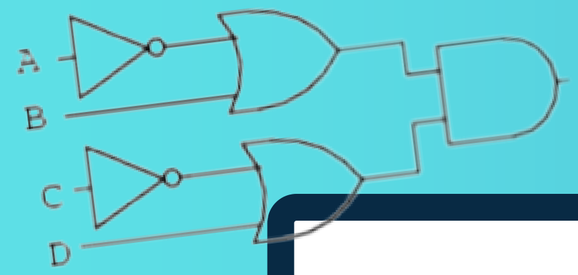
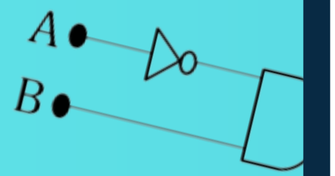
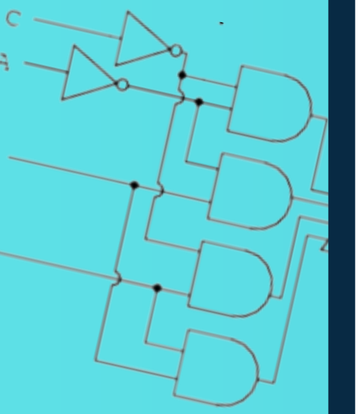
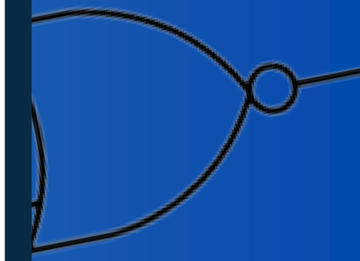
DBM20083-DISCRETE MATHEMATICS

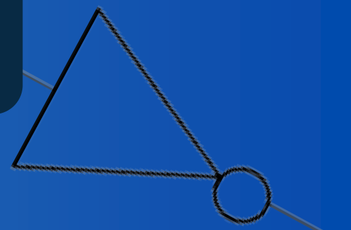
Chapter 2

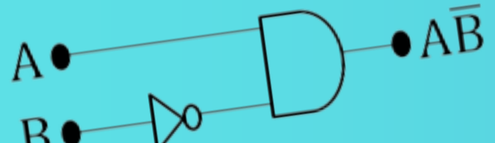


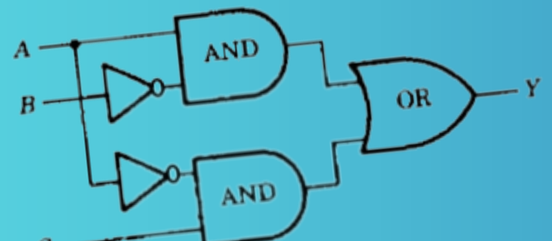


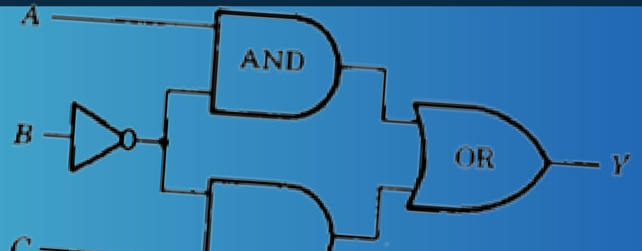












CHAPTER 2: BOOLEAN ALGEBRA

2.1 Carry out Boolean Function

2.1.1 Carry out Boolean Function

2.1.2 Use identities of Boolean algebra

2.2 Construct Logic Gates

2.2.1 Define Logic Gates

BOOLEAN FUNCTION

A **Boolean function** is decribed by an algebraic expression called **Boolean expression** which consists of binary variables, the constants 0 and 1, and the logic operation symbols.

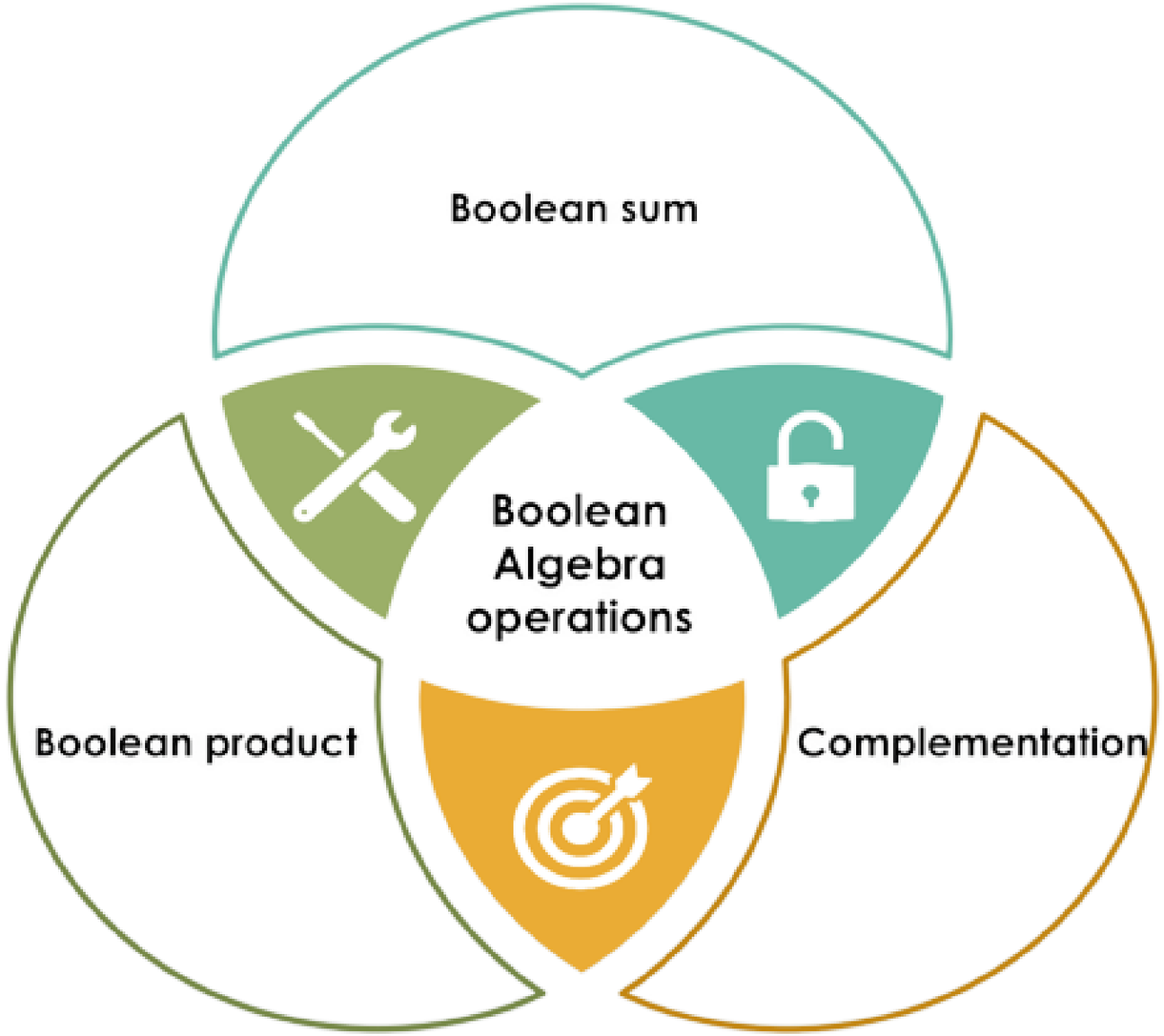
Example:

