EECS 150 - Components and Design Techniques for Digital Systems

Lec 01 – Introduction

8-28-07

David Culler
Electrical Engineering and Computer Sciences
University of California, Berkeley

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

Introductions – CS150 Staff

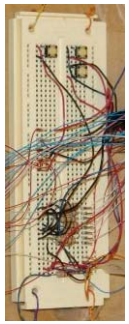
David E. Culler
culler@cs.berkeley.edu
http://www.eecs.berkeley.edu/~culler
627 Soda Hall, 643-7572
Office hours: Tue 3:30-5, Fr 9-11

Allen Lee
leeallen@berkeley.edu
Sarah Bird
slbird@eecs.berkeley.edu

Shah Bawany
shahbawany@gmail.com
Shauki Elassaad
shauki@eecs.berkeley.edu
Udam Singh Saini
usani08@berkeley.edu

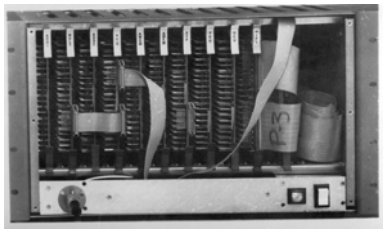
EECS 150 Project in my day

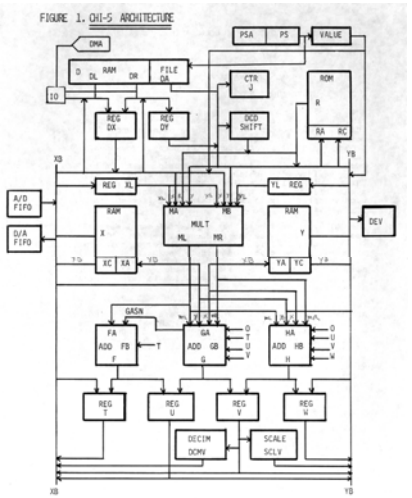
- Row of LEDs
- Bunch of TTL SSI chips
  - TTL cookbook
- Couple of switches for paddle
- Bread board
  - Ground plane would oscillate after wires signal punched through
- Oscilloscope and logic analyzer



