

# Digital Logic

*Gateway to computer science*

**Software**

**Program, Application**

**Programming Language**

**Compiler/Interpreter**

**Operating System**

**Instruction Set Architecture**

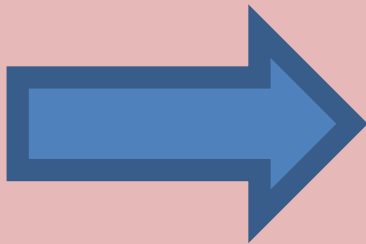
**Microarchitecture**

**Digital Logic**

**Devices (transistors, etc.)**

**Solid-State Physics**

**Hardware**



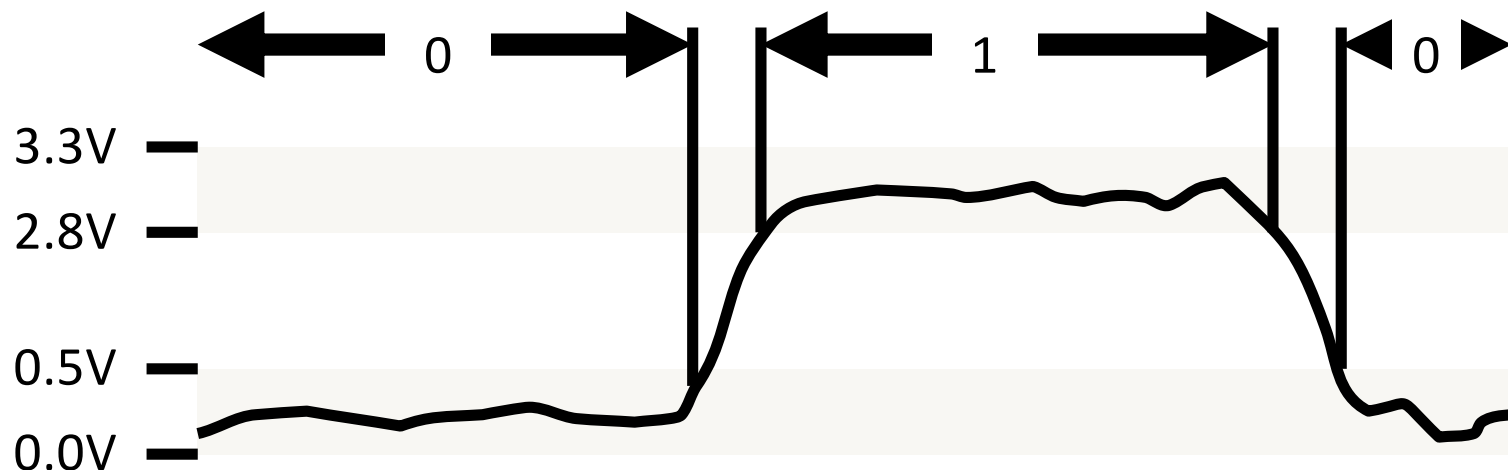
# Digital data/computation = Boolean

Boolean value (*bit*): 0 or 1

Boolean functions (AND, OR, NOT, ...)

Electronically:

bit = high voltage vs. low voltage



Boolean functions = logic gates, built from transistors



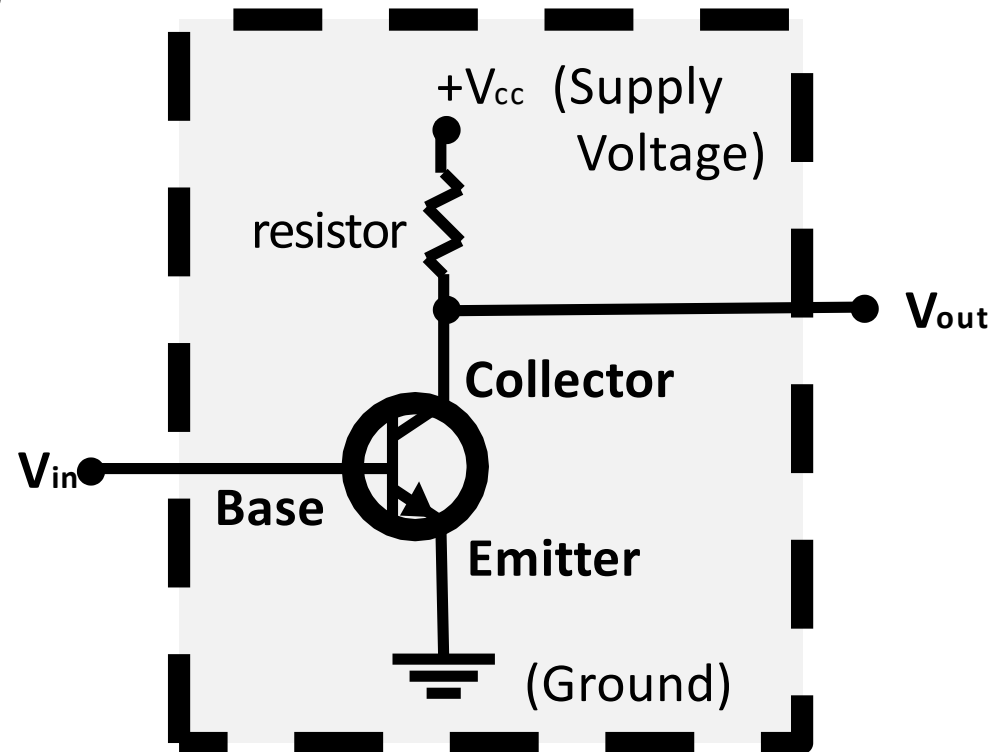
# Transistors (more in lab)

If **Base voltage is high:**

Current may flow freely from *Collector* to *Emitter*.

