MTE 202, 1st Semester 2018-2019
Digital Circuits

Chapter 1: Introduction

- **Teaching Assistants:**
  - Eng.: Mohamed Aziz

- **Grading :**

Midterm + Quiz	40
Attendance	10
Circuits	10
Final Exam	40
Total	100

- Just follow the course notes

  http://bu.edu.eg/staff/motazali3-courses/14758

- **Textbook:**
  - Digital Fundamentals, Thomas Floyd, 11th Edition, Prentice Hall, 2014.

- **Lectures (Video playlist)**

  https://www.youtube.com/watch?v=CvpTpVHEpeY&list=PLJcbpTTZo96tpTEcwrUojt51wQPhHh8IX

Course Contents

- Introduction (Revision).
  - Logic Gates
  - Boolean Algebra
- Combinational Logic Circuits.
- Flip Flops and related devices.
- Counters and Registers.
- Sequential Circuits.

Logic Gates

Binary Logic and Gates

- Binary variables take on one of two values.
- Logical operators operate on binary values and binary variables.
- Basic logical operators are the logic functions AND, OR and NOT.
- Logic gates implement logic functions.
- Boolean Algebra: a useful mathematical system for specifying and transforming logic functions.
- We study Boolean algebra as a foundation for designing and analyzing digital systems!

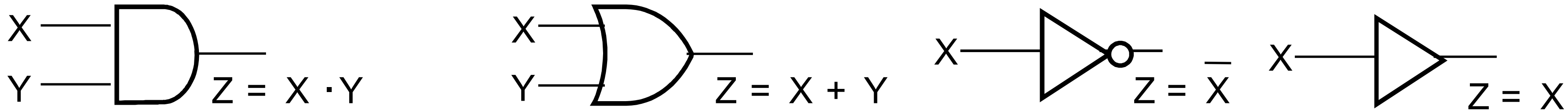
Binary Variables

- Recall that the two binary values have different names:
  - True/False
  - On/Off
  - Yes/No
  - 1/0
- We use 1 and 0 to denote the two values.
- Variable identifier examples:
  - A, B, y, z, or X_1 for now
  - RESET, START_IT, or ADD1 later

Logical Operations

- The three basic logical operations are:
  - AND
  - OR
  - NOT
- AND is denoted by a dot (·).
- OR is denoted by a plus (+).
- NOT is denoted by an over bar (¯).

