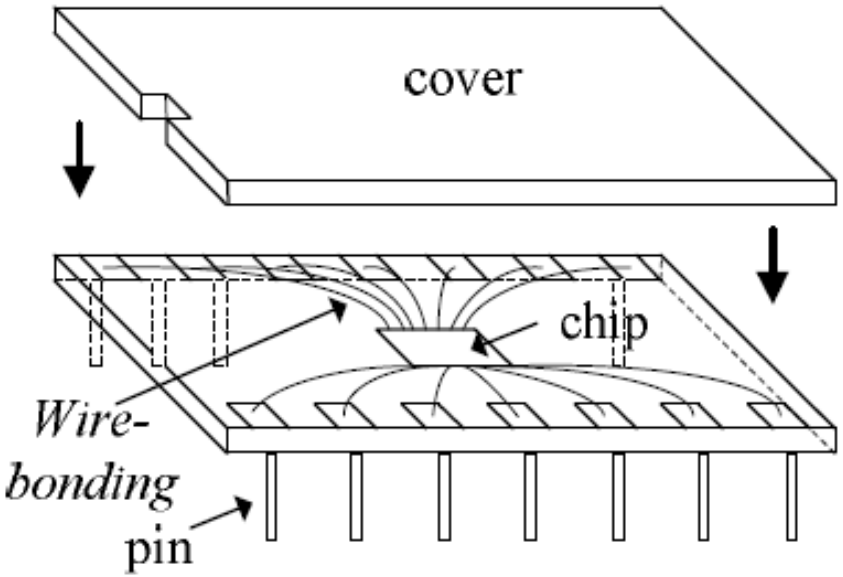
UTM
UNIVERSITI TEKNOLOGI MALAYSIA

SKEE1223: Digital Electronics

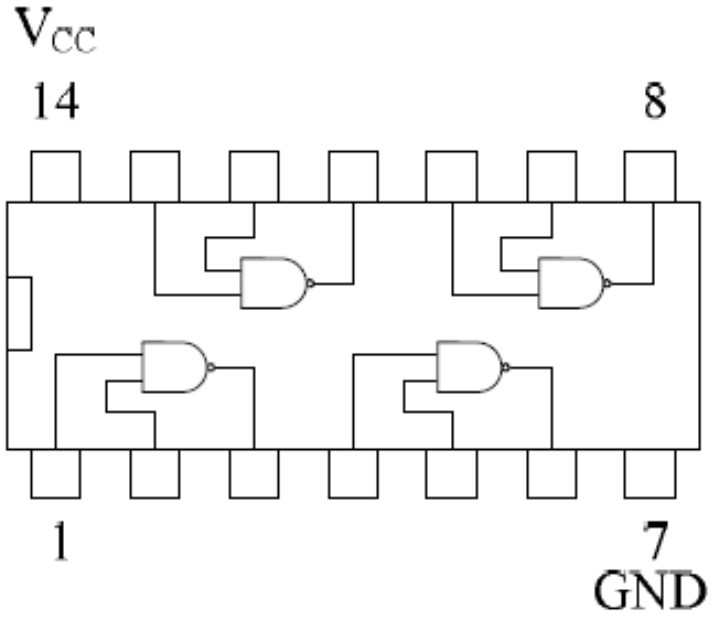
4 – Logic Gates and Boolean Algebra

Dr Michael Tan Loong Peng
PhD (Cambridge)
Senior Lecturer
Faculty of Electrical Engineering
Universiti Teknologi Malaysia