If **Base voltage is low:**

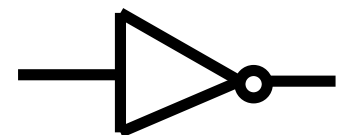
Current may not flow from *Collector* to *Emitter*.



Truth table

$V_{in}$	$V_{out}$	=	in	out	=	in	out
low	high	=	0	1	=	F	T
high	low		1	0		T	F

NOT gate



Abstraction!

# Digital Logic Gates

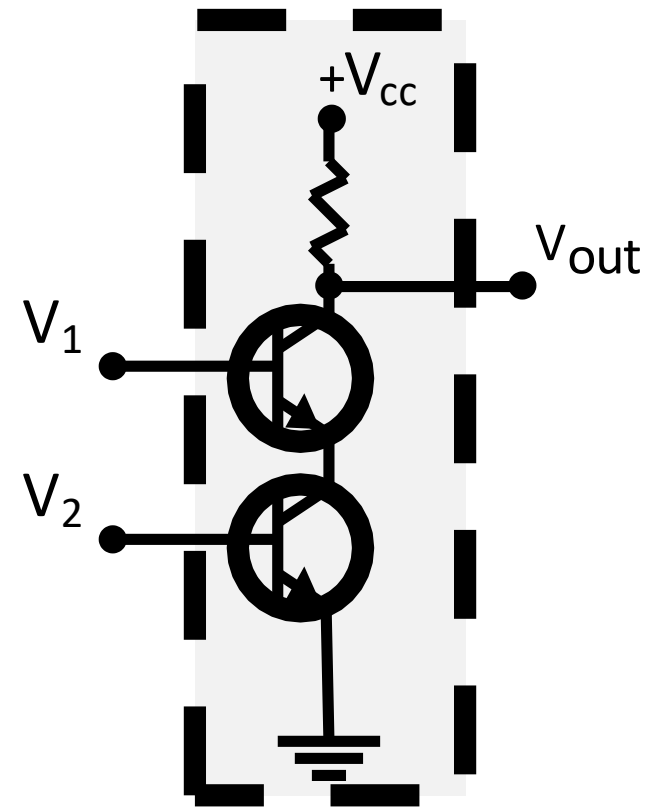
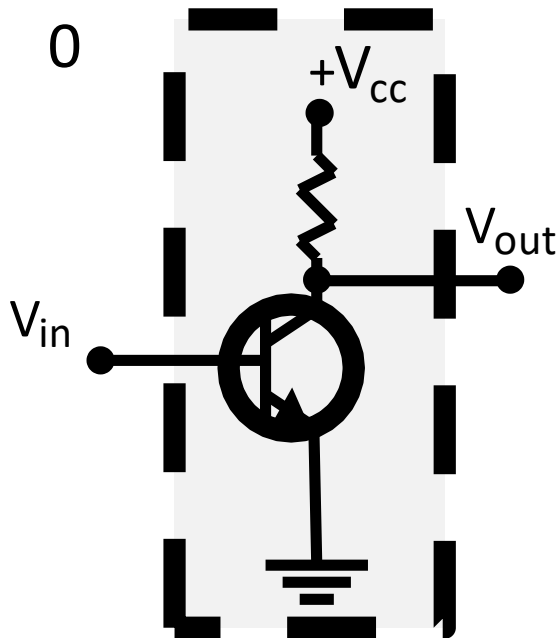


Tiny electronic devices that compute basic Boolean functions.



