CSEE 3827: Fundamentals of Computer Systems

Boolean Logic & Boolean Algebra

Agenda (M&K 2.1-2.2, 2.8-2.9)

- Terminology
- Boolean algebra
- Logic gates
- Circuit fabrication
  - NAND, NOR
- DUAL
- XOR

Terminology

- Digital / Binary / Boolean: 0 = False, 1 = True

- Binary Variable: a symoblic representation of a value that might be 0 or 1, e.g., X, Y, A, B

- Complement (e.g., of a variable X): written $\overline{X}$: the opposite value of X

X	$\overline{X}$
0	1
1	0

- Literal: a boolean variable or its complement (e.g., X, $\overline{X}$, $\overline{Y}$)

Boolean Logic

- All logical functions can be implemented in terms of three logical operations:

NOT

x	$\overline{x}$
0	1
1	0

AND

can omit the "·"

x	y	$x \cdot y$
0	0	0
0	1	0
1	0	0
1	1	1

OR

x	y	$x + y$
0	0	0
0	1	1
1	0	1
1	1	1

Boolean Logic 2

- Precedence rules just like decimal system
- Implied precedence: NOT > AND > OR
- Use parentheses as necessary

$$AB + C = (AB) + C$$

$$(\overline{A} + B)C = ((\overline{A}) + B)C$$

Terminology cont'd

- Expression: a set of literals (possibly with repeats) combined with logic operations (and possibly ordered by parentheses)
    - e.g., 4 expressions: $AB + C$, $(AB) + C$, $(\overline{A} + B)C$, $((\overline{A}) + B)C$
    - Note: can compliment expressions, too, e.g., $\overline{((\overline{A}) + B)C}$
- Equation: expression1 = expression2
    - e.g., $(\overline{A} + B)C = ((\overline{A}) + B)C$
- Function of (possibly several) variables: an equation where the lefthand side is defined by the righthand side $F(A,B,C) = ((\overline{A}) + B)C$

Boolean Logic: Example

Truth Table: all combinations of input variables

k variables ➜ 2^k input combinations

D	X	A	$D\overline{X} + A$
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Boolean Logic: Example 2

X	Y	$XY + \overline{X}\,\overline{Y}$
0	0	
0	1	
1	0	
1	1	

Boolean Algebra: Identities and Theorems

OR	AND	NOT	
$X+0 = X$	$X1 = X$		(identity)
$X+1 = 1$	$X0 = 0$		(null)
$X+X = X$	$XX = X$		(idempotent)
$X+\overline{X} = 1$	$X\overline{X} = 0$		(complementarity)
		$\overline{\overline{X}} = X$	(involution)
$X+Y = Y+X$	$XY = YX$		(commutativity)
$X+(Y+Z) = (X+Y)+Z$	$X(YZ) = (XY)Z$		(associativity)
$X(Y+Z) = XY + XZ$	$X+YZ = (X+Y)(X+Z)$		(distributive)
$\overline{X+Y} = \overline{X}\,\overline{Y}$	$\overline{XY} = \overline{X} + \overline{Y}$		(DeMorgan's theorem)

Boolean Algebra: Example

Simplify this equation using algebraic manipulation.

$$F = \overline{X}YZ + \overline{X}Y\overline{Z} + XZ$$

Boolean Algebra: Example 2

Find the complement of F.

$$F = A\overline{B} + \overline{A}B$$

$$\overline{F} =$$

DeMorgan's Theorem

- Procedure for complementing expressions
- Remove the "big bar" over AND or OR of 2 (or more) functions (e.g., F & G) and replace...
  - AND with OR, OR with AND
  - 1 with 0, 0 with 1
  - function F with $\overline{F}$, $\overline{F}$ with F

$$\overline{FG} = \overline{F} + \overline{G}$$

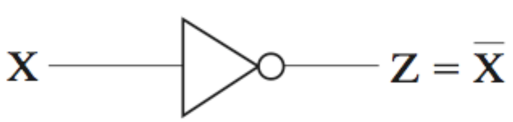
$$\overline{F + G} = \overline{F}\,\overline{G}$$