Gates packaging

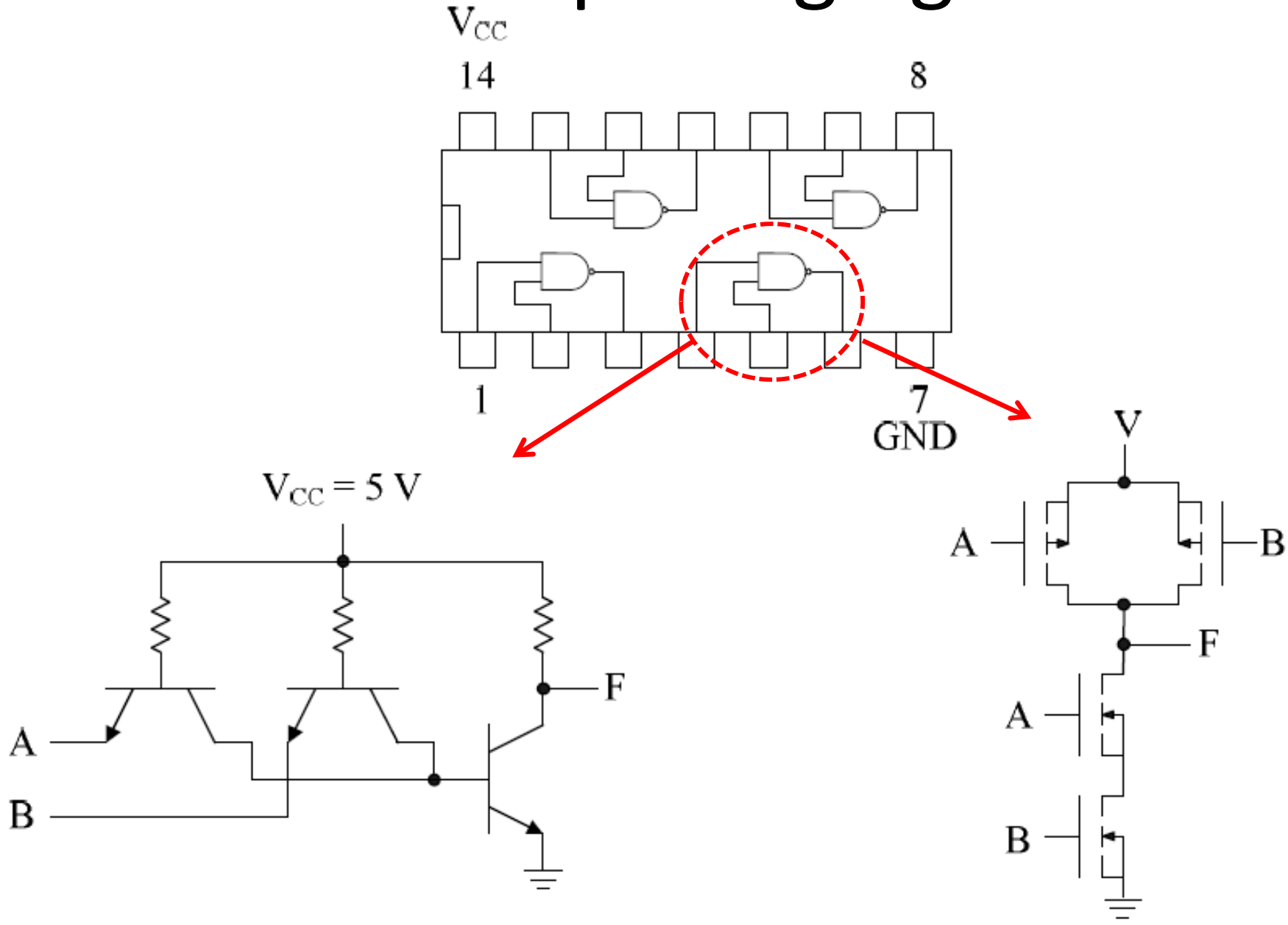


(a) packaging



(b) pin diagram 7400

Gates packaging



Exercise 1: Least Challenging

Complete the sentences:

- If all inputs of an OR gate are low, the output is_____
- If any input of an AND gate is low, the output is _____
- The output of a NOT gate is always the _____ of the input.

Logic Gates and Boolean Algebra

- Logic Gates
  - Inverter, OR, AND, Buffer, NOR, NAND, XOR, XNOR
- Boolean Theorem
  - Commutative, Associative, Distributive Laws
  - Basic Rules
- DeMorgan's Theorem
- Universal Gates
  - NAND and NOR
- Canonical/Standard Forms of Logic
  - Sum of Product (SOP)
  - Product of Sum (POS)
  - Minterm and Maxterm

Boolean Algebra

- Mathematics of digital system
- Important in the analysis of logic circuit
- Binary variable– a symbol (letter) used to represent a logical quantity.
- Example binary variables and their values:
  - A = 0
  - B = 1
- Complement—the inverse of a variable
  - The complement of A is $\overline{A}$ or A'.

Dr Michael Tan, michael@fke.utm.my

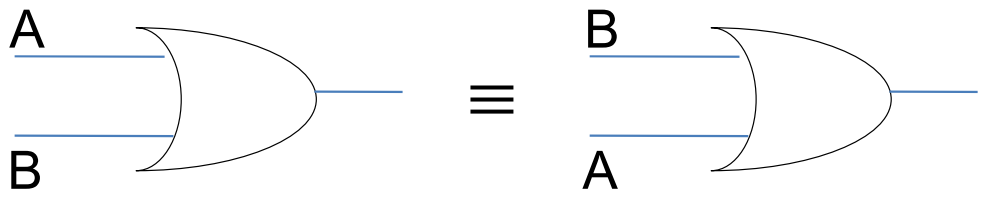
Commutative Law

- Order independent
- For addition : $X+Y = Y+ X$
- For multiplication: $XY = YX$

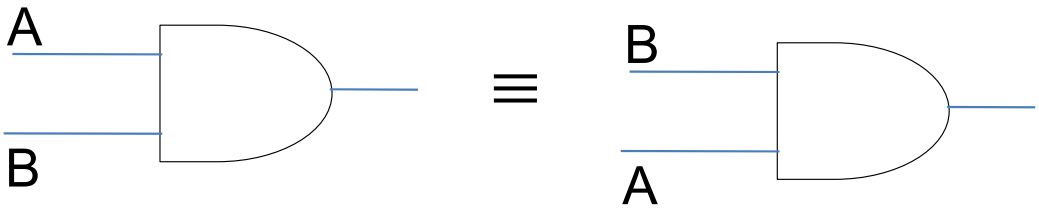
Commutative Law

In terms of the result, the order in which variables are ORed or ANDed makes no difference.

