Introduction to Computer Engineering – EECS 203

http://ziyang.eecs.northwestern.edu/~dickrp/eecs203/

Instructor:	Robert Dick	TA:	Neal Oza
Office:	L477 Tech	Email:	nealoza@u.northwestern.edu
Email:	dickrp@northwestern.edu		
Phone:	847–467–2298	TT:	David Bild
		Email:	d-bild@northwestern.edu

NORTHWESTERN
UNIVERSITY

Outline

Brief course overview

- Hardware design
- Low-level programming

What's your major?

Computer geek already?

- Good!
- You'll still probably see a lot of new things in this course
- Go ahead and ask questions that push beyond the basic material
- If you want to go beyond the normal labs, I'll be happy to make suggestions
- EECS 203 should lay the foundations for logic design and understanding the connections between electrons and software

Not a computer geek yet? Good!

- You're going to be working with computers in almost any field
- Understanding how they work at the lowest levels and knowing how to build them will put you ahead your peers
- If you're not a computer geek yet, sit in the front of the classroom and ask questions!
    - It's the best way to keep the course's pace sane

Backgrounds

- Different backgrounds
- EECS 203 can be a hard course
- However, if you work hard, I'm totally confident that you will learn how to build useful computers
  - TAs and I will help
- In the past, many Materials Science, BME, and IEMS did absolutely amazing work

Rules

- If something in lecture doesn't make sense, please ask
  - If it doesn't make sense to you, others have the same question!
- Do you feel like there is a gap in your background, e.g., forgot about resistance and capacitance?
  - It's O.K. I have handouts and office hours to help but don't fall behind!
- You're paying a huge amount of money for this
- I expect a lot
- However, I'll do whatever I can to make sure you get as much out of this course as you put in

Core course goal

By the end of this course,
I want every one of you to be capable of designing and building simple but useful computer systems from integrated circuits, wires, and assembly language instructions
In fact, it's a requirement

Outline

Administrative stuff

- How to get lab supplies
- How to subscribe to mailing list
- Some good references
- Decide grading policies
- Plan office hours
- Course overview (if time permits)

How to get lab supplies

- Each student is required to pay $20 for lab supplies
  - Integrated circuits, wires, capacitors, resistors, etc.
- Make check out to Northwestern University
- Take the check to Carol Surma in Tech L359
- Take the receipt to Albert Lyerla in CG30 to pick up lab kits

Website

- http://ziyang.eecs.northwestern.edu/~dickrp/eecs203/
- Will use blackboard for grades

References

- **Primary reference:** M. Morris Mano and Charles R. Kime. *Logic and Computer Design Fundamentals*. Prentice-Hall, NJ, fourth edition, 2008
- Zvi Kohavi. *Switching and Finite Automata Theory*. McGraw-Hill Book Company, NY, 1978
- Randy H. Katz. *Contemporary Logic Design*. The Benjamin/Cummings Publishing Company, Inc., 1994
- J. Hennessy and D. Patterson. *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann Publishers, CA, third edition, 2003

Grading scheme

- 15% homeworks
- 35% labs
- 20% midterm exam
- 30% final exam

Late homework assignments

- After the class, on the due date: -5%
- After that, 10% per day penalty
- Three or more working days late: No credit
  - I'll hand out solutions

Late lab assignments

- Late lab verifications will be done at the discretion of the TAs
- In other words, although this will sometimes be possible, I'm not going to force the TA to skip their classes, research work, or meals to hold extra lab verification hours
- Late lab checks (without prior approval): -20%
- Three or more working days late: No credit

When to start labs

- The TAs spend a huge amount of time checking labs
- Having them do lab checks outside of the scheduled hours makes it difficult to keep up in their own classes and research
- Start labs early to see if you have questions
- The TAs and I will be happy to help
- Will need time to finish after pointed in right direction

Labs

- Open labs
- Tech CG30
- The TAs and I may leave a note and go from our offices to CG24 during office hours to answer lab questions
- You will need to sign up for a lab time slot

Lab check times

- New labs will normally be assigned on Mondays
- Lab checks will normally be on Mondays (tentatively)
- First lab much quicker than others
- Need to get go to get kit ASAP

Office hours options

1. Most likely I will have office hours Mon/Tue 15:00–16:00
2. Hopefully these times work for most people
3. TA office hours will be announced soon

Course overview

- Know what is computer engineering is
- Know some reasons to learn computer engineering
- Understand course goals
- Know which future courses EECS 203 can prepare you for
- Know course topics
- Start learning basic logic definitions

What is computer engineering?