$$F(A, B, C, D) = A + \overline{BC} + ADC$$

Boolean Function

- Also known as **output**

Boolean Function


Boolean sum
Boolean
Algebra
operations
Boolean product
Complementation

Boolean Sum

- The Boolean sum, denoted by **+** or by **OR**

Example:

0 **+** 0 = 0

0 **+** 1 = 1

1 **+** 0 = 1

1 **+** 1 = 1 (**the highest output cannot exceed 1**)

- It can be completely described using a truth table. For example, if we have **two variables as the input (X and Y)** and **one output (denoted as F)**, the truth table as shown below:

X	*Y*	*F=X+Y*
0	0	0
0	1	1
1	0	1
1	1	1

Boolean Product

- The Boolean product, denoted by • or by **AND**, has the following values:

Example:

$$0 \cdot 0 = 0$$

$$1 \cdot 1 = 1$$

- It can be completely described using a truth table. For example, if we have **two variables as the input (X and Y)** and **one output (denoted as F)**, the truth table as shown below:

X	Y	$F = X \cdot Y$
0	0	0
0	1	0
1	0	0
1	1	1

COMPLEMENTATION

- The complement of an element also read as **NOT**. The NOT operation is most often designated by a **prime mark (X')** . It is sometimes indicated by an **overbar ($\overline{X}$)**.
- It can be completely described using a truth table. For example, if we have o**ne variable as the input (X)**, the complement of the input X are shown below:

X	X'
0	1
0	0

EXAMPLE

Find the value of

$$1.0 + (\overline{0 + 1})$$

Solution:

By using the definition of Boolean sum, Boolean product and complement;

$$1.0 + (\overline{0 + 1})$$

$$= 0 + (\bar{1})$$

$$= 0 + 0$$

$$= 0$$

IMPORTANT: METHOD FOR CONSTRUCTING TRUTH TABLES FOR BOOLEAN

The number of rows in the truth table is 2^n where n is the number of input/variables.

Example:

If we have two inputs (A and B), therefore, we will have $2^2 = 4$ rows. If we have three inputs(A, B and C), therefore we will have $2^3 = 8$ rows.

Look at the picture below:

Two inputs

A	*B*
0	0
0	1
1	0
1	1

Three inputs

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

EXAMPLE

Find the values of the Boolean function represented by

$$F(x, y, z) = xy + z'$$

Solution:

We need to construct the truth table:

x	y	z	xy	z'	$F = xy + z'$
0	0	0	0	1	1
0	0	1	0	0	0
0	1	0	0	1	1
0	1	1	0	0	0
1	0	0	0	1	1
1	0	1	0	0	0
1	1	0	1	1	1
1	1	1	1	0	1

Exercise A

Find the truth table T for the equivalent Boolean expression:

$$F(A, B, C) = ABC' + BC' + A'B$$

USE IDENTITIES OF BOOLEAN ALGEBRA

Identity	*Name*
$\overline{\overline{x}} = x$	Law of the double complement
$x + x = x$ $x \cdot x = x$	Idempotent laws
$x + 0 = x$ $x \cdot 1 = x$	Identity laws
$x + 1 = 1$ $x \cdot 0 = 0$	Domination laws
$x + y = y + x$ $xy = yx$	Commutative laws
$x + (y + z) = (x + y) + z$ $x(yz) = (xy)z$	Associative laws
$x + yz = (x + y)(x + z)$ $x(y + z) = xy + xz$	Distributive laws
$\overline{(xy)} = \overline{x} + \overline{y}$ $\overline{(x + y)} = \overline{x}\,\overline{y}$	De Morgan's laws
$x + xy = x$ $x(x + y) = x$	Absorption laws
$x + \overline{x} = 1$	Unit property
$x\overline{x} = 0$	Zero property