$A + B = B + A$



$AB = BA$



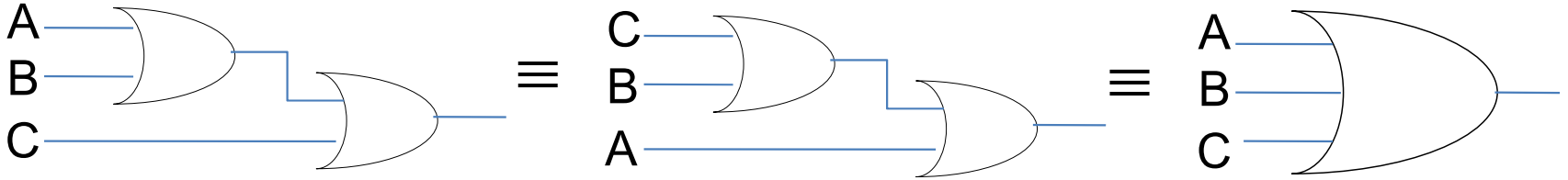
Associative Law

- Grouping is order-independent
- For addition : $X+(Y+Z) = (X+Y)+Z$
- For multiplication: $X(YZ) = (XY)Z$

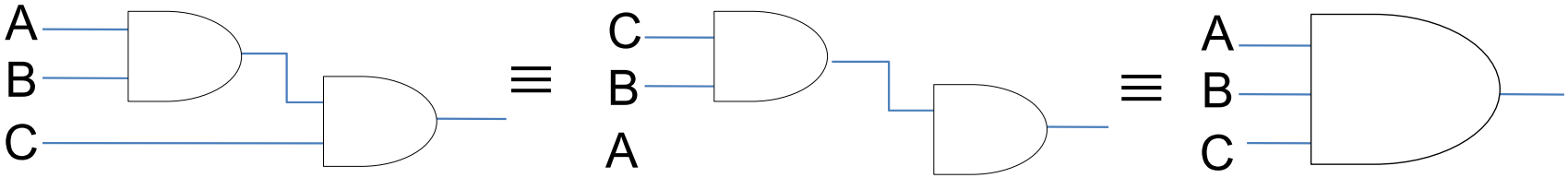
Associative Law

When ORing or ANDing more than two variables, the result is the same regardless of the grouping of the variables.

$$A + (B + C) = (A + B) + C$$



$$A(BC) = (AB)C$$

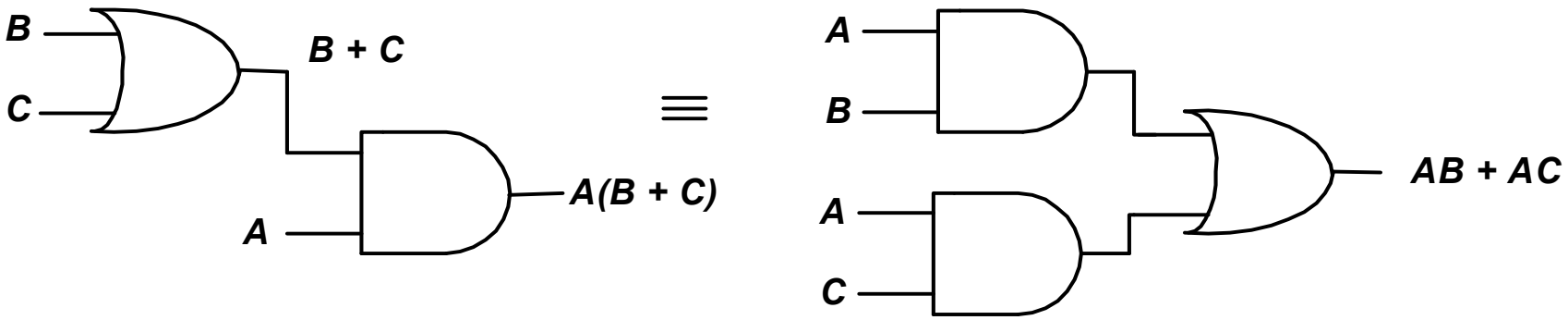


Distributive Law

Expanding an expression by multiplying term by term

X(Y+Z) = XY + XZ

$$A(B + C) = AB + AC$$



Distributive Law

A common variable can be factored from an expression just as in ordinary algebra.

