Gates and Logic: From Transistors to Logic Gates and Logic Circuits

Prof. Hakim Weatherspoon
CS 3410
Computer Science
Cornell University

[Weatherspoon, Bala, Bracy, and Sirer]

Goals for Today

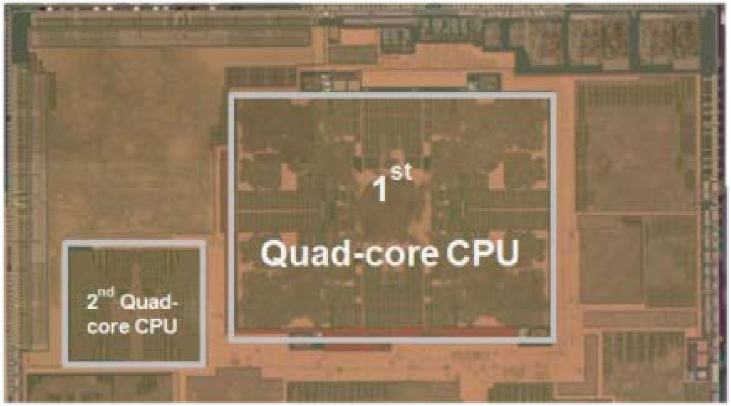
- From Switches to Logic Gates to Logic Circuits
- Understanding the foundations of
  - Computer Systems Organization and Programming

Goals for Today

- From Switches to Logic Gates to Logic Circuits
- Understanding the foundations of
  - Computer Systems Organization and Programming
  - e.g. Galaxy Note 9

Goals for Today

- From Switches to Logic Gates to Logic Circuits
- Understanding the foundations of
  - Computer Systems Organization and Programming
  - e.g. Galaxy Note 9
  - with the big.LITTLE DynamicIQ 8-core ARM processor



Goals for Today

- From Switches to Logic Gates to Logic Circuits
- Understanding the foundations of
  - Computer Systems Organization and Programming
  - e.g. Galaxy Note 9
  - with the big.LITTLE DynamicIQ 8-core ARM processor

	big Quad Core	LITTLE Quad Core
Architecture	ARM v8	ARM v8
Process	Samsung 10nm	Samsung 10nm
Frequency	2.9GHz+	1.9GHz
Area	3.5mm2	
Power-ratio	1	0.17
L1 Cache Size	64 KB I/D Cache	64 KB I/D Cache
L2 Cache Size	2 MB Data Cache	512 KB Data Cache

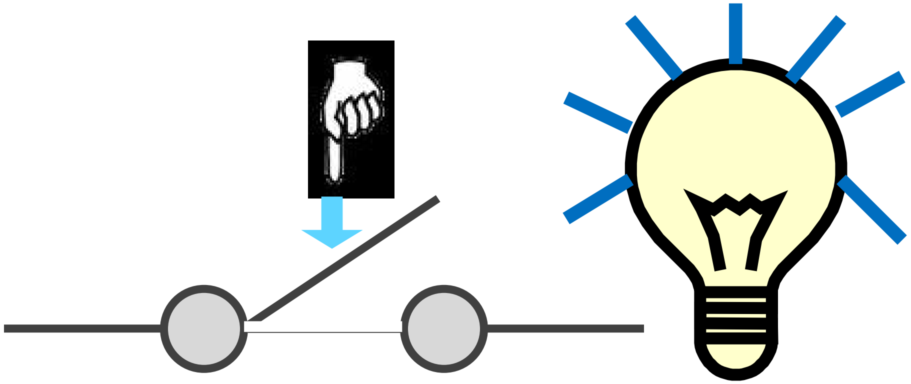
Goals for Today

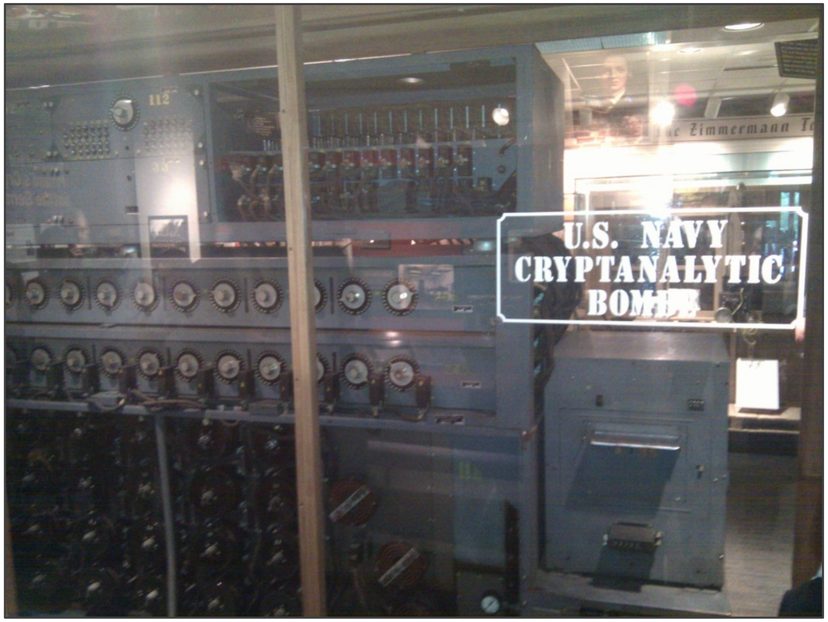
- From Switches to Logic Gates to Logic Circuits
- Logic Gates
  - From switches
  - Truth Tables
- Logic Circuits
  - From Truth Tables to Circuits (Sum of Products)
  - Identity Laws
- Logic Circuit Minimization
  - Algebraic Manipulations
  - Truth Tables (Karnaugh Maps)
- Transistors (electronic switch)

A switch

Acts as a *conductor* or *insulator*.

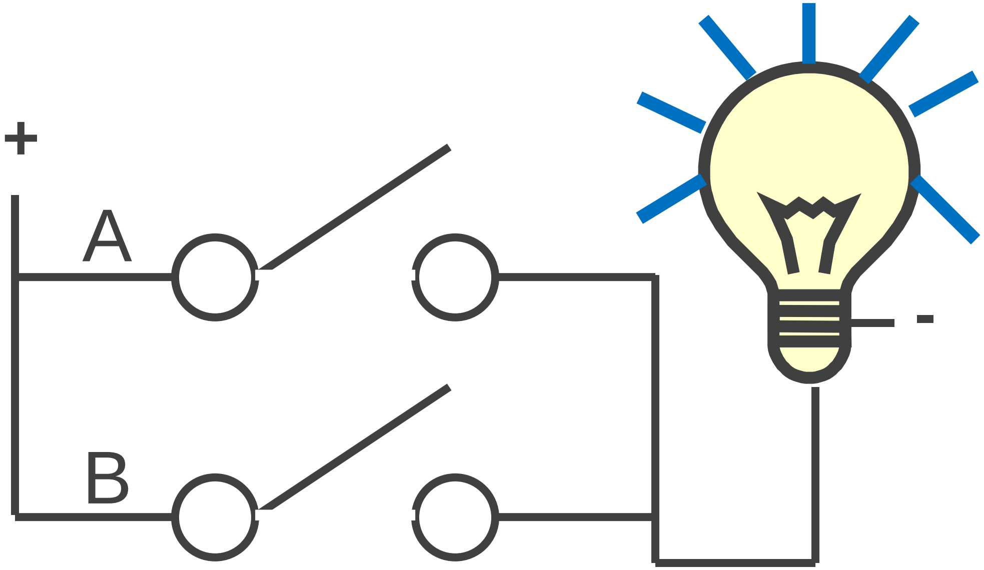
Can be used to build amazing things…





The Bombe used to break the German Enigma machine during World War II

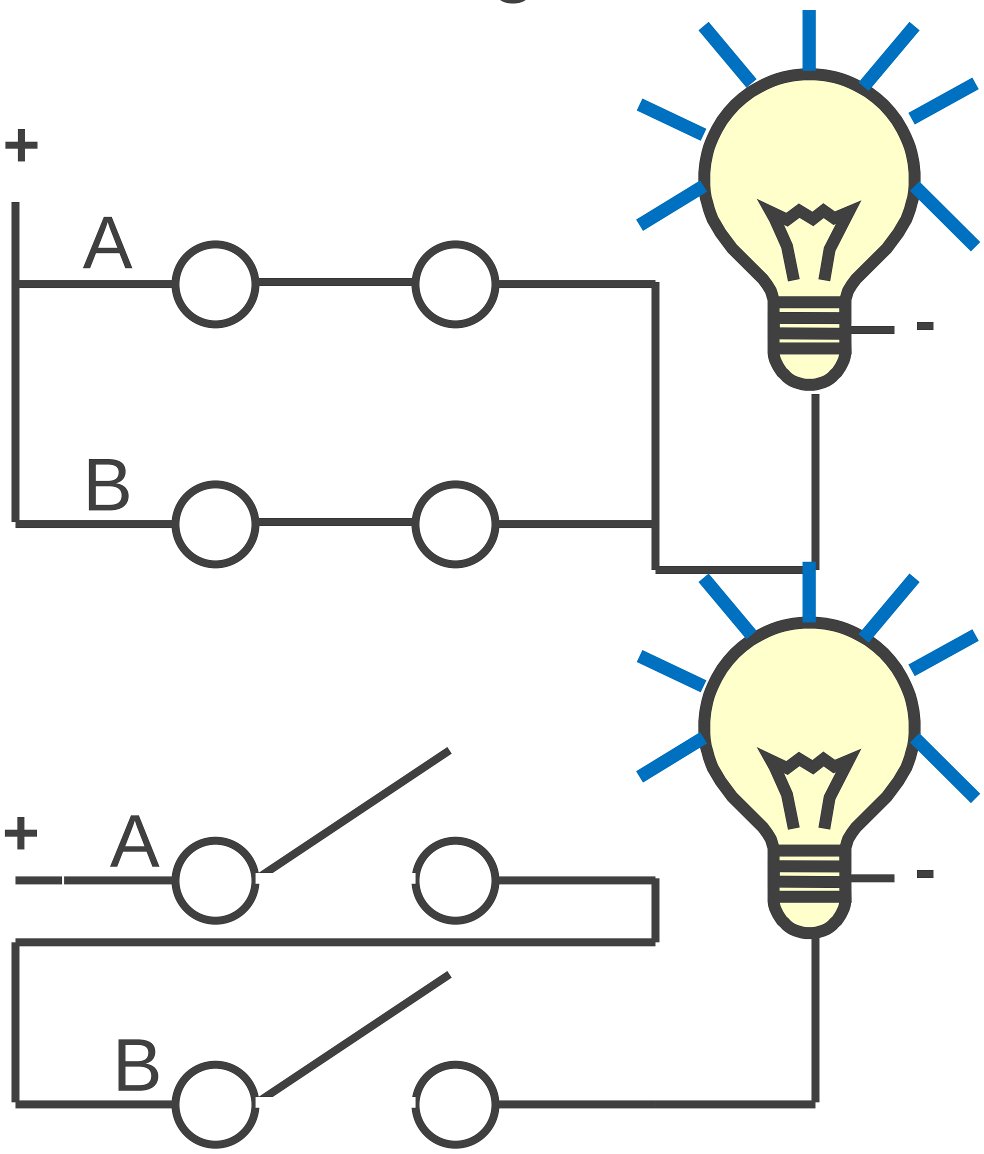
Basic Building Blocks: Switches to Logic Gates



