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

Prof. Anne Bracy

CS 3410

Computer Science

Cornell University

The slides are the product of many rounds of teaching CS 3410 by Professors Weatherspoon, Bala, Bracy, and Sirer.

See: P&H Appendix B.2 and B.3 (Also, see B.1)

Goals for Today

From Switches to Logic Gates to Logic Circuits

Transistors (electronic switch)

Logic Gates

- Truth Tables

Logic Circuits

- Identity Laws
- From Truth Tables to Circuits (Sum of Products)

Logic Circuit Minimization

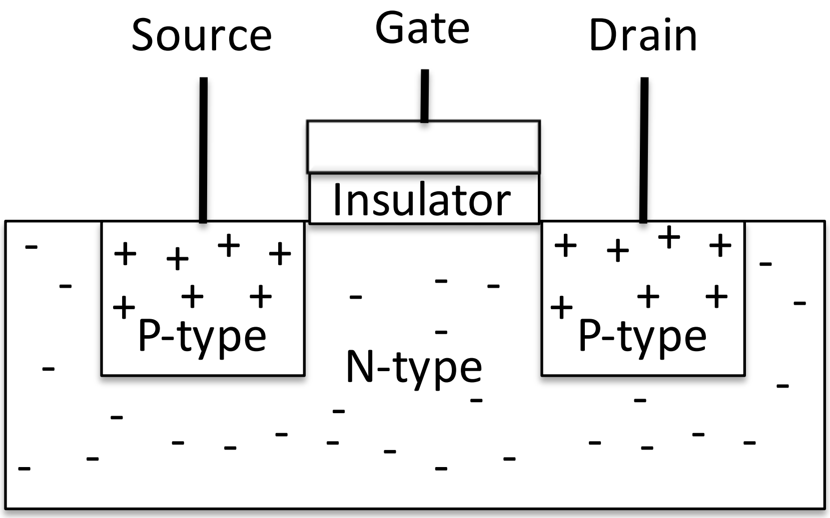
- Algebraic Manipulations
- Karnaugh Maps

Silicon Valley & the Semiconductor Industry

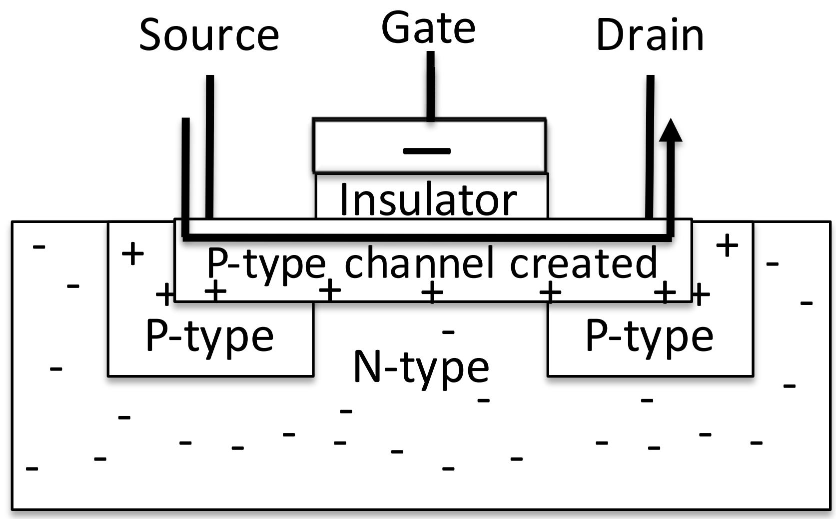
Transistors:

- Youtube video from last week
  https://www.youtube.com/watch?v=IcrBqCFLHIY

Transistors 101



Off



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

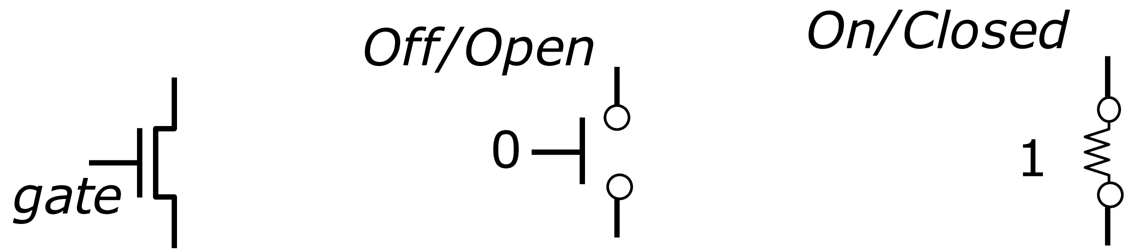
N-Transistor: works the opposite way

Metal-Oxide Semiconductor (Gate-Insulator-Silicon)