$$AB + AC = A(B + C)$$

12 Boolean Rules

1. $A + 0 =$
2. $A + 1 =$
3. $A * 0 =$
4. $A * 1 =$
5. $A + A =$
6. $A + \overline{A} =$
7. $A \bullet A =$
8. $A \bullet \overline{A} =$
9. $\overline{\overline{A}} =$
10. $A+AB =$
11. $A+\overline{A}B =$
12. $(A+B)(A+C) =$

12 Boolean Rules

1. $A + 0 = A$
2. $A + 1 = 1$
3. $A * 0 = 0$
4. $A * 1 = A$
5. $A + A = A$
6. $A + \overline{A} = 1$
7. $A \bullet A = A$
8. $A \bullet \overline{A} = 0$
9. $\overline{\overline{A}} = A$
10. $A+AB = A$
11. $A+\overline{A}B = A+B$
12. $(A+B)(A+C) = A + BC$

Boolean Simplification - Example

- Using boolean theorem, Simplify the expression:

$$AB + A(B + C) + B(B + C)$$

Apply distributive law,

$$AB + AB + AC + BB + BC$$

Apply rule 7 (BB = B), and rule 5 (AB + AB = AB)

$$AB + AC + B + BC$$

Apply rule 10 (B + BC = B)

$$AB + AC + B$$

Boolean Simplification - Example

$$AB + AC + B$$

Apply rule 10 (AB + B = B)

$$B + AC$$

At this point, the expression is simplified as much as possible

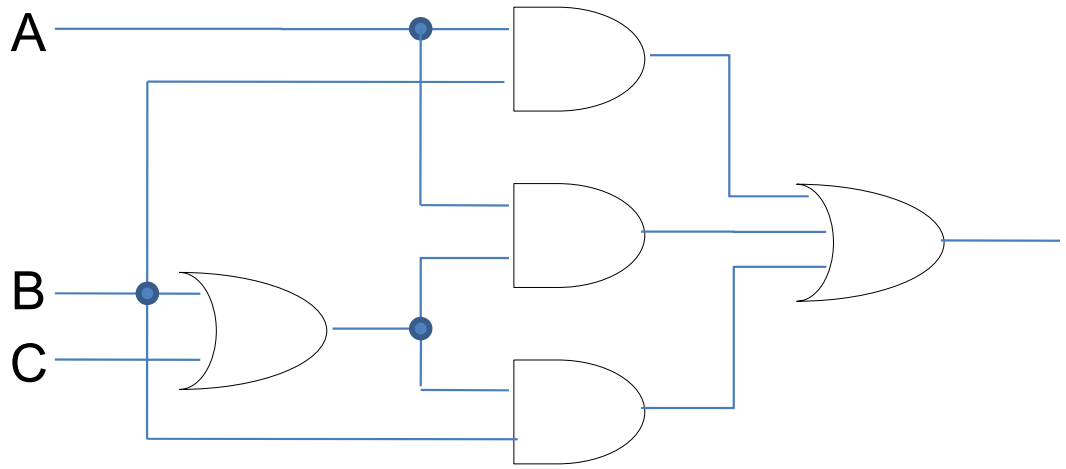
Original expression is $AB + A(B + C) + B(B + C)$

Which is logically equal to $B + AC$

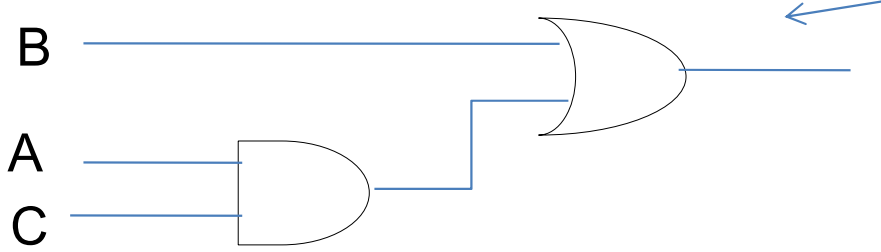
In terms of design, what is the advantage of Boolean simplification?

