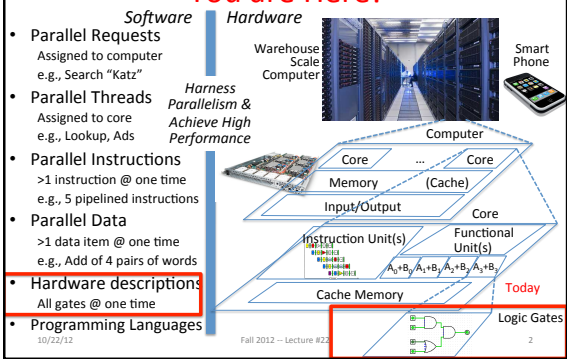
CS 61C:
Great Ideas in Computer Architecture
Switches, Transistors, Gates, and Boolean Logic

Instructors:
Krste Asanovic, Randy H. Katz
http://inst.eecs.Berkeley.edu/~cs61c/fa12

You are Here!



Agenda

- Switching Networks, Transistors
- Gates and Truth Tables for Circuits
- Boolean Algebra
- (Logisim if there is time)
- And in Conclusion, …

Agenda

- Switching Networks, Transistors
- Gates and Truth Tables for Circuits
- Boolean Algebra
- (Logisim if there is time)
- And in Conclusion, …

Hardware Design

- Next several weeks: how a modern processor is built, starting with basic elements as building blocks
- Why study hardware design?
  - Understand capabilities and limitations of hw in general and processors in particular
  - What processors can do fast and what they can't do fast (avoid slow things if you want your code to run fast!)
  - Background for more in depth hw courses (CS 152)
  - Hard to know what will need for next 30 years
  - There is just so much you can do with standard processors: you may need to design own custom hw for extra performance
    - Even some commercial processors today have customizable hardware!

Synchronous Digital Systems

Hardware of a processor, such as the MIPS, is an example of a Synchronous Digital System

Synchronous:

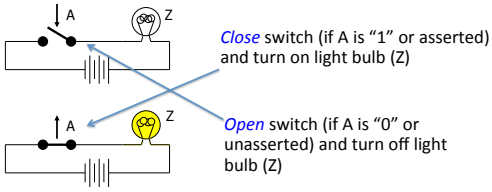
- All operations coordinated by a central clock
  - "Heartbeat" of the system!

Digital:

- Represent all values by two discrete values
- Electrical signals are treated as 1's and 0's
  - 1 and 0 are complements of each other
- High /low voltage for true / false, 1 / 0

Switches: Basic Element of Physical Implementations

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



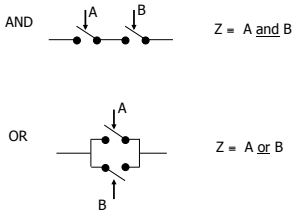
Close switch (if A is "1" or asserted) and turn on light bulb (Z)

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

Z = A

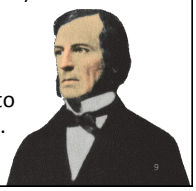
Switches (cont'd)

- Compose switches into more complex ones (Boolean functions):



Historical Note

- Early computer designers built ad hoc circuits from switches
- Began to notice common patterns in their work: ANDs, ORs, …
- Master's thesis (by Claude Shannon) made link between work and 19th Century Mathematician George Boole
  - Called it "Boolean" in his honor
- Could apply math to give theory to hardware design, minimization, …



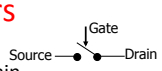
Transistors

- High voltage (V_{dd}) represents 1, or true
- Low voltage (0 volts or Ground) represents 0, or false
- Let threshold voltage (V_{th}) decide if a 0 or a 1
- If switches control whether voltages can propagate through a circuit, can build a computer
- Our switches: CMOS transistors

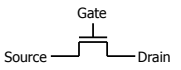
CMOS Transistor Networks

- Modern digital systems designed in CMOS
  - MOS: Metal-Oxide on Semiconductor
  - C for complementary: use *pairs* of normally-open and normally-closed switches
    - Used to be called COS-MOS for complementary-symmetry - MOS
- CMOS transistors act as voltage-controlled switches
  - Similar, though easier to work with, than relay switches from earlier era
  - Use energy primarily when switching