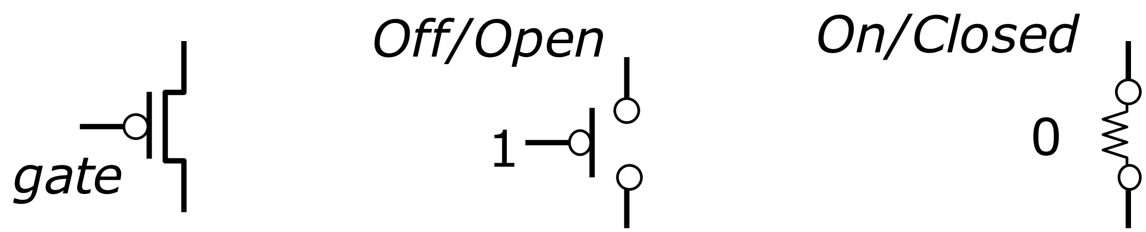
Complementary MOS = **CMOS** technology uses both p- & n-type transistors

CMOS Notation

N-type



P-type



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

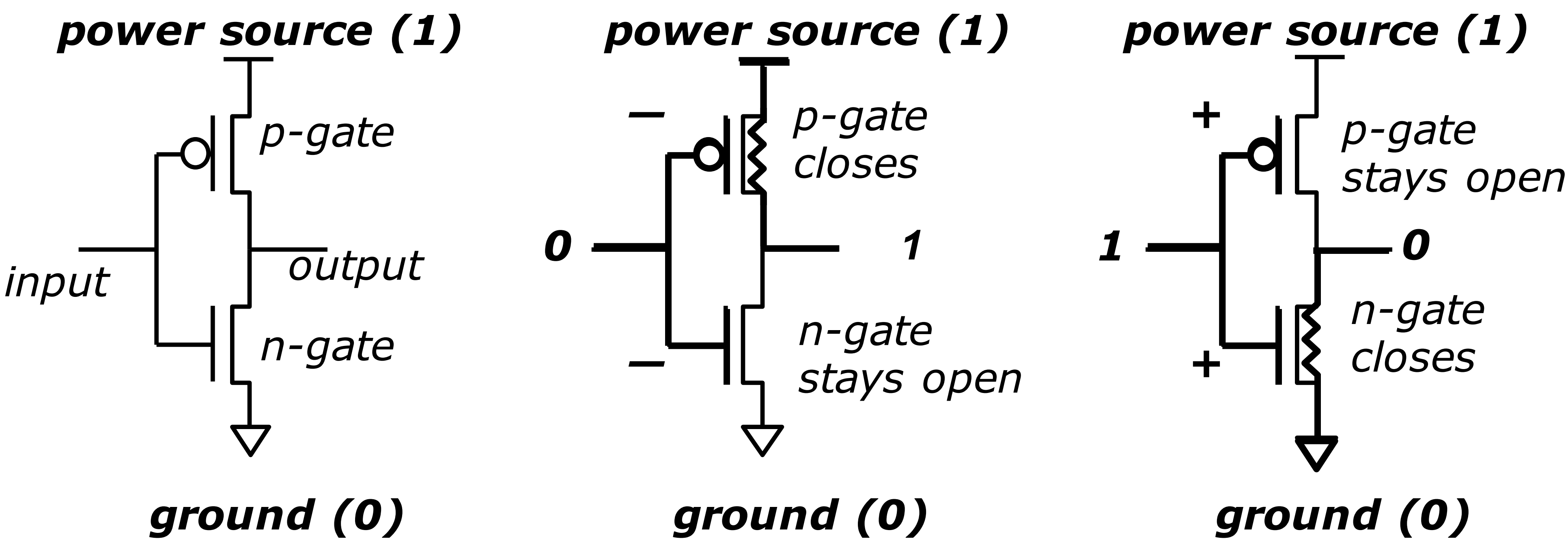
Hint: the "o" bubble of the p-type tells you that this gate wants a 0 to be turned on

2-Transistor Combination: NOT

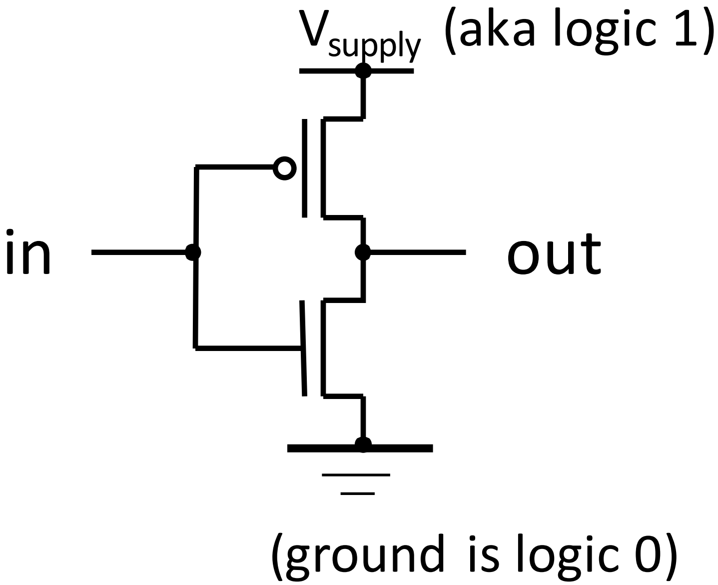
Logic gates are constructed by combining transistors in complementary arrangements