Boolean Simplification - Example

Original expression is $AB + A(B + C) + B(B + C)$



Which is logically equal to $B + AC$



Faster
Compact design
Lower cost

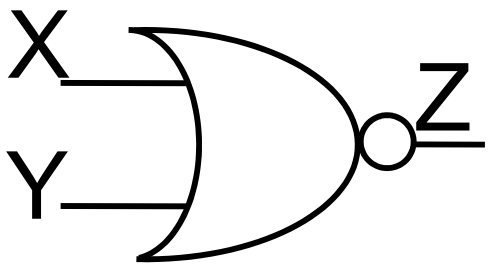
Boolean Simplification - Example

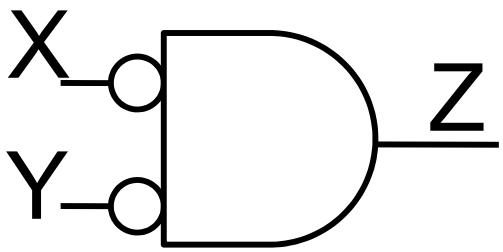
- Applying boolean theorem for logic simplification depends on a thorough knowledge of boolean algebra, with some ingenuity and cleverness
- Please look at Floyd's book examples 4-10, 4-11, and 4-12, as well as some exercises in the book to gain experience

DeMorgan's Theorem

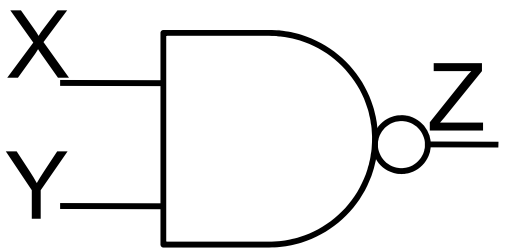
- DeMorgan proposed two theorems that are an important part of Boolean Algebra.

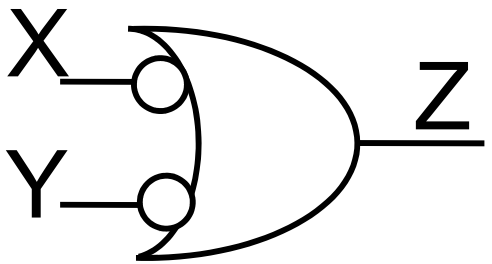
(1) $\overline{X+Y} = \overline{X} \cdot \overline{Y}$





(2) $\overline{X \cdot Y} = \overline{X} + \overline{Y}$





Truth Tables for DeMorgan's

(1) $\overline{X+Y} = \overline{X} \cdot \overline{Y}$

A) X	Y	X + Y	$\overline{X+Y}$	B) X	Y	$\overline{X}$	$\overline{Y}$	$\overline{X} \cdot \overline{Y}$
0	0	0	1	0	0	1	1	1
0	1	1	0	0	1	1	0	0
1	0	1	0	1	0	0	1	0
1	1	1	0	1	1	0	0	0

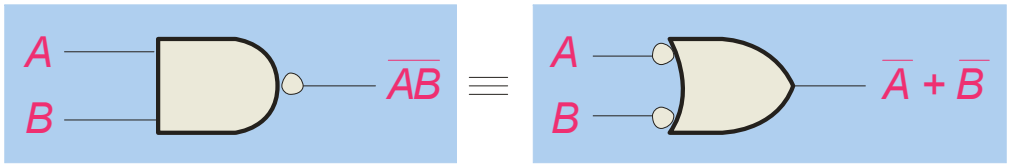
(2) You can try to construct truth table for the other DeMorgan's Theorem.

DeMorgan's Theorem

- The complement of a product of variables is equal to the sum of the complemented variables

Theorem 1

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



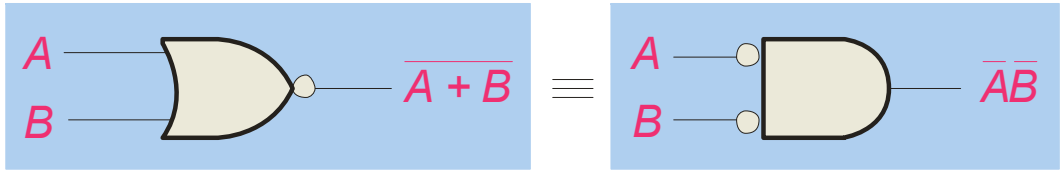
NAND Negative-OR

A	B	$\overline{A \cdot B}$	$\overline{A}+\overline{B}$
0	0	1	1
0	1	1	1
1	0	1	1
1	1	0	0