Truth Table

A	B	Light
OFF	OFF	
OFF	ON	
ON	OFF	
ON	ON	

Basic Building Blocks: Switches to Logic Gates

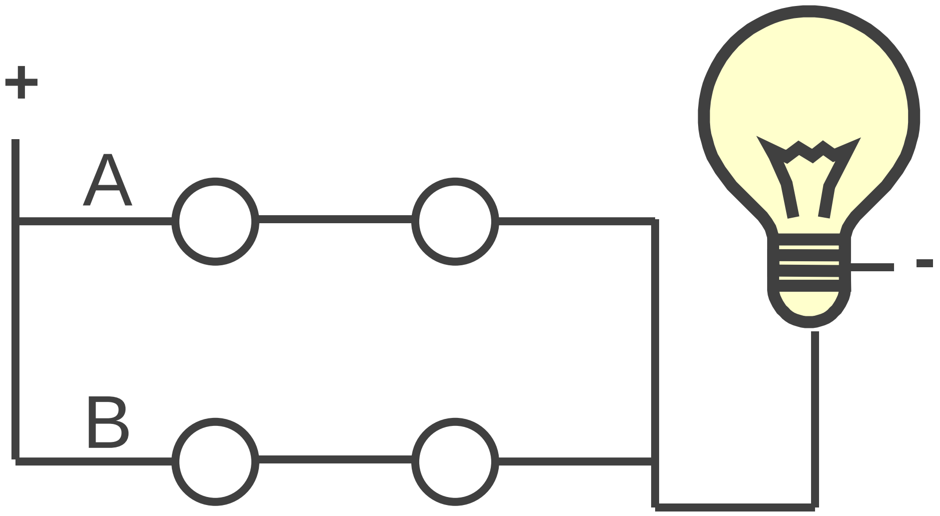


Truth Table

A	B	Light
OFF	OFF	OFF
OFF	ON	ON
ON	OFF	ON
ON	ON	ON

A	B	Light
OFF	OFF	
OFF	ON	
ON	OFF	
ON	ON	

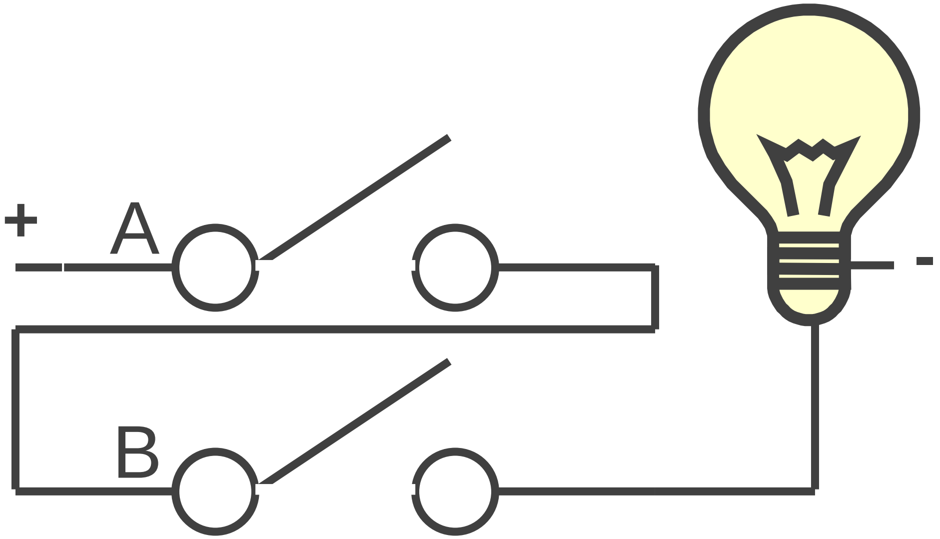
Basic Building Blocks: Switches to Logic Gates



- Either (OR)

Truth Table

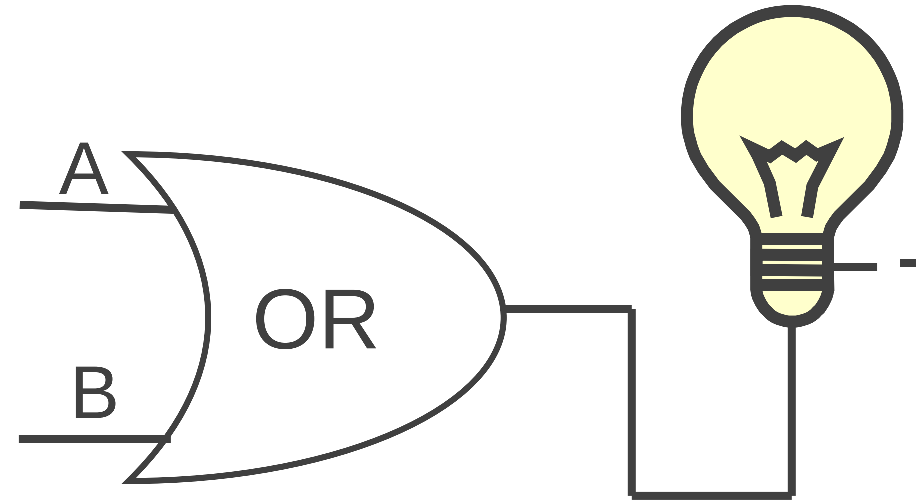
A	B	Light
OFF	OFF	OFF
OFF	ON	ON
ON	OFF	ON
ON	ON	ON



- Both (AND)

A	B	Light
OFF	OFF	OFF
OFF	ON	OFF
ON	OFF	OFF
ON	ON	ON

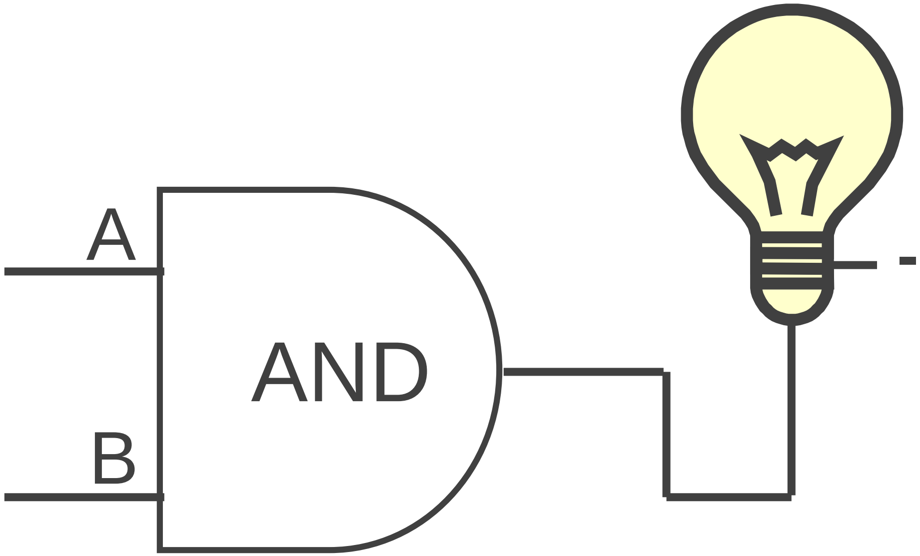
Basic Building Blocks: Switches to Logic Gates



- Either (OR)

Truth Table

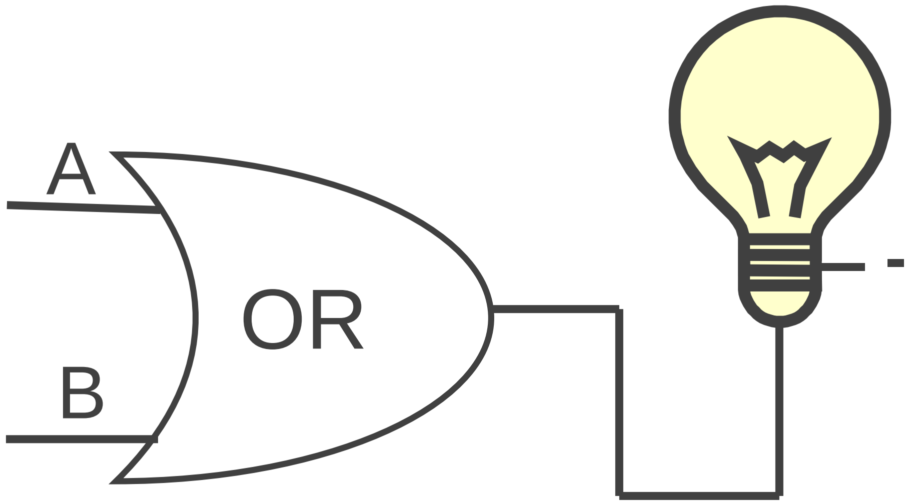
A	B	Light
OFF	OFF	OFF
OFF	ON	ON
ON	OFF	ON
ON	ON	ON



- Both (AND)

A	B	Light
OFF	OFF	OFF
OFF	ON	OFF
ON	OFF	OFF
ON	ON	ON

Basic Building Blocks: Switches to Logic Gates

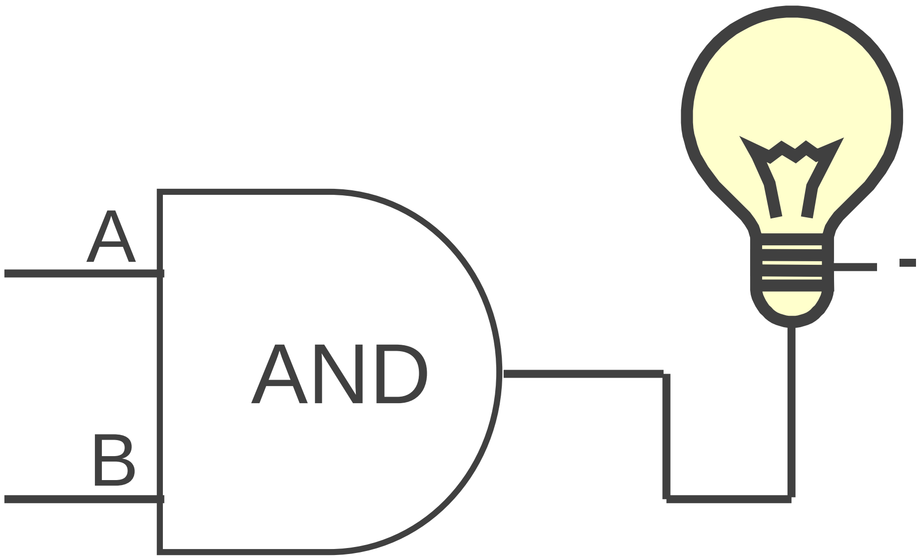


- Either (OR)

Truth Table

A	B	Light
0	0	0
0	1	1
1	0	1
1	1	1

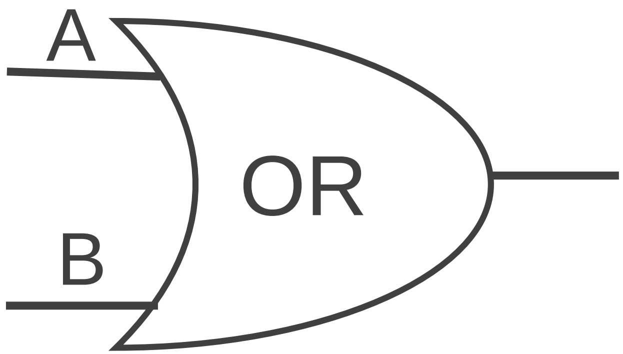
0 = OFF
1 = ON



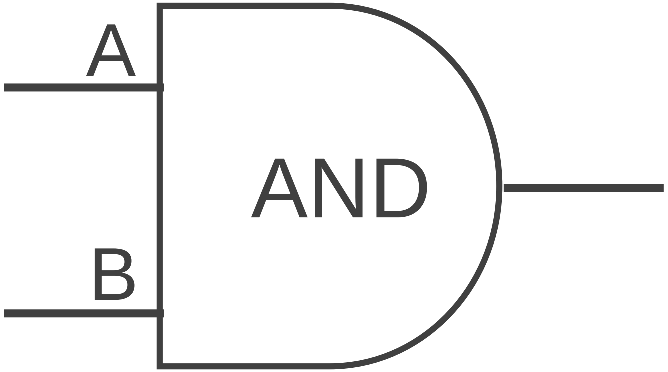
- Both (AND)

A	B	Light
0	0	0
0	1	0
1	0	0
1	1	1

Basic Building Blocks: Switches to Logic Gates



George Boole (1815-1864)



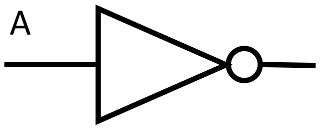
- Did you know?
- George Boole: Inventor of the idea of logic gates. He was born in Lincoln, England and he was the son of a shoemaker in a low class family.