Combine p&n transistors to make a NOT gate:

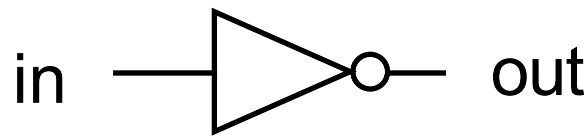
CMOS Inverter :



Inverter



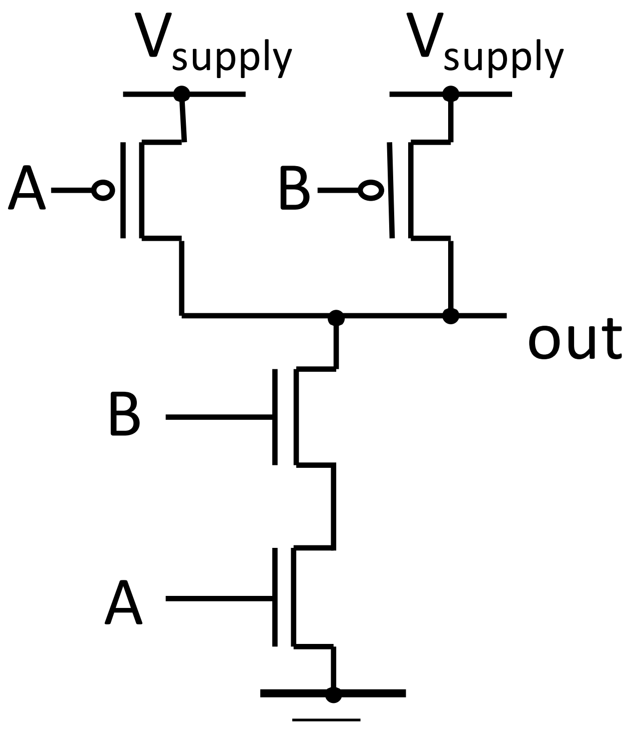
- Function: NOT
- Symbol:



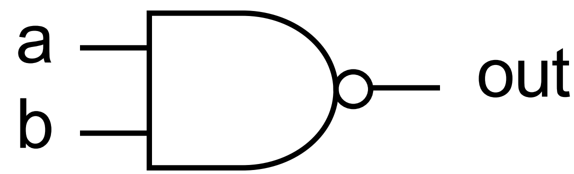
- Truth Table:

In	Out
0	1
1	0

Activity #1: Which Gate is this?



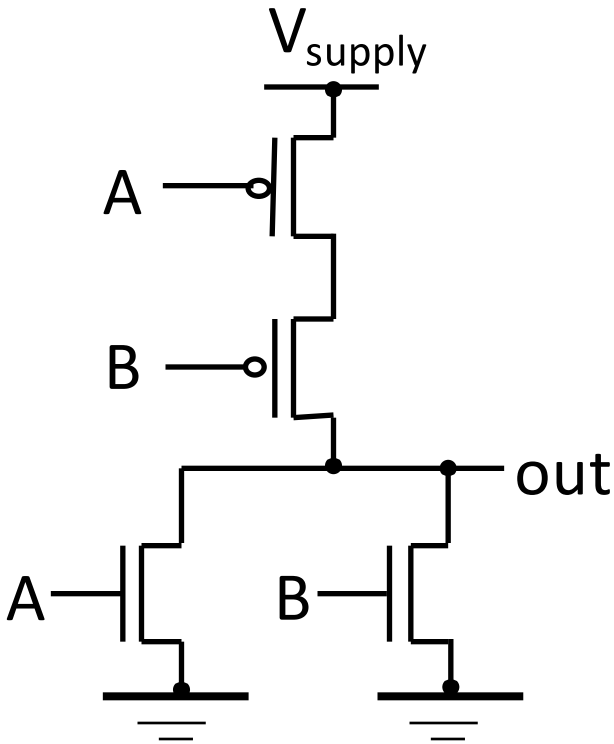
- Function:
- Symbol:



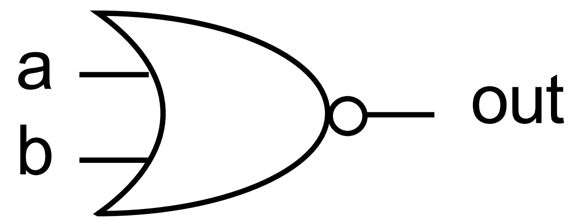
- Truth Table:

A	B	out
0	0	
1	0	
0	1	
1	1	

NOR Gate



- Function: NOR
- Symbol:



- Truth Table:

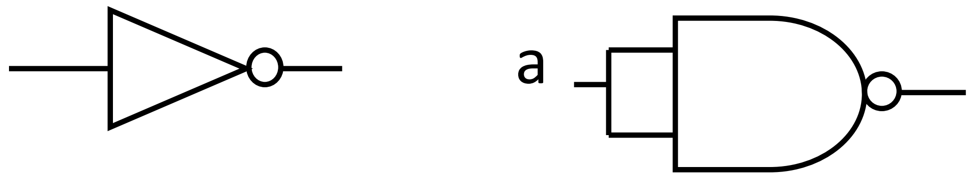
A	B	out
0	0	1
1	0	0
0	1	0
1	1	0

Building Functions

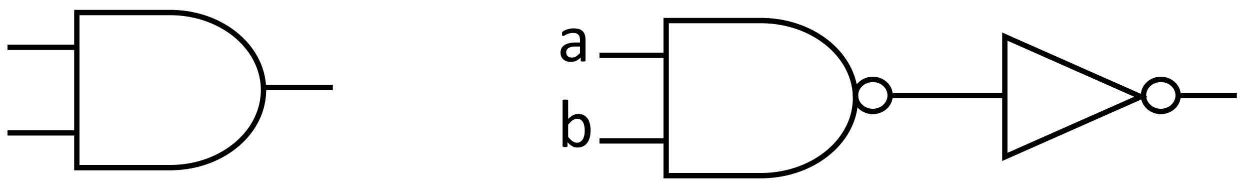
NAND and NOR are universal

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

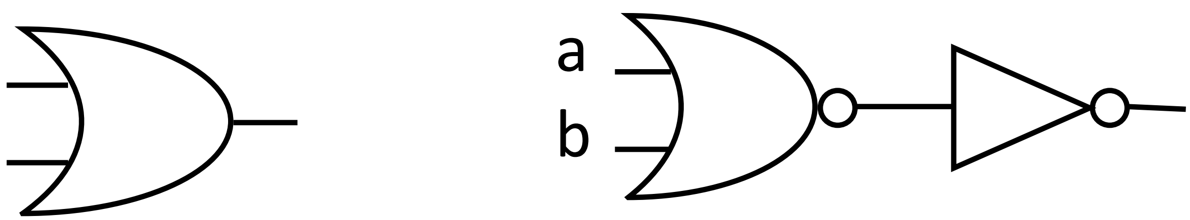
NOT:



AND:



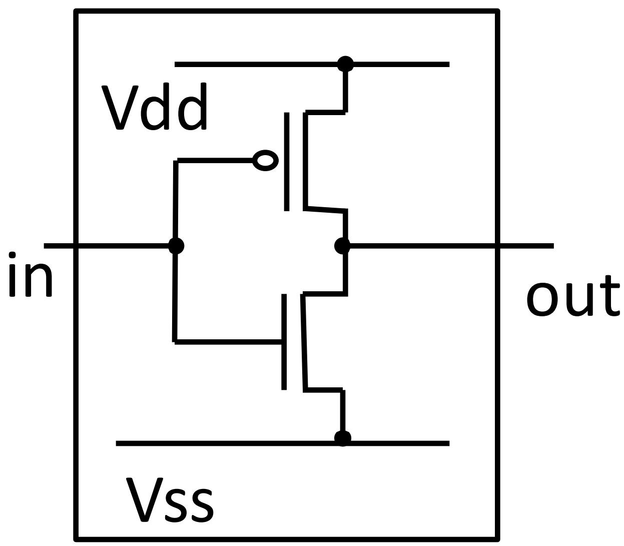
OR:

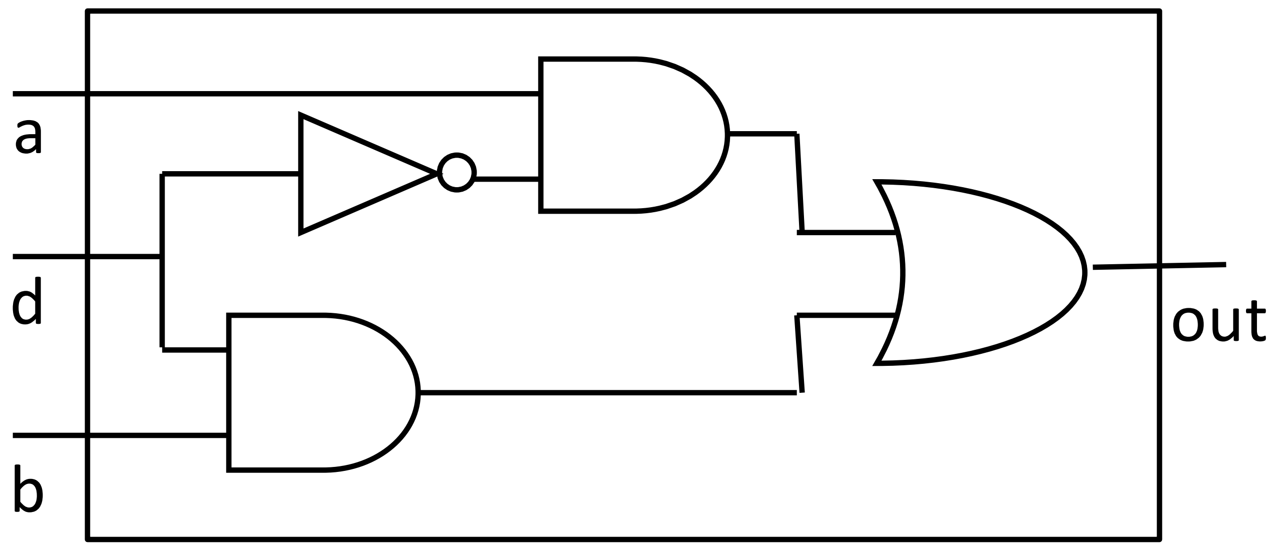


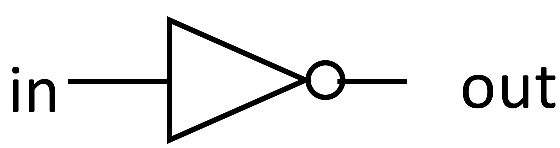
Big Picture: Abstraction

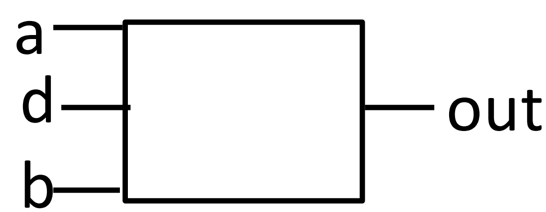
Hide complexity through simple abstractions

- Simplicity
  - Box diagram represents inputs and outputs
- Complexity
  - Hides underlying NMOS- and PMOS-transistors and atomic interactions









Goals for Today

From Switches to Logic Gates to Logic Circuits

Transistors (electronic switch)

Logic Gates

- Truth Tables

Logic Circuits

- Identity Laws
- From Truth Tables to Circuits (Sum of Products)

Logic Circuit Minimization

- Algebraic Manipulations
- Karnaugh Maps

Logic Gates

- Digital circuit that either allows signal to pass through it or not
- Used to build logic functions
- Seven basic logic gates:

  AND,

  OR,

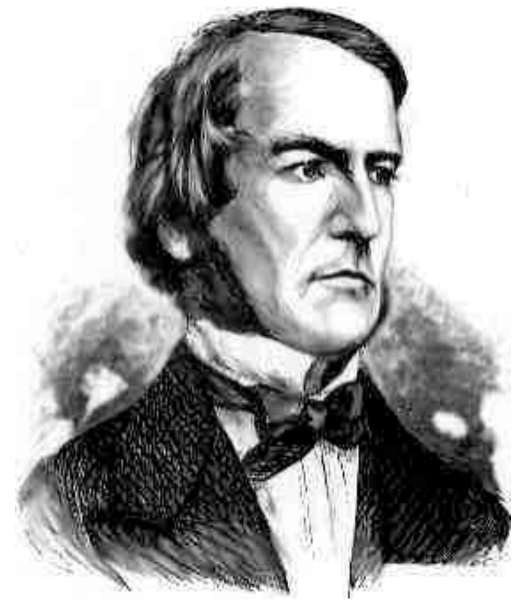
  NOT,

  NAND (not AND),

  NOR (not OR),

  XOR

  XNOR (not XOR)



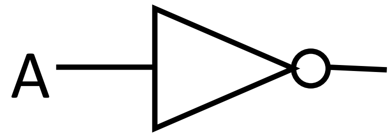
George Boole,(1815-1864)

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.

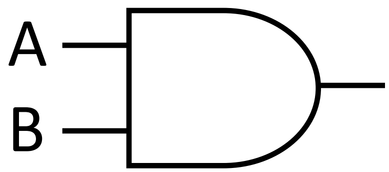
Logic Gates: Names, Symbols, Truth Tables

NOT:



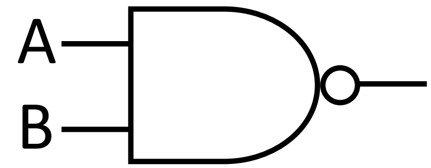
A	Out
0	1
1	0

AND:



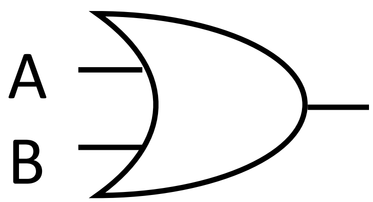
A	B	Out
0	0	0
0	1	0
1	0	0
1	1	1

NAND:



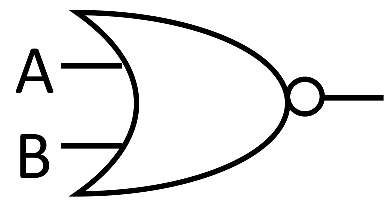
A	B	Out
0	0	1
0	1	1
1	0	1
1	1	0

OR:



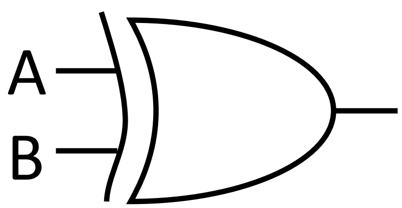
A	B	Out
0	0	0
0	1	1
1	0	1
1	1	1

NOR:



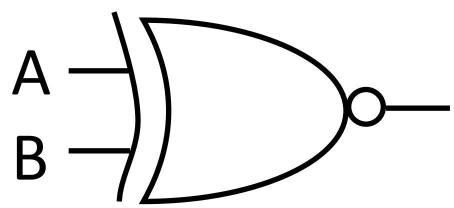
A	B	Out
0	0	1
0	1	0
1	0	0
1	1	0

XOR:.



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

XNOR:



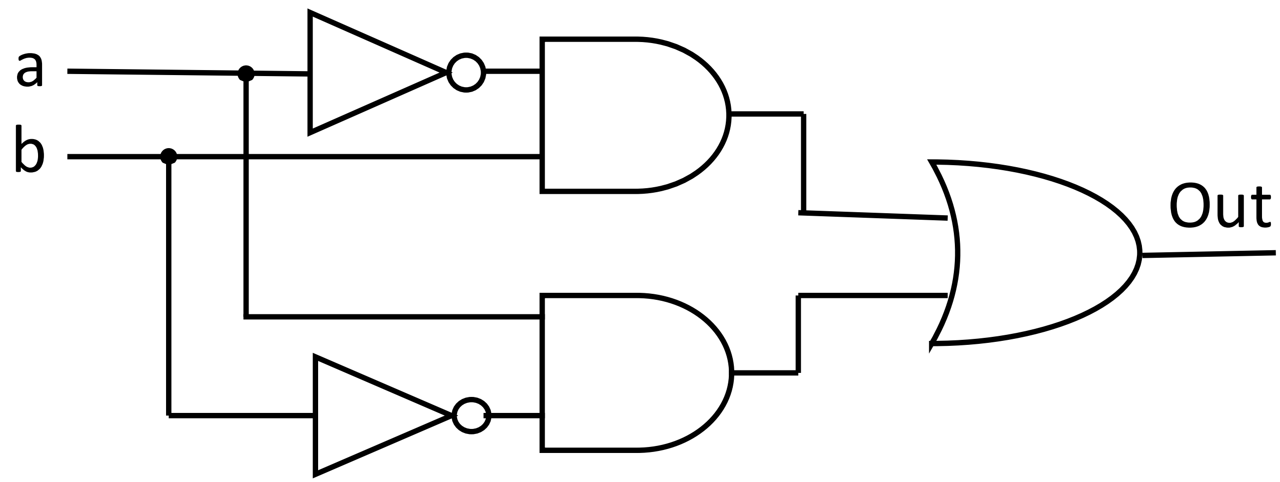
A	B	Out
0	0	1
0	1	0
1	0	0
1	1	1

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**	**Out**

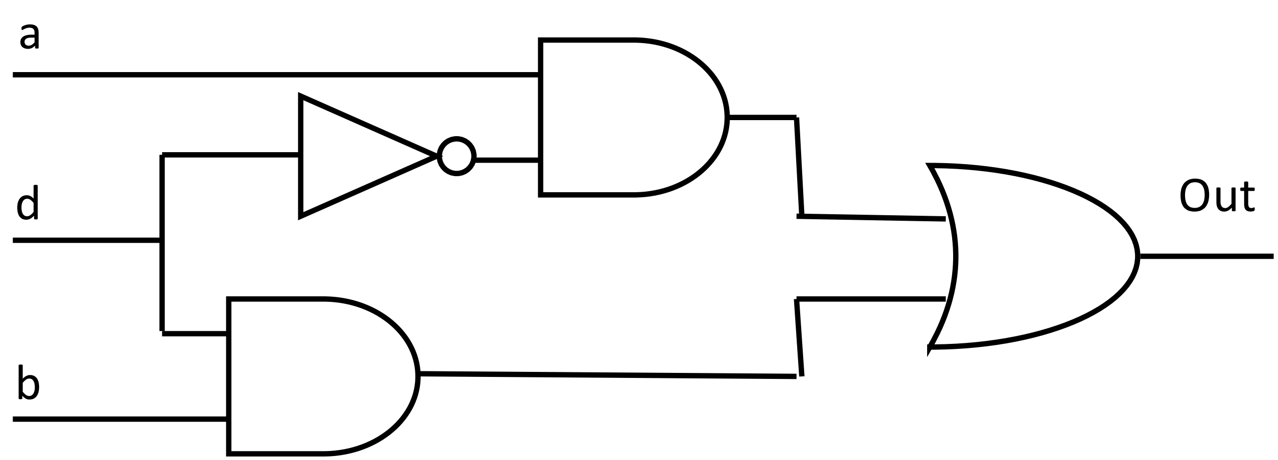


Activity #3: Logic Gates

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	



Goals for Today

From Switches to Logic Gates to Logic Circuits

Transistors (electronic switch)

