DIGITAL ELECTRONICS: ECE 213

Topic: Boolean Algebra and Logic Gates

UNIT II: Combinational Logic System

Lecture No.: 9

Prepared By: Irfan Ahmad Pindoo

Assistant Professor

VLSI Design, ECE

School of Computer Science and Engineering

Introduction

- Because binary logic is used in all of today's digital computers and devices, the cost of the circuits that implement it is an important factor addressed by designers.

- Finding simpler and cheaper, but equivalent, realizations of a circuit can reap huge payoffs in reducing the overall cost of the design.

Logic Gates

- Digital systems are said to be constructed by using logic gates.
- These gates are the AND, OR, NOT, NAND, NOR, EXOR and EXNOR gates.
- The basic operations are described below with the aid of truth tables
- Truth tables are used to help show the function of a logic gate

- Boolean functions practically implemented by using electronic gates

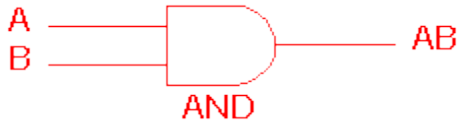
- Generally logic gate have 2 input 1 output
- Gate **INPUTS** are driven by voltages having two nominal values,

  0V logic 0 and 5V logic 1
- The **OUTPUT** of a gate provides two nominal values of voltage only

  0V logic 0 and 5V logic 1

Logic Gates

AND gate



2 Input AND gate		
A	B	A.B
0	0	0
0	1	0
1	0	0
1	1	1

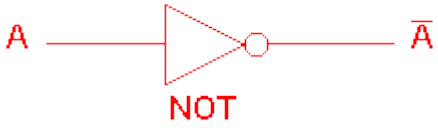
high output (1) only if **all** its inputs are high

OR gate

2 Input OR gate		
A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

high output (1) if **one or more** of its inputs are high.

NOT gate



NOT gate	
A	$\overline{A}$
0	1
1	0

produces an inverted version of the input at its output.

Logic Gates

NAND GATE

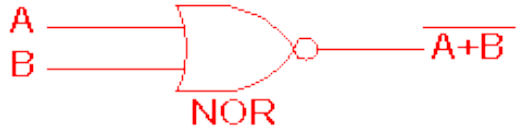
AND gate followed by a NOT gate.

2 Input NAND gate		
A	B	$\overline{A.B}$
0	0	1
0	1	1
1	0	1
1	1	0

high output (1) only if **all** its inputs are low

NOR GATE

OR gate followed by a NOT gate.



2 Input NOR gate		
A	B	$\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

Low output (0) if **one or more** of its inputs are high.