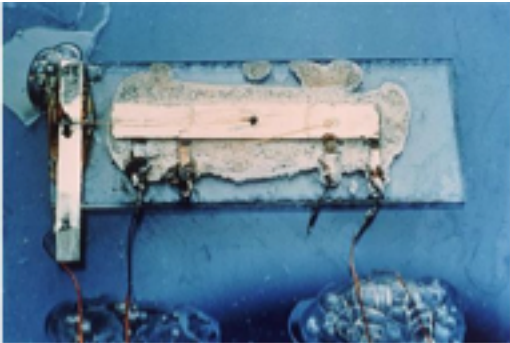
$V_{in}$	$V_{out}$
0	1
1	0

	$V_2$	
	0	1
$V_1$	0	
	1	

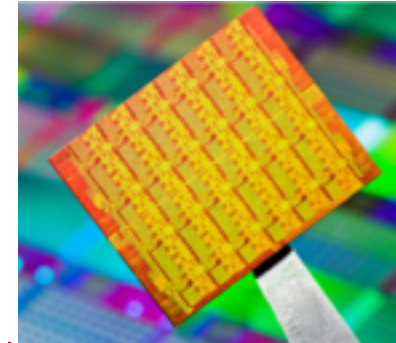


# Integrated Circuits (1950s - )

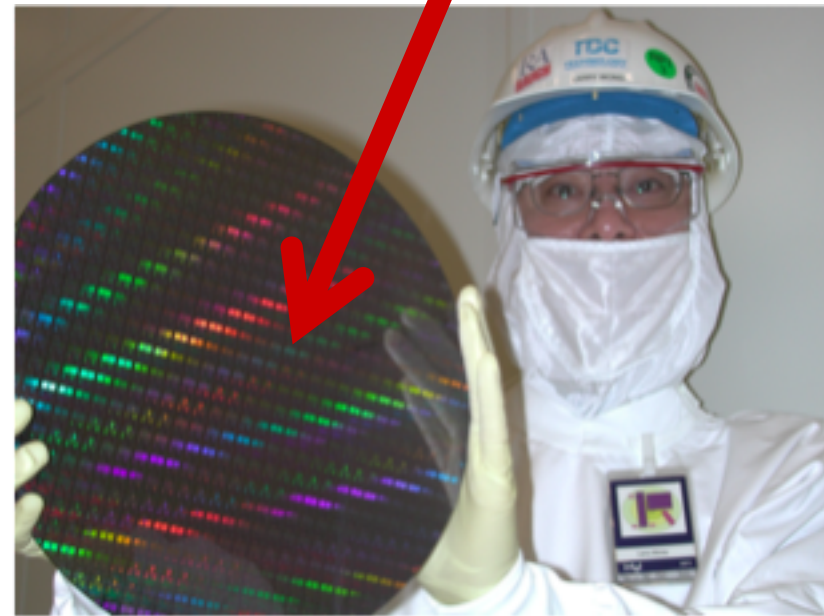
Early (first?) transistor



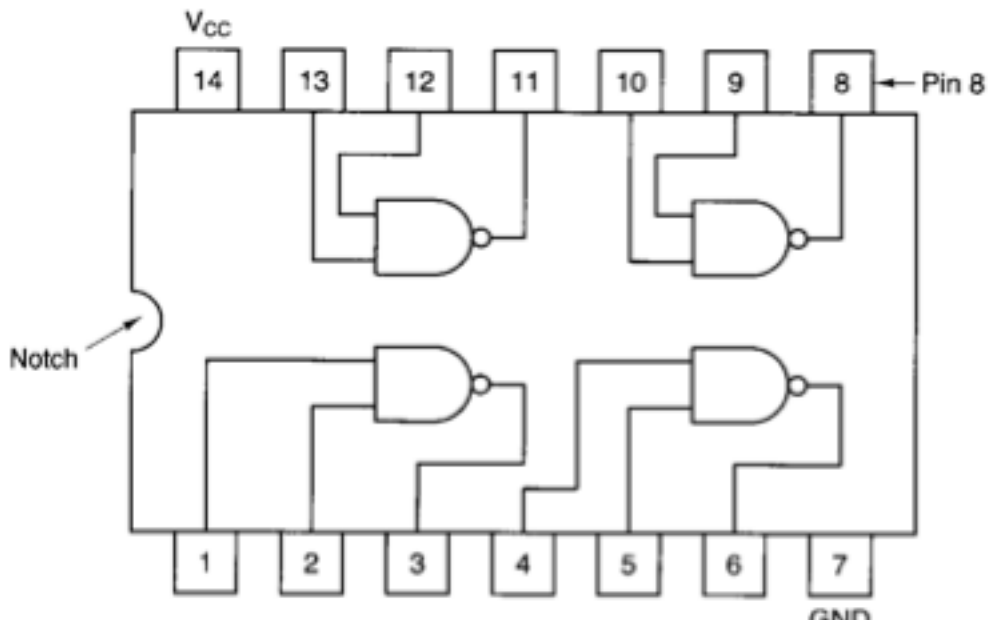
Chip



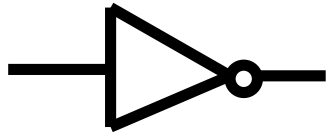
Wafer



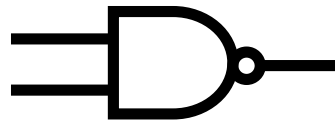
Small integrated circuit



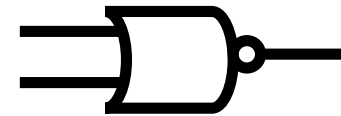
# Five basic gates: define with truth tables