Logic Gates

- Truth Tables

Logic Circuits

- From Truth Tables to Circuits (Sum of Products)
- Identity Laws

Logic Circuit Minimization

- Algebraic Manipulations
- Karnaugh Maps

Next Goal

Given a Logic function → create a Logic Circuit that implements the Logic Function…

Logic Implementation

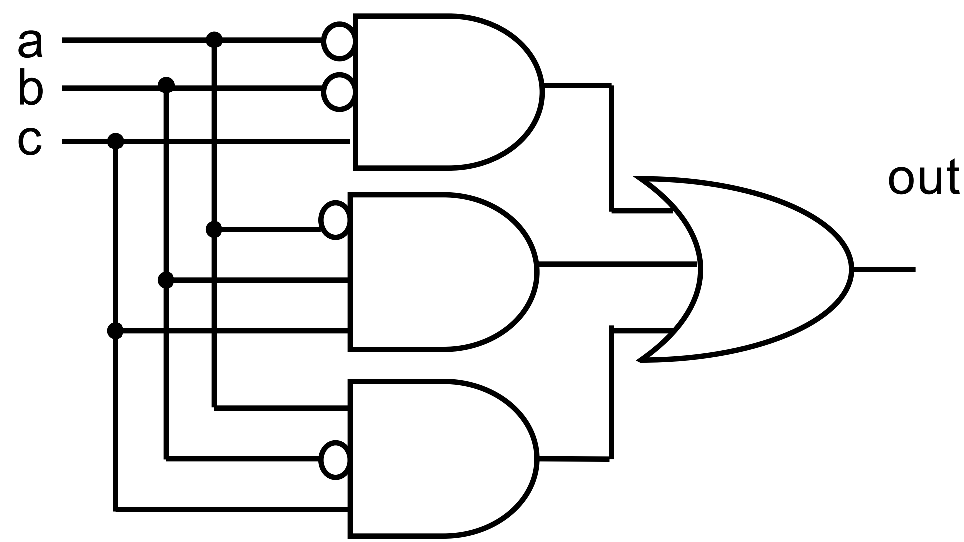
How to implement a desired logic function?

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

1) Write minterms
2) Write sum of products:

   OR of all minterms where out=1

$$out = \overline{a}\overline{b}c + \overline{a}bc + a\overline{b}c$$



Any combinational circuit *can be* implemented in two levels of logic (ignoring inverters)

Logic Equations

NOT: $= \bar{a}$ $= !a$ $= \neg a$

AND: $= a \cdot b$ $= a \,\&\, b$ $= a \wedge b$

OR: $= a + b$ $= a \mid b$ $= a \vee b$

XOR: $= a \oplus b = a\bar{b} + \bar{a}b$

NAND: $\overline{(a \bullet b)} = !(a \,\&\, b) = \neg (a \wedge b)$

NOR: $\overline{(a + b)} = !(a \mid b) = \neg (a \vee b)$

XNOR: $\overline{(a \oplus b)} = \overline{a\bar{b} + \bar{a}b}$

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

$$a + 0 = a$$

$$a + 1 = 1$$

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

$$a \cdot 0 = 0$$

$$a \cdot 1 = a$$

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

Identities

Identities useful for manipulating logic equations

- For optimization & ease of implementation

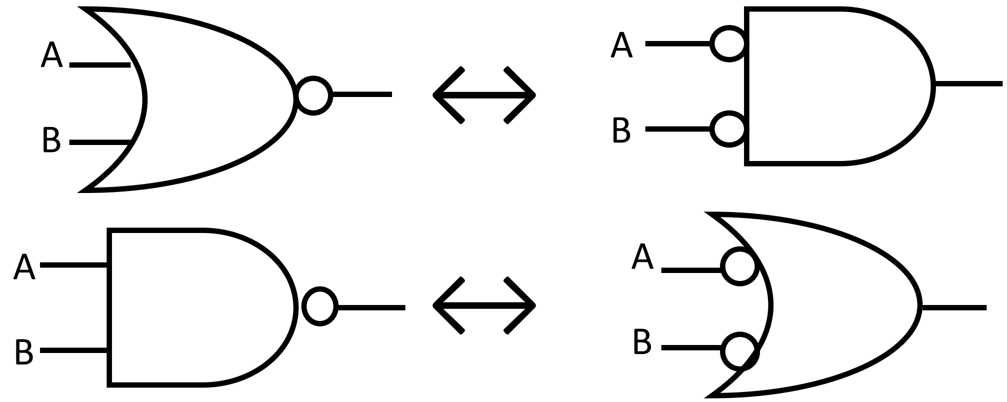
$$\overline{(a+b)} = \overline{a} \bullet \overline{b}$$