Truth Tables



AND gate — OR gate — NOT gate or inverter — Buffer

AND		
X	Y	Z = X·Y
0	0	0
0	1	0
1	0	0
1	1	1

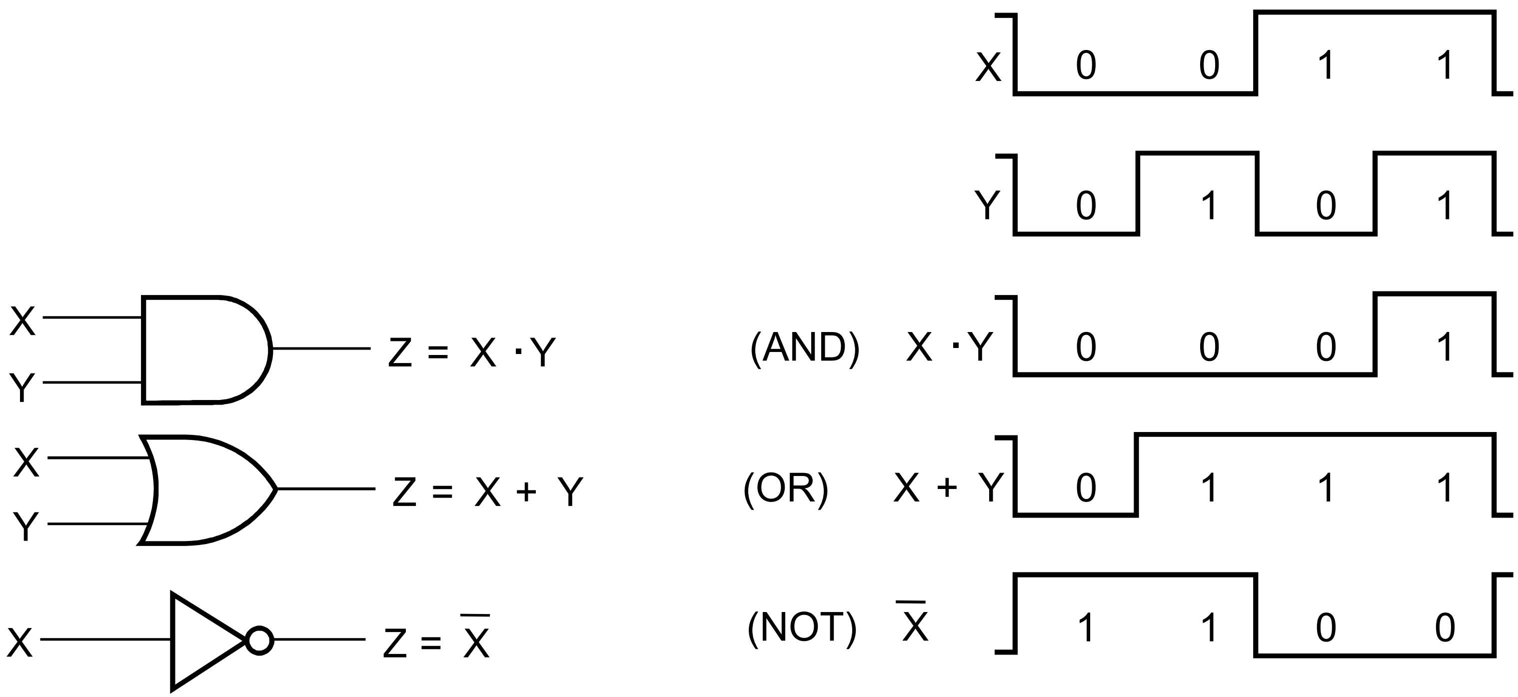
OR		
X	Y	Z = X+Y
0	0	0
0	1	1
1	0	1
1	1	1

NOT	
X	$Z = \overline{X}$
0	1
1	0

Buffer	
X	Z = X
0	0
1	1

Logic Gate Behavior

- A waveform in time domain can be applied to logic gate



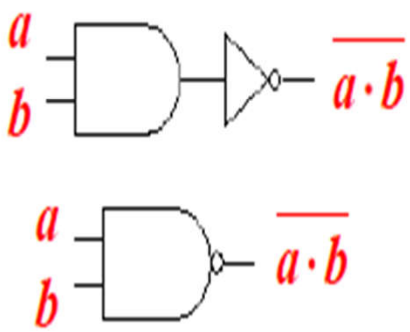
NAND & NOR Logic Gates

- Both NAND and NOR are composite functions, but we discuss them here, since they are commercially available.
- The NAND function is inverted AND; NOR is inverted OR.
- Mathematically, we use a horizontal bar to indicate the inversion.

  i.e.: a NAND b is written as $\overline{a.b}$, and a NOR b is written as $\overline{a+b}$
- As NAND is the reverse of AND, the NAND output is 1 unless all inputs are 1. Likewise, NOR outputs 0 unless all inputs are 0.