DeMorgan’s Practice

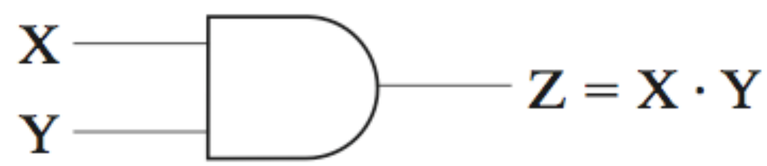
$$\overline{\overline{A\overline{B}C} + \overline{ACD} + B\overline{C}}$$

Circuit Representation

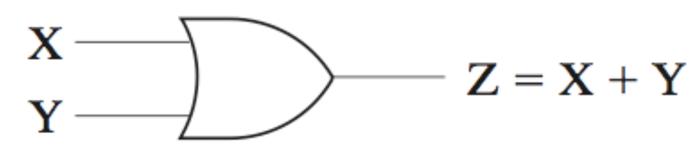
- Information flows from left to right
- Input(s) all the way on the left, output(s) on the right



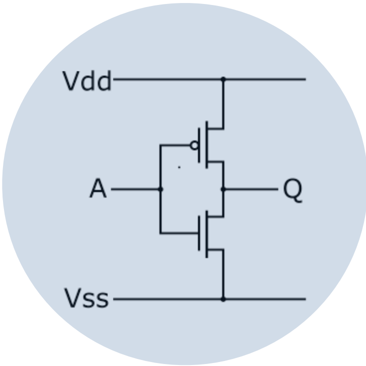
NOT gate or inverter

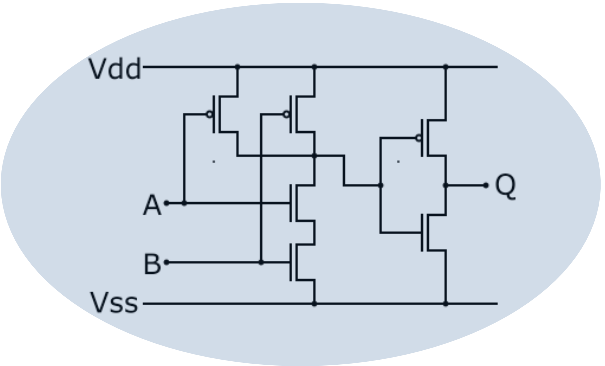


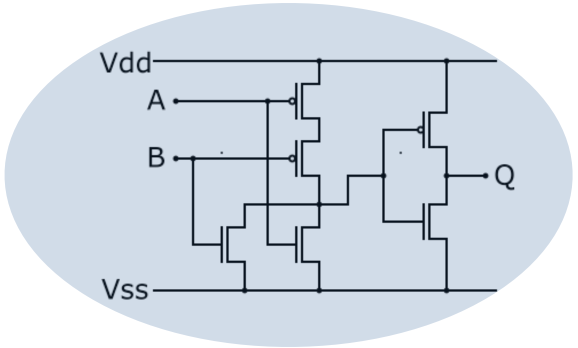
AND gate



OR gate







These circuits consume area, power, and time

Goal: minimize the amount of circuitry to compute the desired function

We simplify to reduce required circuitry...