My post-EECS150 summer job

CHI-5 16-bit Digital Speech Processor

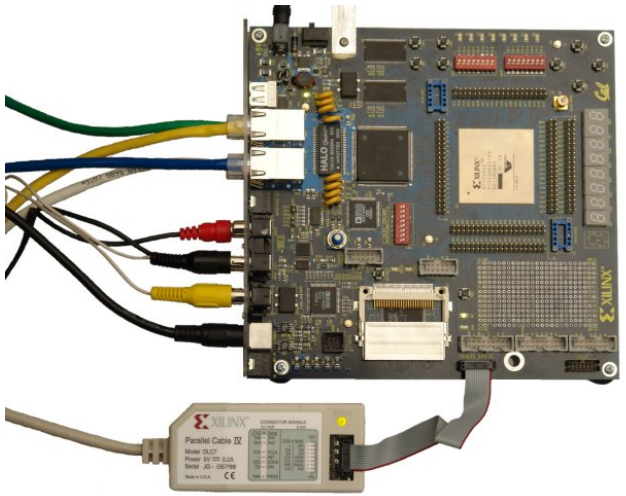




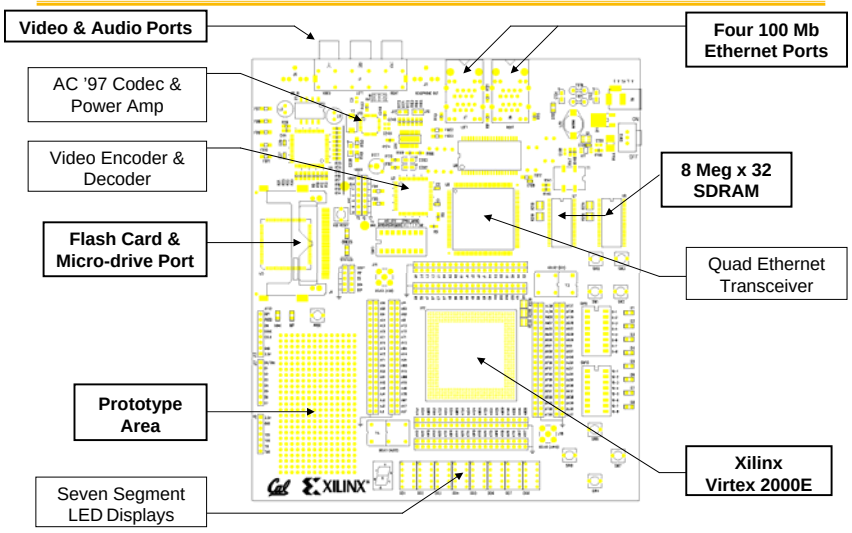
Digital Design in Your Life…

Your ideas here …

CaLinx2 – Your EECS150 …

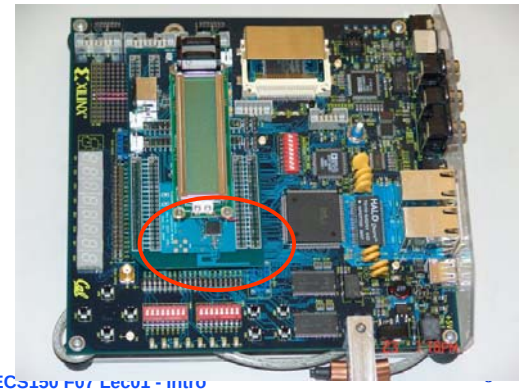


CaLinx II - Class Lab/Project Board

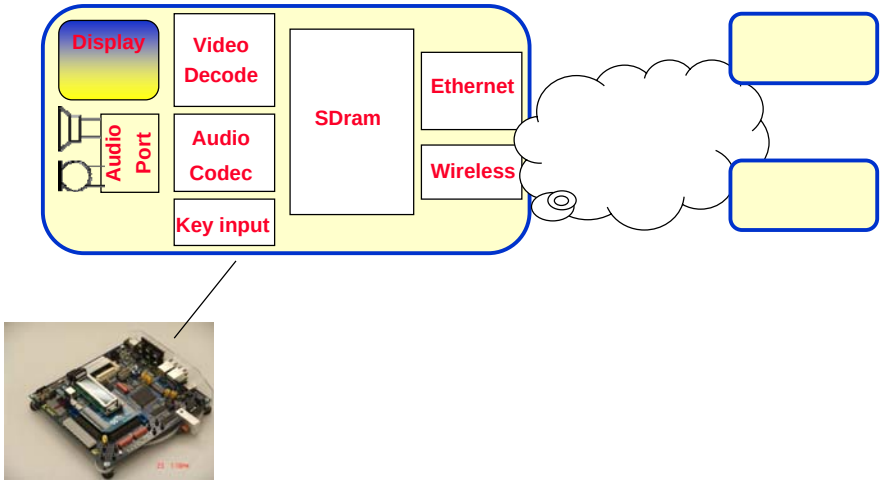


Wireless network + …

- **IEEE 802.15.4 Personal Area Network**
- **ADC channels**
- **Simple display**
- **Serial interface**



Fa07 Course Project – i50phone

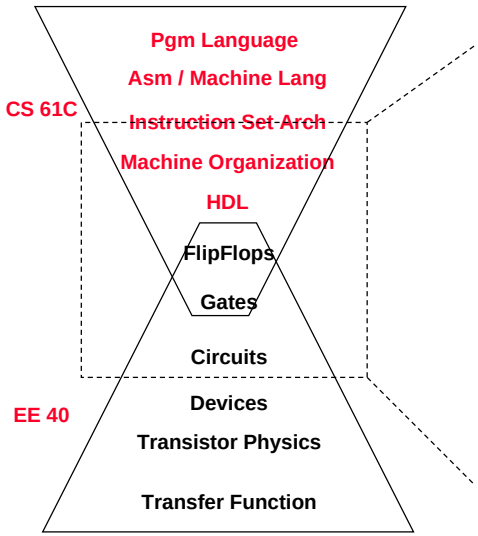


Outline

- Introductions
- Project Teaser
- Course Content
- Administrivia
  - Enrollment & Attendance
  - Course Structure & Grading
- A Few Basic Principles of Digital Design
- Summary

- Reading: Katz&Boriello, Ch 1

What is EECS150 about?