EXAMPLE

Simplify the following Boolean expression by using the suitable identities (refer to the table):

$$F\,(A, B) = A + \bar{A}B$$

Solution:

$F\,(A, B) = A + \bar{A}B$

$= (A + AB) + \bar{A}B$ **Absorption Law**

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

$= A + B(A + \bar{A})$ Factorizing the common factor B **Distributive Law**

$= A + B(1)$ **Distributive Law**

$= A + B$

Unit property

EXAMPLE

Simplify the following Boolean expression by using Boolean identities (refer to the table);

$$F\ A, B, C = (A + B)\ (A + C)$$

Solution:

$(A + B)\ (A + C) = AA + AC + BA + BC$ Expansion of the bracket **Distributive Law**

$= A + AC + BA + BC$ **Idempotent Law**

$= A(1 + C) + B(A + C)$ Factorizing the common factor A,B

$= A(1) + B(A + C)$ **Unit property** → **Distributive Law**

$= A + AB + BC$ Expansion of the bracket **Distributive Law**

$= A(1 + B) + BC$ Factorizing the common factor A

$= A(1) + BC$ **Unit property** → **Distributive Law**

$= A + BC$

Exercise B

Simplify the following Boolean expression:

a) $C + \overline{BC}$

b) $\overline{AB}(\overline{A} + B)(\overline{B} + B)$

c) $(x + y)(xz + xz') + xy + y$

d) $\overline{x}(x + y) + (y + xx)(x + \overline{y})$

2.2.1 Define Logic Gates

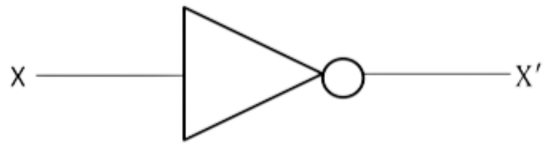
A logic gate is a building block of a digital circuit. Most logic gates have **two inputs and one output** and are **based on Boolean algebra.**

At any given moment, every terminal is in one of the two binary conditions:

- Low (0)
- High (1)

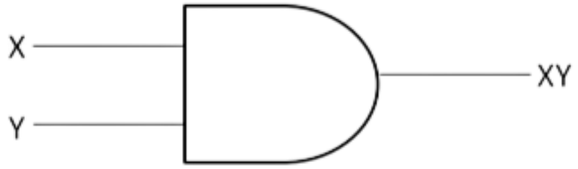
Logic Gates

The Inverter



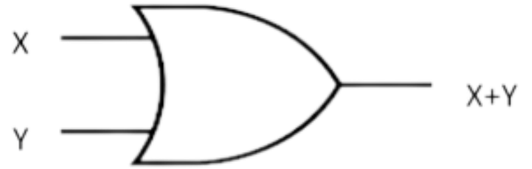
Performs the Boolean **NOT** operation **(refer to the slide no. 7)**.

The AND Gate



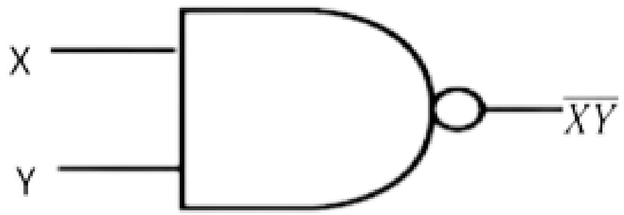
Performs the Boolean **AND** operation **(refer to the slide no. 4)**.

The OR Gate



Performs the Boolean **OR** operation **(refer to the slide no. 5)**.

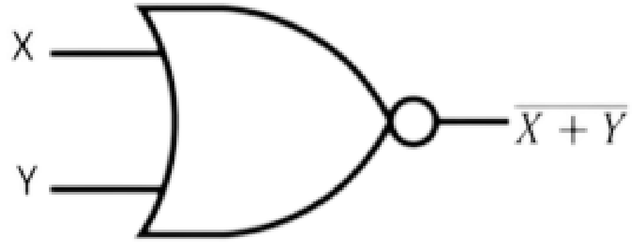
The NAND Gate



Performs the Boolean NOT-AND operation. It produces a **LOW output (0) when all inputs are HIGH (1)**. For a 2-input gate, the truth table is:

X	Y	$F=\overline{XY}$
0	0	1
0	1	1
1	0	1
1	1	0

The NOR Gate



Performs the Boolean **NOT-OR** operation. It produces a **LOW output (0) when ANY input is HIGH (1)**. For a 2-input gate, the truth table is:

X	Y	$F=\overline{X+Y}$
0	0	1
0	1	0
1	0	0
1	1	0

CHAPTER 2: BOOLEAN ALGEBRA

2.2 Construct Logic Gates

2.2.1 Construct combination of Gates

2.3 Customize minimization of circuits

2.3.1 Define minimization of circuits

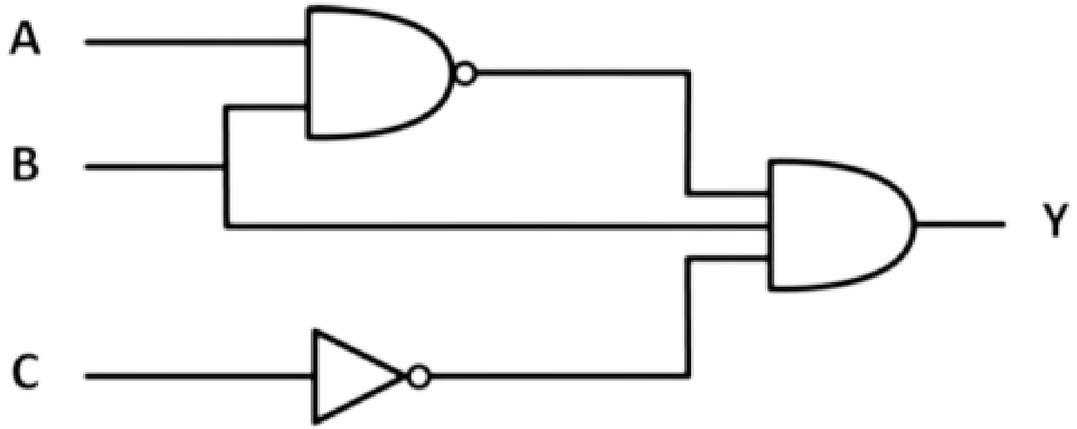
2.3.2 Use the Karnaugh Map method in two or three variables