NAND & NOR Logic Gates

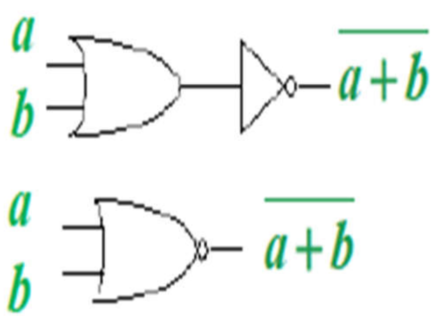
NAND



NAND Truth Table

a	b	a NAND b
0	0	1
0	1	1
1	0	1
1	1	0

NOR

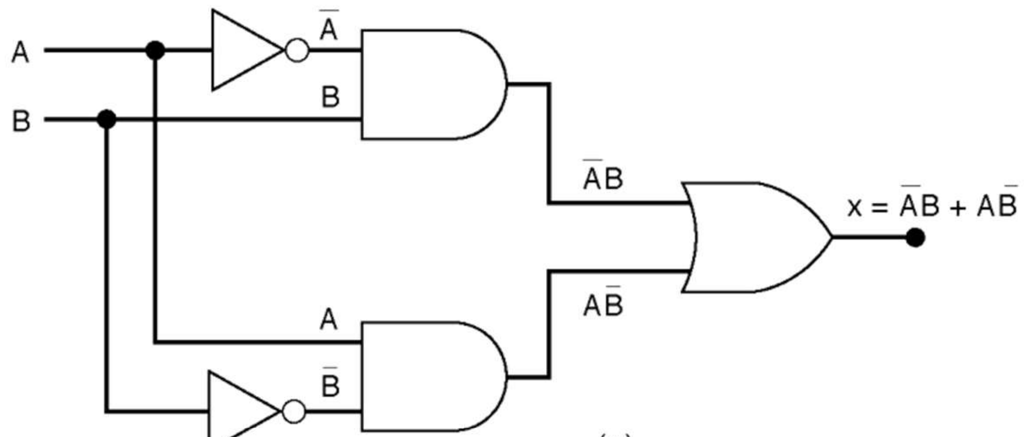


NOR Truth Table

a	b	a NOR b
0	0	1
0	1	0
1	0	0
1	1	0

Exclusive-OR (XOR)

- Exclusive-OR (XOR) produces a HIGH output whenever the two inputs are at opposite levels.

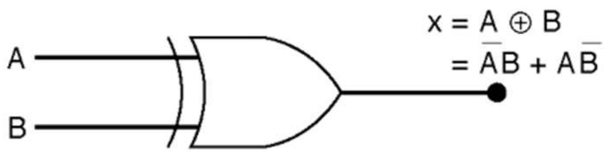


(a)

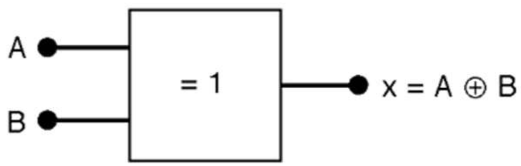
A	B	x
0	0	0
0	1	1
1	0	1
1	1	0

$X = A \oplus B$

XOR gate symbols



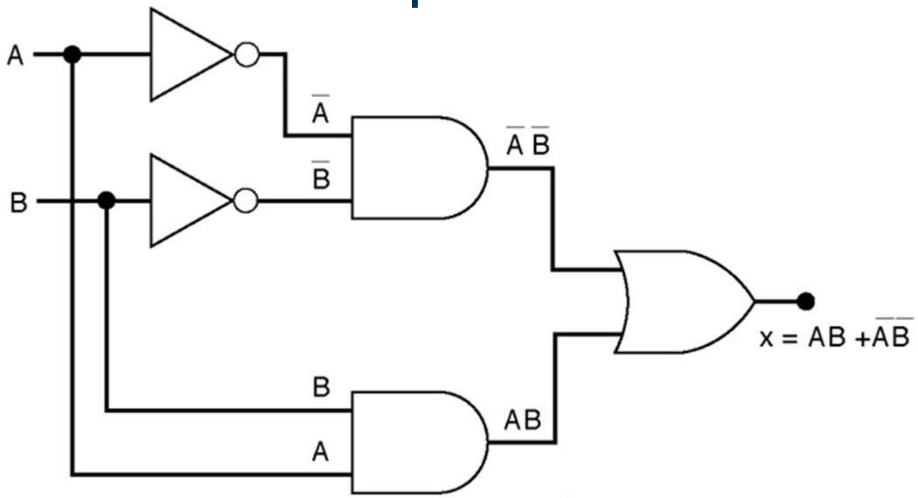
(b)



(c)

Exclusive-NOR (XNOR)

- Exclusive-NOR (XNOR) produces a HIGH output whenever the two inputs are at the same level.