$$F = \overline{X}YZ + \overline{X}Y\overline{Z} + XZ$$

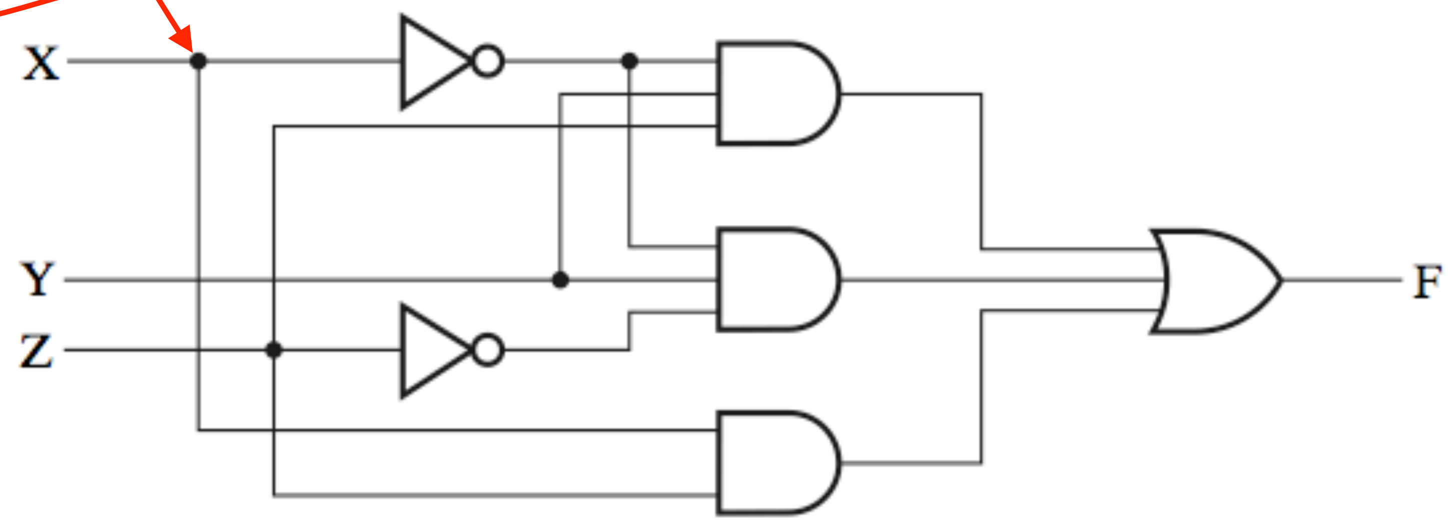
$\overline{X}Y(Z + \overline{Z}) + XZ$ (by reverse distribution)

$\overline{X}Y1 + XZ$ (by complementarity)

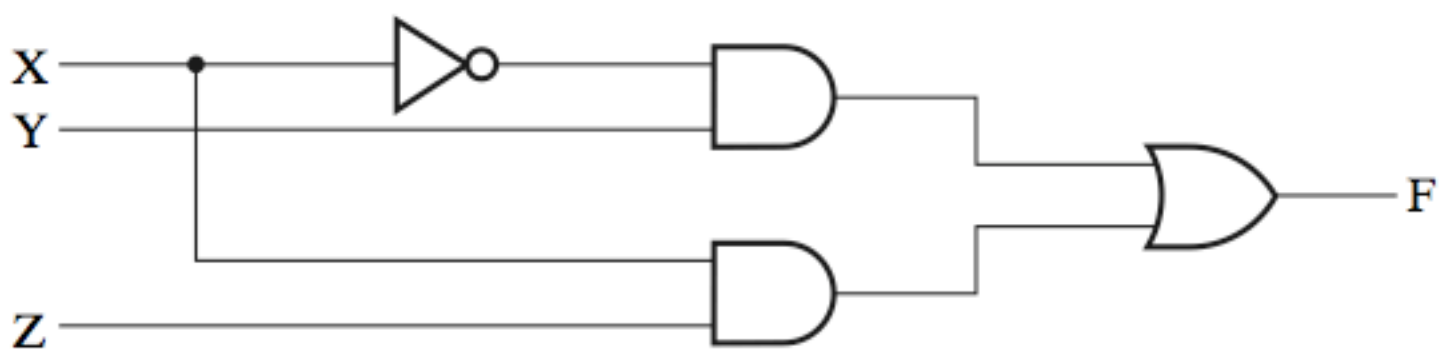
$\overline{X}Y + XZ$ (by identity)

Circuit view

wire connector: black dot signifies wires are connected



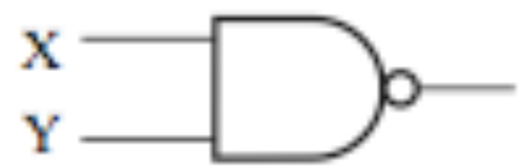
(a) $F = \overline{X}YZ + \overline{X}Y\overline{Z} + XZ$



(b) $F = \overline{X}Y + XZ$

Universal gates: NAND, NOR

x	y	$z = \overline{xy}$
0	0	1
0	1	1
1	0	1
1	1	0



$\overline{XY}$

Note: the “o” in a circuit represents a NOT (inverter)

Different from “●” which represents wire connector

x	y	$z = \overline{x+y}$
0	0	1
0	1	0
1	0	0
1	1	0

$\overline{X+Y}$

NAND and NOR universal because...