2.3.3 Apply the Karnaugh Map in minimization of the circuits

BOOLEAN FUNCTION

- In a Combinational Logic Circuit, **the output is dependent at all times on the combination of its inputs**.
- **Combinational Logic Circuits are made up from basic logic NAND, NOR and NOT gates** that are “combined” or connected together to produce more complicated switching circuits.
- The **NAND and NOR gates** are also known as **“universal” gates**.
- The three main ways of specifying the function of a combination logic circuit are:
- **Boolean algebra**
- **Truth table**
- **Logic Diagram** (graphical representation of logic circuit)

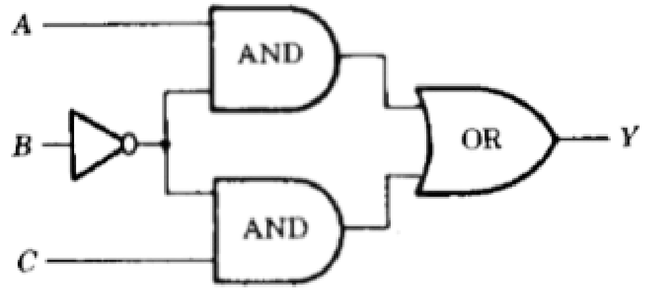
A logic circuit diagram uses the graphical representation or description of logic gates to represent a logic expression. **An example of logic circuit diagram**, shows below with **three inputs (A, B, and C) and one output (Y)**.



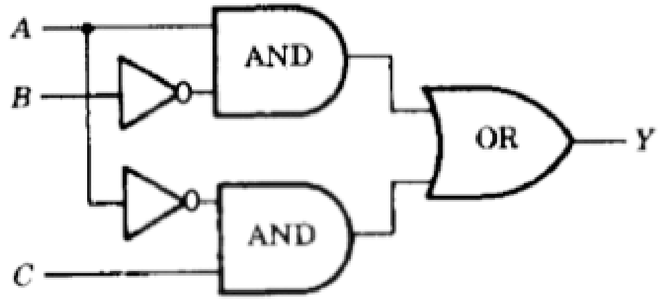
Example 2C

- Express the output Y as a Boolean expression in the inputs A, B and C for the logic circuit below:

a)



b)



2. Draw a logic circuit corresponding to the Boolean expression:

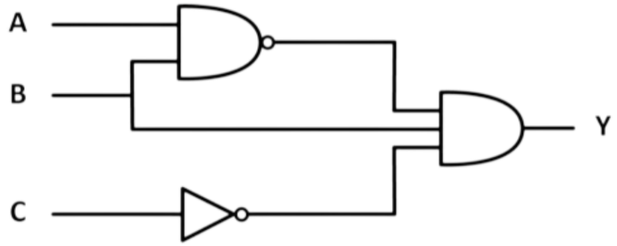
a) $Y = AB+\overline{AC}$

b) $Y=(x + y)\overline{x}$

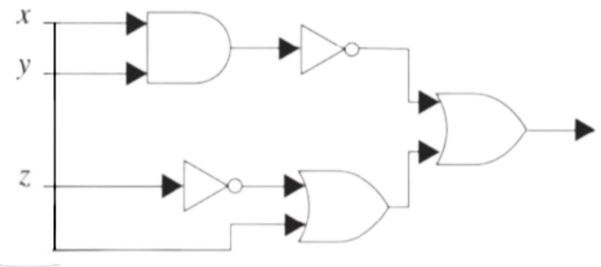
EXERCISE C

- Find the output of the given circuits:

a)



b)



2.Draw a logic circuit corresponding to the Boolean expression:

a) Y = (A+B)C

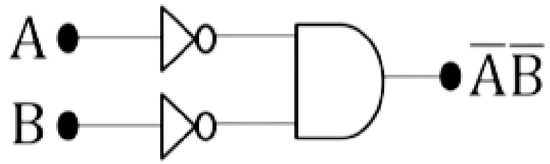
b) $Y = A + BC + \overline{D}$

c) $\overline{A + BC} + B$

d) $Y = \overline{A'B} + \overline{A + C}$

For any of the **four possible input** conditions, we can generate **a HIGH output (1)** by using an **AND gate and inverter** with the appropriate inputs to generate the requires AND product. Refer to the **FOUR examples** below:

Example 1

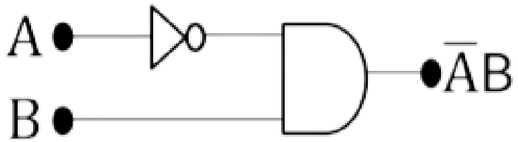


- Inputs: A and B; Output:$\overline{A}\overline{B}$
  ***AND gate** means the **product** for BOOLEAN operation
- The truth table:

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

- **BOTH low input (0)** by using AND gate and inverter generate **HIGH output (1)**