- Design and implementation of computer systems
- Hardware and software design
- Related to electrical engineering and computer science with an emphasis on digital circuits
- The best computer engineers are also good at electrical engineers and computer science
- Knowing fundamentals helps in fields where computers are used

What is computer engineering?

- You need something solid to stand on
- Applications make more sense if you understand programming
- Programming makes more sense if you understand processors
- Processor make more sense if you understand logic design
- Logic design makes more sense if you understand circuits and discrete math
- Circuits make more sense if you understand transistors
- Every understanding rests on others
- Computer engineering requires understanding the many levels and the ways they fit together

Why computer engineering

- Why are you taking this class?
- What do you want to learn?
- What kind of background do you have?
  - When you see something cool do you reach for a screwdriver?
  - Who was electrocuted as a young child trying to figure out how something works?
  - Who has written code?
  - Who has designed something complicated for the fun of it?

Why computer engineering? Fun

- Computers are almost magical
  - You'll learn how they work and how to build new ones
- You'll learn (discrete) math, semiconductor physics, and the theory of algorithms
- You'll be able to use your knowledge creatively

Why computer engineering? Fun

- In the end, your creations will be tested against unforgiving physical laws in the real world
- There are many right ways (ways that work) to design a computer but there are also many wrong ways (ways that don't work)
  - There are measurable and clear differences between the quality of different designs
- You'll spend a lot of time with hard-working people who share your interest in designing machines that make life better

Why computer engineering? Flexible

Learn hardware and software design, can move in either direction

- Embedded system design
- Computer architecture
- VLSI design
- Digital circuit design
- Software engineering
- Algorithm design
- Information technology

If you finish a Ph.D., many other doors also open

Why computer engineering? Money

Highest 2006–2007 salaries reported by National Association of Colleges and Employers, February 2007

Field	Average salary ($)
Chemical engineering	60,054
Computer engineering	54,877
Electrical engineering	54,599
Mechanical engineering	54,587
Economics	51,631
Computer science	51,070
Finance	47,905
Civil engineering	47,145
Accounting	46,508
Business administration/management	43,523
Marketing/marketing management	41,323

Why computer engineering? Money

- Money alone isn't a good reason to pick a major
- Do what you love!
  - . . . but if you love computer engineering, the financial stuff might make it easier to justify to your relatives

Future courses

- Advanced digital logic design
- Computer architecture
- Design and analysis of algorithms
- Fundamentals of computer system software
- Introduction to computer networks

Future courses

- Introduction to VLSI CAD
- Introduction to mechatronics
- Microprocessor system design
- Programming for computer engineers
- VLSI systems design

Course topics in context

- Logic gates
  - Basic units of digital logic design
- Truth tables
  - Simple Boolean function representation
- Boolean algebra
  - Another way of representing and manipulating Boolean functions
- Two-level logic forms

Course topics

- Logic minimization: Boolean algebra, Karnaugh maps, and Quine-McCluskey's method (if time permits)
  - Reduce area, power consumption, or improve performance
- Hazards
- Implementation in CMOS
- Number systems: decimal, binary, octal, hex, and Gray codes
- Signed and unsigned numbers

Course topics

- Arithmetic circuits, decoders, encoders, and multiplexers
- Sequential logic: Latches, flip-flops
- Finite state machines
- Assembly language programming

Course topics

- Overview of compilation of higher-level languages
- Computer organization
- Microcontrollers

Software

- Easy to change and design
- Usually has low performance compared to hardware implementation
- High power consumption
- General-purpose processor
- Digital signal processor (DSP)
- Field programmable gate array (FPGA)
- Application specific integrated circuit (ASIC)

Hardware

- Usually difficult to design and implement compared to software
  - Hardware description languages can make this easier
- Necessary (all software runs on hardware)
- High performance
- Low power

Hardware/software rules of thumb

- If you can do it in software, do it in software
  - However, some things can't be done in a sane way with software
- If you can't do it in software but you can do it with an HDL, do it with an HDL
  - Sometimes the results of automation aren't good enough
- If you're tired, don't do hardware implementation
  - Software design errors usually mean wasted time
  - Hardware design errors often mean fried chips

Embedded systems

- Special-purpose computers, computers within devices which are generally not seen to be computers
- Larger market than general-purpose computers by volume and monetary value
- Microcontrollers rule
- Cool application-specific optimizations
  - Power
  - Size
  - Reliability
  - Hard deadlines