- NOT, AND, OR can each be implemented using only NAND gates
- NOT, AND, OR can each be implemented using only NOR gates

$\overline{A} = A \text{ NAND } A$	$\overline{A} = A \text{ NOR } A$
$AB = \overline{A \text{ NAND } B}$	$A+B = \overline{A \text{ NOR } B}$
$A+B = \overline{A} \text{ NAND } \overline{B}$	$AB = \overline{A} \text{ NOR } \overline{B}$

Duals

Duals

- All boolean expressions have duals
- Any theorem you can prove, you can also prove for its dual
- To form a dual...
  - replace AND with OR, OR with AND
  - replace 1 with 0, 0 with 1

What is the dual of this theorem?

$$\overline{X} + \overline{Y} = \overline{XY}$$

Duals and Complements

$$
\begin{tikzcd}
\overline{X}+\overline{Y}=\overline{XY} \arrow[r, leftrightarrow, "\text{complement}"] \arrow[d, leftrightarrow, "\text{dual}"] & XY=\overline{\overline{X}+\overline{Y}} \arrow[d, leftrightarrow, "\text{dual}"] \\
\overline{X}\,\overline{Y}=\overline{X+Y} \arrow[r, leftrightarrow, "\text{complement}"] & X+Y=\overline{\overline{X}\,\overline{Y}}
\end{tikzcd}
$$

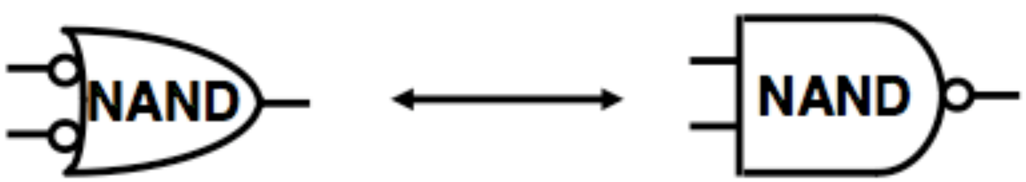
Note: to complement a function, compute its dual and complement literals

“Complement using Dual” example

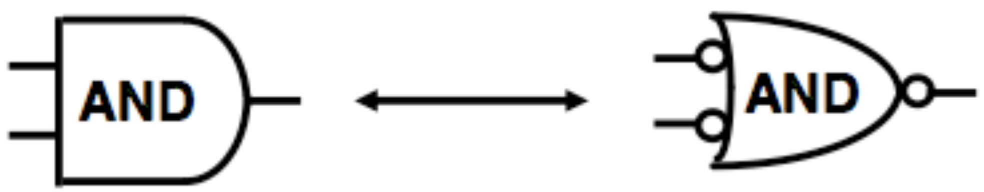
$$F = X + A\,(Z + \overline{X}\,(Y + W) + \overline{Y}\,(Z + W))$$

Can be used for gate manipulation.

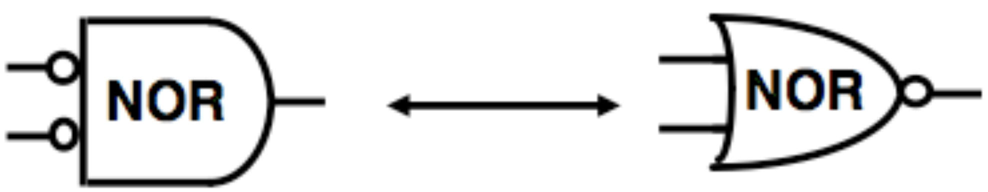
$\overline{X}+\overline{Y}=\overline{XY}$



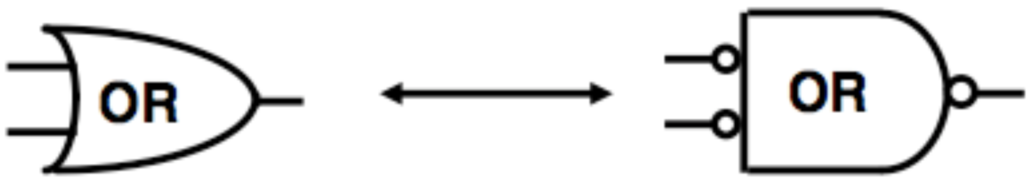
$XY = \overline{\overline{X}+\overline{Y}}$