NAND and NOR gates are called *universal*

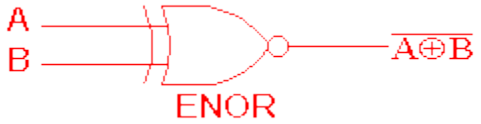
Logic Gates

EXOR GATE/XOR

2 Input EXOR gate		
A	B	A⊕B
0	0	0
0	1	1
1	0	1
1	1	0

high output (1) for different input

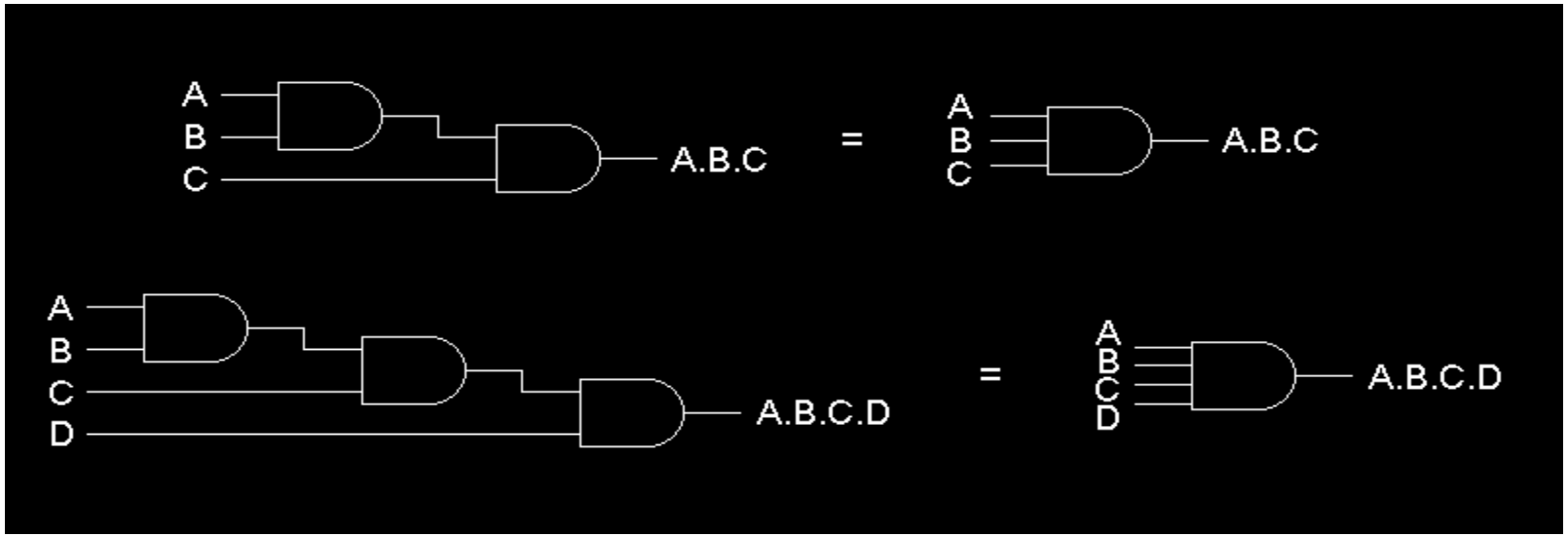
EXNOR GATE/XNOR

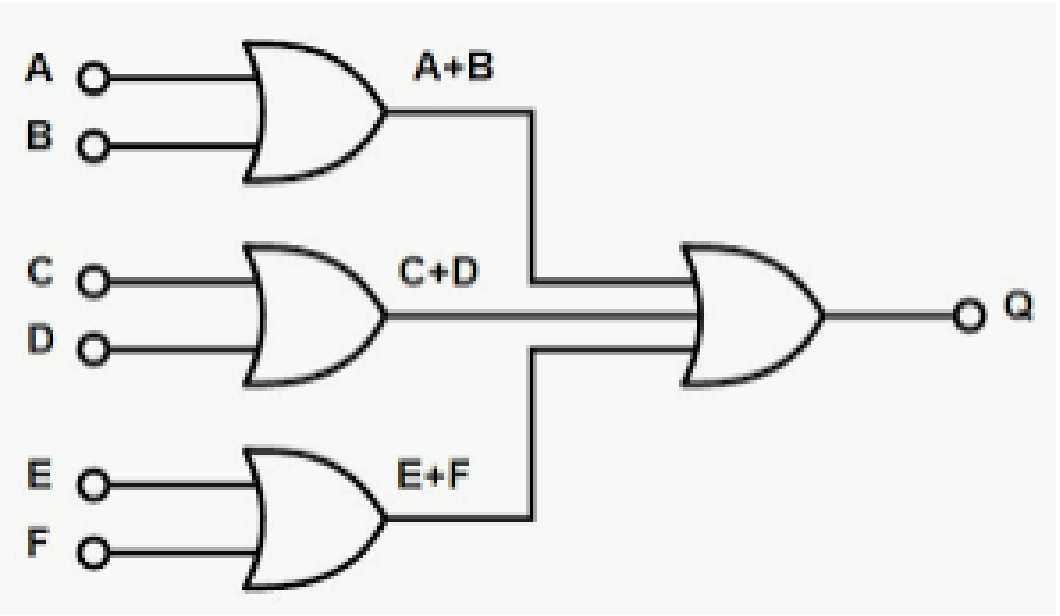


2 Input EXNOR gate		
A	B	$\overline{A⊕B}$
0	0	1
0	1	0
1	0	0
1	1	1

High output (1) for same input

Multiple Input Logic Gates





QUICK QUIZ (POLL QUESTION)

Which one of the following truth tables represents the behavior a NAND gate?