Example 2

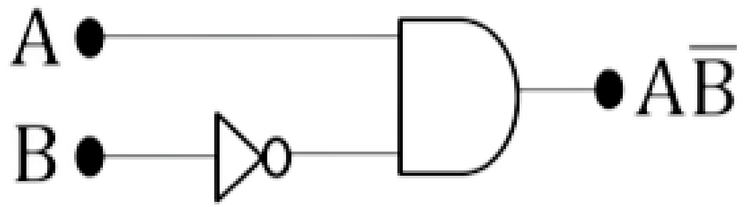


- Inputs: A and B; Output:$\overline{A}B$
  ***AND gate** means the **product** for BOOLEAN operation
- The truth table:

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

- **ONE low input (0)** by using AND gate and inverter generate **HIGH output (1)**

Example 3



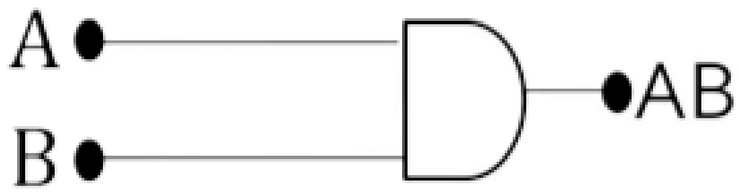
- Inputs: A and B; Output:AB

  ***AND gate** means the **product** for BOOLEAN operation
- The truth table:

A	*B*	$A\overline{B}$
1	0	1

- **ONE low input (0)** by using AND gate and inverter generate **HIGH output (1)**

Example 4



- Inputs: A and B; Output:AB

  ***AND gate** means the **product** for BOOLEAN operation
- The truth table:

A	*B*	*AB*
0	0	1

- **BOTH high input (1)** by using AND gate and inverter generate **HIGH output (1)**

EXAMPLE

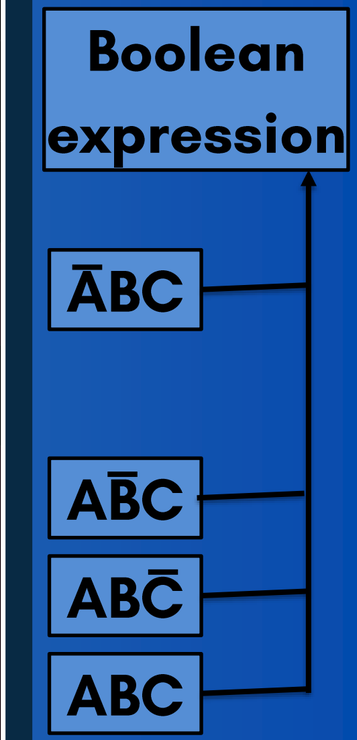
Design a logic circuit that has three inputs, A, B, and C, and whose output will be HIGH only when a majority of the inputs are HIGH.

Solution:

Step 1

Set up the truth table. On the basis of the problem statement, **the output X should be 1 whenever two or more inputs are 1**; for all other cases, the output should be 0.

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



Step 2

Write the AND term for each case where the output is a 1. There are four such cases. Refer to the TRUTH TABLE above.

Step 3. **Write the sum-of-products expression** for the output.

$$X = \bar{A}BC + A\bar{B}C + AB\bar{C} + ABC$$

Step 4. **Simplify the output expression**.

This expression can be simplified in several ways. Perhaps the quickest way is to realize that the last term ABC has two variables in common with each of the other terms. Thus, we can use the ABC term to pair with each of the other terms. **The expression is rewritten with the ABC term occurring three times**.

$$X = \bar{A}BC + ABC + A\bar{B}C + ABC + AB\bar{C} + ABC$$

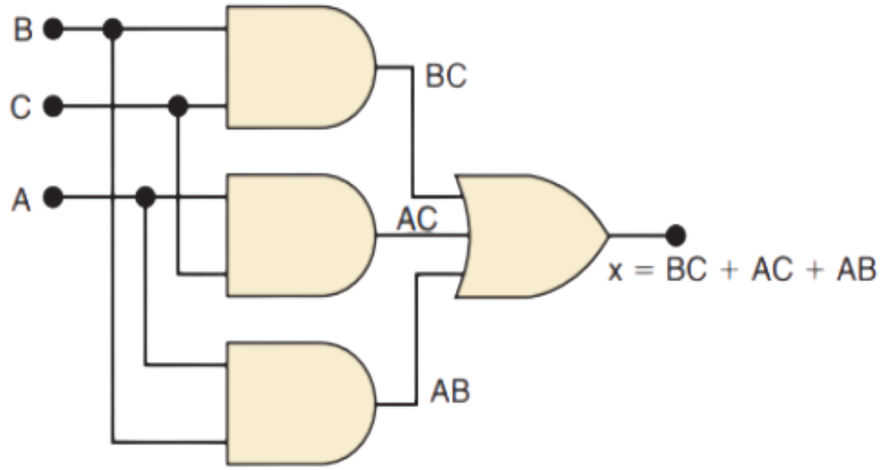
Factoring the appropriate pairs of terms, we have

$$X = BC(\bar{A} + A) + AC(\bar{B} + B) + AB(\bar{C} + C)$$

Each term in parentheses is equal to 1, so we have

$$X = BC + AC + AB$$

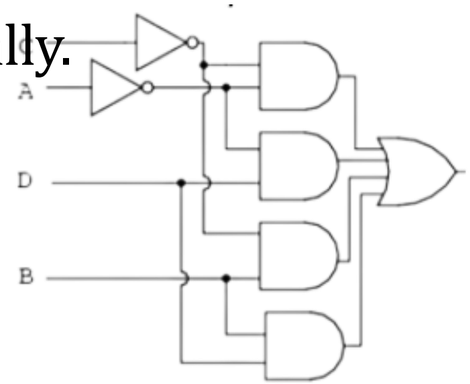
Step 5. Implement the circuit for the final expression.