Takeaway

- Binary (two symbols: true and false) is the basis of Logic Design

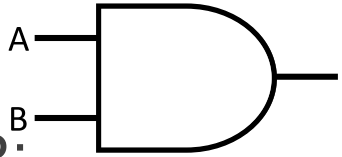
Building Functions: Logic Gates

- NOT:



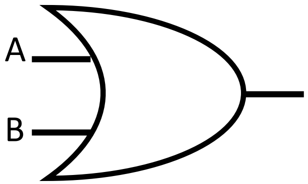
A	Out

- AND:



A	B	Out
0	0	0
0	1	0
1	0	0
1	1	1

- OR:



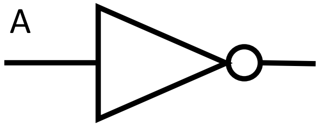
A	B	Out
0	0	0
0	1	1
1	0	1
1	1	1

- Logic Gates
  - digital circuit that either allows a signal to pass through it or not.
  - Used to build logic functions
  - There are seven basic logic gates:

    AND, OR, **NOT**,

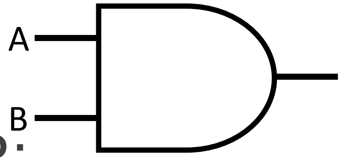
Building Functions: Logic Gates

- NOT:



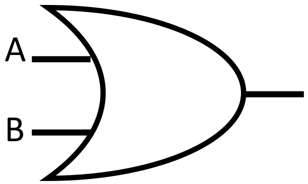
A	Out
0	1
1	0

- AND:



A	B	Out
0	0	0
0	1	0
1	0	0
1	1	1

- OR:



A	B	Out
0	0	0
0	1	1
1	0	1
1	1	1

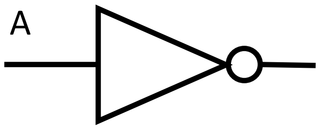
- Logic Gates
  - digital circuit that either allows a signal to pass through it or not.
  - Used to build logic functions
  - There are seven basic logic gates:

    AND, OR, **NOT**,

    NAND (not AND), NOR (not OR), XOR, and XNOR (not XOR) [later]

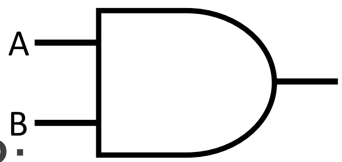
Building Functions: Logic Gates

- NOT:



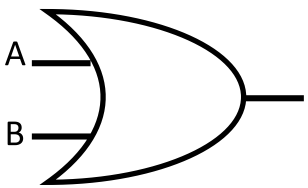
A	Out
0	1
1	0

- AND:



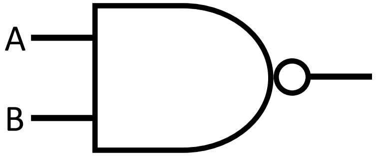
A	B	Out
0	0	0
0	1	0
1	0	0
1	1	1

- OR:



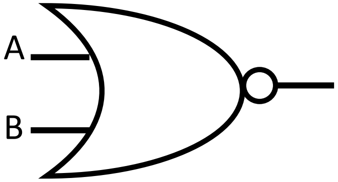
A	B	Out
0	0	0
0	1	1
1	0	1
1	1	1

NAND:



A	B	Out
0	0	1
0	1	1
1	0	1
1	1	0

NOR:



A	B	Out
0	0	1
0	1	0
1	0	0
1	1	0

- Logic Gates
  - digital circuit that either allows a signal to pass through it or not.
  - Used to build logic functions
  - There are seven basic logic gates:

    AND, OR, **NOT**,

    NAND (not AND), NOR (not OR), XOR, and XNOR (not XOR) [later]

Which Gate is this?

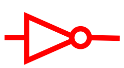
iClicker Question

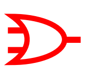
Function:

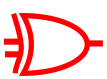
Symbol:

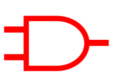
Truth Table:

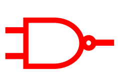
a	b	Out

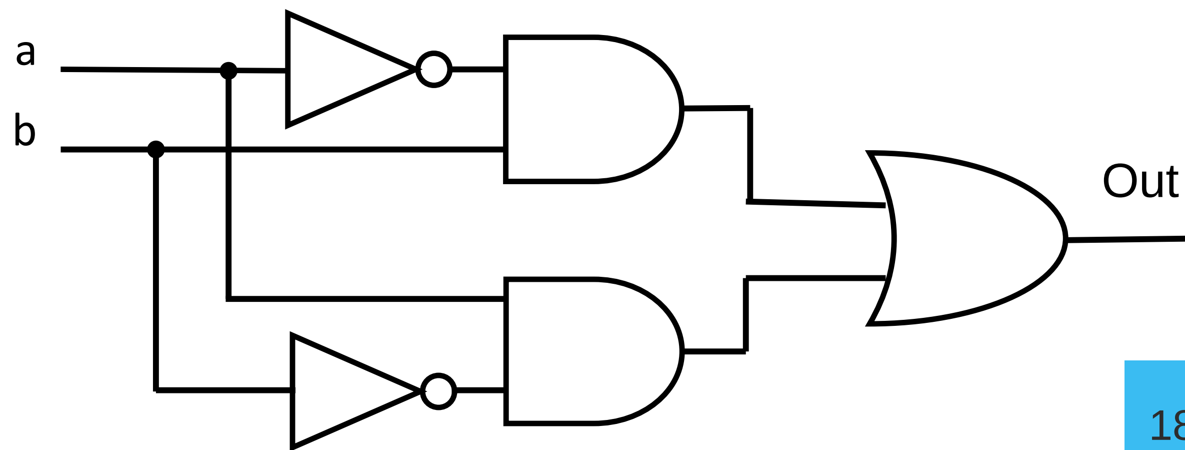
(A) NOT 

(B) OR 

(C) XOR 

(D) AND 

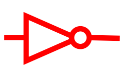
(E) NAND 

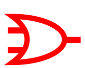


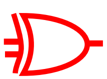
Which Gate is this? iClicker Question

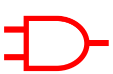
- XOR: out = 1 if a or b is 1, but not both;
- out = 0 otherwise.
- out = 1, only if a = 1 AND b = 0
- OR a = 0 AND b = 1

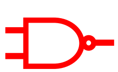
a	b	Out
0	0	0
0	1	1
1	0	1
1	1	0

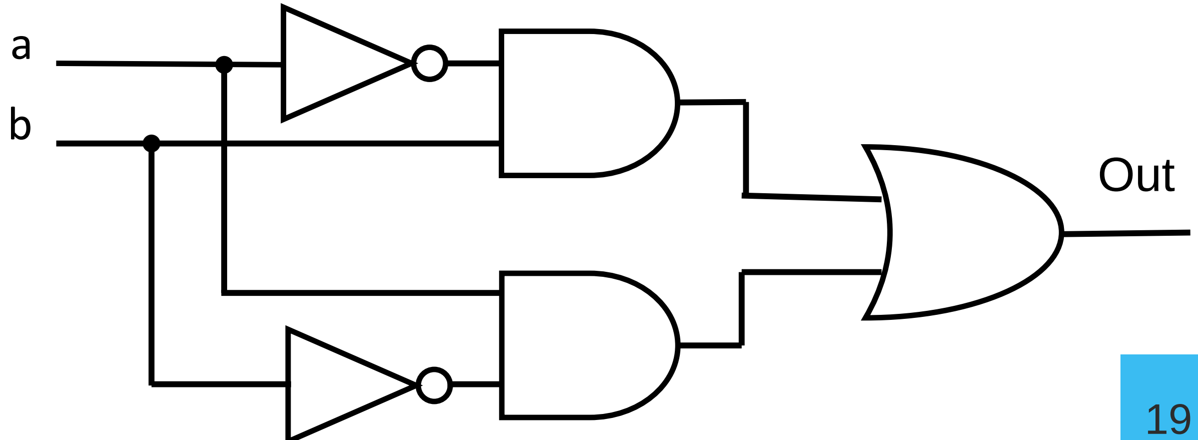
(A) NOT 

(B) OR 

(C) XOR 

(D) AND 

(E) NAND 

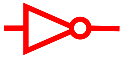
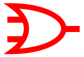
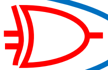
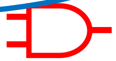
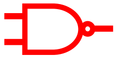


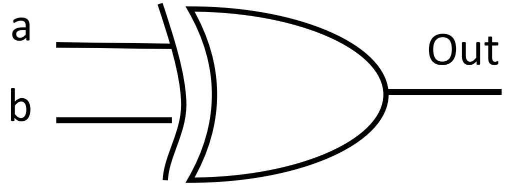
Which Gate is this?

iClicker Question

- XOR: out = 1 if a or b is 1, but not both;
- out = 0 otherwise.
- out = 1, only if a = 1 AND b = 0
- OR a = 0 AND b = 1

a	b	Out
0	0	0
0	1	1
1	0	1
1	1	0

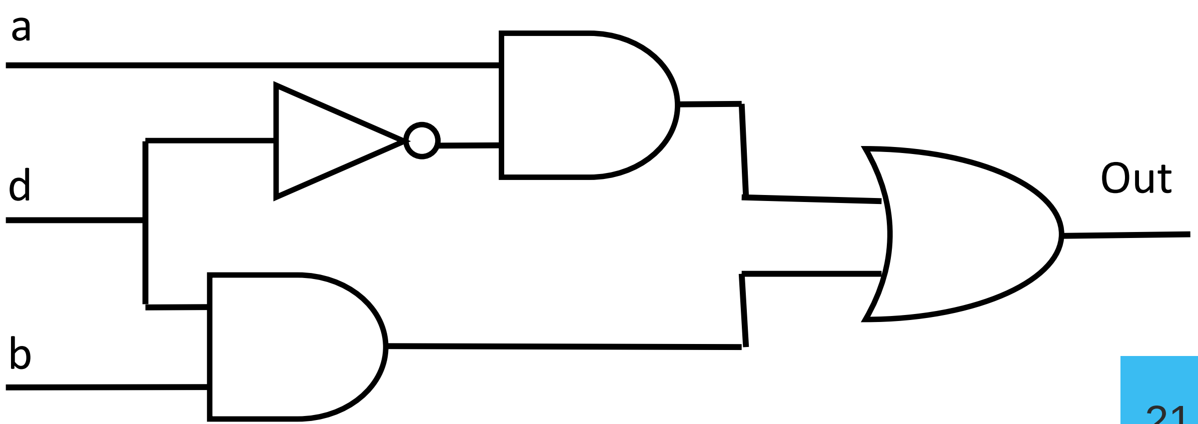
(A) NOT 
(B) OR 
(C) XOR 
(D) AND 
(E) NAND 



Activity#2: Logic Gates

- Fill in the truth table, given the following Logic Circuit made from Logic AND, OR, and NOT gates.
- What does the logic circuit do?

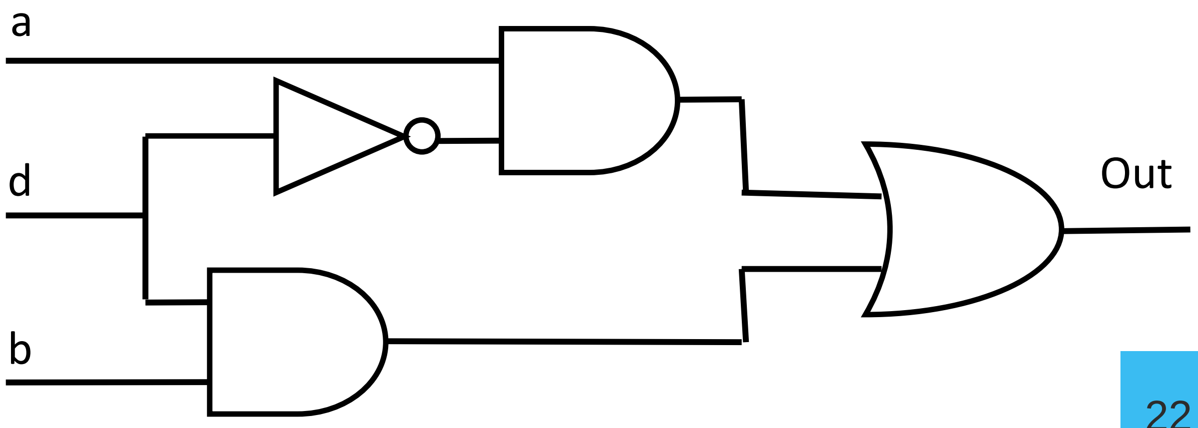
a	b	d	Out
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	



Activity#2: Logic Gates

- Multiplexor: select (d) between two inputs (a and b) and set one as the output (out)?
- out = a, if d = 0
- out = b, if d = 1

a	b	d	Out
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1



Goals for Today

- From Switches to Logic Gates to Logic Circuits
- Logic Gates
  - From switches
  - Truth Tables
- Logic Circuits
  - From Truth Tables to Circuits (Sum of Products)
  - Identity Laws
- Logic Circuit Minimization
  - Algebraic Manipulations
  - Truth Tables (Karnaugh Maps)
- Transistors (electronic switch)

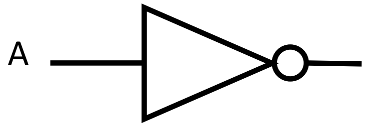
Next Goal

- Given a Logic function, create a Logic Circuit that implements the Logic Function…
- …and, *with the minimum number of logic gates*

- Fewer gates: A cheaper ($$$) circuit!