DeMorgan's Theorem

Theorem 2

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

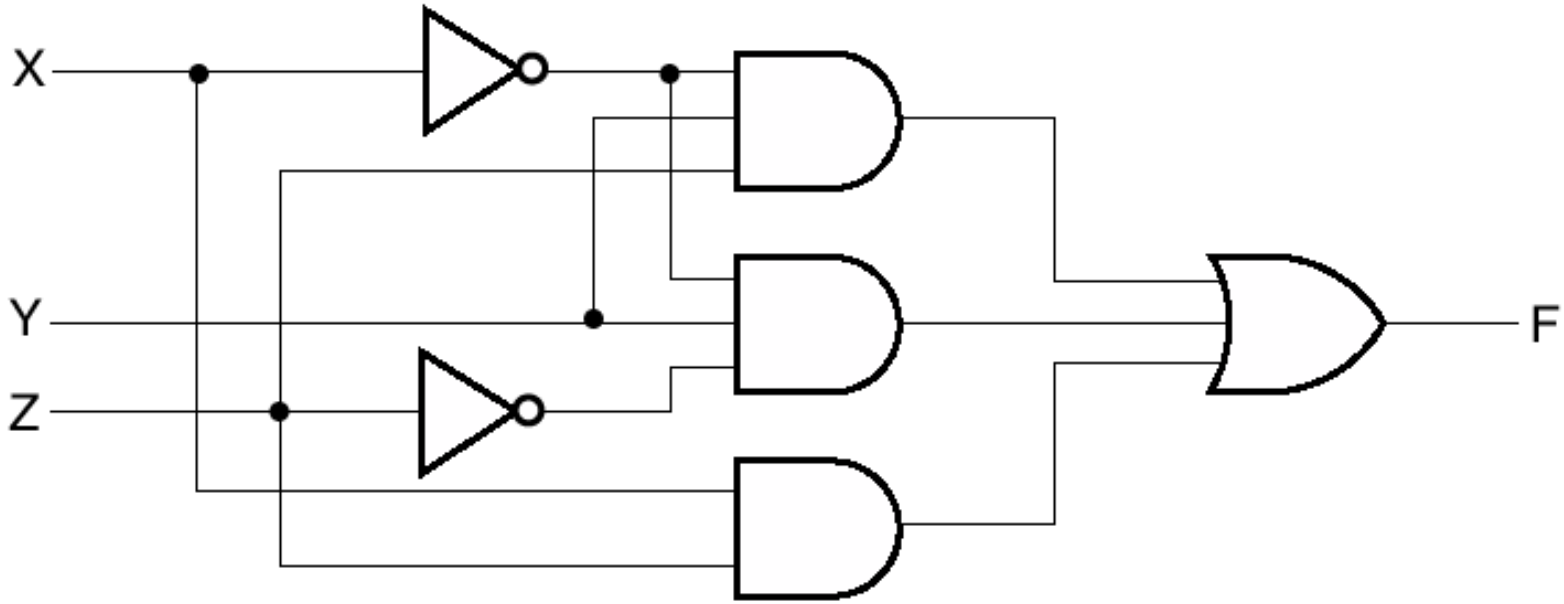


NOR Negative-AND

A	B	$\overline{A+B}$	$\overline{A} \cdot \overline{B}$
0	0	1	1
0	1	0	0
1	0	0	0
1	1	0	0