Deep Digital Design Experience

- Fundamentals of Boolean Logic
- Synchronous Circuits
- Finite State Machines
- Timing & Clocking
- Device Technology & Implications
- Controller Design
- Arithmetic Units
- Bus Design
- Encoding, Framing
- Testing, Debugging
- Hardware Architecture
- HDL, Design Flow (CAD)

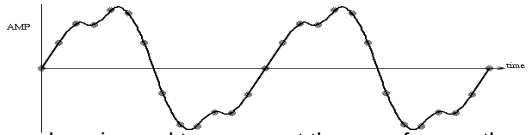
Course Content

Components and Design Techniques for Digital Systems

Synchronous Digital Hardware Systems

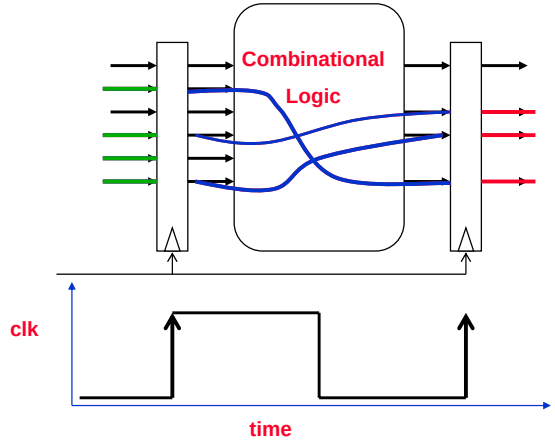
- **Synchronous: "Clocked" - all changes in the system are controlled by a global clock and happen at the same time (not asynchronous)**
- **Digital: All inputs/outputs and internal values (signals) take on discrete values (not analog).**

Example digital representation: acoustic waveform



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

What makes Digital Systems tick?



What determines the systems performance?

Course Content

- Hardware Architectures
- Arithmetic units, controllers
- Memory elements, logic gates, busses
- Transistor-level circuits
- 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.
- Not a course on computer architecture or the architecture of other systems. Although we will look at these as examples.

We Will Learn in EECS 150 …

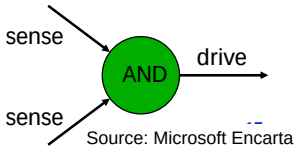
- Language of logic design
  - Logic optimization, state, timing, CAD tools
- Concept of state in digital systems
  - Analogous to variables and program counters in software systems
- Hardware system building
  - Datapath + control = digital systems
- Hardware system design methodology
  - Hardware description languages: *Verilog*
  - Tools to simulate design behavior: output = function (inputs)
  - Logic compilers synthesize hardware blocks of our designs
  - Mapping onto programmable hardware (code generation)
- Contrast with software design
  - Both map specifications to physical devices
  - Both must be flawless …

What is Logic Design?

- What is design?
  - Given problem spec, solve it with available components
  - While meeting quantitative (size, cost, power) and qualitative (beauty, elegance)
- What is logic design?
  - Choose digital logic components to perform specified control, data manipulation, or communication function and their interconnection
  - Which logic components to choose?
    Many implementation technologies (fixed-function components, *programmable devices*, individual transistors on a chip, etc.)
  - Design optimized/transformed to meet design constraints

What is Digital Hardware?

- **Devices that sense/control wires carrying digital values (physical quantity interpreted as "0" or "1")**
  - Digital logic: voltage < 0.8v is "0", > 2.0v is "1"
  - Pair of wires where "0"/"1" distinguished by which has higher voltage (differential)
  - Magnetic orientation signifies "0" or "1"
- **Primitive digital hardware devices**
  - Logic computation devices (sense and drive)
    - » Two wires both "1" - make another be "1" (AND)
    - » At least one of two wires "1" - make another be "1" (OR)
    - » A wire "1" - then make another be "0" (NOT)
  - Memory devices (store)
    - » Store a value
    - » Recall a value previously stored

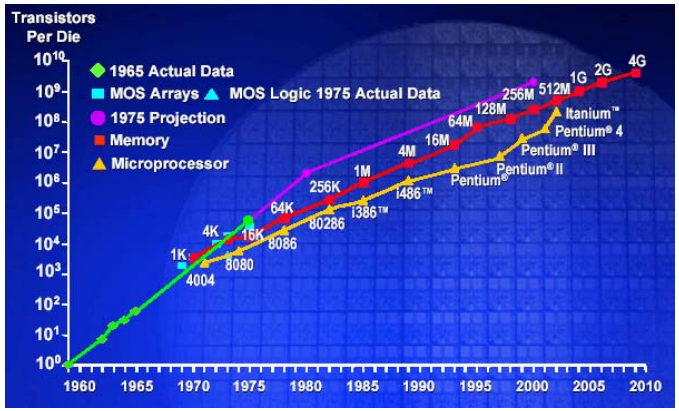


Source: Microsoft Encarta

Current State of Digital Design

- **Changes in industrial practice**
  - Larger designs
  - Shorter time to market
  - Cheaper products
- **Scale**
  - Pervasive use of computer-aided design tools over hand methods
  - Multiple levels of design representation
- **Time**
  - Emphasis on abstract design representations
  - Programmable rather than fixed function components
  - Automatic synthesis techniques
  - Importance of sound design methodologies
- **Cost**
  - Higher levels of integration
  - Use of simulation to debug designs
- **Power**
  - Critical at the high performance end
  - Critical at the portable end