Market

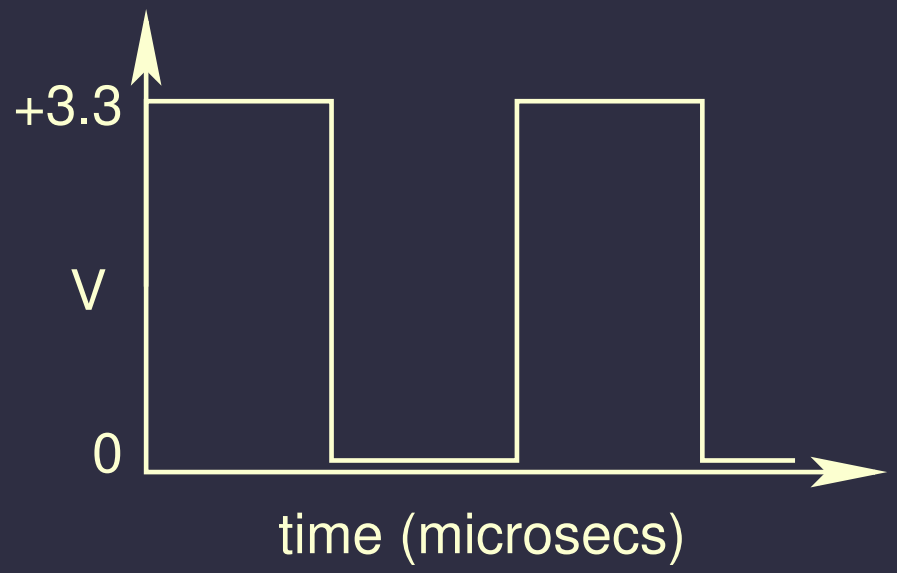
- How large is the semiconductor market?

Market

- How large is the semiconductor market?
- $\$270.3 \times 10^9$ for 2007 and growing fast
  - Semiconductor Industry Association