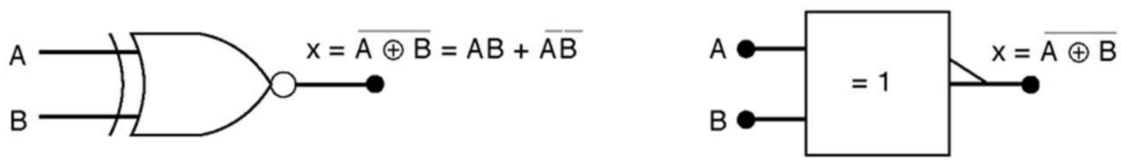


(a)

A	B	x
0	0	1
0	1	0
1	0	0
1	1	1

$$X = \overline{A \oplus B}$$

XNOR gate symbols



(b) (c)

Boolean Algebra

- Invented by George Boole in 1854.
- An algebraic structure defined by a set B = {0, 1}, together with two binary operators (+ and ·) and a unary operator ($\overline{\ }$).
- Set of axioms and theorems to simplify Boolean equations.
- Like regular algebra, but in some cases simpler because variables can have only two values (1 or 0).
- Axioms and theorems obey the principles of duality:
  - ANDs and ORs interchanged, 0's and 1's interchanged

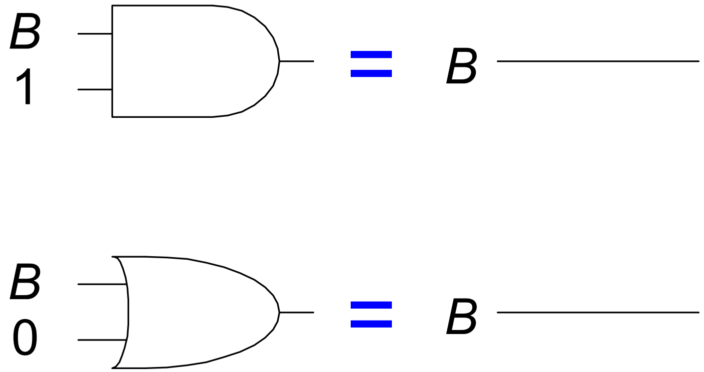
Boolean Axioms

	Axiom		Dual	Name
A1	$B = 0$ if $B \neq 1$	A1′	$B = 1$ if $B \neq 0$	Binary field
A2	$\overline{0} = 1$	A2′	$\overline{1} = 0$	NOT
A3	$0 \bullet 0 = 0$	A3′	$1 + 1 = 1$	AND/OR
A4	$1 \bullet 1 = 1$	A4′	$0 + 0 = 0$	AND/OR
A5	$0 \bullet 1 = 1 \bullet 0 = 0$	A5′	$1 + 0 = 0 + 1 = 1$	AND/OR

	Theorem		Dual	Name
T1	$B \bullet 1 = B$	T1′	$B + 0 = B$	Identity
T2	$B \bullet 0 = 0$	T2′	$B + 1 = 1$	Null Element
T3	$B \bullet B = B$	T3′	$B + B = B$	Idempotency
T4		$\overline{\overline{B}} = B$		Involution
T5	$B \bullet \overline{B} = 0$	T5′	$B + \overline{B} = 1$	Complements