Moore's Law – 2x stuff per 1-2 yr

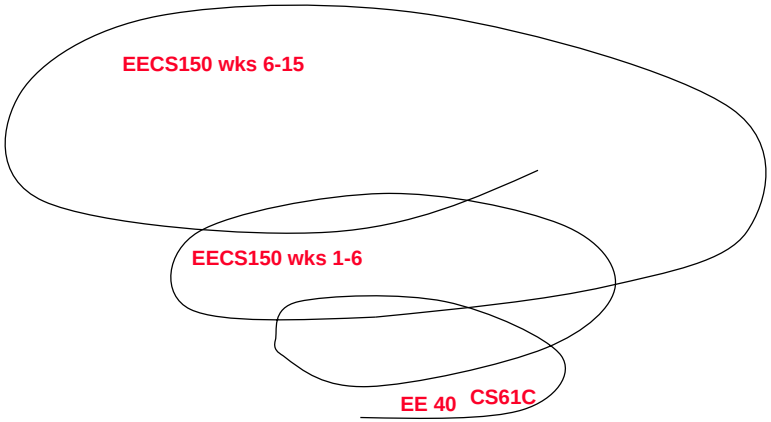


CS 150: Concepts/Skills/Abilities

- **Basics of logic design (concepts)**
- **Sound design methodologies (concepts)**
- **Modern specification methods (concepts)**
- **Familiarity with full set of CAD tools (skills)**
- **Appreciation for differences and similarities (abilities) in hardware and software design**
- **Hands-on experience with non-trivial design**

New ability: perform logic design with computer-aided design tools, validating that design via simulation, and mapping its implementation into programmable logic devices;
Appreciating the advantages/disadvantages hw vs. sw implementation

Traversing Digital Design



Administrative Issues

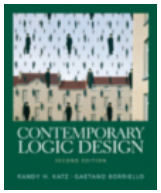
- See inst.eecs.berkeley.edu/~cs150 every day
- Lab lectures and discussions in 125 Cory
  - Dis 101 Th 3-4 Cancelled (0 enrolled)
- If you are enrolled OR plan to take the course you must attend your lab section this week.
- If you need to ADD the class, see me after class.
- Lab sections *will* be held this week.
- No discussion sections this week
- Lab lecture on Friday.

Attendance

- Attend regular lectures and ask questions.
  - *No webcast*
- Attend weekly "lab lecture" (Friday 2-3).
  - Probably webcast
- Attend your lab section. You must stick with the same lab section all semester.
  - 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. Attendance is optional, but useful.
- The instructor and TAs hold regular office hours (see class webpage). Please take advantage of this opportunity!

Course Materials

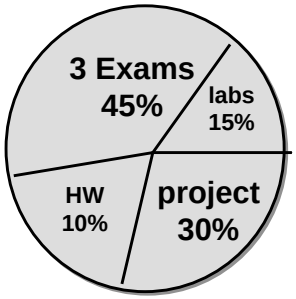
- **Textbook:** R. H. Katz, G. Borriello, *Contemporary Logic Design, 2nd Ed.*, Prentice Hall/Pearson Publishing.



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

  http://inst.eecs.berkeley.edu/~cs150
  - Check the class webpage and newsgroup often!
  - You are responsible for checking the class webpage at least once every 24 hours.

Course Grading



- Three exams of approximately equal weight
  - Possibly some quizzes
- Weekly homework based on reading and lectures.
  - Out by Th lecture, due Friday 2:00 next week
- Lab exercises for weeks 1-6, followed by project checkpoints and final checkoff.
- Labs and checkpoints due within the first 30 minutes of your next lab session.