$\overline{X}\,\overline{Y} = \overline{X+Y}$

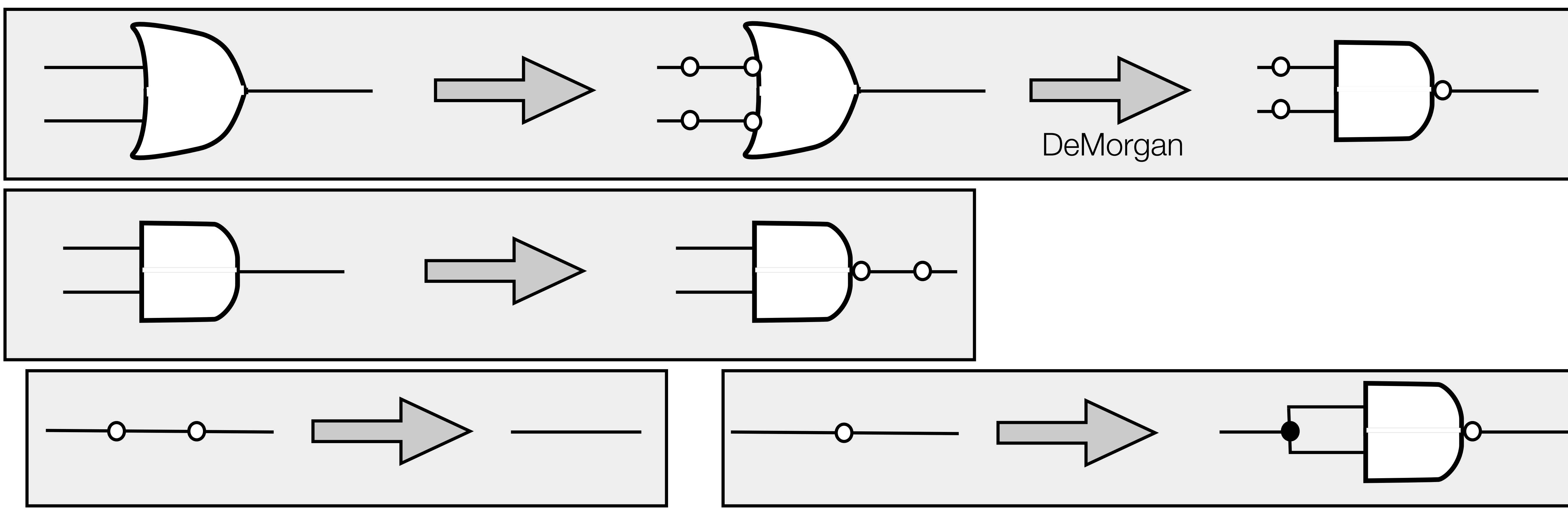


$X+Y = \overline{\overline{X}\,\overline{Y}}$

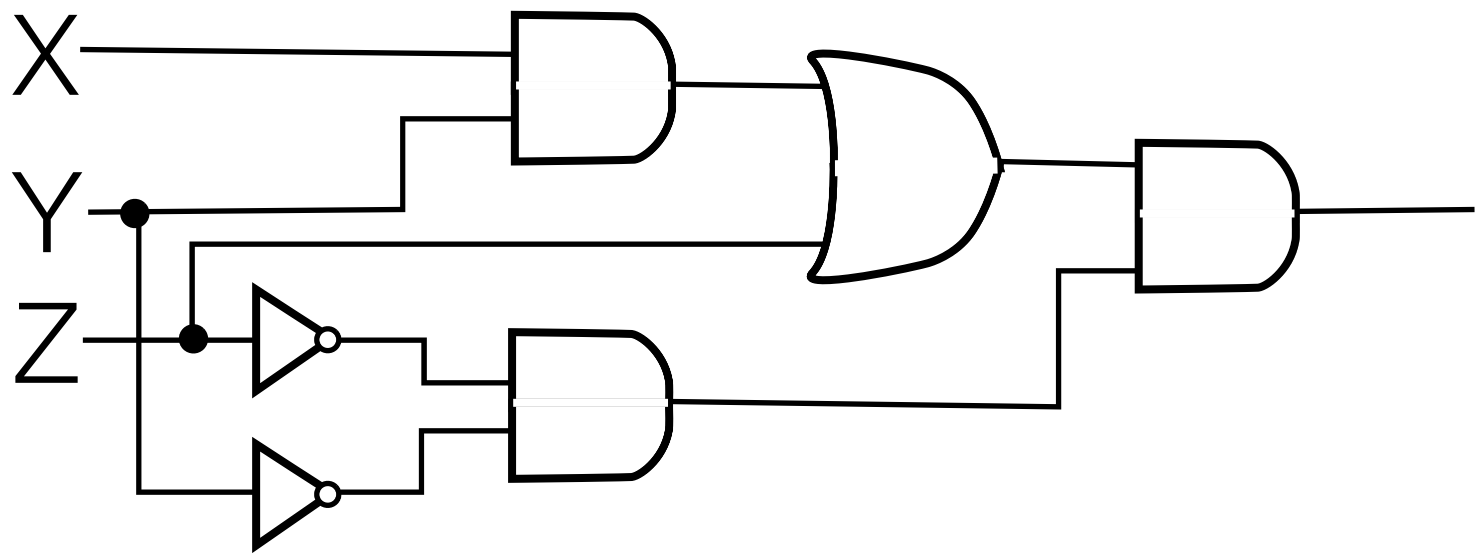


Converting circuits to all-NAND (or all-NOR)

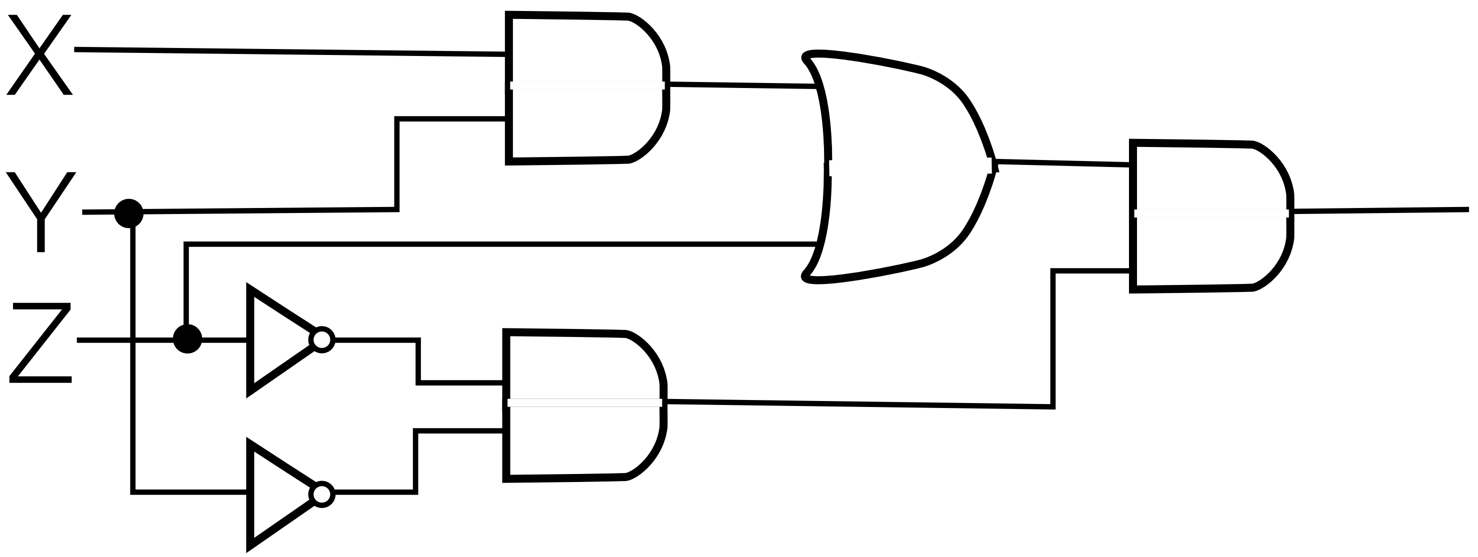
- Work from right to left
- When manipulating an (AND or OR) gate, stick in pairs of NOT gates to get it in “appropriate” form
- Isolated NOT gates are easily implemented as a NAND (NOR) gate
- example manipulations (for NAND gates)