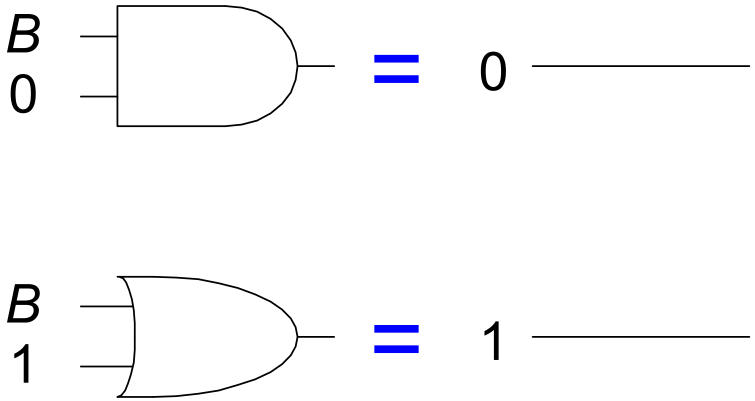
T1: Identity Theorem

- $B \bullet 1 = B$
- $B + 0 = B$



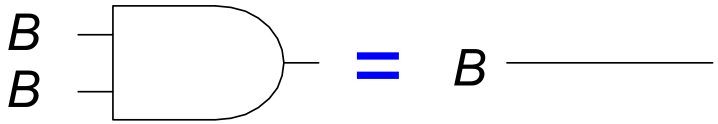
T2: Null Element Theorem

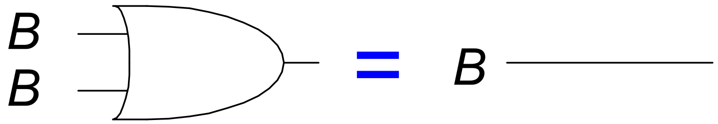
- $B \bullet 0 = 0$
- $B + 1 = 1$



T3: Idempotency Theorem

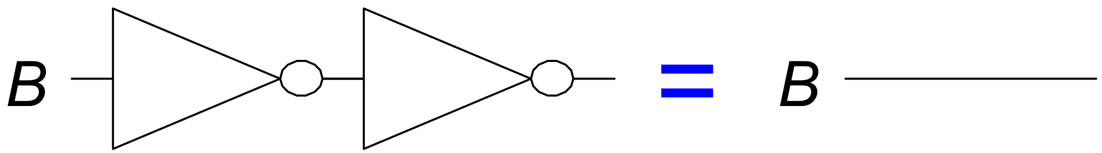
- $B \bullet B = B$
- $B + B = B$





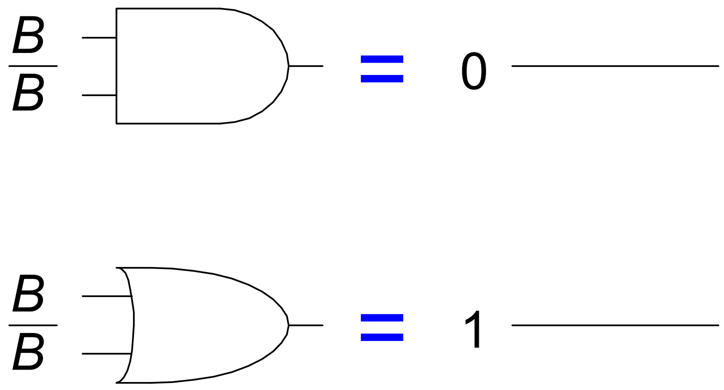
T4: Involution Theorem

- $\overline{\overline{B}} = B$



T5: Complement Theorem

- $B \bullet \overline{B} = 0$
- $B + \overline{B} = 1$



Review of Boolean Algebra & Functions

AND

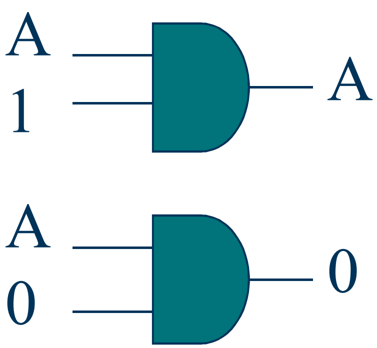
A	B	C
0	0	0
0	1	0
1	0	0
1	1	1

OR

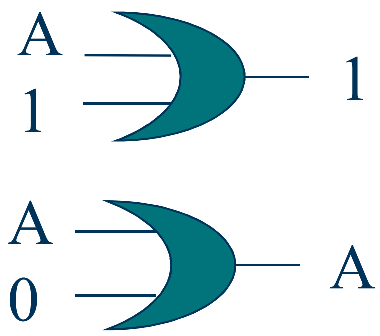
A	B	C
0	0	0
0	1	1
1	0	1
1	1	1

NOT

A	C
0	1
1	0



0 dominates in AND



1 dominates in OR