Algebraic Manipulation

- Consider the following function



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

Simplify Function

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

Apply $X(Y+Z) = XY + XZ$

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

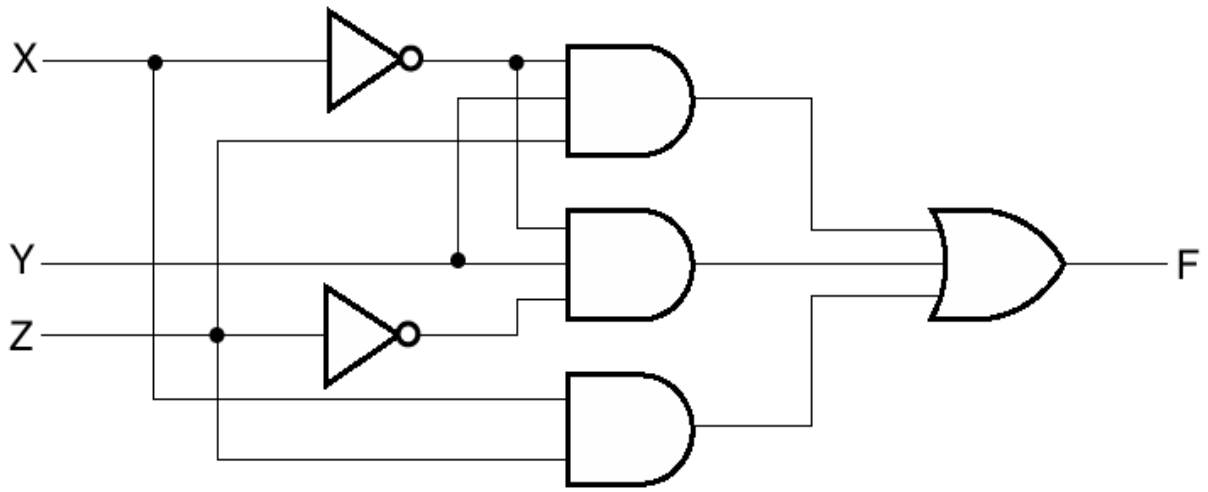
Apply $X + \overline{X} = 1$

$$F = \overline{X}Y \bullet 1 + XZ$$

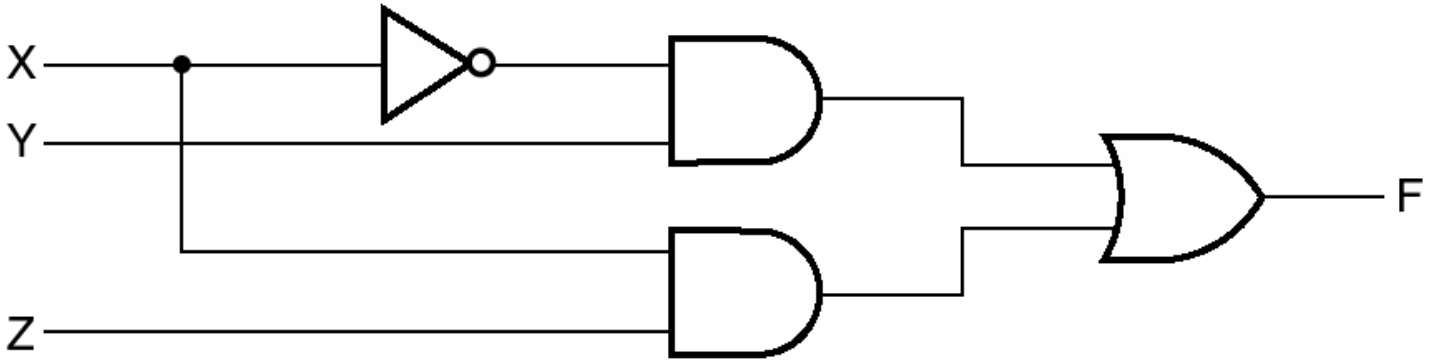
Apply $X \cdot 1 = X$

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

Fewer Gates



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



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

Implementation of Boolean Function with Gates

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

Priority in Simplifying Functions

- Parentheses
- NOT
- AND
- OR

Exercises

1. Apply DeMorgan's theorems to each of the following expressions:

   a. $\overline{(A+B+C)D}$

   b. $\overline{ABC+DEF}$

   c. $\overline{A\overline{B}+\overline{C}D+EF}$

Exercises

2. Simplify the following using Boolean algebra techniques:

$a.\quad A\overline{B} + AB$

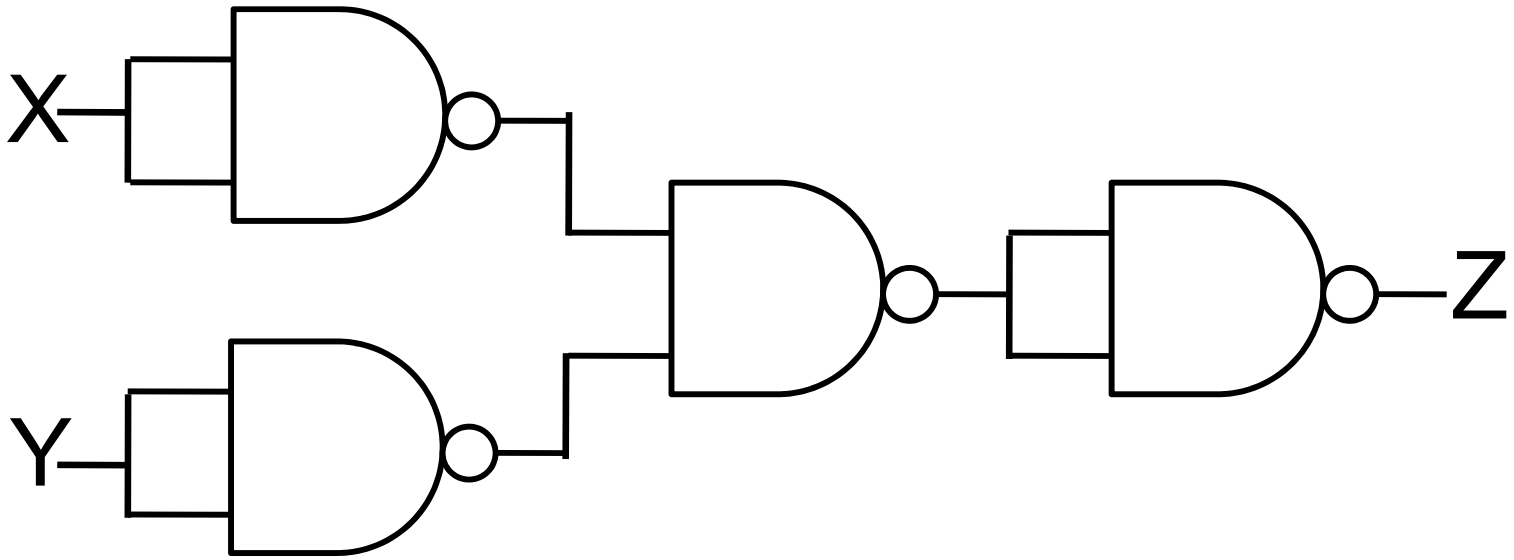
$b.\quad AB + A(B + C) + B(B + C)$

$c.\quad C[\overline{A}\,\overline{B} + A\overline{B}(C + BD)]$