A

2 Input NAND gate		
A	B	$\overline{A.B}$
0	0	1
0	1	0
1	0	0
1	1	0

B

2 Input NAND gate		
A	B	$\overline{A.B}$
0	0	0
0	1	1
1	0	1
1	1	0

C

2 Input NAND gate		
A	B	$\overline{A.B}$
0	0	0
0	1	0
1	0	0
1	1	1

D

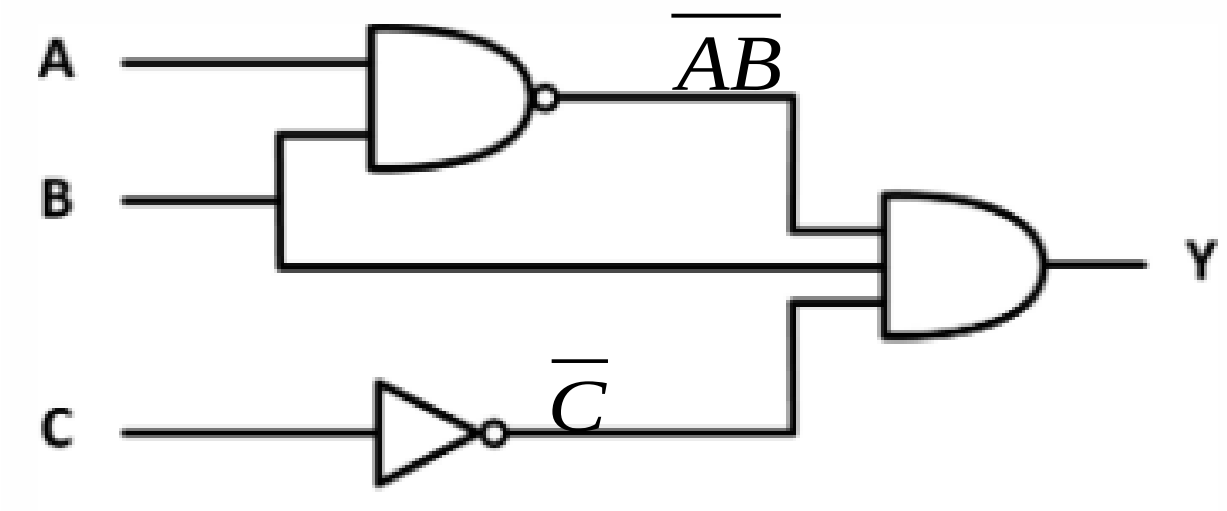
2 Input NAND gate		
A	B	$\overline{A.B}$
0	0	1
0	1	1
1	0	1
1	1	0

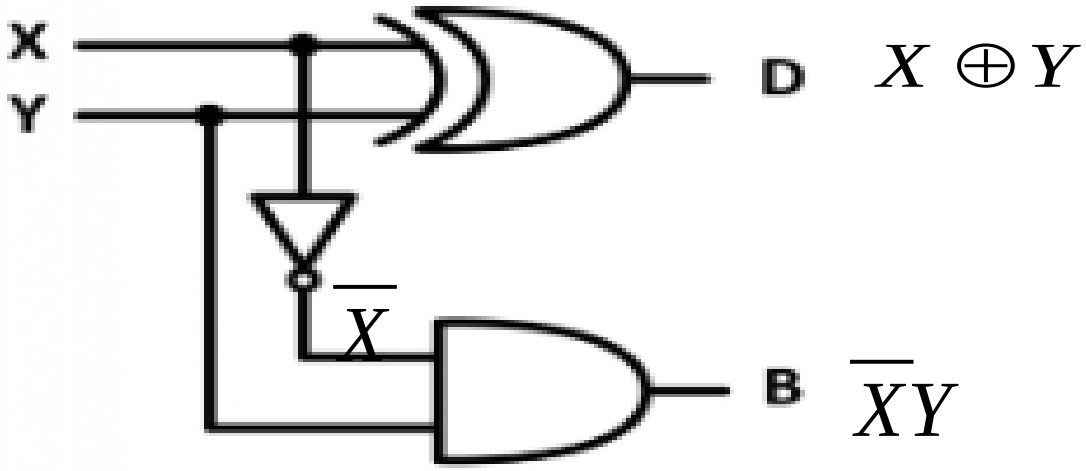
QUICK QUIZ (POLL QUESTION)

The output of an AND gate with three inputs, A, B, and C, is HIGH when _____.