NOT	
0	1
1	0



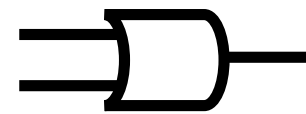
NAND	0	1
0	1	1
1	1	0



NOR	0	1
0		
1		



AND	0	1
0		
1		

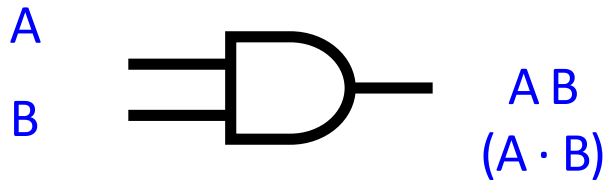


OR	0	1
0		
1		

# Boolean Algebra

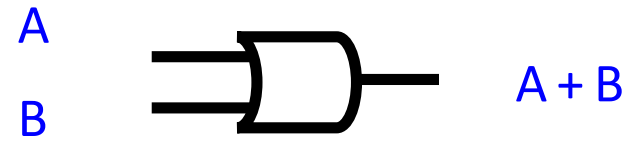
for combinational logic

*inputs* = *variables*  
*wires* = *expressions*  
*gates* = *operators/functions*  
*circuits* = *functions*



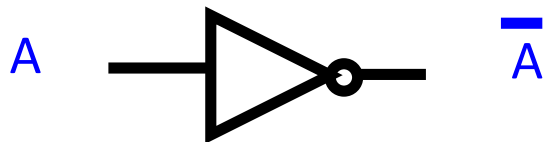
AND = Boolean product

$\cdot$	<b>0</b>	<b>1</b>
<b>0</b>	0	0
<b>1</b>	0	1



OR = Boolean sum

$+$	<b>0</b>	<b>1</b>
<b>0</b>	0	1
<b>1</b>	1	1



NOT = inverse or complement

	<b>0</b>	<b>1</b>
<b>0</b>	1	
<b>1</b>		0



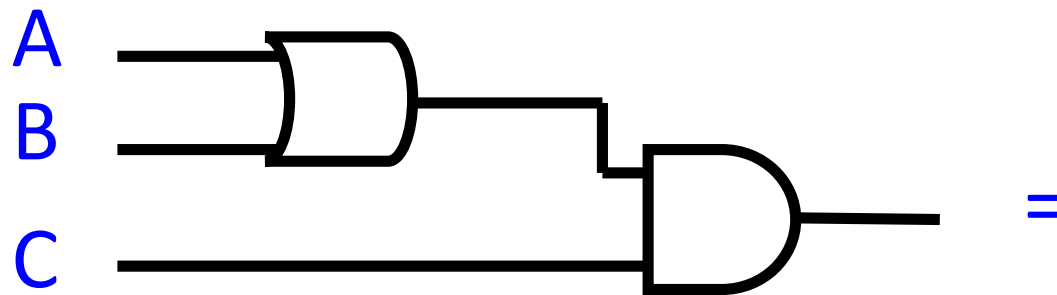
wire = identity

	<b>0</b>	<b>0</b>
<b>0</b>	0	
<b>1</b>		1



# Circuits

Connect inputs and outputs of gates with wires.  
Crossed wires touch **only if** there is a dot.



What is the output if  $A=1$ ,  $B=0$ ,  $C=1$ ?

What is the truth table of this circuit?

What is an equivalent Boolean expression?



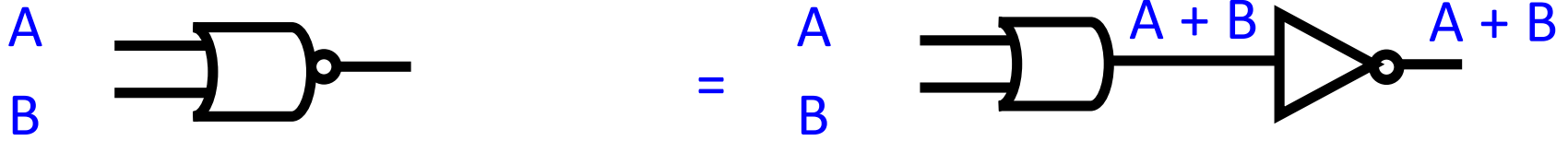
# Translation

Connect gates to implement these functions. Check with truth tables.  
Use a direct translation -- it is straightforward and bidirectional.

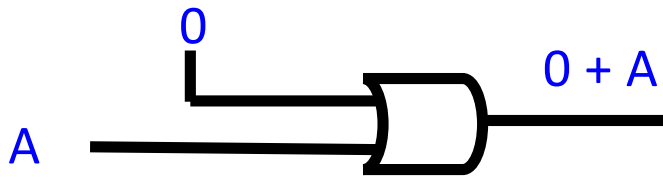
$$F = (A\bar{B} + C)D$$

$$Z = \bar{W} + (X + \bar{W}Y)$$

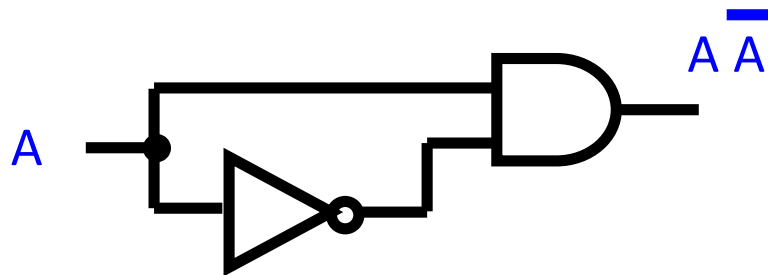
Note on notation: bubble = inverse/complement