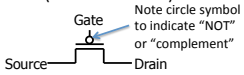
CMOS Transistors



- Three terminals: source, gate, and drain
  - Switch action:
    if voltage on gate terminal is (some amount) higher/lower than source terminal then conducting path established between drain and source terminals (switch is closed)



n-channel transitor
open when voltage at Gate is low
closes when:
voltage(Gate) > voltage (Threshold)
(High resistance when gate voltage Low, Low resistance when gate voltage High)



p-channel transistor
closed when voltage at Gate is low
opens when:
voltage(Gate) > voltage (Threshold)
(Low resistance when gate voltage Low, High resistance when gate voltage High)

CMOS Circuit Rules

- Don't pass weak values => Use Complementary Pairs
  - N-type transistors pass weak 1's (V_{dd} - V_{th})
  - N-type transistors pass strong 0's (ground)
  - Use N-type transistors only to pass 0's (N for negative)
  - Converse for P-type transistors: Pass weak 0s, strong 1s
    - Pass weak 0's (V_{th}), strong 1's (V_{dd})
    - Use P-type transistors only to pass 1's (P for positive)
  - Use pairs of N-type and P-type to get strong values
- Never leave a wire undriven
  - Make sure there's always a path to V_{dd} or gnd
- Never create a path from V_{dd} to gnd (ground)

Administrivia

- Regrade window opens today and closes by next Wednesday's lecture
  - Best to return to your lab/discussion TA (parallel processing!)
- Coming labs and HWs: build up for Project #3
  - Last week's lab: Cache blocking
  - This week's lab: SIMD instruction set
  - This week's HW: Cache blocking and SIMD practice
  - Next week's lab: OpenMP thread programming

EECS Grading Policy

- http://www.eecs.berkeley.edu/Policies/ugrad.grading.shtml
  "A typical GPA for courses in the lower division is 2.7. This GPA would result, for example, from 17% A's, 50% B's, 20% C's, 10% D's, and 3% F's. A class whose GPA falls outside the range 2.5 - 2.9 should be considered atypical."
- Spring 2011: GPA 2.85
  24% A's, 49% B's, 18% C's, 6% D's, 3% F's
- Job/Intern Interviews: They grill you with technical questions, so it's what you say, not your GPA
  (61c gives you good stuff to say)

	Fall	Spring
2011	2.72	2.85
2010	2.81	2.81
2009	2.71	2.81
2008	2.95	2.74
2007	2.67	2.76

CS61c in the News
Computers meet Cameras

http://gigaplan.com

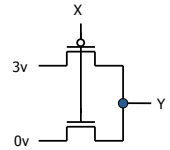
- http://mlb.mlb.com/photos/gigapan/?gpId=a2be33bf9acd8730e8a086e8c1f26d8d&ps=1

Agenda

- Switching Networks, Transistors
- Gates and Truth Tables for Circuits
- Boolean Algebra
- (Logisim if there is time)
- And in Conclusion, …

MOS Networks

p-channel transistor
closed when voltage at Gate is low
opens when:
voltage(Gate) > voltage (Threshold)



n-channel transitor
open when voltage at Gate is low
closes when:
voltage(Gate) > voltage (Threshold)

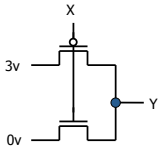
what is the relationship between x and y?

x	y
0 volts (gnd)	
3 volts (Vdd)	

Called an *invertor* or *not gate*

MOS Networks

n-channel transitor
open when voltage at Gate is low
closes when voltage(Gate) > voltage (Source) + ε



p-channel transistor
closed when voltage at Gate is low
opens when voltage(Gate) < voltage (Source) – ε

what is the relationship between x and y?

x	y
0 volts (gnd)	3 volts (Vdd)
3 volts (Vdd)	0 volts (gnd)