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

a + a b = a

a (b+c) = ab + ac

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



Goals for Today

From Switches to Logic Gates to Logic Circuits

Transistors (electronic switch)

Logic Gates

- Truth Tables

Logic Circuits

- Identity Laws
- From Truth Tables to Circuits (Sum of Products)

Logic Circuit Minimization

- Algebraic Manipulations
- Karnaugh Maps

Next Goal

Implement the Logic Function…

with the minimum number of logic gates

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

How to standardize this minimizing?

Activity #4: Logic Minimization

$a + 0 = a$

$a + 1 = 1$

$a + \bar{a} = 1$

$a \cdot 0 = 0$

$a \cdot 1 = a$

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

$a + a\,b = a$

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

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

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

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

Minimize this logic equation:

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

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					

Minimization in Practice

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

Building a Karnaugh Map

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 =

$$\overline{a}\overline{b}c + \overline{a}bc + a\overline{b}\overline{c} + a\overline{b}c$$

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

K-maps identify which inputs are relevant to the output

Minimization with K-Maps

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

(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

Karnaugh Minimization Tricks (1)

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

Minterms can overlap

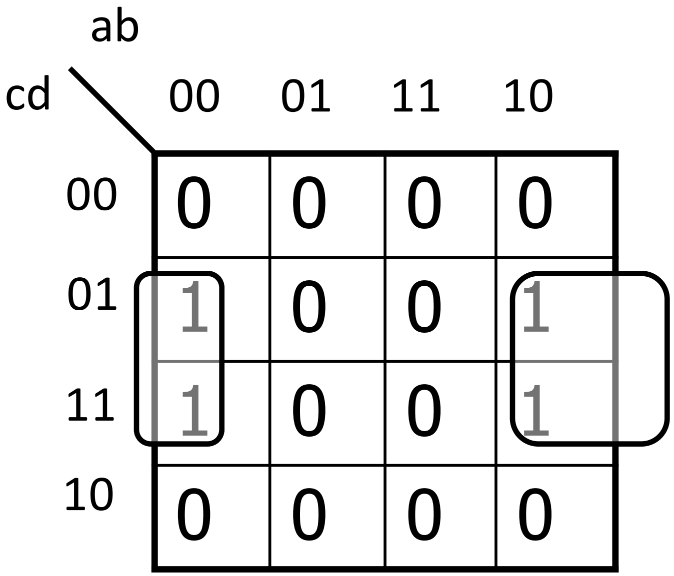
$$out = b\overline{c} + a\overline{c} + ab$$

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

Minterms can span 2, 4, 8 or more cells

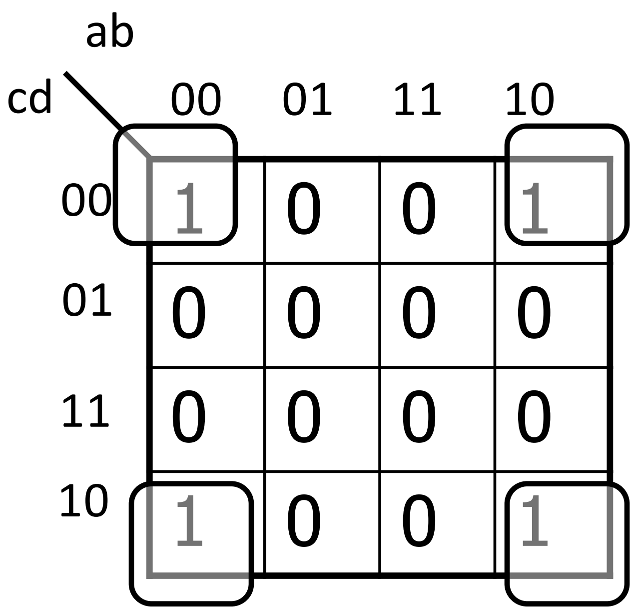
$$out = \overline{c} + ab$$

Karnaugh Minimization Tricks (2)



The map wraps around

$\text{out} = \bar{b}d$



$\text{out} = \bar{b}\,\bar{d}$

Don’t Cares

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

“Don’t care” values can be interpreted individually in whatever way is convenient

- assume all x’s = 1

→ out = d

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

- assume middle x’s = 0 (ignore them)
- assume 4^{th} column x = 1

→ out = $\overline{b}\ \overline{d}$

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.

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