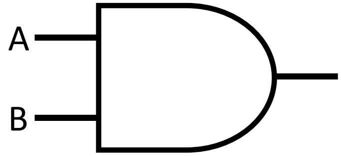
Logic Gates

NOT:



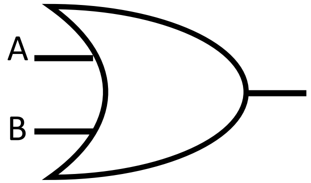
A	Out
0	1
1	0

AND:



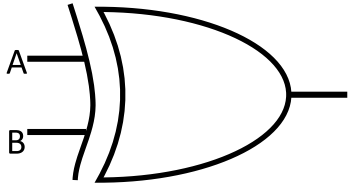
A	B	Out
0	0	0
0	1	0
1	0	0
1	1	1

OR:



A	B	Out
0	0	0
0	1	1
1	0	1
1	1	1

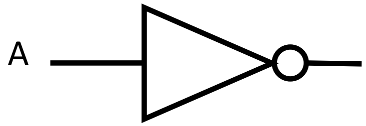
XOR:



A	B	Out
0	0	0
0	1	1
1	0	1
1	1	0

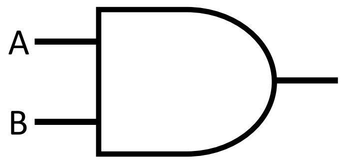
Logic Gates

NOT:



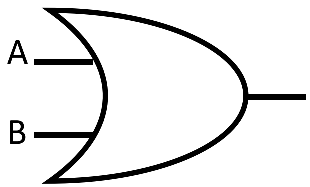
A	Out
0	1
1	0

AND:



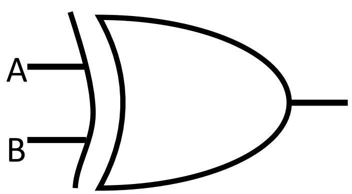
A	B	Out
0	0	0
0	1	0
1	0	0
1	1	1

OR:



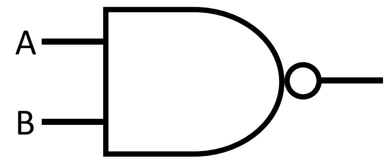
A	B	Out
0	0	0
0	1	1
1	0	1
1	1	1

XOR:



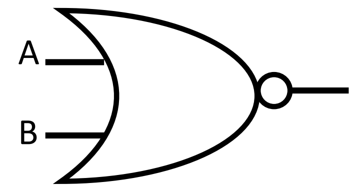
A	B	Out
0	0	0
0	1	1
1	0	1
1	1	0

NAND:



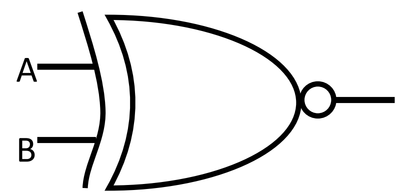
A	B	Out
0	0	1
0	1	1
1	0	1
1	1	0

NOR:



A	B	Out
0	0	1
0	1	0
1	0	0
1	1	0

XNOR:



A	B	Out
0	0	1
0	1	0
1	0	0
1	1	1

Logic Implementation

- How to implement a desired logic function?

a	b	c	out
0	**0**	**0**	**0**
0	**0**	**1**	**1**
0	**1**	**0**	**0**
0	**1**	**1**	**1**
1	**0**	**0**	**0**
1	**0**	**1**	**1**
1	**1**	**0**	**0**
1	**1**	**1**	**0**

Logic Implementation

- How to implement a desired logic function?

a	b	c	out	minterm
0	**0**	**0**	**0**	$\bar{a}\,\bar{b}\,\bar{c}$
0	**0**	**1**	**1**	$\bar{a}\,\bar{b}\,c$
0	**1**	**0**	**0**	$\bar{a}\,b\,\bar{c}$
0	**1**	**1**	**1**	$\bar{a}\,b\,c$
1	**0**	**0**	**0**	$a\,\bar{b}\,\bar{c}$
1	**0**	**1**	**1**	$a\,\bar{b}\,c$
1	**1**	**0**	**0**	$a\,b\,\bar{c}$
1	**1**	**1**	**0**	$a\,b\,c$

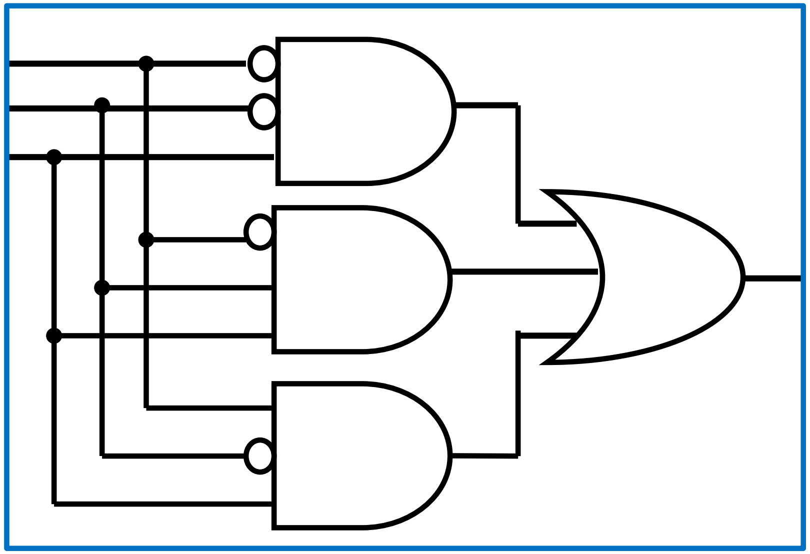
1) Write minterms
2) sum of products:
   - OR of all minterms where out=1

Logic Implementation

- How to implement a desired logic function?

a	b	c	out	minterm
0	**0**	**0**	**0**	$\bar{a}\,\bar{b}\,\bar{c}$
0	**0**	**1**	**1**	$\bar{a}\,\bar{b}\,c$
0	**1**	**0**	**0**	$\bar{a}\,b\,\bar{c}$
0	**1**	**1**	**1**	$\bar{a}\,b\,c$
1	**0**	**0**	**0**	$a\,\bar{b}\,\bar{c}$
1	**0**	**1**	**1**	$a\,\bar{b}\,c$
1	**1**	**0**	**0**	$a\,b\,\bar{c}$
1	**1**	**1**	**0**	$a\,b\,c$

1) Write minterms
2) sum of products:
   - OR of all minterms where out=1
     - E.g. out = $\overline{ab}c + \bar{a}bc + a\bar{b}c$



corollary: *any* combinational circuit *can be* implemented in two levels of logic (ignoring inverters)

Logic Equations

- NOT:
  - out = $\bar{a}$ = !a = $\neg$a
- AND:
  - out = $a \cdot b$ = a & b = $a \wedge b$
- OR:
  - out = $a + b$ = a | b = $a \vee b$
- XOR:
  - out = $a \oplus b = a\bar{b} + \bar{a}b$
- Logic Equations
  - Constants: true = 1, false = 0
  - Variables: a, b, out, …
  - Operators (above): AND, OR, NOT, etc.

Logic Equations

- NOT:
  - out = $\bar{a}$ = !a = ¬a
- AND:
  - out = a · b = a & b = a ∧ b
- OR:
  - out = a + b = a | b = a ∨ b
- XOR:
  - out = a ⊕ b = $a\bar{b} + \bar{a}b$

NAND:
- out = $\overline{a \cdot b}$ = !(a & b) = ¬ (a ∧ b)

NOR:
- out = $\overline{a + b}$ = !(a | b) = ¬ (a ∨ b)

XNOR:
- out = $\overline{a \oplus b}$ = ab + $\overline{ab}$

- Logic Equations
  - Constants: true = 1, false = 0
  - Variables: a, b, out, …
  - Operators (above): AND, OR, NOT, etc.

Identities

Identities useful for manipulating logic equations

- For optimization & ease of implementation

$a + 0 =$

$a + 1 =$

$a + \bar{a} =$

$a \cdot 0 =$

$a \cdot 1 =$

$a \cdot \bar{a} =$

Identities

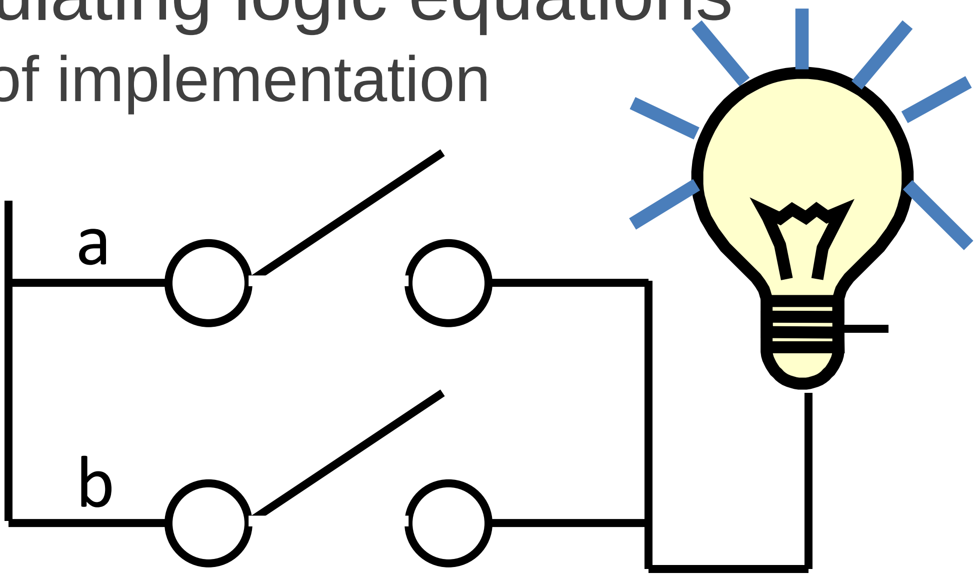
Identities useful for manipulating logic equations