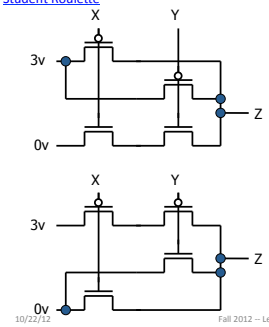
Called an *invertor* or *not gate*

P = ½ C V² f

- Dynamic Energy (when switching) is proportional to Capacitance * Voltage²
- Since pulse is 0 -> 1 -> 0 or 1 -> 0 -> 1, Energy of a single transition is proportional to ½ *Capacitance * Voltage²
- Power is just energy per transition times frequency of transitions: proportional to ½ * Capacitance * Voltage² * Frequency

Two Input Networks

Student Roulette

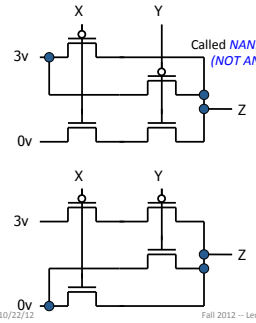


what is the relationship between x, y and z?

x	y	z
0 volts	0 volts	
0 volts	3 volts	
3 volts	0 volts	
3 volts	3 volts	

x	y	z
0 volts	0 volts	
0 volts	3 volts	
3 volts	0 volts	
3 volts	3 volts	

Two Input Networks: Peer Instruction



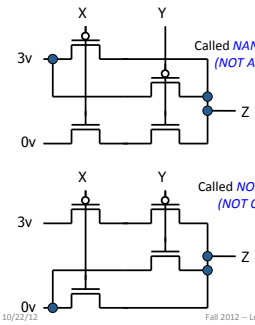
Called *NAND gate (NOT AND)*

what is the relationship between x, y and z?

x	y	z
0 volts	0 volts	3 volts
0 volts	3 volts	3 volts
3 volts	0 volts	3 volts
3 volts	3 volts	0 volts

x	y	z
0 volts	0 volts	A B C D 0 0 3 3 volts
0 volts	3 volts	0 3 0 3 volts
3 volts	0 volts	0 3 0 3 volts
3 volts	3 volts	3 3 0 0 volts

Two Input Networks



Called *NAND gate (NOT AND)*

what is the relationship between x, y and z?

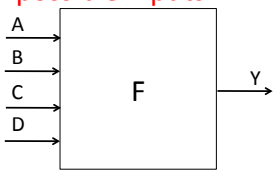
x	y	z
0 volts	0 volts	3 volts
0 volts	3 volts	3 volts
3 volts	0 volts	3 volts
3 volts	3 volts	0 volts

Called *NOR gate (NOT OR)*

x	y	z
0 volts	0 volts	3 volts
0 volts	3 volts	0 volts
3 volts	0 volts	0 volts
3 volts	3 volts	0 volts

Truth Tables

List outputs for all possible inputs



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)
0	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)

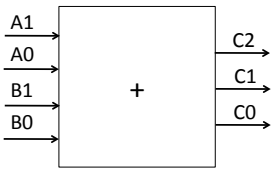
Truth Table Example #1: y= F(a,b): 1 iff a ≠ b

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

Truth Table Example #2: 2-bit Adder

A a_1a_0	B b_1b_0	C $c_2c_1c_0$

How Many Rows?



Truth Table Example #3: 32-bit Unsigned Adder

A	B	C
000 ... 0	000 ... 0	000 ... 00
000 ... 0	000 ... 1	000 ... 01
.	.	.
.	.	.
.	.	.
111 ... 1	111 ... 1	111 ... 10

How Many Rows?

Truth Table Example #4: 3-input Majority Circuit

Y =

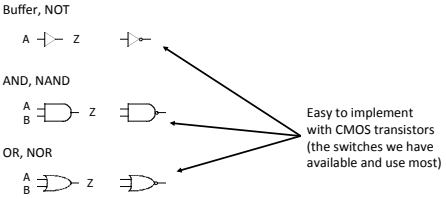
This is called *Sum of Products* form; Just another way to represent the TT as a logical expression

More simplified forms (fewer gates and wires)

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

Combinational Logic Symbols

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



Easy to implement with CMOS transistors (the switches we have available and use most)