$d.\quad \overline{AB + AC} + \overline{A}\,\overline{B}C$

Exercises

3. Simplify the following circuit.

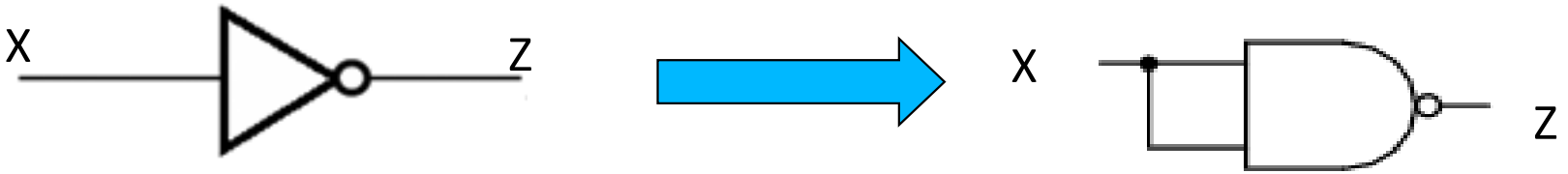


Universal Gates

- NAND and NOR gates are known as Universal gates because all logic gates can be represented by NAND and NOR
- NAND and NOR are the cheapest and smallest to manufacture in Integrated Circuits compared to AND and OR
- Therefore NAND and NOR are almost always used in practical circuit design

NAND Universal Gates

- How to represent inverter using NAND gates?



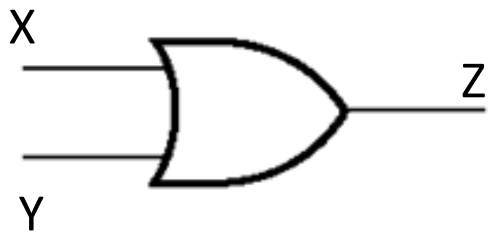
NAND gate truth table

X	Y	$Z=\overline{X\cdot Y}$
0	0	1
~~0~~	~~1~~	~~1~~
~~1~~	~~0~~	~~1~~
1	1	0

Short the inputs together

NAND Universal Gates

- How to represent OR gate using NAND gates?

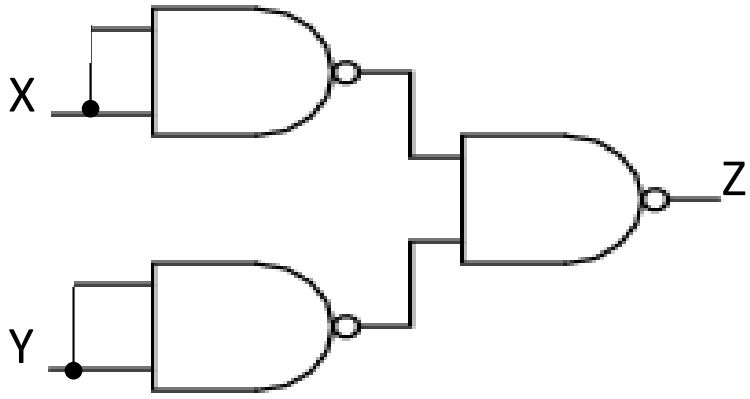


Logic expression for OR gate:

$$X + Y \equiv \overline{\overline{X + Y}}$$

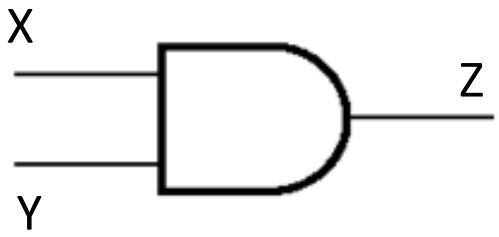
Using DeMorgan's Theorem,

$$\overline{\overline{X} \cdot \overline{Y}}$$



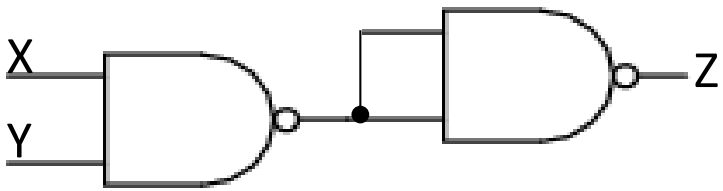
NAND Universal Gates

- How to represent AND gate using NAND gates?



Logic expression for AND gate:

$$X \cdot Y \equiv \overline{\overline{X \cdot Y}}$$



NOR Universal Gates