– For optimization & ease of implementation

a + 0 = a

a + 1 = 1

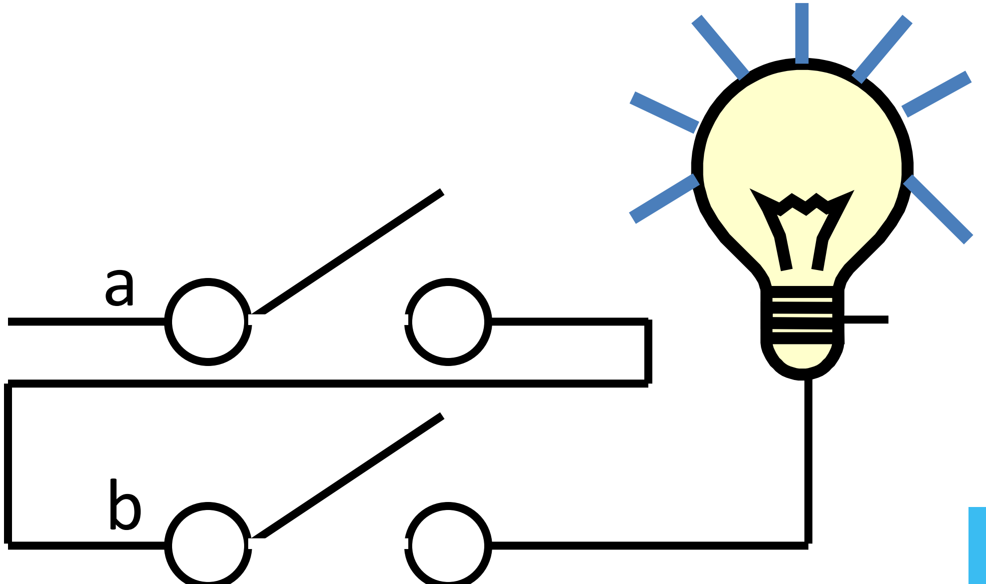
a + ā = 1



a · 0 = 0

a · 1 = a

a · ā = 0



Identities

Identities useful for manipulating logic equations

- For optimization & ease of implementation

$\overline{(a + b)}$ =

$\overline{(a \cdot b)}$ =

a + a b =

a(b+c) =

$\overline{a(b + c)}$ =

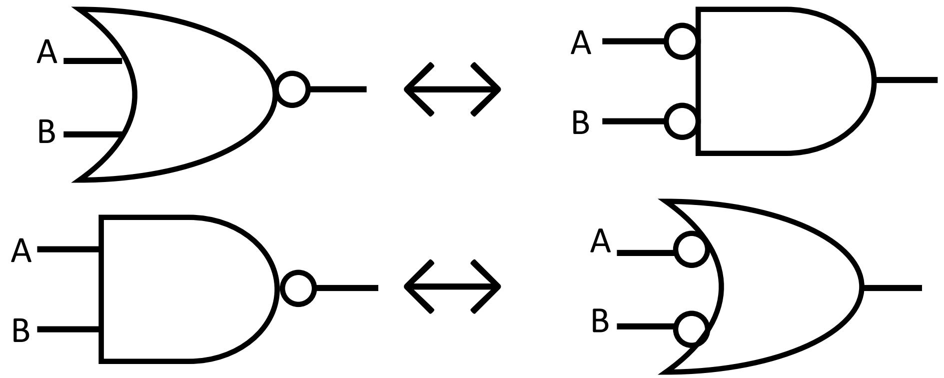
Identities

Identities useful for manipulating logic equations

– For optimization & ease of implementation

$\overline{(a + b)} = \bar{a} \cdot \bar{b}$

$\overline{(a \cdot b)} = \bar{a} + \bar{b}$



$a + a\,b = a$

$a(b+c) = ab + ac$

$\overline{a(b + c)} = \bar{a} + \bar{b} \cdot \bar{c}$

Goals for Today

- From Switches to Logic Gates to Logic Circuits
- Logic Gates
  - From switches
  - Truth Tables
- Logic Circuits
  - From Truth Tables to Circuits (Sum of Products)
  - Identity Laws
- **Logic Circuit Minimization – *why?***
  - Algebraic Manipulations
  - Truth Tables (Karnaugh Maps)
- Transistors (electronic switch)

Minimization Example

$a + 0 = a$
$a + 1 = 1$
$a + \bar{a} = 1$
$a \cdot 0 = 0$
$a \cdot 1 = a$
$a \cdot \bar{a} = 0$

$a + a b = a$
$a (b+c) = ab + ac$

$$\overline{(a+b)} = \overline{a} \bullet \overline{b}$$
$$\overline{(ab)} = \overline{a} + \overline{b}$$
$$\overline{a(b+c)} = \overline{a} + \overline{b} \bullet \overline{c}$$

Minimize this logic equation:

$(a+b)(a+c) = (a+b)a + (a+b)c$
$= aa + ba + ac + bc$
$= a + a(b+c) + bc$
$= a + bc$

iClicker Question

a + 0 = a
a + 1 = 1
a + ā = 1
a · 0 = 0
a · 1 = a
a · ā = 0

a + a b = a
a (b+c) = ab + ac

$$\overline{(a+b)} = \overline{a} \bullet \overline{b}$$

$$\overline{(ab)} = \overline{a} + \overline{b}$$

$$\overline{a(b+c)} = \overline{a} + \overline{b} \bullet \overline{c}$$

(a+b)(a+c) → a + bc

How many gates were required before and after?

BEFORE	AFTER
(A) 2 OR	1 OR
(B) 2 OR, 1 AND	2 OR
(C) 2 OR, 1 AND	1 OR, 1 AND
(D) 2 OR, 2 AND	2 OR
(E) 2 OR, 2 AND	2 OR, 1 AND

iClicker Question

a + 0 = a
a + 1 = 1
a + ā = 1
a · 0 = 0
a · 1 = a
a · ā = 0

a + a b = a
a (b+c) = ab + ac

$$\overline{(a+b)} = \overline{a} \bullet \overline{b}$$

$$\overline{(ab)} = \overline{a} + \overline{b}$$

$$\overline{a(b+c)} = \overline{a} + \overline{b} \bullet \overline{c}$$

(a+b)(a+c) → a + bc

How many gates were required before and after?

BEFORE	AFTER
(A) 2 OR	1 OR
(B) 2 OR, 1 AND	2 OR
(C) 2 OR, 1 AND	1 OR, 1 AND
(D) 2 OR, 2 AND	2 OR
(E) 2 OR, 2 AND	2 OR, 1 AND

Checking Equality w/Truth Tables

circuits ↔ truth tables ↔ equations

Example: (a+b)(a+c) = a + bc

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

Checking Equality w/Truth Tables

circuits ↔ truth tables ↔ equations

Example: $(a+b)(a+c) = a + bc$

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

Checking Equality w/Truth Tables

circuits ↔ truth tables ↔ equations

Example: (a+b)(a+c) = a + bc

a	b	c	a+b	a+c	LHS	bc	RHS
0	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	1	1	1	1
1	0	0	1	1	1	0	1
1	0	1	1	1	1	0	1
1	1	0	1	1	1	0	1
1	1	1	1	1	1	1	1

Takeaway

- Binary (two symbols: true and false) is the basis of Logic Design

- More than one Logic Circuit can implement same Logic function. Use Algebra (Identities) or Truth Tables to show equivalence.

Goals for Today

- From Switches to Logic Gates to Logic Circuits
- Logic Gates
  - From switches
  - Truth Tables
- Logic Circuits
  - From Truth Tables to Circuits (Sum of Products)
  - Identity Laws
- Logic Circuit Minimization
  - Algebraic Manipulations
  - Truth Tables (Karnaugh Maps)
- Transistors (electronic switch)

Karnaugh Maps

How does one find the most efficient equation?

- Manipulate algebraically until…?
- Use Karnaugh Maps (optimize visually)
- Use a software optimizer

For large circuits

- Decomposition & reuse of building blocks

Minimization with Karnaugh maps (1)

a	b	c	out
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	0
1	1	1	0

- Sum of minterms yields
  - out = -

Minimization with Karnaugh maps (2)

a	b	c	out
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	0
1	1	1	0

c \ ab	00	01	11	10
0				
1				

- Sum of minterms yields
  - out = $\bar{a}\bar{b}c + \bar{a}bc + a\bar{b}\bar{c} + a\bar{b}c$
- Karnaugh maps identify which inputs are (ir)relevant to the output

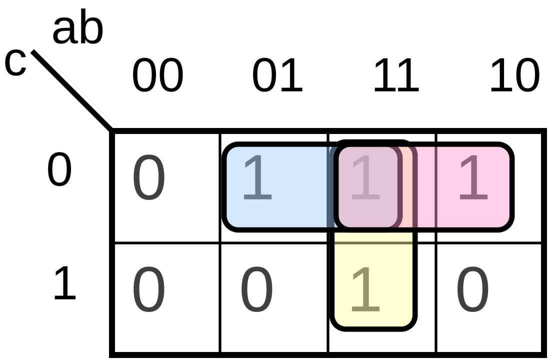
Minimization with Karnaugh maps (2)

a	b	c	out
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	0
1	1	1	0

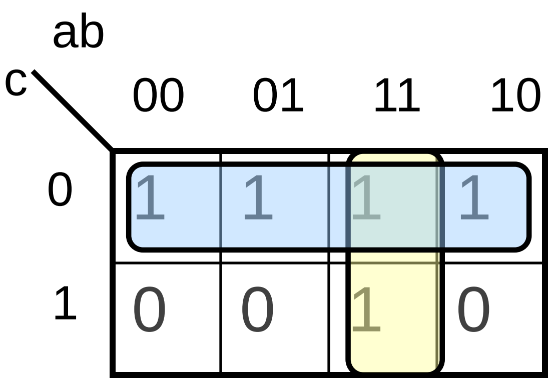
c \ ab	00	01	11	10
0	0	0	0	1
1	1	1	0	1

- Sum of minterms yields
  - out = $\overline{ab}c + \bar{a}bc + a\overline{bc} + a\bar{b}c$
- Karnaugh map minimization
  - Cover all 1's
  - Group adjacent blocks of 2^n 1's that yield a rectangular shape
  - Encode the common features of the rectangle
    - out =

Karnaugh Minimization Tricks (1)



- Minterms can overlap
  - out =



- Minterms can span 2, 4, 8 or more cells
  - out =

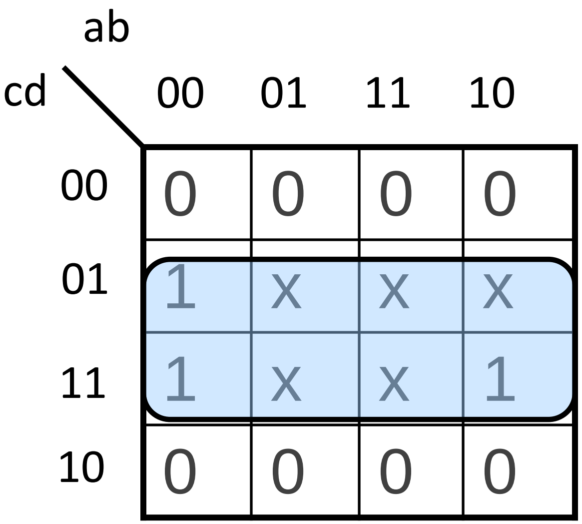
Karnaugh Minimization Tricks (2)

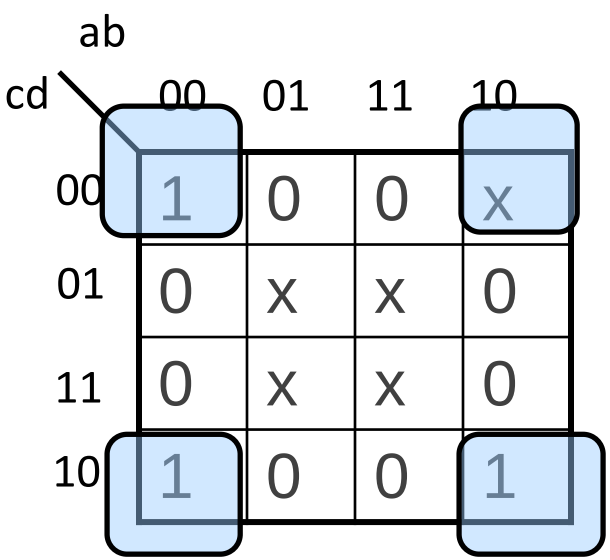
cd \ ab	00	01	11	10
00	0	0	0	0
01	1	0	0	1
11	1	0	0	1
10	0	0	0	0

cd \ ab	00	01	11	10
00	1	0	0	1
01	0	0	0	0
11	0	0	0	0
10	1	0	0	1

- The map wraps around
  - out =

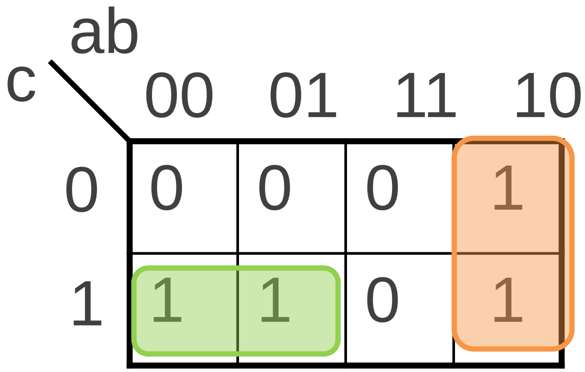
Karnaugh Minimization Tricks (3)