Since the expression is in SOP(Sum of product) form, the circuit consists of a group of AND gates working into a single OR gate.

2.3.1 Define the minimization of circuits

The process of simplifying the algebraic expression of a Boolean function is called **minimization**. It is clear from the following image that the **minimized** version of the expression takes a less number of **logic gates** and also reduces the complexity the **circuit** substantially.



Output = $\bar{A}\bar{C} + \bar{A}D + \bar{C}B$ + BD

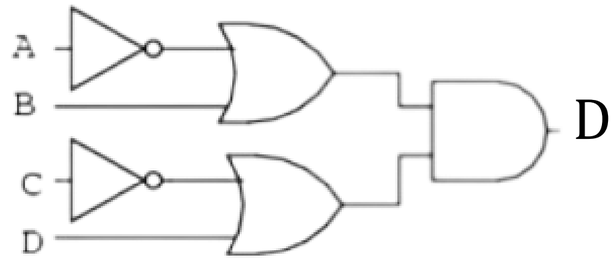
By simplifying the output using Boolean identities:

$\bar{A}\bar{C} + \bar{A}D + \bar{C}B$ + BD

$= \bar{A}(\bar{C} + \mathrm{D}) + B(\bar{C} + \mathrm{D})$

$= (\bar{A} + \mathrm{B})(\bar{C} + \mathrm{D})$

Therefore, we can draw **new LOGIC DIAGRAM** as below:



Output = $(\bar{A} + \mathrm{B})(\bar{C} + \mathrm{D})$

2.2.1 Use the Karnaugh Map method in Two or Three Variables

KARNAUGH MAPS

- Used to facilitate converting between Truth Tables and Boolean Expressions
- Make deriving a Boolean Expression much easier because they are graphical
- **Map TWO or MORE inputs to one output**
- Number of cells = 2^n. Example: if we have 2 inputs, $2^2 = 4$ cells

Example 1: K Map of 2 variables/inputs

A	B	X
0	0	$1 \rightarrow \bar{A}\bar{B}$
0	1	0
1	0	0
1	1	$1 \rightarrow AB$

$$\left\{ x = \bar{A}\bar{B} + AB \right\}$$

	$\bar{B}$	B
$\bar{A}$	1	0
A	0	1

Example 2: K Map of 3 variables/inputs

A	B	C	X
0	0	0	$1 \rightarrow \bar{A}\bar{B}\bar{C}$
0	0	1	$1 \rightarrow \bar{A}\bar{B}C$
0	1	0	$1 \rightarrow \bar{A}B\bar{C}$
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	$1 \rightarrow AB\bar{C}$
1	1	1	0

$$\left\{ X = \bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}C + \bar{A}B\bar{C} + AB\bar{C} \right\}$$

	$\bar{C}$	C
$\bar{A}\bar{B}$	1	1
$\bar{A}B$	1	0
$A\bar{B}$	1	0
$A\bar{B}$	0	0

2.3.3 Apply the Karnaugh Map in Minimization of the Circuits

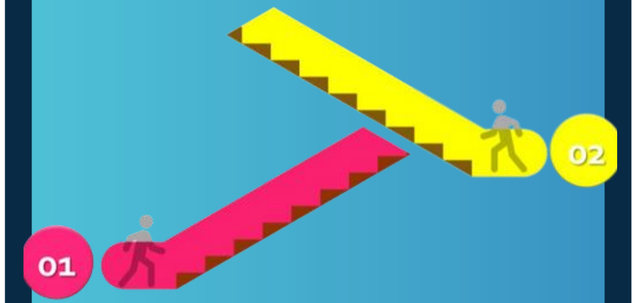
There are **7** rules used for the simplication of Boolean expressions. Refer to the explanation below:

RULE 1:

Any cell containing a **zero cannot be grouped**.

A \ BC	00	01	11	10
0	1	0	1	0
1	0	1	1	1

WRONG GROUPING



RULE 2:

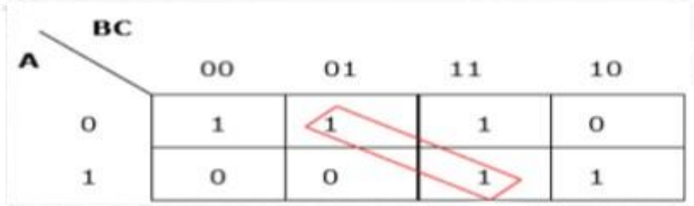
Groups must contain **2n cells (n starting from 1)**.

A \ BC	00	01	11	10
0	1	0	1	0
1	0	1	1	1

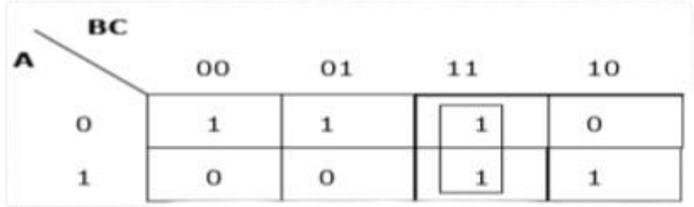
WRONG GROUPING

RULE 3:

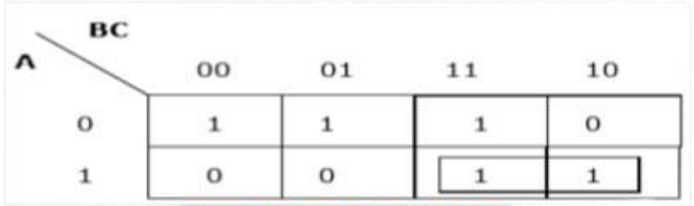
Grouping must be horizontal or vertical, but must not be diagonal.