## Identity law, inverse law



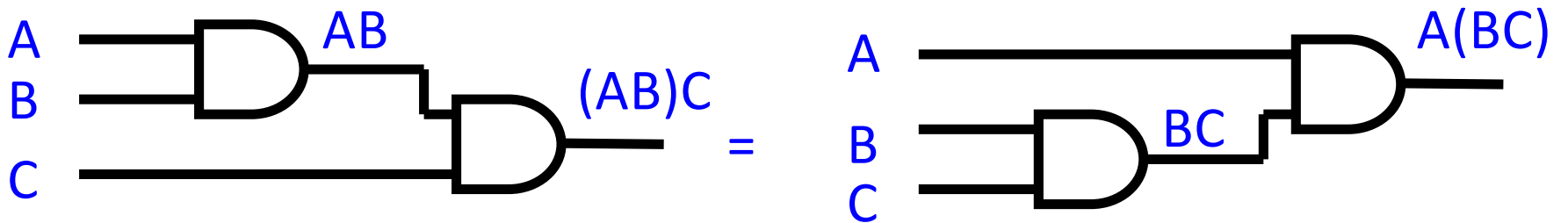
=



=



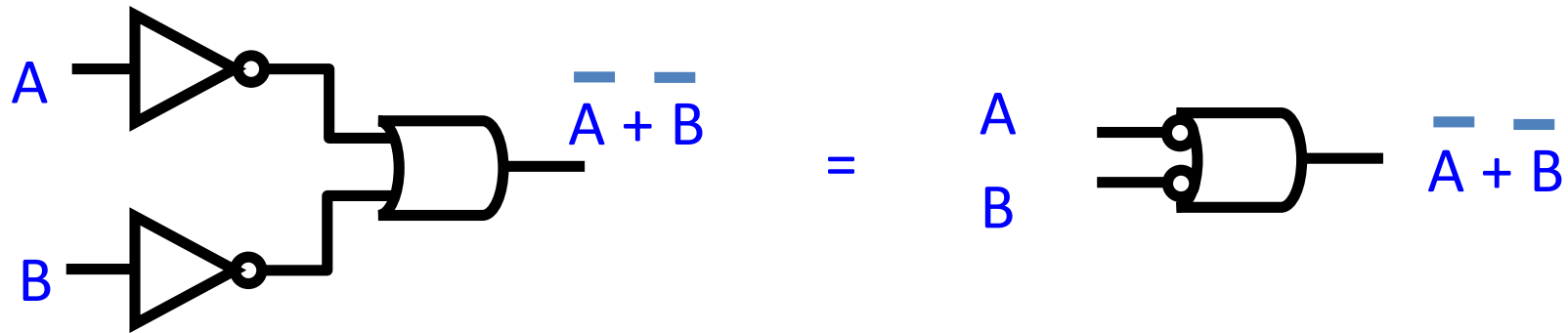
# Commutativity, Associativity



# Idempotent law, Null/Zero law

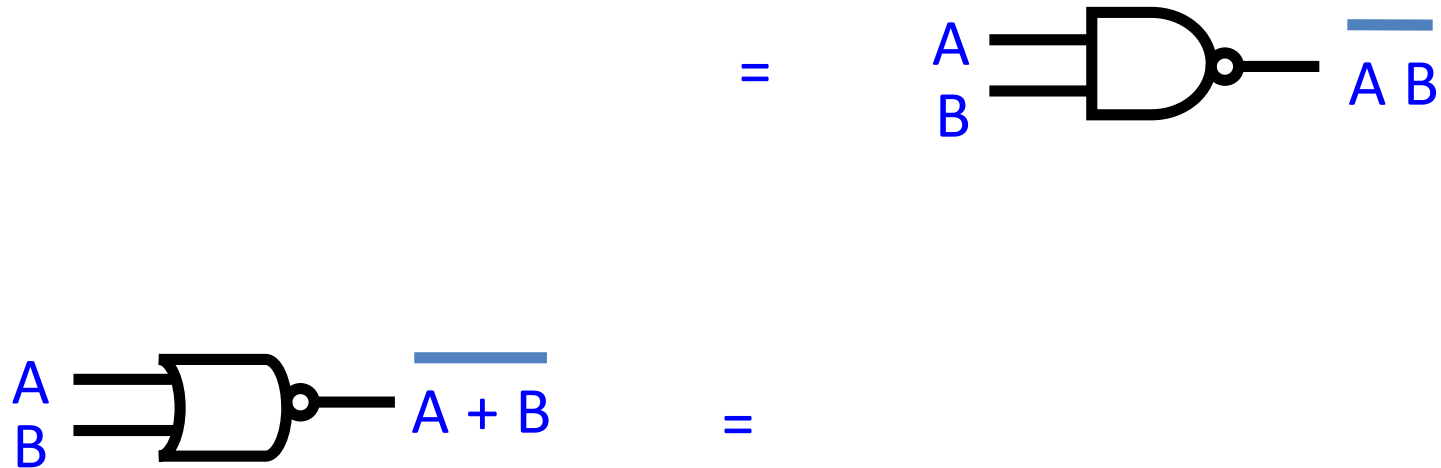


Note on notation: bubble = inverse/complement



## DeMorgan's Law

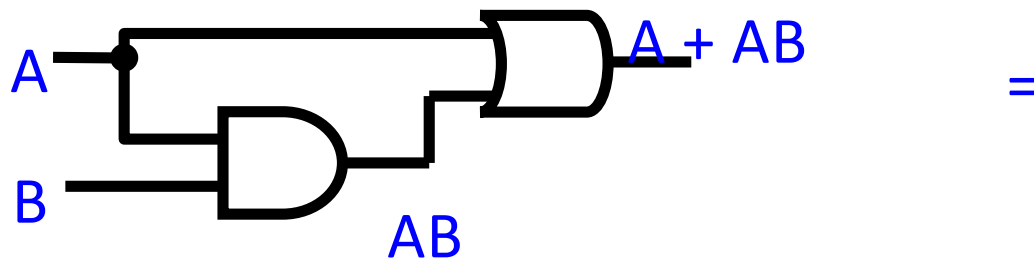
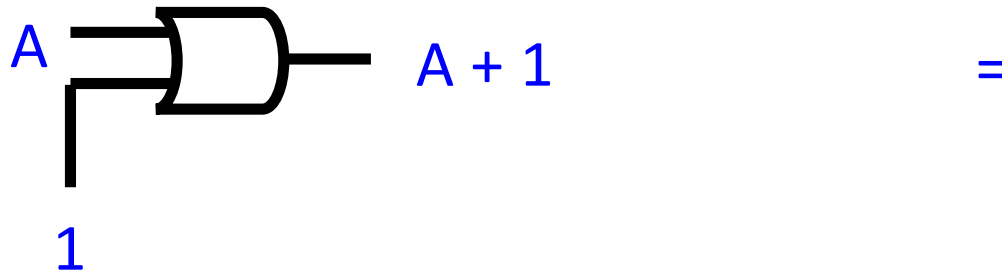
(double bubble, toil and trouble, in Randy's words...)





# One law, Absorption law

Write truth tables. Do they correspond to simpler circuits?





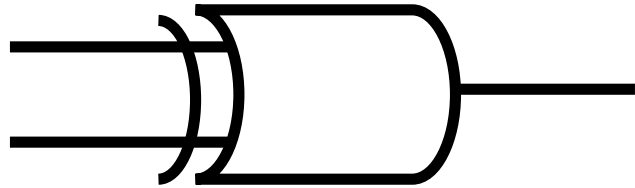
**NAND is *universal*.** 

All Boolean functions can be implemented using only NANDs.  
Build NOT, AND, OR, NOR, using only NAND gates.





# XOR: Exclusive OR



Output = 1 if exactly one input = 1.