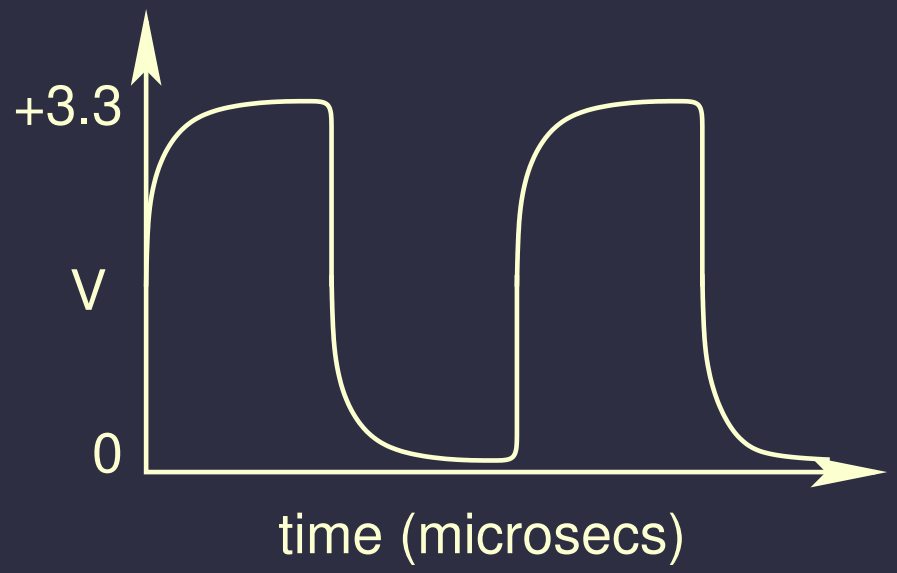
Outline

Digital and analog signals



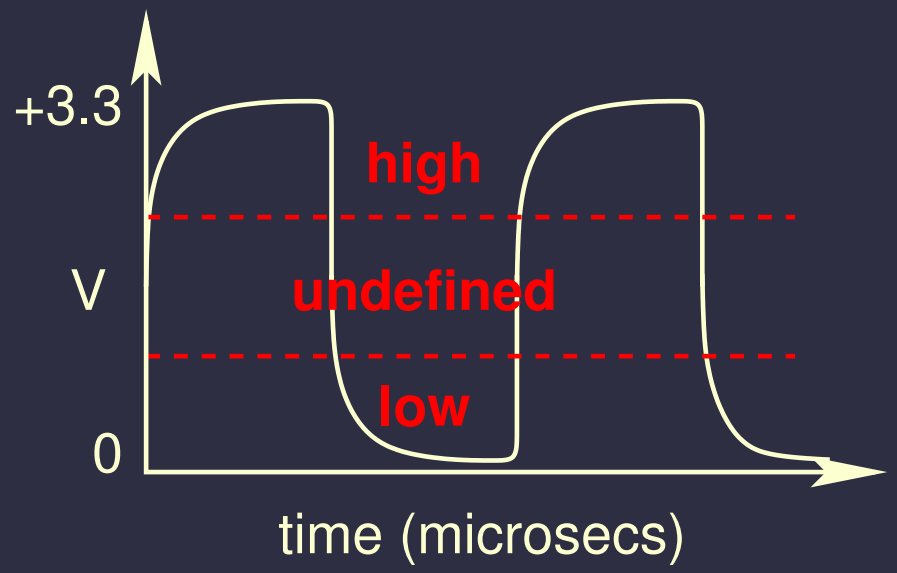
- Analog: Continuously varying signal

Digital and analog signals



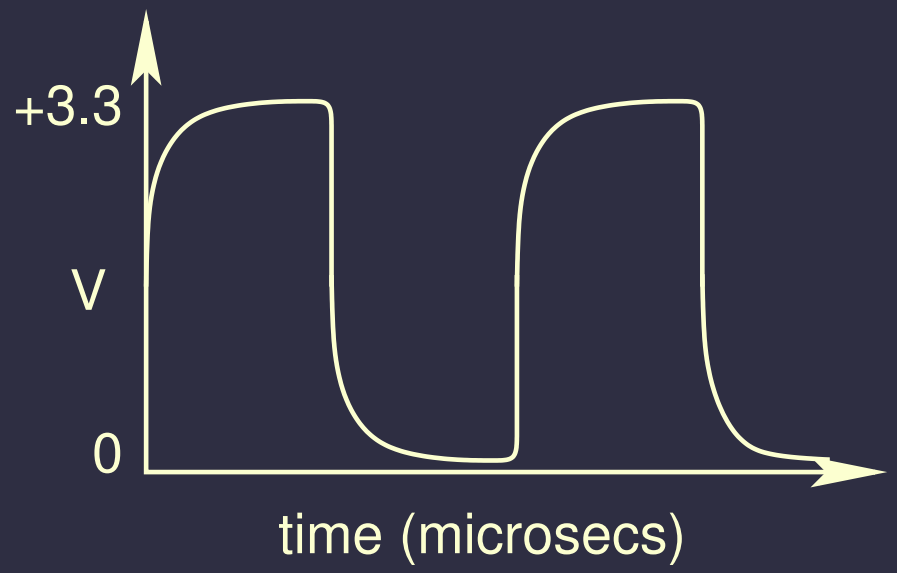
- Analog: Continuously varying signal
- Digital: Discrete values, assumed instantaneous transition

Digital and analog signals



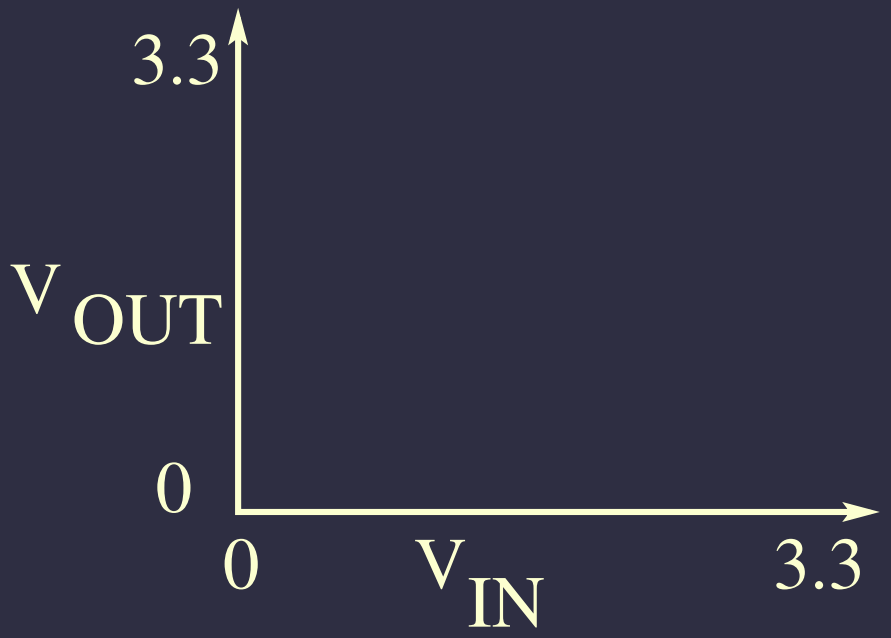
- Analog: Continuously varying signal
- Digital: Discrete values, assumed instantaneous transition
- In reality, digital assumption is approximation