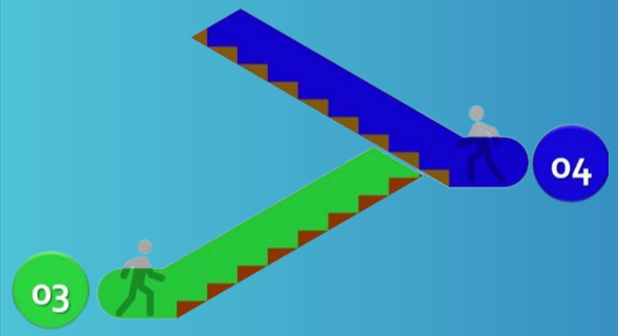
WRONG GROUPING !!!



PROPER GROUPING

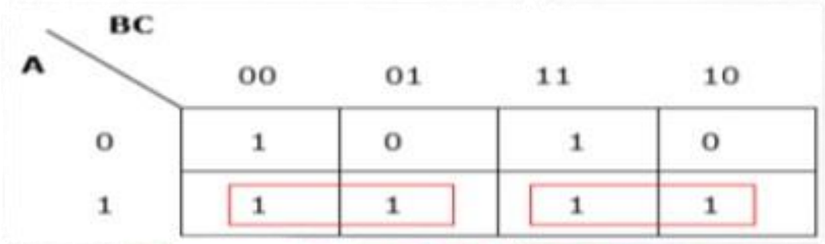


PROPER GROUPING

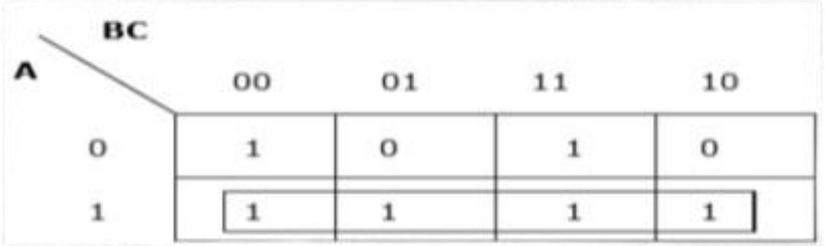


RULE 4:

Groups must be covered as largely aspossible.



INSUFFICIENT GROUPING !!!



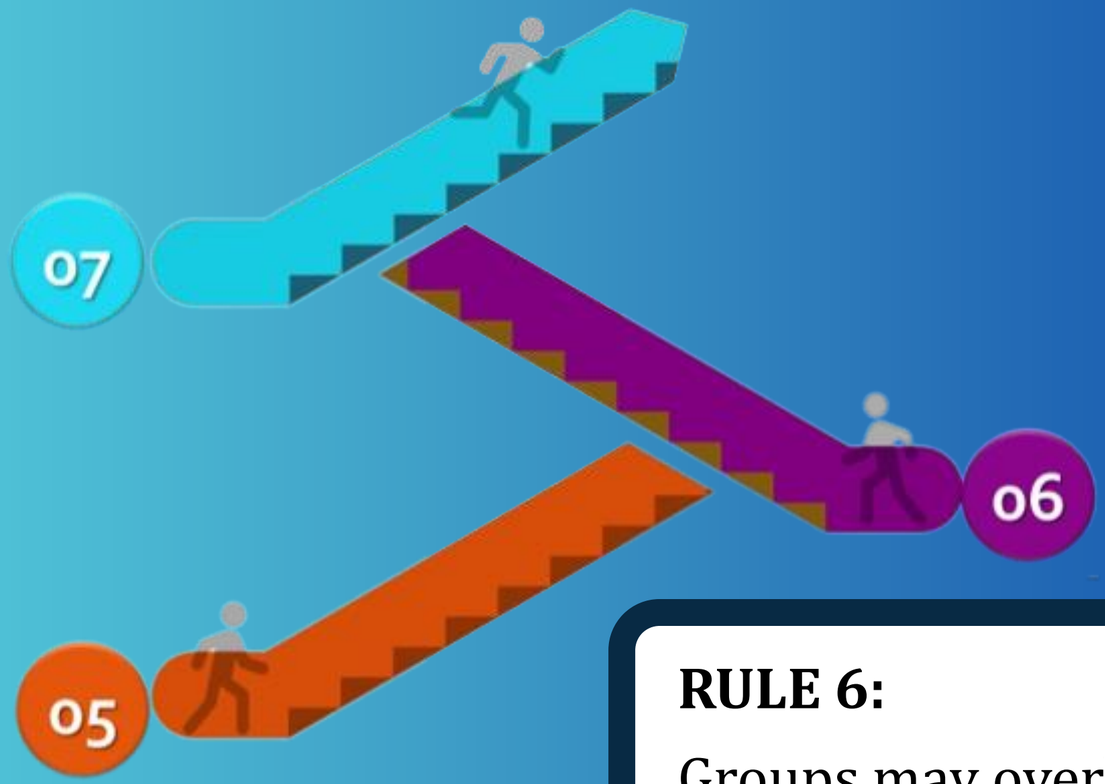
PROPER GROUPING

RULE 7:

The leftmost cell/cells can be grouped with the rightmost cell/cells and the topmost cell/cells can be grouped with the bottommost cell/cells.

A \ BC	00	01	11	10
0	1	0	0	1
1	1	0	0	1

PROPER GROUPING



RULE 5:

If 1 of any cell cannot be grouped with any other cell, it will act as a group itself.

A \ BC	00	01	11	10
0	1	0	1	0
1	0	1	0	1

PROPER GROUPING

RULE 6:

Groups may overlap but there should be as few groups as possible.