- "Don't care" values can be interpreted individually in whatever way is convenient
  - assume all x's = 1
  - out =
  - assume middle x's = 0
  - assume 4th column x = 1
  - out =

Minimization with K-Maps



(1) Circle the 1’s (see below)

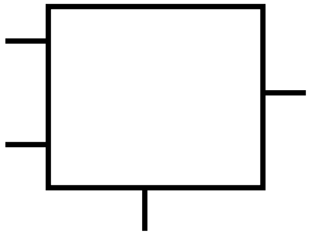
(2) Each circle is a logical component of the final equation

$= a\bar{b} + \bar{a}c$

Rules:

- Use fewest circles necessary to cover all 1’s
- Circles must cover *only* 1’s
- Circles span rectangles of size power of 2 (1, 2, 4, 8…)
- Circles should be as large as possible (all circles of 1?)
- Circles may wrap around edges of K-Map
- 1 may be circled multiple times *if* that means fewer circles

Multiplexer

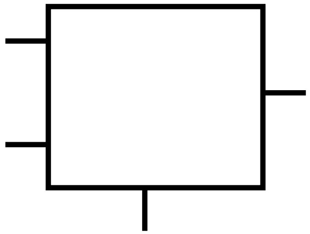


- A multiplexer selects between multiple inputs
  - out = a, if d = 0
  - out = b, if d = 1

a	b	d	out
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

- Build truth table
- Minimize diagram
- Derive logic diagram

Multiplexer Implementation

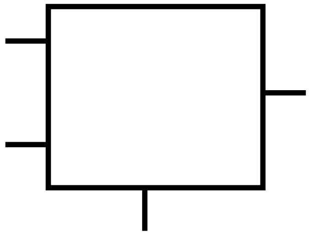


- Build a truth table

  $out = \bar{a}bd + a\overline{bd} + ab\bar{d} + abd$

a	b	d	out
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Multiplexer Implementation

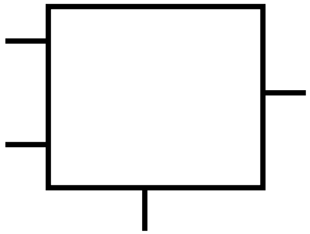


- Build the Karnaugh map

a	b	d	out
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

d \ ab	00	01	11	10
0				
1				

Multiplexer Implementation

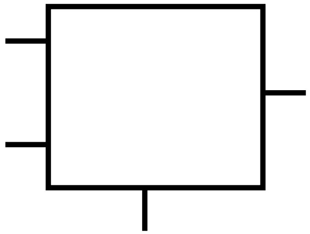


- Build the Karnaugh map

a	b	d	out
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

d \ ab	00	01	11	10
0	0	0	1	1
1	0	1	1	0

Multiplexer Implementation

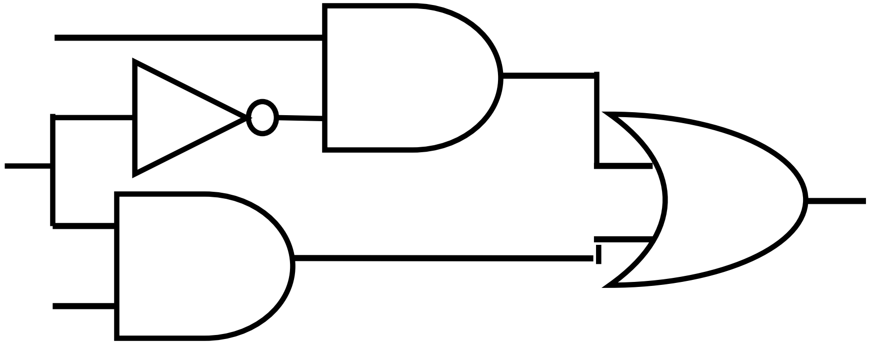


a	b	d	out
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

- Derive Minimal Logic Equation

d \ ab	00	01	11	10
0	0	0	1	1
1	0	1	1	0

- out = $a\bar{d} + bd$



Takeaway

- Binary (two symbols: true and false) is the basis of Logic Design
- More than one Logic Circuit can implement same Logic function. Use Algebra (Identities) or Truth Tables to show equivalence.
- Any logic function can be implemented as “sum of products”. Karnaugh Maps minimize number of gates.

Administrivia

- **Dates to keep in Mind**
  - Prelims: Tue Mar 5th and Thur May 2nd
  - Proj 1: Due Fri Feb 15th
  - Proj 2: Due Fri Mar 11th
  - Proj 3: Due Thur Mar 28th before Spring break
  - Final Project: Due Tue May 16th

iClicker

- Attempt to balance the iClicker graph

- Register iClicker
  - http://atcsupport.cit.cornell.edu/pollsrvc/

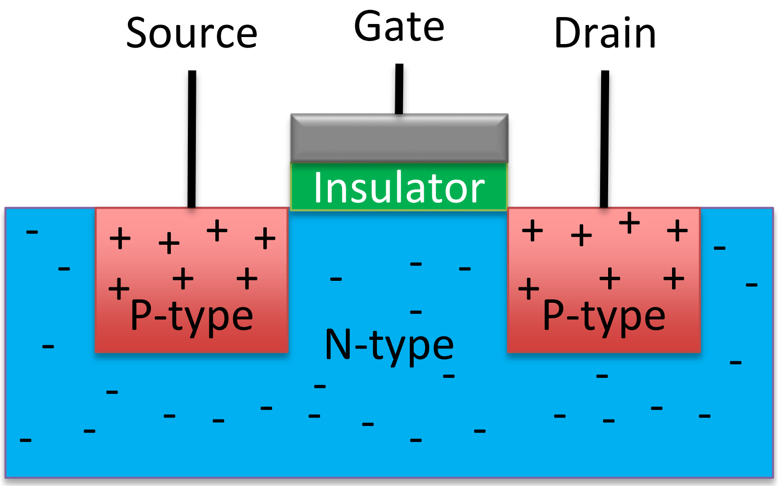
Goals for Today

- From Switches to Logic Gates to Logic Circuits
- Logic Gates
  - From switches
  - Truth Tables
- Logic Circuits
  - From Truth Tables to Circuits (Sum of Products)
  - Identity Laws
- Logic Circuit Minimization
  - Algebraic Manipulations
  - Truth Tables (Karnaugh Maps)
- Transistors (electronic switch)

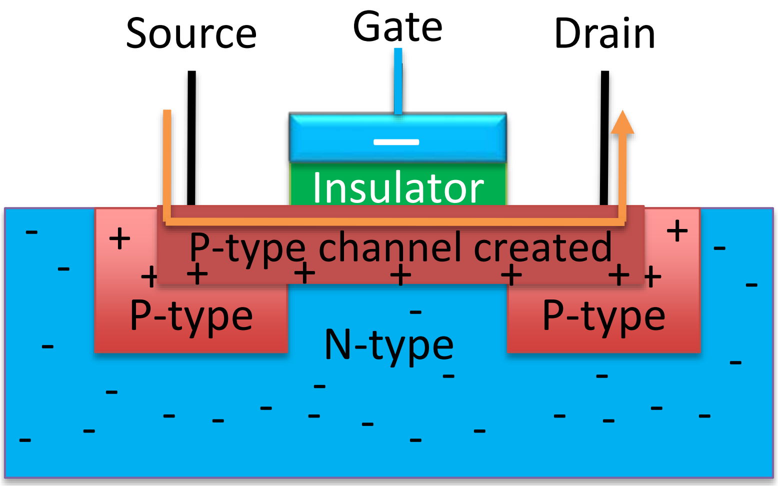
Silicon Valley & the Semiconductor Industry

- Transistors:
- Youtube video “How does a transistor work”
  https://www.youtube.com/watch?v=IcrBqCFLHIY
- Break: show some Transistor, Fab, Wafer photos

Transistors 101



P-Transistor Off



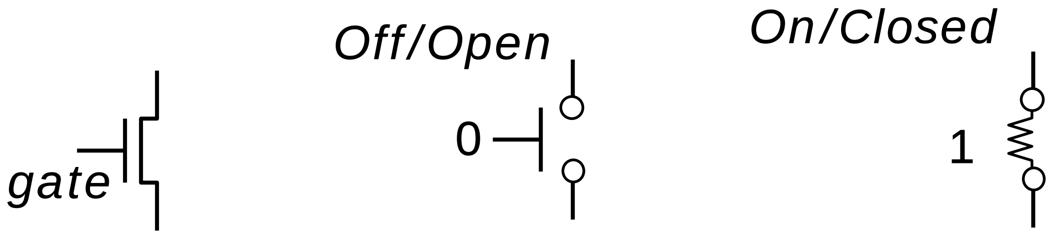
P-Transistor On

N-Type Silicon: negative free-carriers (electrons)
P-Type Silicon: positive free-carriers (holes)
P-Transistor: negative charge on gate generates electric field that creates a (+ charged) p-channel connecting source & drain
N-Transistor: works the opposite way
Metal-Oxide Semiconductor (Gate-Insulator-Silicon)

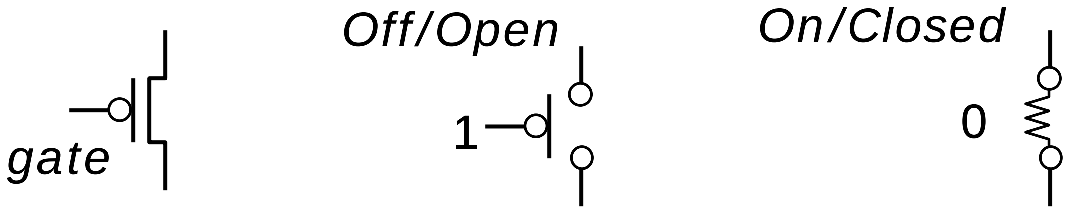
- Complementary MOS = **CMOS** technology uses both p- & n-type transistors

CMOS Notation

N-type



P-type



Gate input controls whether current can flow between the other two terminals or not.

Hint: the "o" bubble of the p-type tells you that this gate wants a 0 to be turned on

iClicker Question

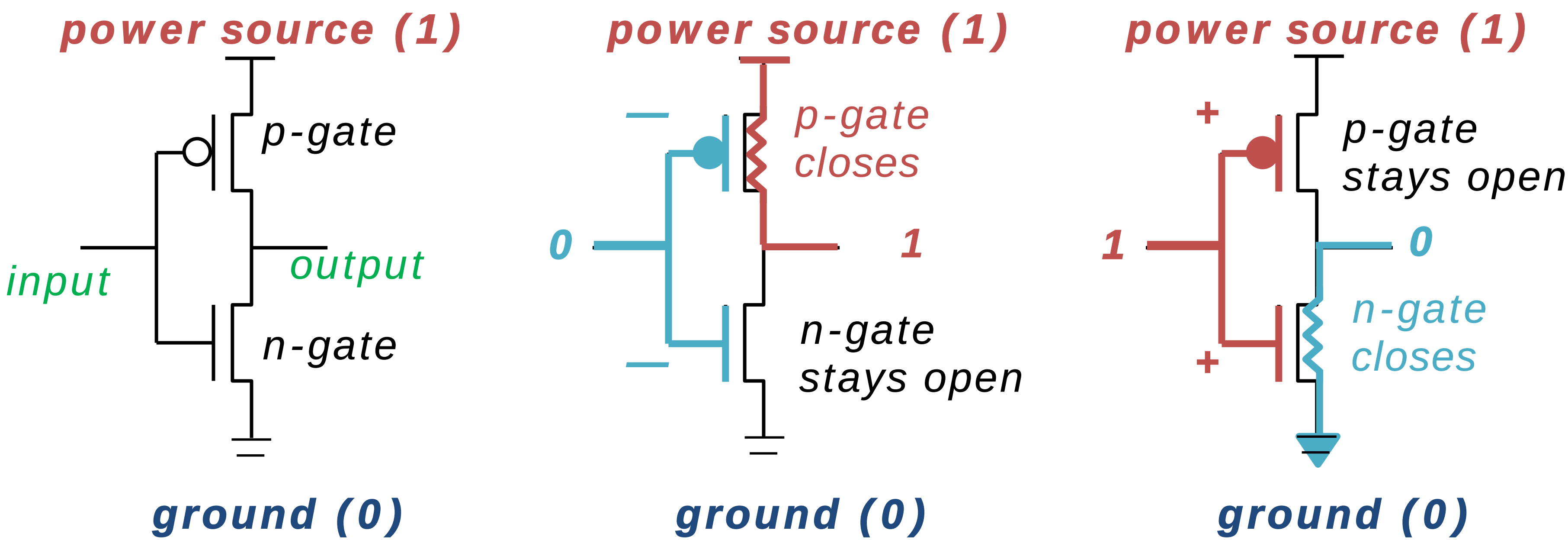
Which of the following statements is **false**?