Digital and analog signals

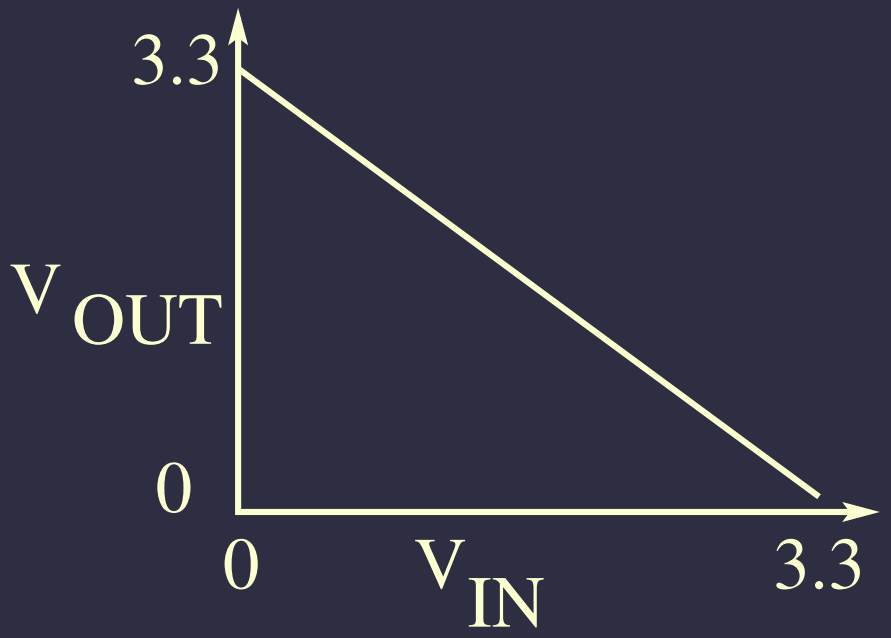


- Analog: Continuously varying signal
- Digital: Discrete values, assumed instantaneous transition
- In reality, digital assumption is approximation
- Use thresholds

Digital voltage regeneration

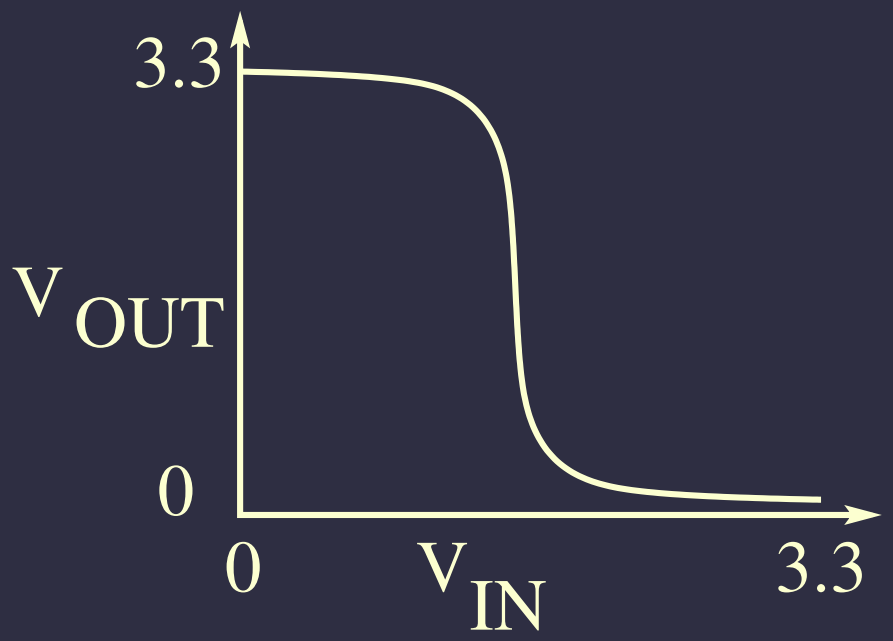


Digital voltage regeneration



- Error in input appears on output

Digital voltage regeneration



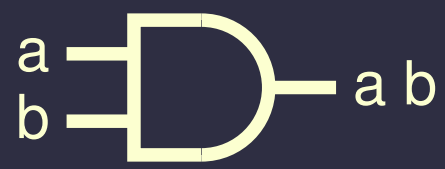
- Error in input appears on output
- Voltage regeneration hides input variation