Cheating

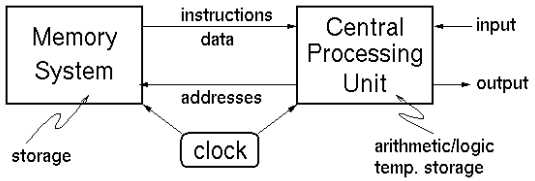
- **Any act that gives you unfair advantage at the expense of another classmate.**
- **Examples:**
  - **copying on exams, homework,**
  - **copying design data,**
  - **modifying class CAD software,**
  - **modifying or intentionally damaging lab equipment.**
- **If you ever have a question about what will be considered cheating, please ask.**
- **What should the penalty be?**
  - **Fail the course. Report to student affairs.**
  - **Fail the assignment / exam / project. (first time) Report.**
  - **Fail the disputed entity.**
- **Key is time management. Avoid desperation.**

Lecture format

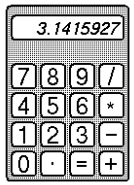
- **Outline**
- **Quick review of key points from previous time**
- **Main Topic**
- **Administrative issues & Break**
- **Additional depth or additional topic**
- **Summary of key points**

Example Digital Systems

- **Computer**



  - **Usually design to maximize performance. "Optimized for speed"**
- **Calculator**



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

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

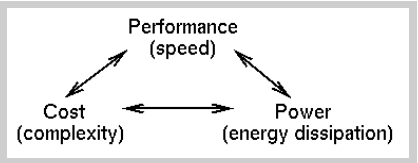
Example Digital Systems

- **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

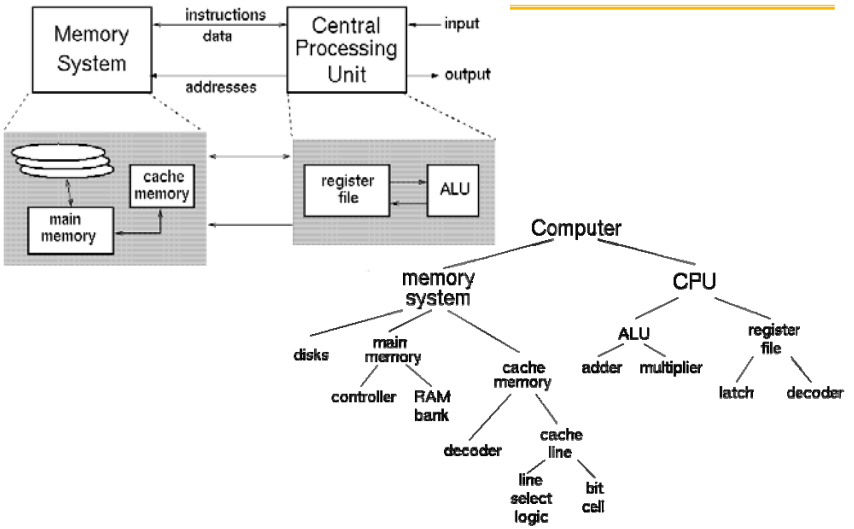


- **You can usually improve on one at the expense of one or both of the others.**
- **These tradeoffs exist at every level in the system design - every sub-piece and component.**
- **Design Specification -**
  - **Functional Description.**
  - **Performance, cost, power constraints.**
- **As a designer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

To design is to represent…

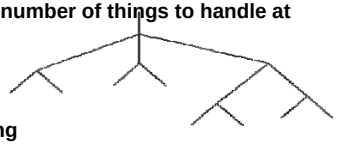
- **How is design and engineering different from craftsmanship?**
- **What is the result of the design process?**

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

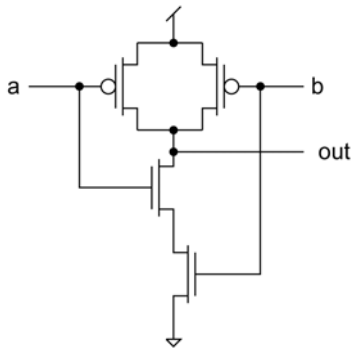


- Top-Down Design
  - Starts at the top (root) and works down by successive refinement.
- Bottom-up Design
  - Starts at the leaves & puts pieces together to build up the design.
- Which is better?
  - In practice both are needed & used.
    - » Need top-down divide and conquer to handle the complexity.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

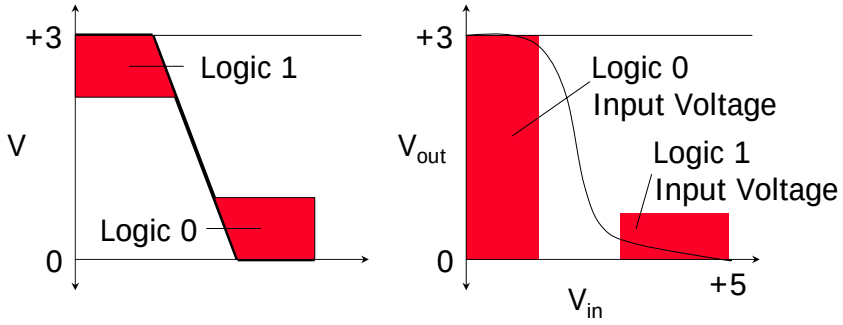
Building Blocks of Digital Logic

Interactive Background Quiz

1. What is the truth table for logic implemented by this circuit?
2. What is the name of this logic gate?
3. What is its symbol?
4. How would you make an inverter out of it?
5. Which is faster, the rise time or the fall time?