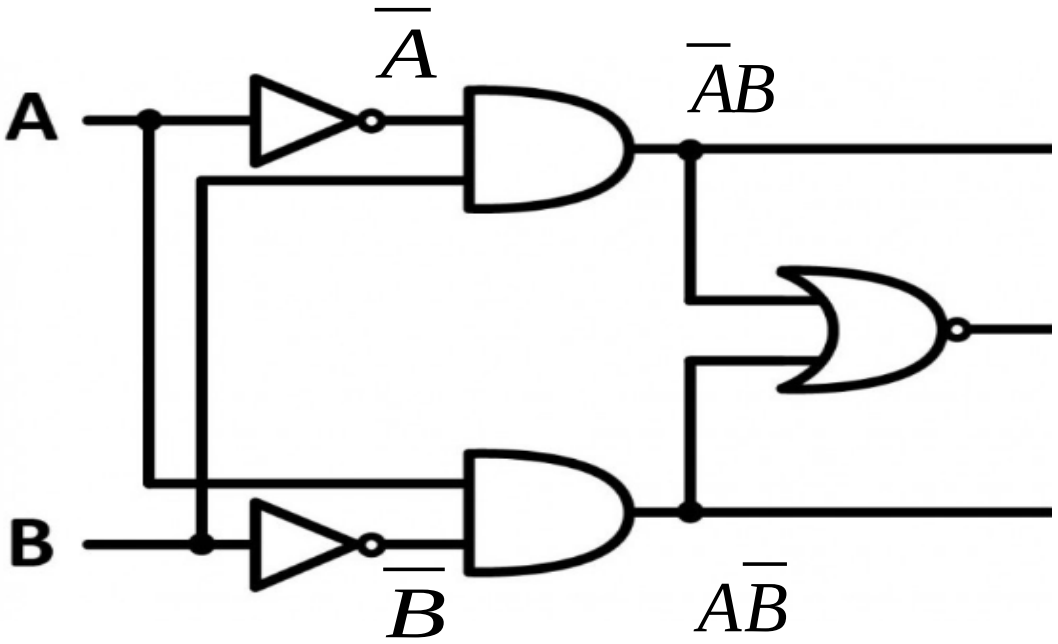
A. A = 1, B = 1, C = 0
B. A = 0, B = 0, C = 0
C. A = 1, B = 1, C = 1
D. A = 1, B = 0, C = 1