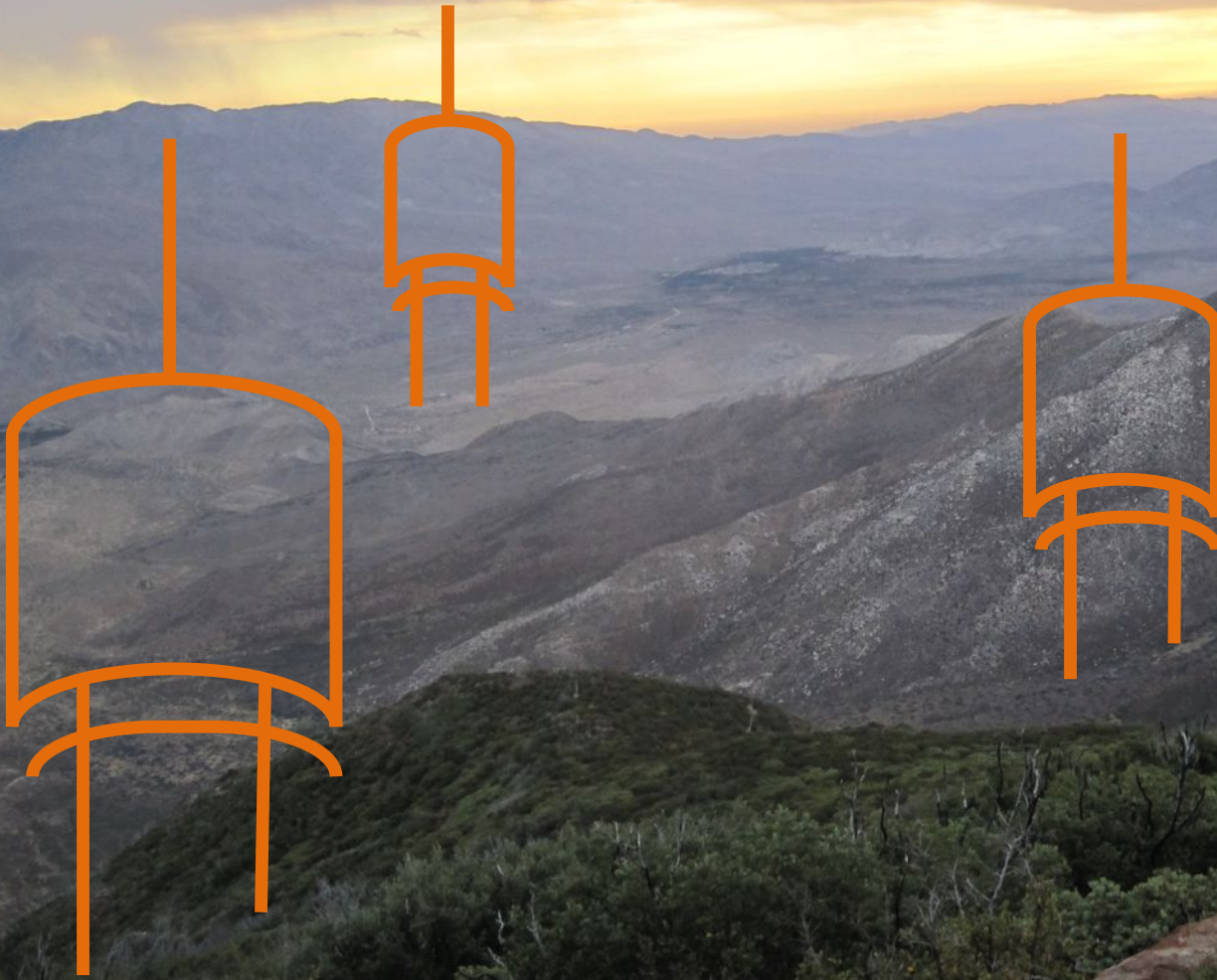
Truth table:

Build from earlier gates:

Often used as a one-bit comparator.

*Video game designers, Halloween costumers extraordinaire, sci-fi/fantasy screenwriters,  
I have an idea...*