Boolean algebra

- The only values are 0 (or false) and 1 (or true)
- One can define operations/functions/gates
  - Boolean values as input and output
- A truth table enumerates output values for all input value combinations

AND

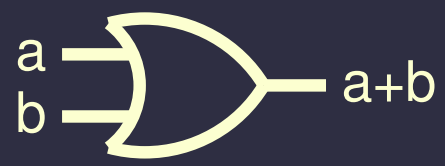
a	b	a · b
0	0	0
0	1	0
1	0	0
1	1	1



$$\mathrm{a}\ \mathit{AND}\ \mathrm{b} = \mathrm{a} \wedge \mathrm{b} = \mathrm{a} \cdot \mathrm{b} = \mathrm{a}\ \mathrm{b}$$

OR

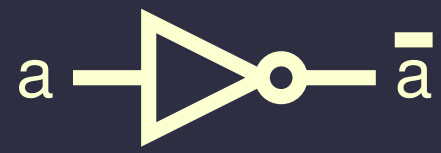
a	b	a + b
0	0	0
0	1	1
1	0	1
1	1	1



$$a\ \mathit{OR}\ b = a \vee b = a + b$$

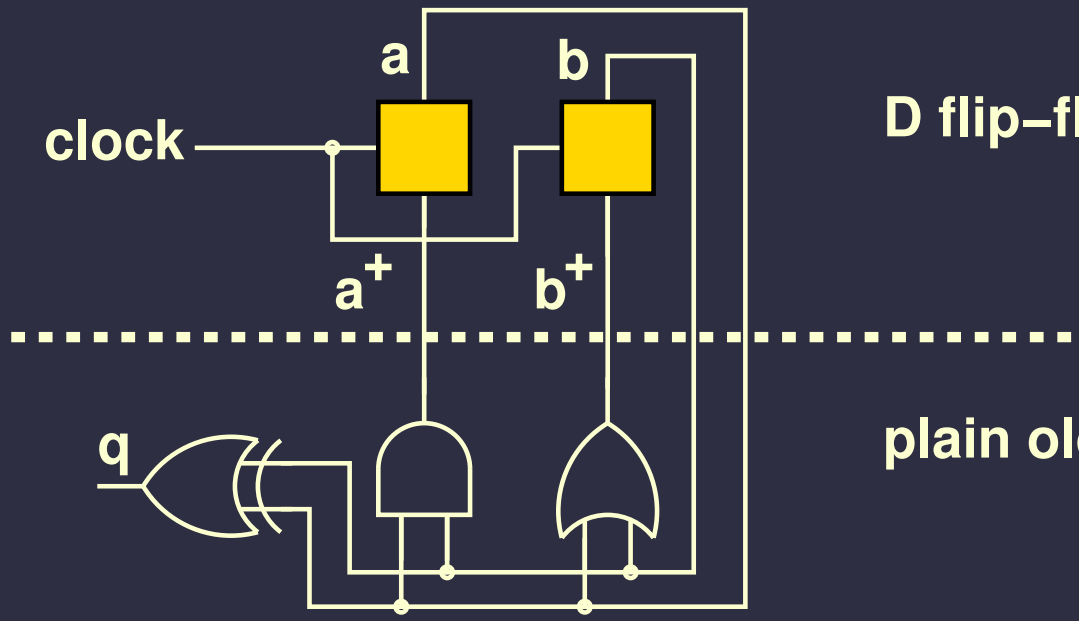
NOT

a	$\bar{a}$
0	1
1	0



$NOT\ \mathrm{a} = \mathrm{a'} = \bar{\mathrm{a}}$

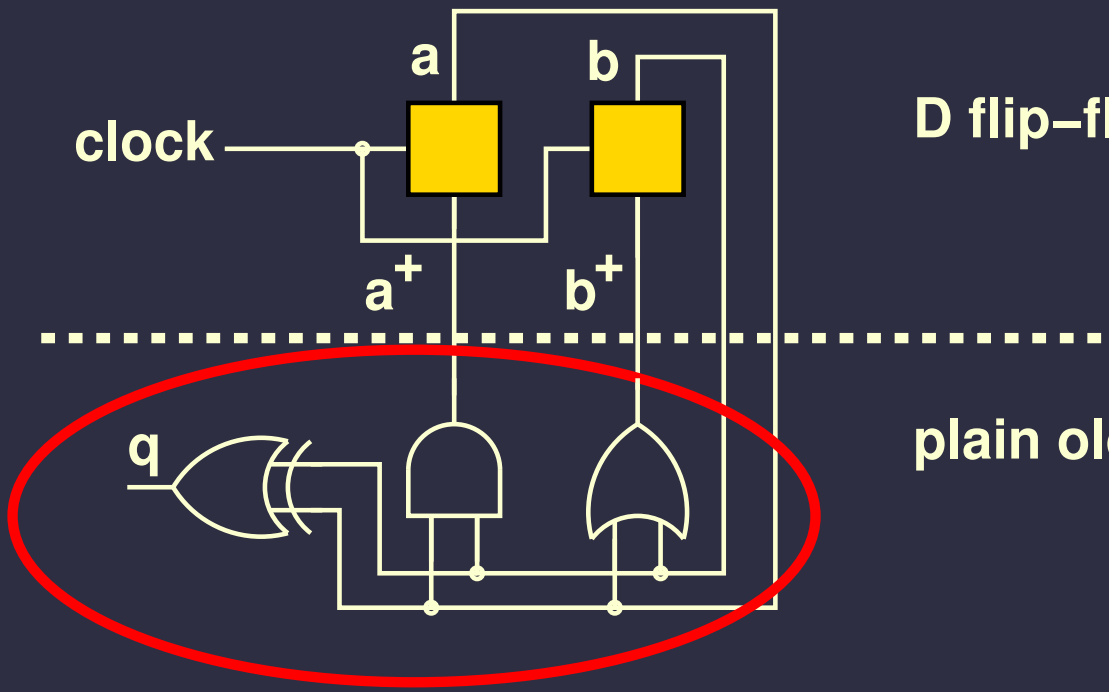
Combinational vs. sequential logic



0.65

- No feedback between inputs and outputs – combinational
  - Outputs a function of the current inputs, only

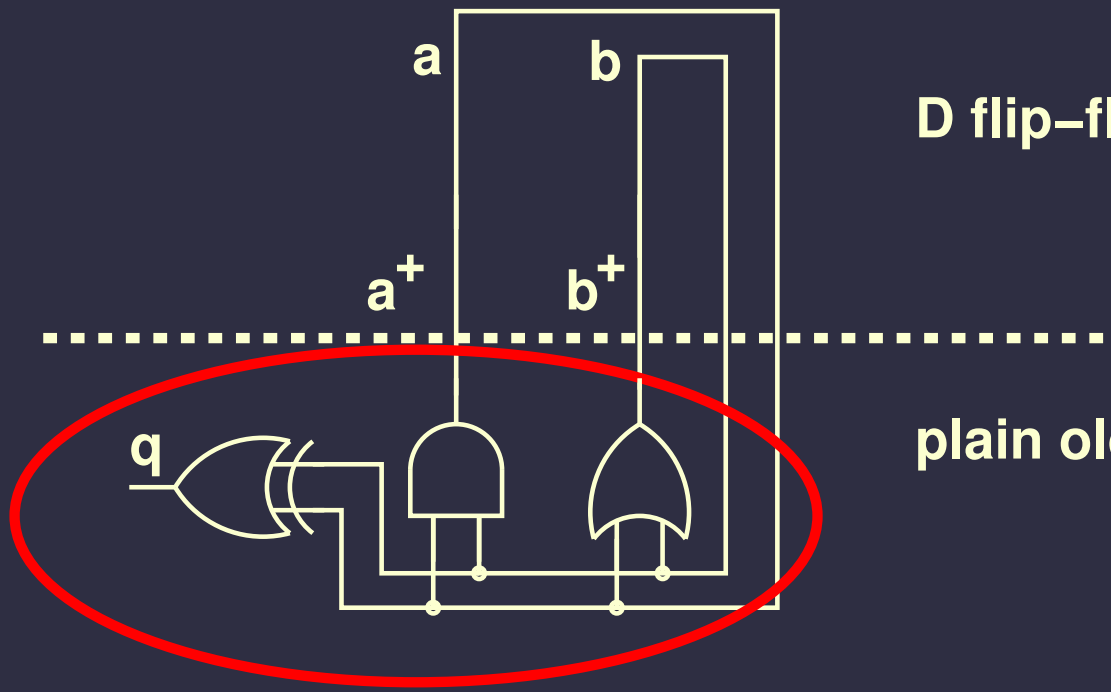
Combinational vs. sequential logic



0.65

- No feedback between inputs and outputs – combinational
  - Outputs a function of the current inputs, only
- Feedback – sequential
  - Outputs a function of the current and *previous* inputs