Logical Values



- **Threshold**
  - Logical 1 (true) : V > Vdd –V th
  - Logical 0 (false) : V < Vth

in	out
F	T
T	F

not(out, in)

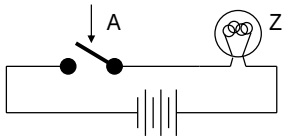
Mapping from physical world to binary world

Technology	State "0"	State "1"
Relay logic	Circuit Open	Circuit Closed
CMOS logic	0.0-1.0 volts	2.0-3.0 volts
Transistor transistor logic (TTL)	0.0-0.8 volts	2.0-5.0 volts
Fiber Optics	Light off	Light on
Dynamic RAM	Discharged capacitor	Charged capacitor
Nonvolatile memory (erasable)	Trapped electrons	No trapped electrons
Programmable ROM	Fuse blown	Fuse intact
Bubble memory	No magnetic bubble	Bubble present
Magnetic disk	No flux reversal	Flux reversal
Compact disc	No pit	Pit

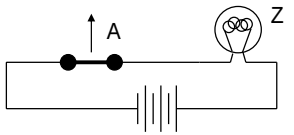
Sense the logical value, manipulate in s systematic fashion.

Switches: basic element of physical implementations

- **Implementing a simple circuit (arrow shows action if wire changes to "1"):**



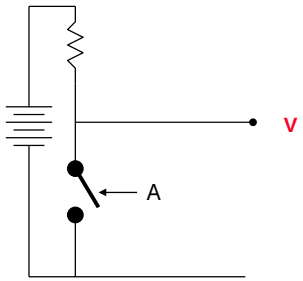
close switch (if A is "1" or asserted) and turn on light bulb (Z)



open switch (if A is "0" or unasserted) and turn off light bulb (Z)

Z ≡ A

What is this?



- **1-bit Analog to Digital Converter**

Boolean Algebra/Logic Circuits

- **Why are they called "logic circuits"?**
- **Logic: The study of the principles of reasoning.**
- **The 19th Century Mathematician, George Boole, developed a math. system (algebra) involving logic, Boolean Algebra.**
- **His variables took on TRUE, FALSE**
- **Later Claude Shannon (father of information theory) showed (in his Master's thesis!) how to map Boolean Algebra to digital circuits:**
- **Primitive functions of Boolean Algebra:**

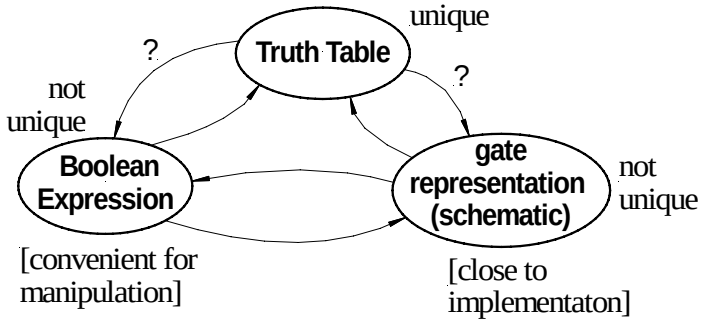
a b	AND
0 0	0
0 1	0
1 0	0
1 1	1

a b	OR
0 0	0
0 1	1
1 0	1
1 1	1

a	NOT
0	0
1	0

Relationship Among Representations

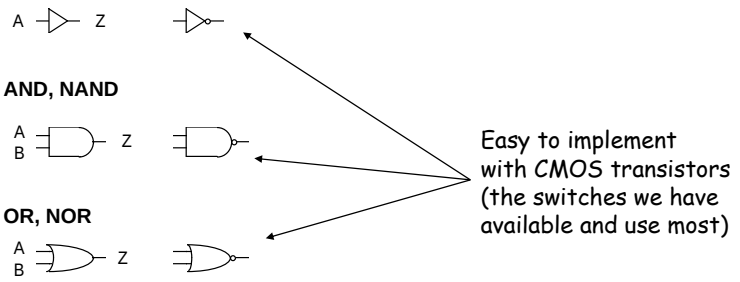
* **Theorem: Any Boolean function that can be expressed as a truth table can be written as an expression in Boolean Algebra using AND, OR, NOT.**



How do we convert from one to the other?