(A) P- and N-type transistors are both used in CMOS designs.
(B) As transistors get smaller, the frequency of your processor will keep getting faster.
(C) As transistors get smaller, you can fit more and more of them on a single chip.
(D) Pure silicon is a semi conductor.
(E) Experts believe that Moore’s Law will soon end.

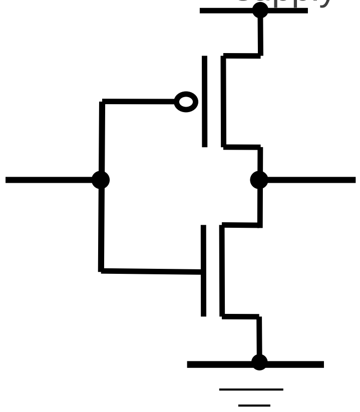
2-Transistor Combination: NOT

- Logic gates are constructed by combining transistors in complementary arrangements
- Combine p&n transistors to make a NOT gate:

CMOS Inverter :

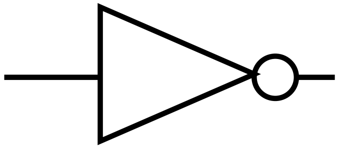


Inverter



Function: NOT

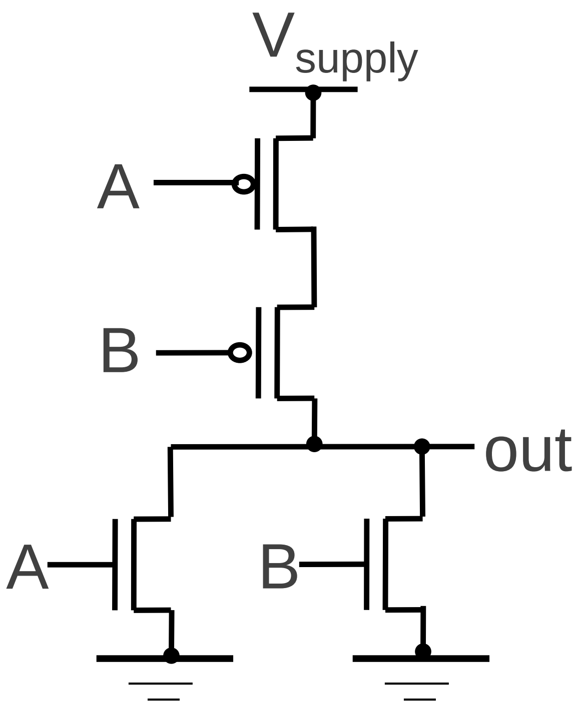
Symbol:



Truth Table:

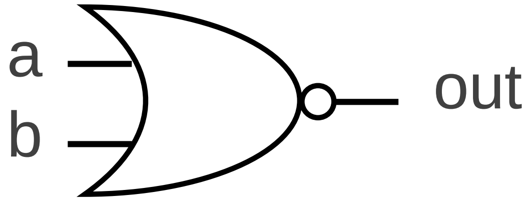
In	Out
0	1
1	0

NOR Gate



Function: NOR

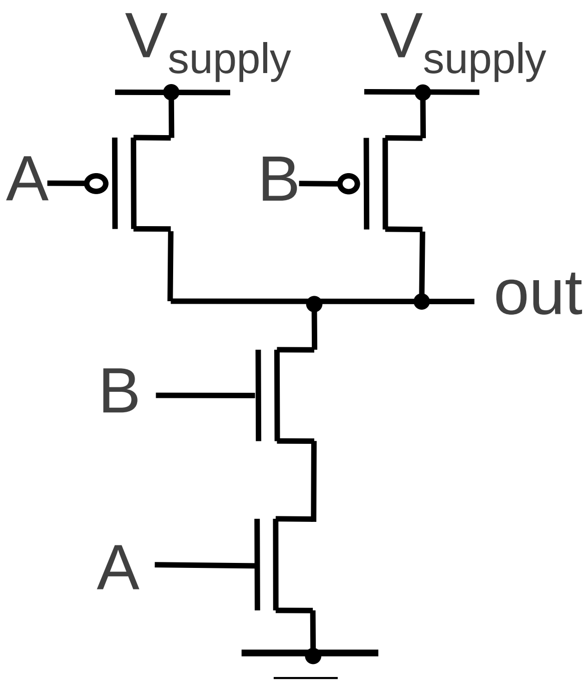
Symbol:

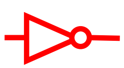


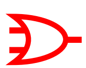
Truth Table:

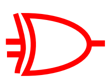
A	B	out
0	0	1
0	1	0
1	0	0
1	1	0

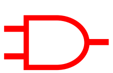
Which Gate is this?

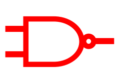


(A) NOT 

(B) OR 

(C) XOR 

(D) AND 

(E) NAND 

Function:

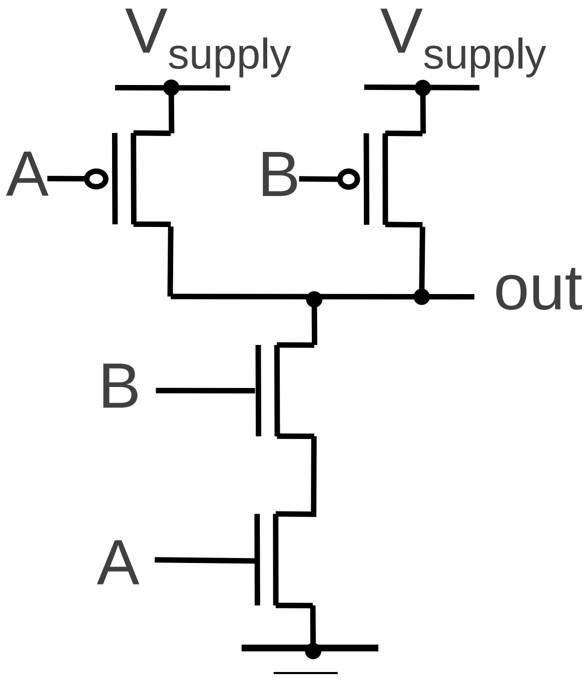
Symbol:

Truth Table:

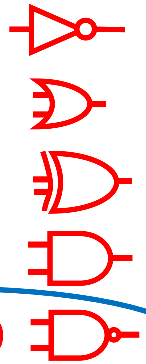
A	B	out
0	0	
0	1	
1	0	
1	1	

iClicker Question

Which Gate is this?



(A) NOT
(B) OR
(C) XOR
(D) AND
(E) NAND



Function:

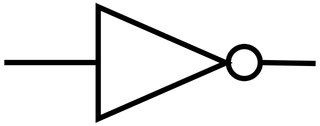
Symbol:

Truth Table:

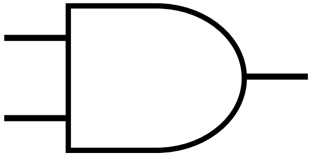
A	B	out
0	0	1
0	1	1
1	0	1
1	1	0

Building Functions (Revisited)

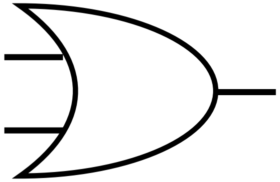
- NOT:



- AND:



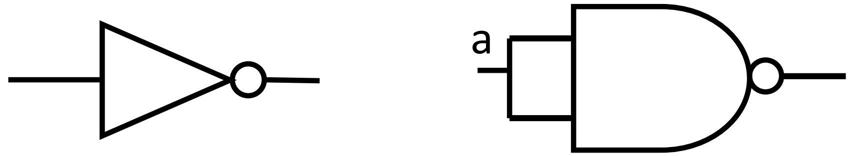
- OR:



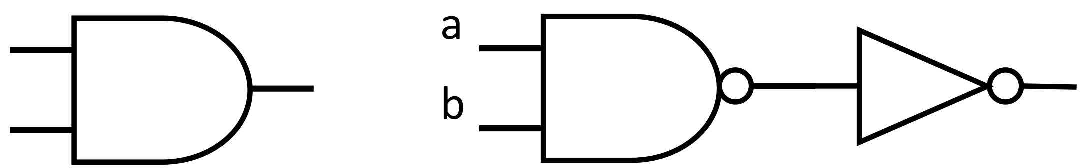
- NAND and NOR are universal
  - Can implement ***any*** function with NAND or just NOR gates
  - useful for manufacturing

Building Functions (Revisited)

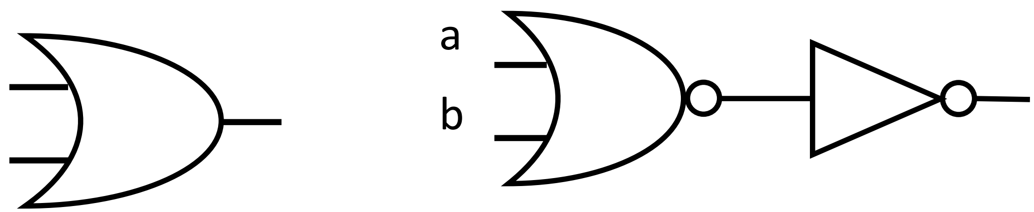
- NOT:



- AND:

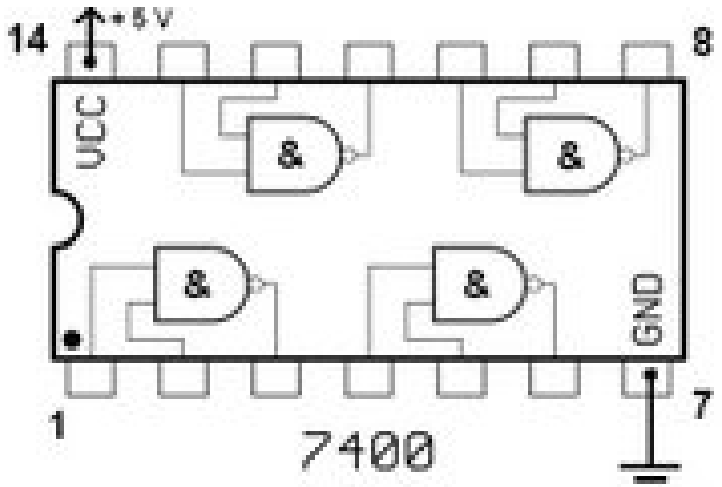


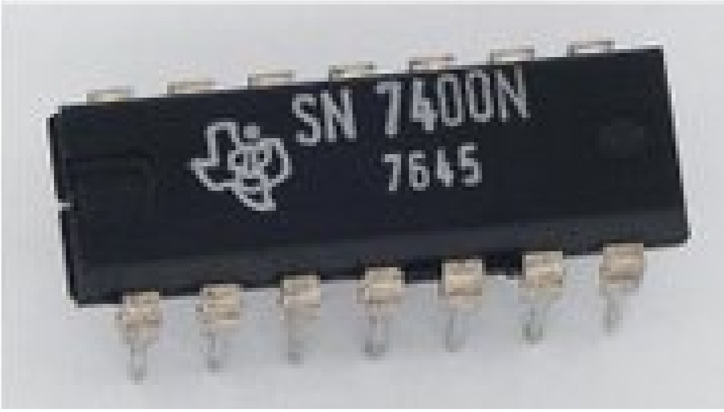
- OR:



- NAND and NOR are universal
  - Can implement ***any*** function with NAND or just NOR gates
  - useful for manufacturing

Logic Gates

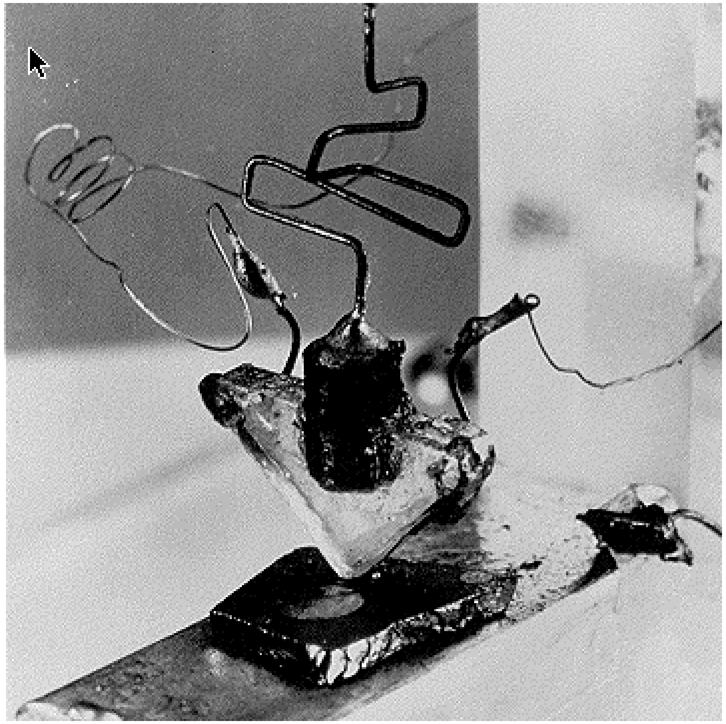


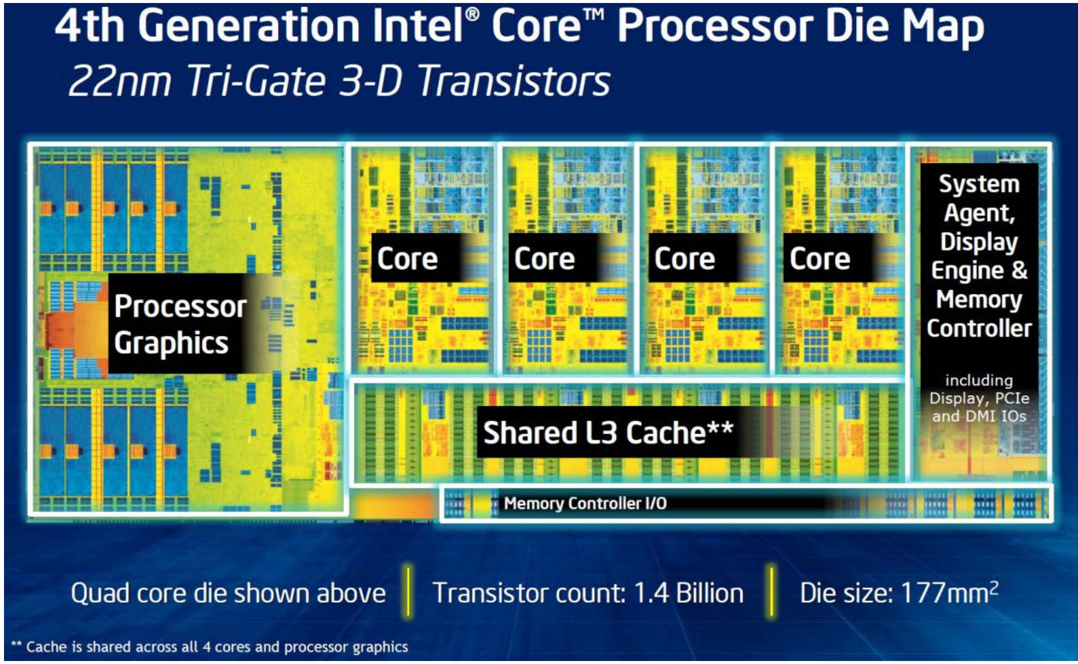


- One can buy gates separately
  - ex. 74xxx series of integrated circuits
  - cost ~$1 per chip, mostly for packaging and testing

- Cumbersome, but possible to build devices using gates put together manually

Then and Now



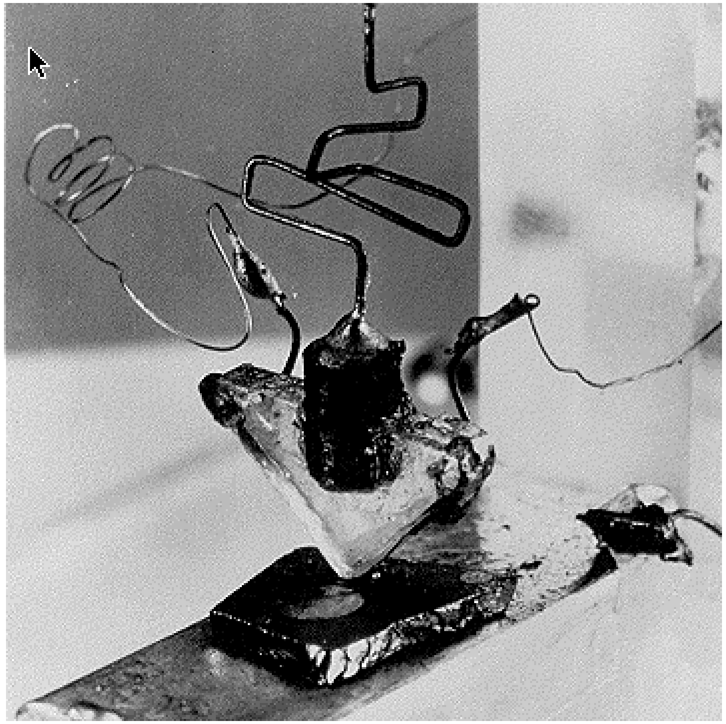


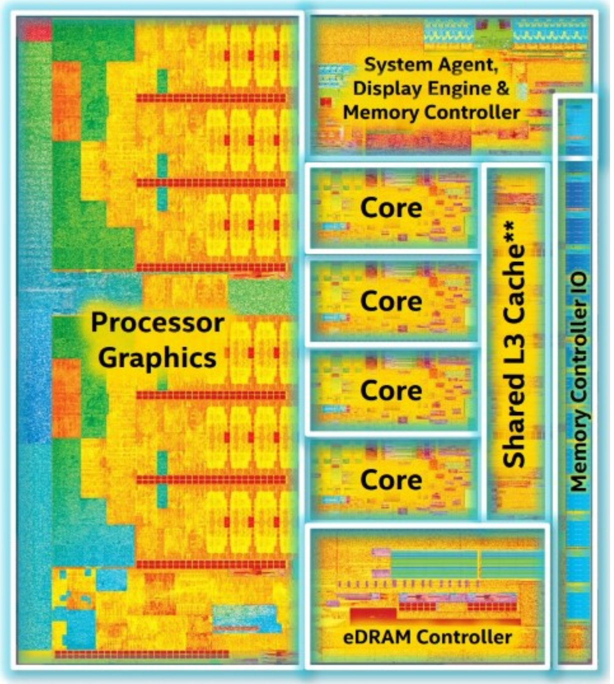
http://techguru3d.com/4th-gen-intel-haswell-processors-architecture-and-lineup/

- The first transistor
  - One workbench at AT&T Bell Labs
  - 1947
  - Bardeen, Brattain, and Shockley
- Intel Haswell
  - 1.4 billion transistors, 22nm
  - 177 square millimeters
  - Four processing cores

https://en.wikipedia.org/wiki/Transistor_count

Then and Now





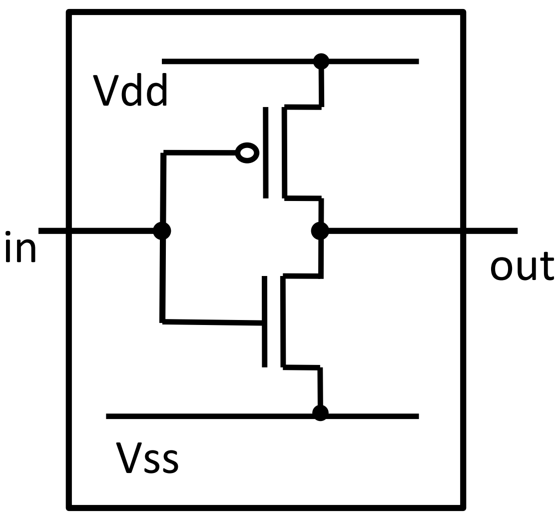
https://www.computershopper.com/computex-2015/performance-preview-desktop-broadwell-at-computex-20

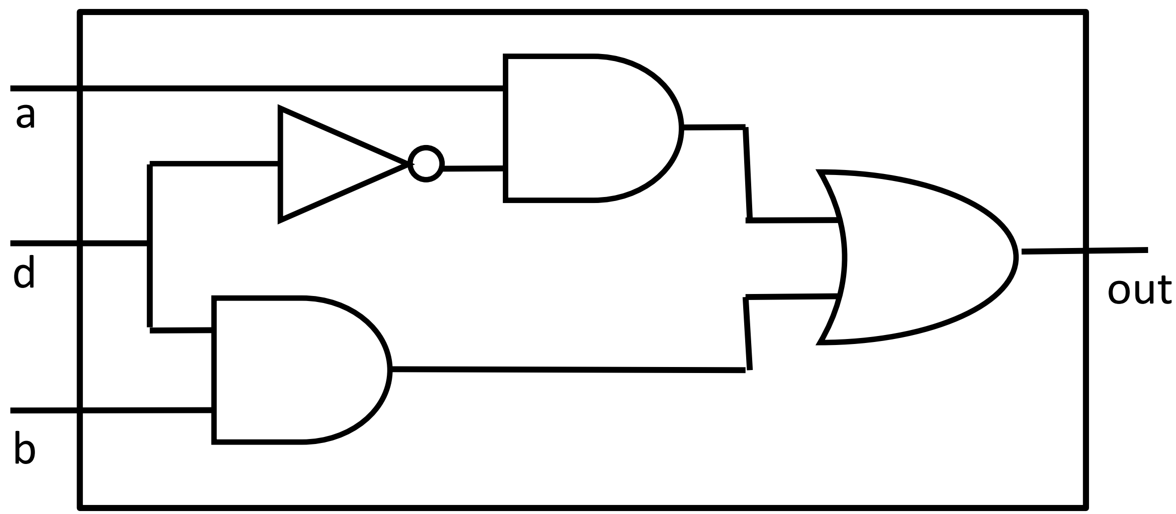
- The first transistor
  - One workbench at AT&T Bell Labs
  - 1947
  - Bardeen, Brattain, and Shockley
- Intel Broadwell
  - 7.2 billion transistors, 14nm
  - 456 square millimeters
  - Up to 22 processing cores

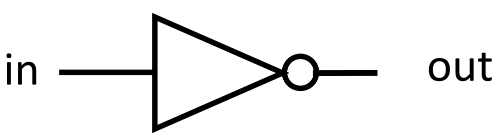
https://en.wikipedia.org/wiki/Transistor_count

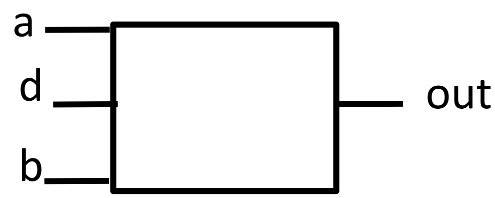
Big Picture: Abstraction

- Hide complexity through simple abstractions
  - Simplicity
    - Box diagram represents inputs and outputs
  - Complexity
    - Hides underlying NMOS- and PMOS-transistors and atomic interactions









Summary

- Most modern devices made of billions of transistors
  - You will build a processor in this course!
  - Modern transistors made from semiconductor materials
  - Transistors used to make logic gates and logic circuits
- We can now implement any logic circuit
  - Use P- & N-transistors to implement NAND/NOR gates
  - Use NAND or NOR gates to implement the logic circuit
  - *Efficiently*: use K-maps to find required minimal terms