Boolean Theorems: Summary

	Theorem		Dual	Name
T1	$B \bullet 1 = B$	T1′	$B + 0 = B$	Identity
T2	$B \bullet 0 = 0$	T2′	$B + 1 = 1$	Null Element
T3	$B \bullet B = B$	T3′	$B + B = B$	Idempotency
T4		$\overline{\overline{B}} = B$		Involution
T5	$B \bullet \overline{B} = 0$	T5′	$B + \overline{B} = 1$	Complements

Useful Theorems

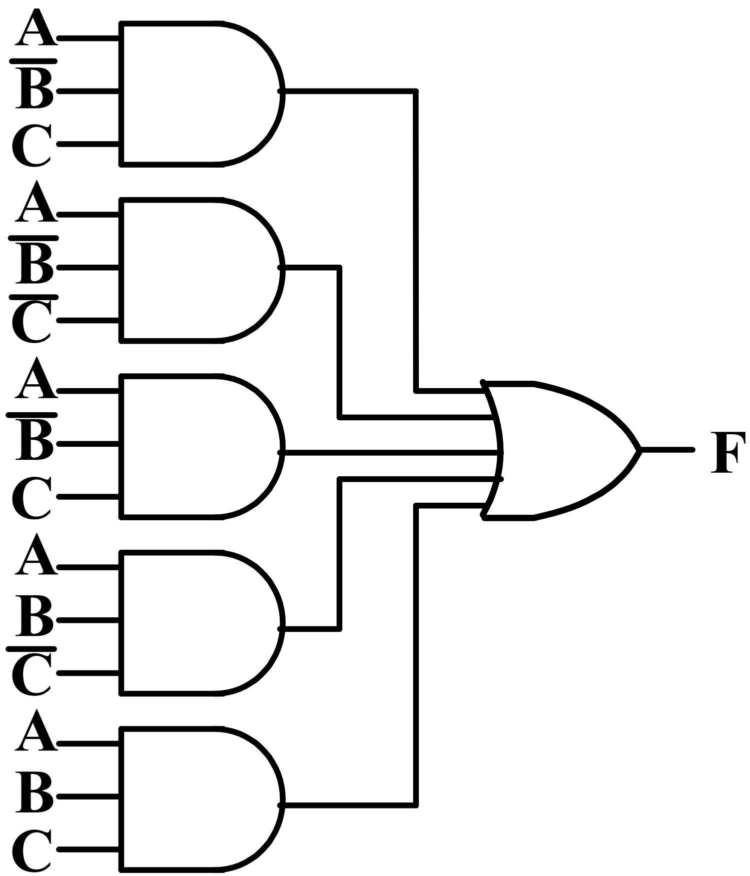
- Minimization
  $$X Y + \overline{X} Y = Y$$
- Minimization (dual)
  $$(X+Y)(\overline{X}+Y) = Y$$
- Absorption
  $$X + X Y = X$$
- Absorption (dual)
  $$X \cdot (X + Y) = X$$
- Simplification
  $$X + \overline{X} Y = X + Y$$
- Simplification (dual)
  $$X \cdot (\overline{X} + Y) = X \cdot Y$$
- DeMorgan's
  $$\overline{X + Y} = \overline{X} \cdot \overline{Y}$$
- DeMorgan's (dual)
  $$\overline{X \cdot Y} = \overline{X} + \overline{Y}$$