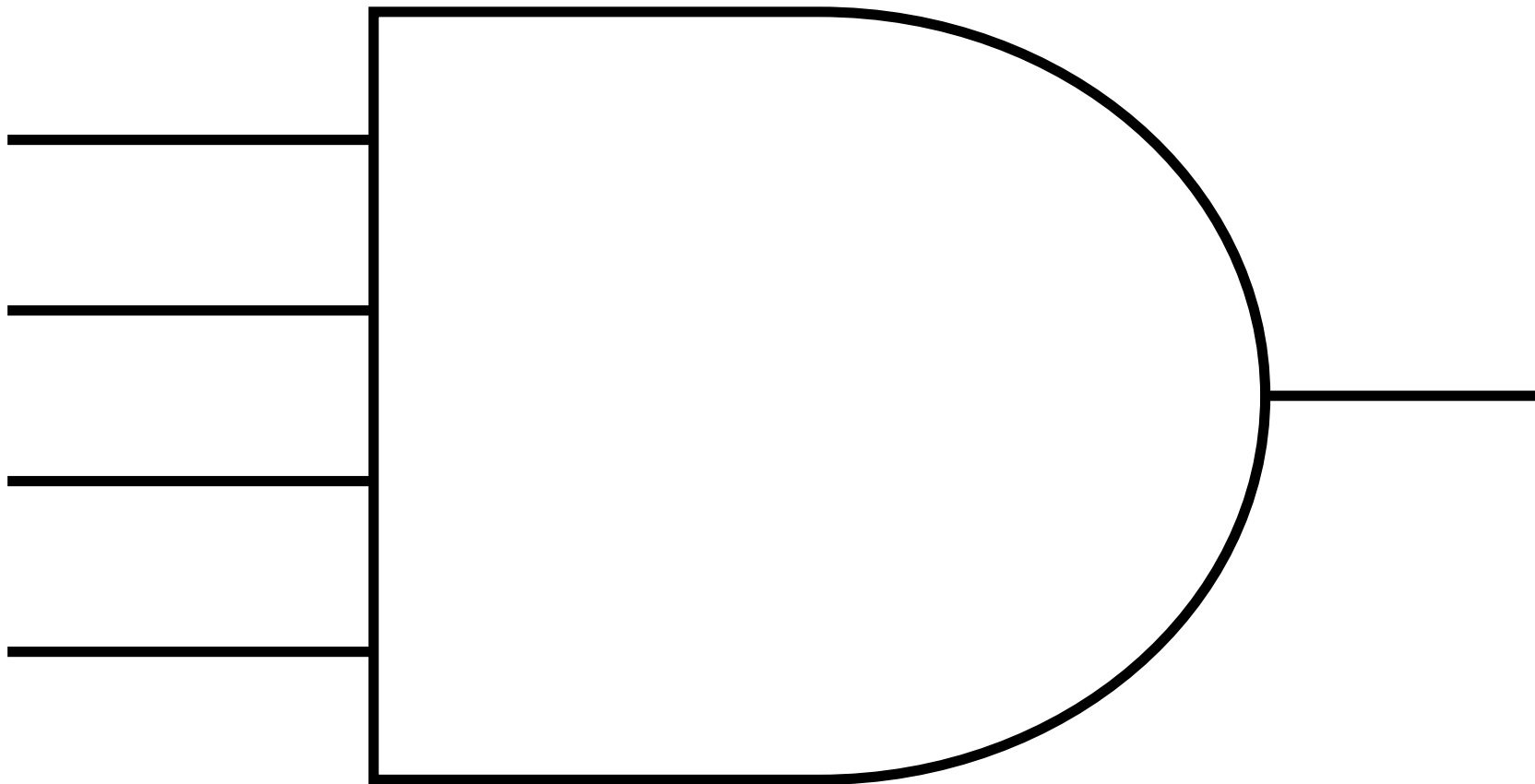
# GATES OF XOR





# Larger gates

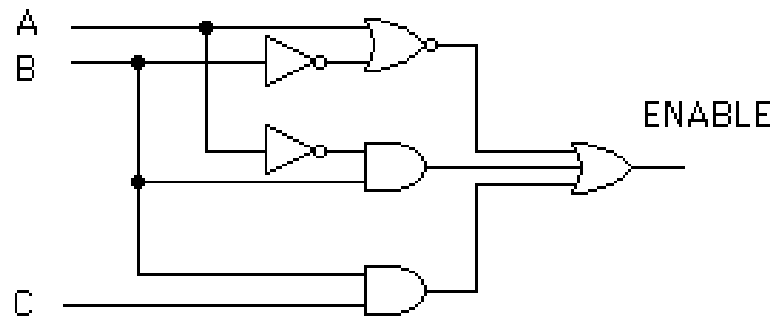
Build a 4-input AND gate using any number of 2-input gates.





# Circuit simplification

Is there a simpler circuit that performs the same function?



Start with an equivalent Boolean expression, then simplify with algebra.

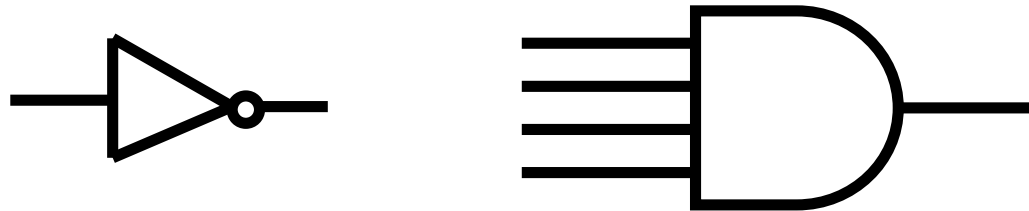
$$F(A, B, C) =$$

Check the answer with a truth table.



# Circuit derivation: *code detectors*

AND gate + NOT gates = code detector, recognizes exactly one input code.



Design a 4-input code detector to output 1 if ABCD = 1001, and 0 otherwise.



Design a 4-input code detector to accept two codes (ABCD=1001, ABCD=1111) and reject all others. (accept = 1, reject = 0)



# Circuit derivation: *sum-of-products* form

logical sum (OR)

of products (AND)

of inputs or their complements (NOT)

**Draw the truth table and design a sum-of-products circuit** for a 4-input code detector to accept two codes (ABCD=1001, ABCD=1111) and reject all others.

**How are the truth table and the sum-of-products circuit related?**



# Voting machines

A **majority circuit** outputs 1 if and only if a majority of its inputs equal 1.

Design a majority circuit for three inputs. **Use a sum of products.**

A	B	C	Majority
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

## Triply redundant computers in spacecraft

- Space program also hastened Integrated Circuits.

# Computers

- Manual calculations
- powered all early US **space** missions.
- Facilitated transition to digital computers.



**Mary Jackson**



**Katherine Johnson**

- Supported Mercury, Apollo, Space Shuttle, ...

**Dorothy Vaughn**

- First black supervisor within NACA
- Early self-taught FORTRAN programmer for NASA move to digital computers.





# Early pioneers in reliable computing

## Apollo 11 code print-out



### Katherine Johnson

- Calculated first US human space flight trajectories
- Mercury, Apollo 11, Space Shuttle, ...
- Reputation for accuracy in manual calculations, verified early code
- Called to verify results of code for launch calculations for first US human in orbit
- Backup calculations helped save Apollo 13
- Presidential Medal of Freedom 2015



### Margaret Hamilton

- Led software team for Apollo 11 Guidance Computer, averted mission abort on first moon landing.
- Coined “software engineering”, developed techniques for correctness and reliability.
- Presidential Medal of Freedom 2016