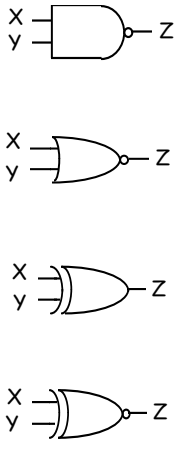
Combinational Logic Symbols

- **Common combinational logic systems have standard symbols called logic gates**
  - **Buffer, NOT**
  - **AND, NAND**
  - **OR, NOR**



more Boolean Expressions to Logic Gates

- **NAND**
- **NOR**
- **XOR**
  X ⊕ Y
- **XNOR**
  X = Y



X	Y	Z
0	0	1
0	1	1
1	0	1
1	1	0

X	Y	Z
0	0	1
0	1	0
1	0	0
1	1	0

X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	0

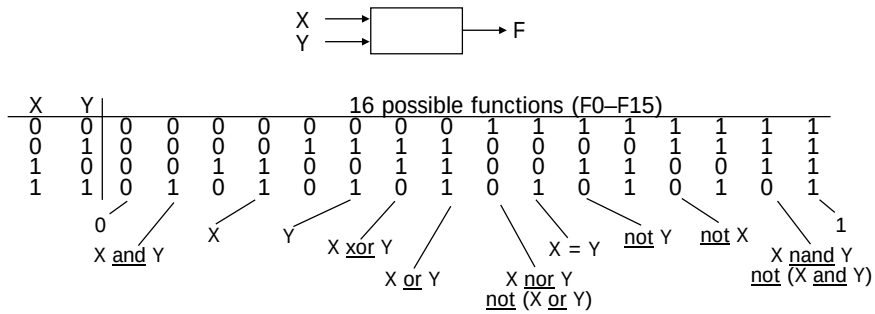
X	Y	Z
0	0	1
0	1	0
1	0	0
1	1	1

X xor Y = X Y' + X' Y
X or Y but not both
("inequality", "difference")

X xnor Y = X Y + X' Y'
X and Y are the same
("equality", "coincidence")

Possible Logic Functions of Two Variables

- **16 possible functions of 2 input variables:**
  - **2**(2**n) functions of n inputs**



Logic Functions and Boolean Algebra

- **Any logic function that can be expressed as a truth table can be written as an expression in Boolean algebra using the operators: ', +, and •**

X	Y	X • Y
0	0	0
0	1	0
1	0	0
1	1	1

X	Y	X'	X' • Y
0	0	1	0
0	1	1	1
1	0	0	0
1	1	0	0

X	Y	X'	Y'	X • Y	X' • Y'	(X • Y) + (X' • Y')
0	0	1	1	0	1	1
0	1	1	0	0	0	0
1	0	0	1	0	0	0
1	1	0	0	1	0	1

(X • Y) + (X' • Y') ≡ X = Y

Boolean expression that is true when the variables X and Y have the same value and false, otherwise

X, Y are Boolean algebra variables

Minimal set of functions

- **Implement any logic functions from NOT, NOR, and NAND?**
  - **For example, implementing X and Y is the same as implementing not (X nand Y)**
- **Do it with only NOR or only NAND**
  - **NOT is just a NAND or a NOR with both inputs tied together**

X	Y	X nor Y
0	0	1
1	1	0

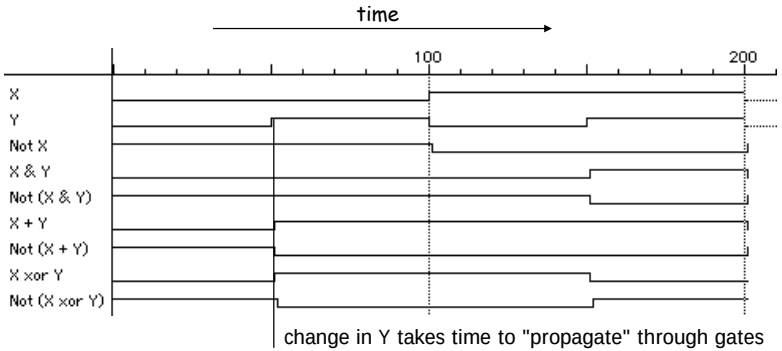
X	Y	X nand Y
0	0	1
1	1	0

  - **and NAND and NOR are "duals", i.e., easy to implement one using the other**

X nand Y ≡ not ((not X) nor (not Y))
X nor Y ≡ not ((not X) nand (not Y))