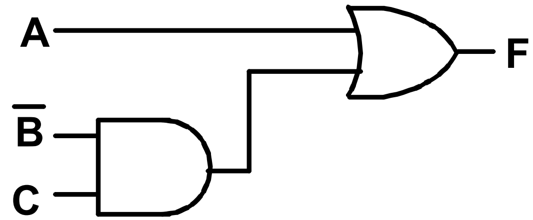
A Simplification Example:

- Writing the minterm expression:
  $F = \overline{A}\,\overline{B}\,C + A\,\overline{B}\,\overline{C} + A\,\overline{B}\,C + AB\overline{C} + ABC$
- Simplifying:
  $F = \overline{A}\,\overline{B}\,C + A\,(\overline{B}\,\overline{C} + \overline{B}\,C + B\,\overline{C} + B\,C)$
  $F = \overline{A}\,\overline{B}\,C + A\,(\overline{B}\,(\overline{C} + C) + B\,(\overline{C} + C))$
  $F = \overline{A}\,\overline{B}\,C + A\,(\overline{B} + B)$
  $F = \overline{A}\,\overline{B}\,C + A$
  $F = \overline{B}\,C + A$
- Simplified F contains 3 literals compared to 15

AND/OR Two-Level Implementation

- The two implementations for F are shown below





It is quite apparent which is simpler!

Karnaugh Maps (K-maps)

- Karnaugh maps provide an alternative way of simplifying logic circuits.
- Instead of using Boolean algebra simplification techniques, you can transfer logic values from a Boolean statement or a truth table into a Karnaugh map.
- The arrangement of 0's and 1's within the map helps you to visualise the logic relationships between the variables and leads directly to a simplified Boolean statement.

Karnaugh Maps (K-maps)

- Each **minterm** in a truth table corresponds to a cell in the K-Map.
- K-Map cells are labeled such that both horizontal and vertical movement differ only by one variable.
- Since the adjacent cells differ by only one variable, they can be grouped to create simpler terms in the sum-of-products expression.
- The **sum-of-products** expression for the logic function can be obtained by OR-ing together the cells or group of cells that contain 1s.

Karnaugh Maps (K-maps)

- Karnaugh maps, or K-maps, are often used to simplify logic problems with 2, 3 or 4 variables.

Number of Cells = 2^n ,where n is a number of variables

Two variables: n=2:
AB

A \ B	0	1
0	00 **0**	01 **1**
1	10 **2**	11 **3**

Karnaugh Maps (K-maps)

- 3 variables (ABC) Karnaugh map

Cells = 2^3=8

A \ BC	00	01	11	10
0	0 $\overline{A}\,\overline{B}\,\overline{C}$	1 $\overline{A}\,\overline{B}\,C$	3 $\overline{A}BC$	2 $\overline{A}B\overline{C}$
1	4 $A\overline{B}\,\overline{C}$	5 $A\overline{B}C$	7 ABC	6 $AB\overline{C}$

Karnaugh Maps (K-maps)

- 4 variables (ABCD) Karnaugh map

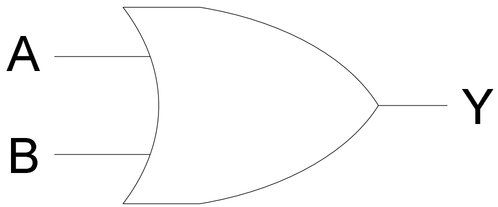
AB \ CD	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	12	13	15	14
10	8	9	11	10

K-maps Simplification

1. Construct a label for the K-Map. **Place 1s in cells corresponding to the 1s in the truth table**. Place 0s in the other cells.
2. **Identify and group** all <u>isolated 1's</u>. Isolated 1's are ones that cannot be grouped with any other one, or can only be grouped with one other adjacent one.
3. Group any hex.
4. Group any octet, even if it contains <u>some</u> 1s already grouped but not enclosed in a hex.
5. Group any quad, even if it contains <u>some</u> 1s already grouped but not enclosed in a hex or octet.
6. Group any pair, even if it contains <u>some</u> 1s already grouped but not enclosed in a hex, octet, or quad.
7. **OR together** all terms to generate the SOP equation.