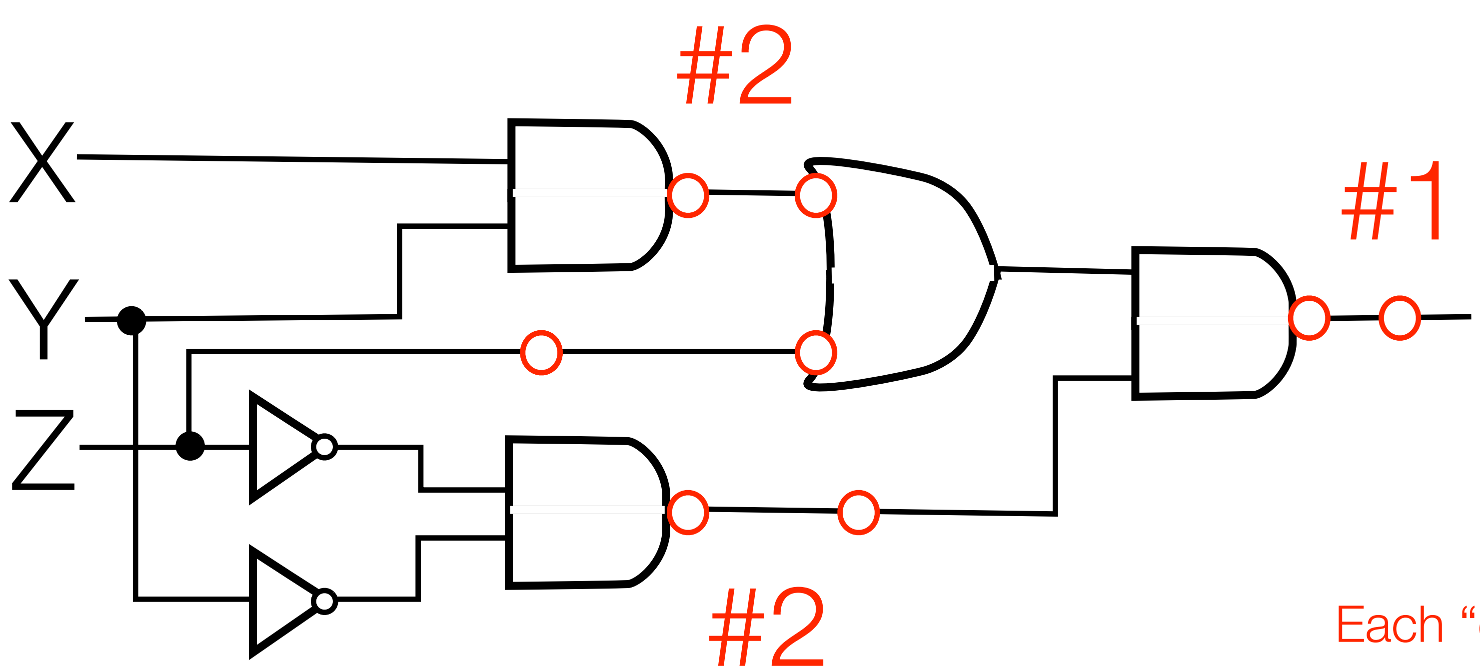


Convert-to-all-NAND example



Convert-to-all-NAND example





Each “o” by itself represents a NOT gate

XOR: the parity operation

- $X \oplus Y = X\overline{Y} + \overline{X}Y$

X	Y	$X \oplus Y$
0	0	0
0	1	1
1	0	1
1	1	0

- In general, represents parity, i.e.,
- $X_1 \oplus X_2 \oplus X_3 \oplus \ldots \oplus X_k = 1$ when an odd number of $X_i = 1$