A \ BC	00	01	11	10
0	0	0	1	1
1	1	1	1	1

PROPER GROUPING

EXAMPLE

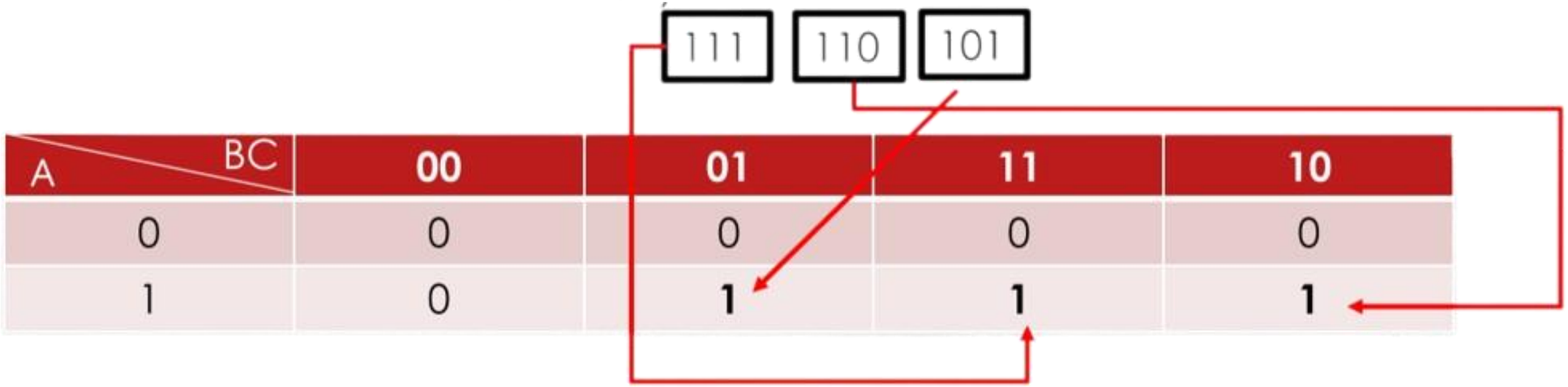
Minimize the following Boolean expression by using K-map.

$$F(A,B,C)=ABC+ABC'+AB'C$$

Solution:

First, we need to SOP (Sum of Product) for the Boolean expression given:

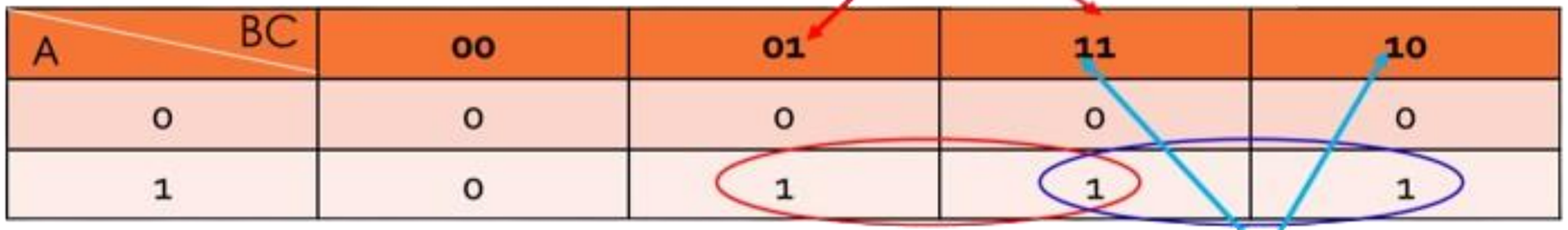
$$F(A,B,C)= ABC + ABC' + AB'C$$



A \ BC	00	01	11	10
0	0	0	0	0
1	0	1	1	1

***OTHER THAN THAT, WE PUT 0 IN THE TABLE.**

For the red circle:
Value of C remains
unchanged



A \ BC	00	01	11	10
0	0	0	0	0
1	0	1	1	1

For the blue circle:
Value of B remains
unchanged

Therefore; F(A,B,C) = AC + AB

Red circle **Blue circle**

Exercise D

- Here is a truth table for a specific three input logic circuit.

A	B	C	OUTPUT
0	0	0	1
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

Draw a K Map according to the values found in the truth table.

Exercise D cont...

2. Use Karnaugh maps to find the minimal form for each expression.

a) xy + xy’

b) xy + x’y + x’y’

c) xy’ + x’y’

d) xyz’ + xy’z + xy’z’ + x’yz + x’yz’ + x’y’z

e) xyz + xyz’ + x’yz + x’y’z

f) xyz + xyz’ + xy’z + xy’z’ + x’y’z

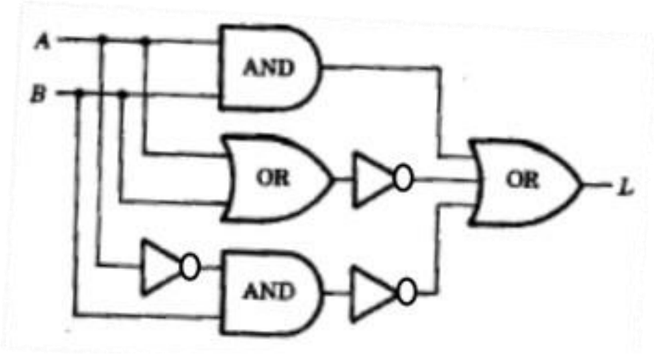
3. Design a minimal AND-OR circuit which yields the following truth table:

$$T = [A = 00001111,\ B = 00110011,$$
$$C = 01010101,\ L = 10101001]$$

EXERCISE D CONT...

4. Redesign the following circuit so that it becomes a minimal AND-OR circuit

a)



b)