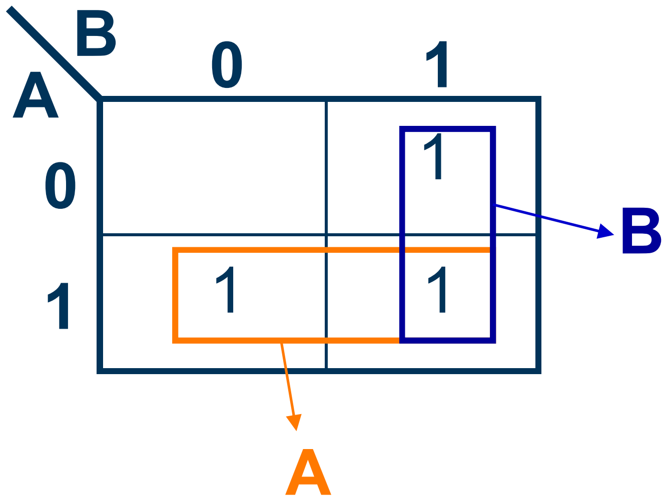
K-maps

Example 1:



A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

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



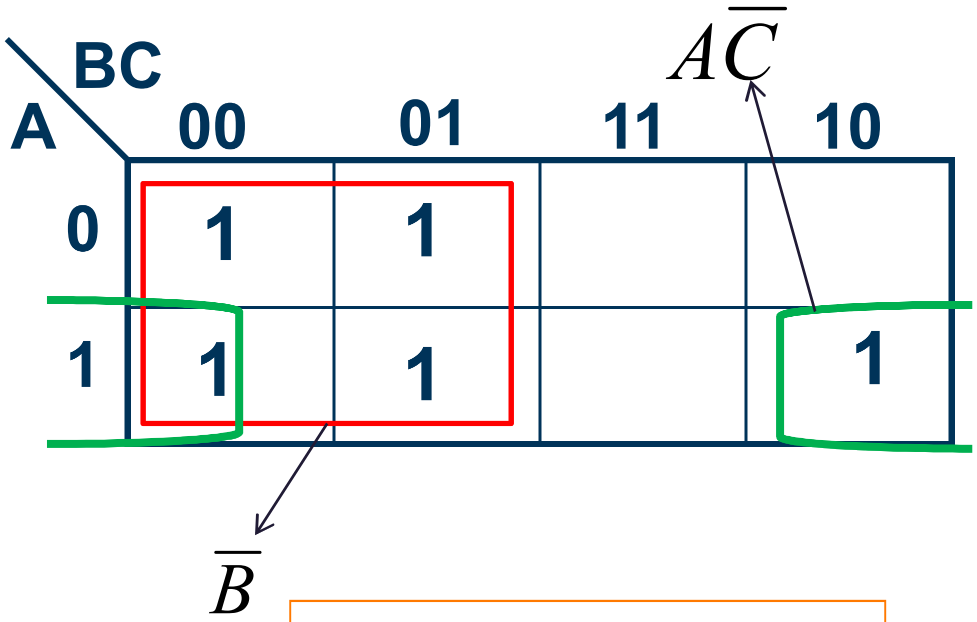
$$\boxed{Y=A+B}$$

K-maps

Example 2:

$$Y = \overline{A}\,\overline{B}\,\overline{C} + \overline{A}\,\overline{B}C + A\overline{B}\,\overline{C} + A\overline{B}C + AB\overline{C}$$

A	B	C	Y
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0



$$Y = \overline{B} + A\overline{C}$$