Combinational vs. sequential logic



D flip–flops

plain old combinational logic

0.65

- No feedback between inputs and outputs – combinational
  - Outputs a function of the current inputs, only
- Feedback – sequential
  - Outputs a function of the current and *previous* inputs

Sequential logic

- Outputs depend on current state and (maybe) current inputs
- Next state depends on current state and input
- For implementable machines, there are a finite number of states
- Synchronous
  - State changes upon clock event (transition) occurs
- Asynchronous
  - State changes upon inputs change, subject to circuit delays

Summary

- Brief overview
- Administrative stuff
- Introduction and definitions

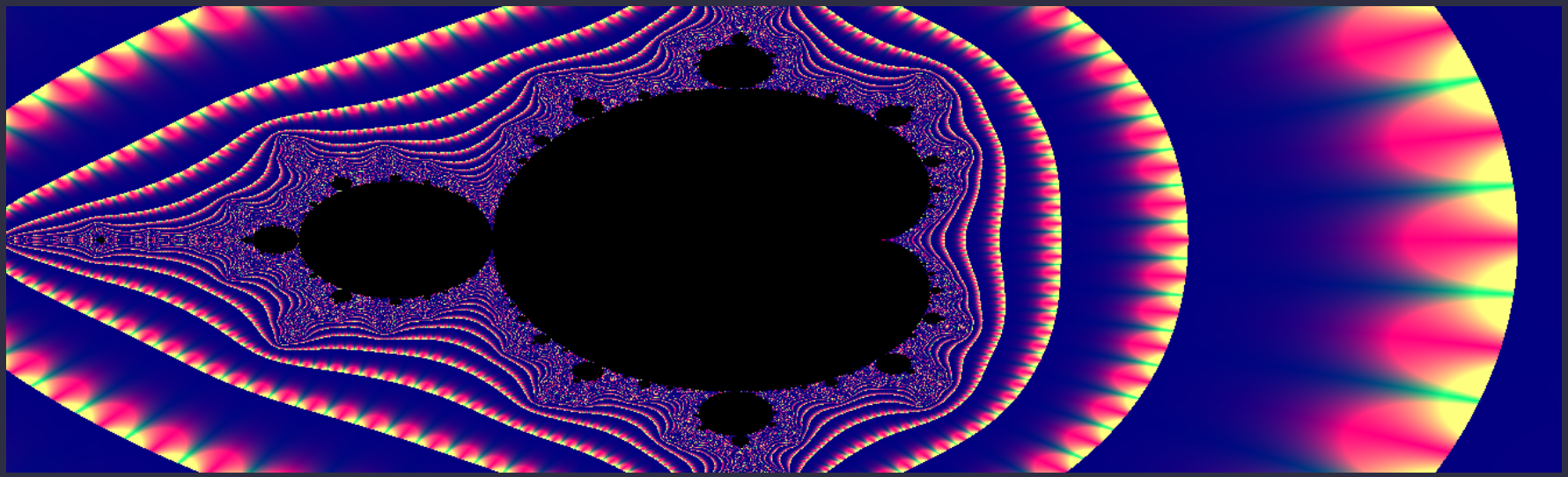
Mandelbrot set display code

```
threshold = 1000
iter = 500

def f(c, iter):
  a = (0 + 0j)
  for i in xrange(iter):
    a = a**2 + c
    if abs(a) > threshold:
     break
  return a

desc = [[complex(x, y) for x in xfrange(-2, 2, 0.0015)]
  for y in xfrange(1, -1, -0.0015)]
```

Mandelbrot set image produced by code



Computers enabled many inventions

- Simulation, automation, knowledge discovery
- In astrophysics, chemistry, biology, medicine, etc.

Outline

Reading assignment (for next class)

- M. Morris Mano and Charles R. Kime. *Logic and Computer Design Fundamentals*. Prentice-Hall, NJ, fourth edition, 2008
- Sections 1.1, 2.1, and 2.2
- CMOS handout from M. Morris Mano and Charles R. Kime. *Web supplements to Logic and Computer Design Fundamentals*. Prentice-Hall, NJ. http://www.writphotec.com/mano/Supplements
- Read these as soon as possible

Computer geek culture references

- $Z_{i+1} = Z_i^2 + K$