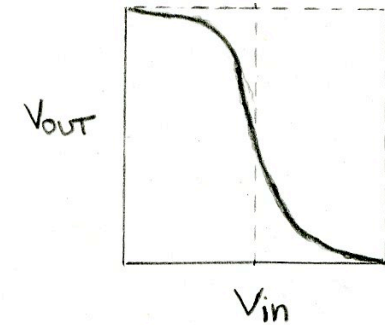
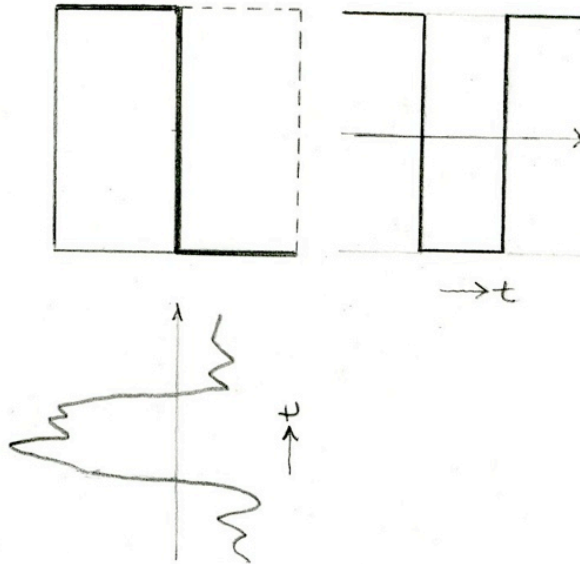


Inverter Example of Restoration

Example (look at 1-input gate, to keep it simple):



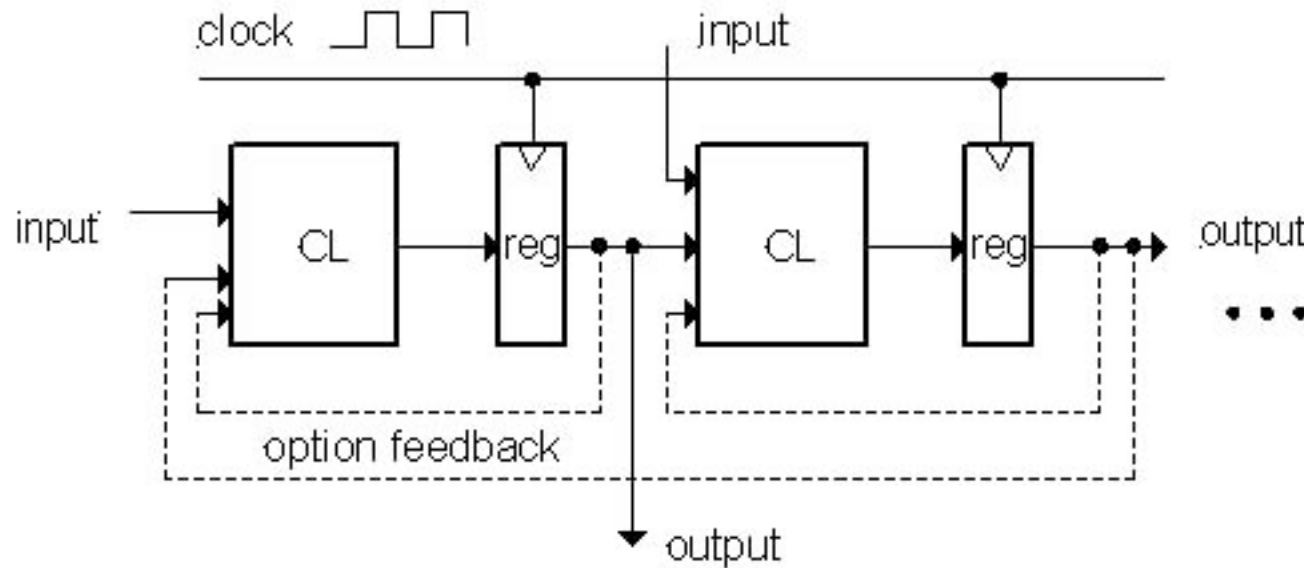
Idealize Inverter



Actual Inverter

- Inverter acts like a “non-linear” amplifier
- The non-linearity is critical to restoration
- Other logic gates act similarly with respect to input/output relationship.

General Model for Synchronous Systems



- All synchronous digital systems fit this model:
 - Collections of combinational logic blocks and state elements connected by signal wires. These form a directed graph with only two types of nodes (although the graph need not be bi-partite.)
 - Instead of simple registers, sometimes the state elements are large memory blocks.