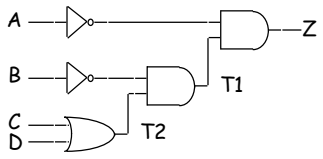
Waveform View of Logic Functions

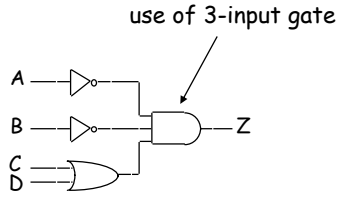
- **Just a sideways truth table**
  - **But note how edges don't line up exactly**
  - **It takes time for a gate to switch its output!**



From Boolean Expressions to Logic Gates

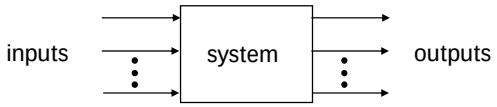
- **More than one way to map expressions to gates**
  - **e.g., $Z = A' \cdot B' \cdot (C + D) = (A' \cdot (\overbrace{B' \cdot \overbrace{(C + D)}^{T1}}^{T2}))$**





Combinational vs. Sequential Digital Circuits

- Simple model of a digital system is a unit with inputs and outputs:



- Combinational means "memory-less"
  - Digital circuit is combinational if its output values only depend on its inputs

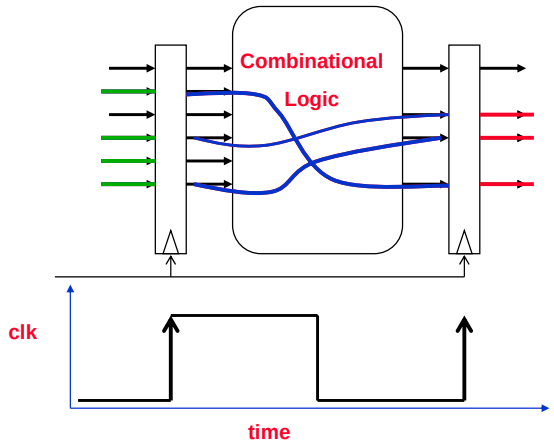
Sequential Logic

- Sequential systems
  - Exhibit behaviors (output values) that depend on current as well as previous inputs
- Time response of real circuits are sequential
  - Outputs do not change instantaneously after an input change
  - Why not, and why is it then sequential?
- Fundamental abstraction of digital design is to reason (mostly) about steady-state behaviors
  - Examine outputs only after sufficient time has elapsed for the system to make its required changes and settle down

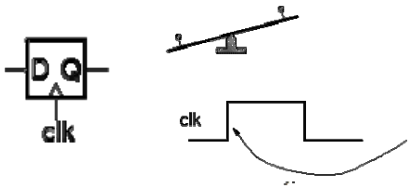
Synchronous sequential digital systems

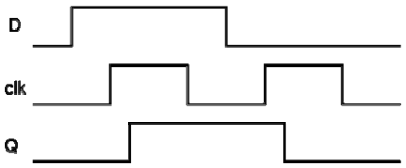
- Combinational circuit outputs depend *only* on current inputs
  - After sufficient time has elapsed
- Sequential circuits have *memory*
  - Even after waiting for transient activity to finish
- Steady-state abstraction: most designers use it when constructing sequential circuits:
  - Memory of system is its state
  - Changes in system state only allowed at specific times controlled by an external periodic signal (the *clock*)
  - Clock period is elapsed time between state changes sufficiently long so that system reaches steady-state before next state change at end of period

Recall: What makes Digital Systems tick?

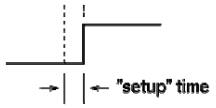


D-type edge-triggered flip-flop





- The edge of the clock is used to *sample* the "D" input & send it to "Q" (positive edge triggering).
  - At all other times the output Q is independent of the input D (just stores previously sampled value).
  - The input must be stable for a short time before the clock edge.



Summary: Digital Design

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 generate them.**
  2. **Learn about design representation.**
  3. **Learn formal methods to optimally manipulate the representations.**
  4. **Look at design examples.**
  5. **Use trial and error - CAD tools and prototyping.**
- ***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*