Agenda

- Switching Networks, Transistors
- Gates and Truth Tables for Circuits
- Boolean Algebra
- (Logisim if there is time)
- And in Conclusion, …

Boolean Algebra

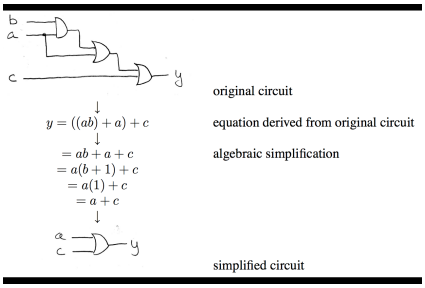
- Use plus for OR
  - "logical sum"
- Use product for AND (a•b or implied via ab)
  - "logical product"
- "Hat" to mean complement (NOT)
- Thus

  ab + a + $\overline{c}$

  = a•b + a + $\overline{c}$

  = (a AND b) OR a OR (NOT c)

Boolean Algebra: Circuit & Algebraic Simplification



Laws of Boolean Algebra

X $\overline{X}$ = 0	X + $\overline{X}$ = 1	Complementarity
X 0 = 0	X + 1 = 1	Laws of 0's and 1's
X 1 = X	X + 0 = X	Identities
X X = X	X + X = X	Idempotent Laws
X Y = Y X	X + Y = Y + X	Commutativity
(X Y) Z = Z (Y Z)	(X + Y) + Z = Z + (Y + Z)	Associativity
X (Y + Z) = X Y + X Z	X + Y Z = (X + Y) (X + Z)	Distribution
X Y + X = X	(X + Y) X = X	Uniting Theorem
$\overline{X}$ Y + X = X + Y	($\overline{X}$ + Y) X = X Y	United Theorem v. 2
$\overline{X\,Y}$ = $\overline{X}$ + $\overline{Y}$	$\overline{X + Y}$ = $\overline{X}$ $\overline{Y}$	DeMorgan's Law

Boolean Algebraic Simplification Example

$$y = ab + a + c$$

Boolean Algebraic Simplification Example

$$y = ab + a + c$$

$$= a(b + 1) + c \quad \textit{distribution, identity}$$

$$= a(1) + c \quad \textit{law of 1's}$$

$$= a + c \quad \textit{identity}$$

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

Agenda

- Switching Networks, Transistors
- Gates and Truth Tables for Circuits
- Boolean Algebra
- And in Conclusion, …

Agenda

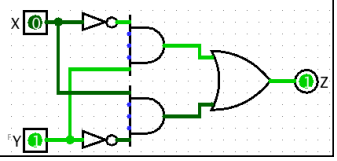
- Switching Networks, Transistors
- Gates and Truth Tables for Circuits
- Boolean Algebra
- (Logisim if there is time)
- And in Conclusion, …

Logisim

- Free schematic capture/logic simulation program in Java
  - "A graphical tool for designing and simulating logic circuits"
  - Search and download version 2.7.1, online tutorial
  - *ozark.hendrix.edu/~burch/**logisim**/*
- Drawing interface based on toolbar
  - Color-coded wires aid in simulating and debugging a circuit
  - Wiring tool draws horizontal and vertical wires, automatically connecting to components and to other wires.
- Circuit layouts used as "subcircuits" of other circuits, allowing hierarchical circuit design
- Included circuit components: inputs and outputs, gates, multiplexers, arithmetic circuits, flip-flops, RAM memory

Logisim Wires

- Blue wires: value at that point is "unknown"
- Gray wires: not connected to anything
- OK when in process of building a circuit
- When finished => wires not be blue or gray
- If connected, all wires should be green
  - Bright green a 1
  - Dark green a 0



Common Mistakes in Logisim

- Connecting wires together
- Using input for output
- Connecting to edge without connecting to actual input
  - Unexpected direction of input

And in Conclusion, …

- Real world voltages are analog, but are quantized to represent logic 0 and logic 1
- Transistors are just switches, combined to form gates: AND, OR, NOT, NAND, NOR
- Truth table can be mapped to gates for combinational logic design
- Boolean algebra allows minimization of gates