Combination of Gates

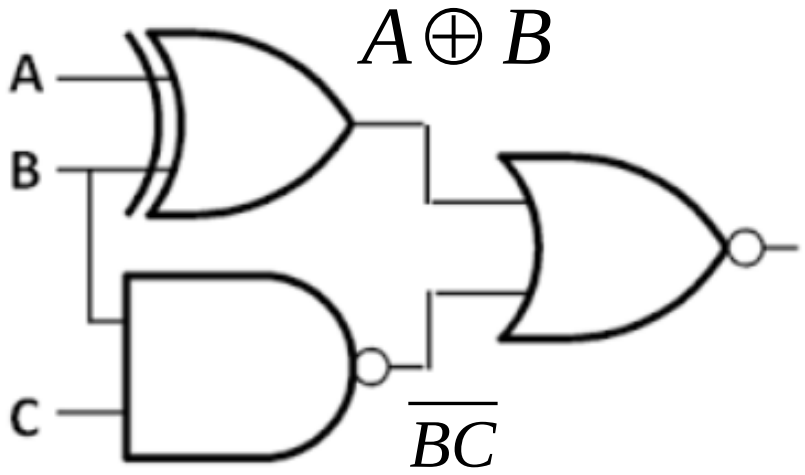




Combination of Gates

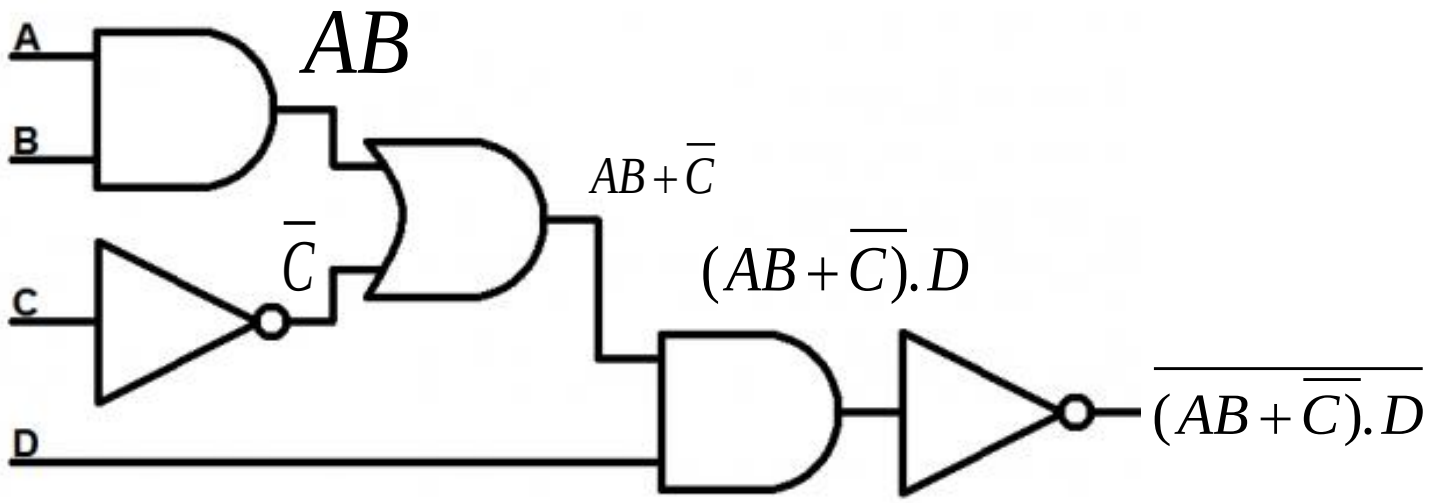


$$\overline{\overline{A}B + A\overline{B}}$$



$$(A \oplus B) + \overline{BC}$$

Combination of Gates



Draw circuit for $Y = AB + \overline{A}C$

Boolean Algebra Rules

Analyze and simplify the digital (logic) circuits

Variable used can have only two values. Binary 1 for HIGH and Binary 0 for LOW

Commutative law

(i) $A.B = B.A$ (ii) $A + B = B + A$

Associative law

(i) $(A.B).C = A.(B.C)$ (ii) $(A + B) + C = A + (B + C)$

Distributive law

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

AND law

(i) $A.0 = 0$ (ii) $A.1 = A$

(iii) $A.A = A$ (iv) $A.\overline{A} = 0$

OR law

(i) $A + 0 = A$ (ii) $A + 1 = 1$

(iii) $A + A = A$ (iv) $A + \overline{A} = 1$

INVERSION law

$\overline{\overline{A}} = A$

Boolean Algebra Rules

Basic Rules of Boolean Algebra

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

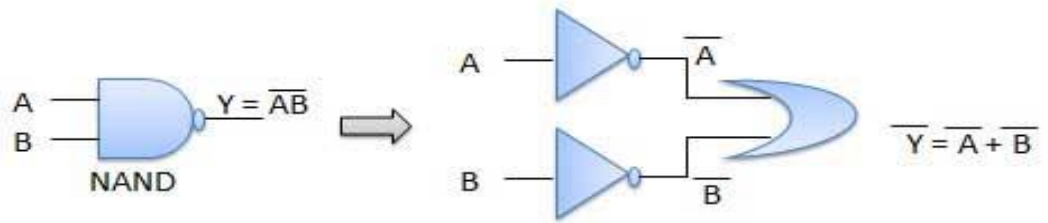
DeMorgan's Theorem

$$\overline{(AB)} = (\overline{A} + \overline{B}) \qquad \overline{(A + B)} = (\overline{A}\ \overline{B})$$

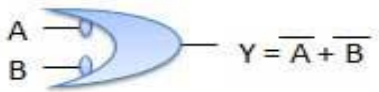
De’ Morgan Laws:

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

NAND = Bubbled OR



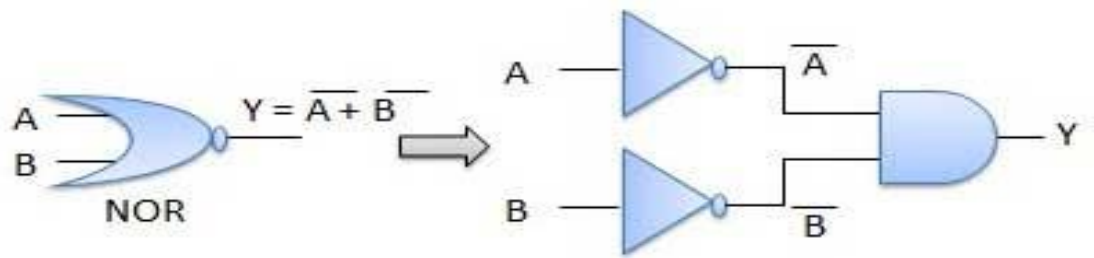
NAND ≡ Bubbled OR



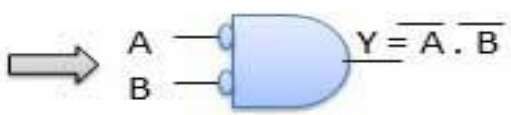
Bubbled OR

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

NOR = Bubbled AND



NOR ≡ Bubbled AND



Bubbled AND

QUICK QUIZ (POLL QUESTION)

Which of the following Boolean theorem is incorrect?

A. $A + A = A$
B. $A + 1 = 1$
C. $A + 0 = A$
D. $A + A' = 0$

Simplification using Boolean Algebra

$$Simplify \quad C + \overline{BC}$$

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

$$(C + \overline{C}) + [\overline{B}]$$

$$1 + \overline{B}$$

$$1$$

$$Simplify\ F = ABC + A + A\overline{B}C$$

$$AC(B + \overline{B}) + A$$

$$AC + A$$

$$A(C + 1)$$

$$A$$

Simplification using Boolean Algebra

Practice Question:

1. Simplify the following Boolean expression:

$$\overline{\overline{AB} + \bar{A} + AB}$$

Simplification using Boolean Algebra

Practice Question:

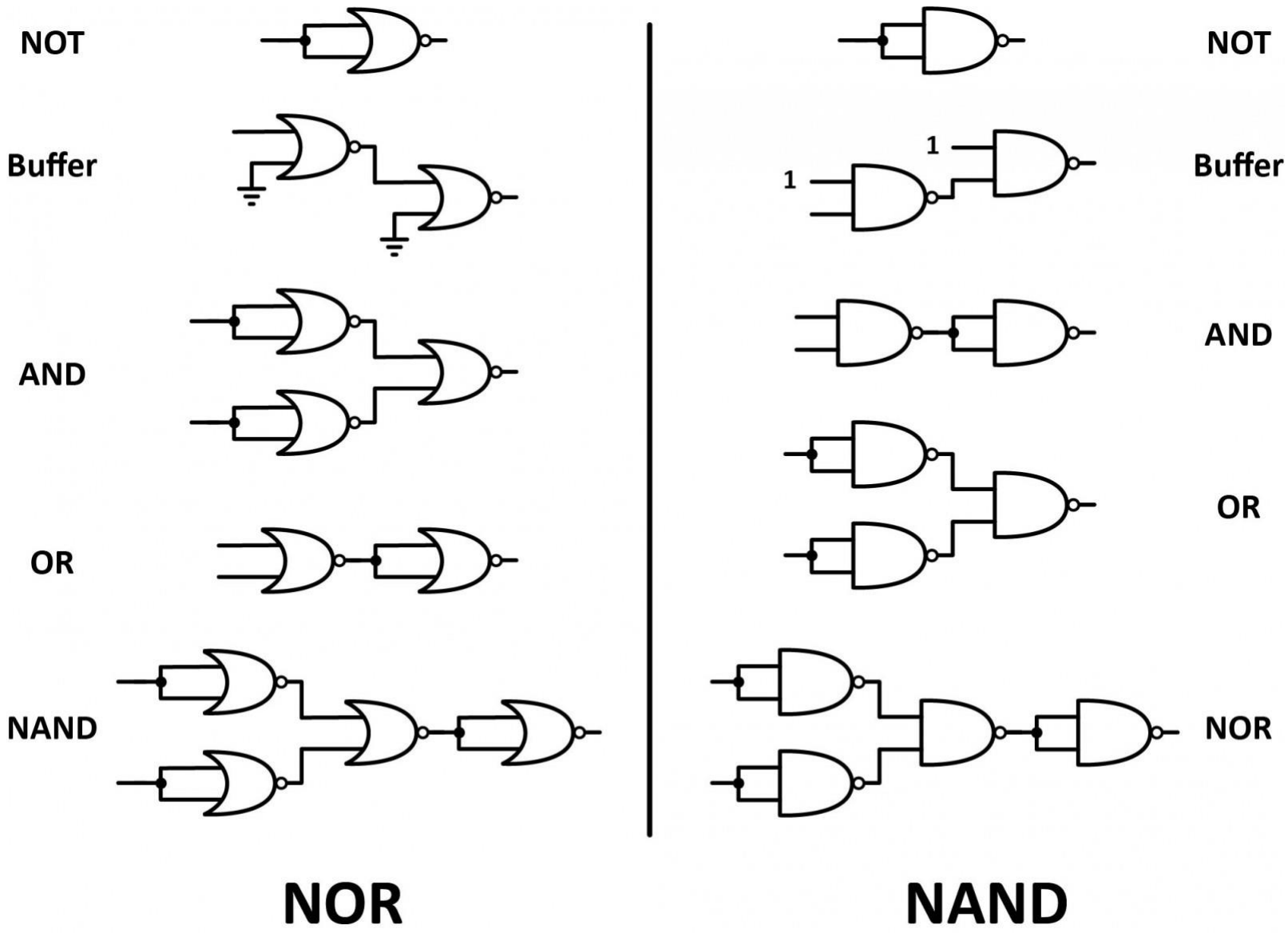
1. Simplify the following Boolean expression:

$$A\left[B + \bar{C}\left(\overline{AB + A\bar{C}}\right)\right]$$

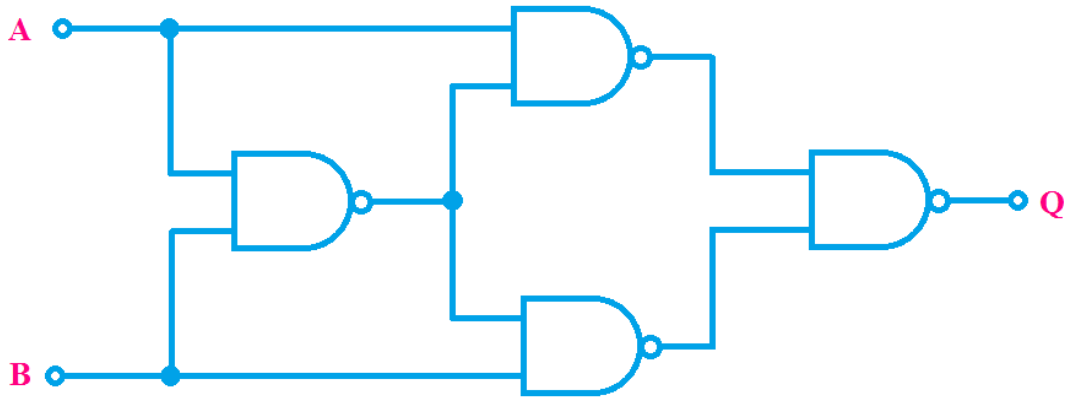
Advantages of Boolean Algebra

1. To obtain simpler expression
2. To reduce the number of gates in the circuit
3. To reduce the number of inputs in the circuit
4. To reduce complexity
5. To reduce cost.

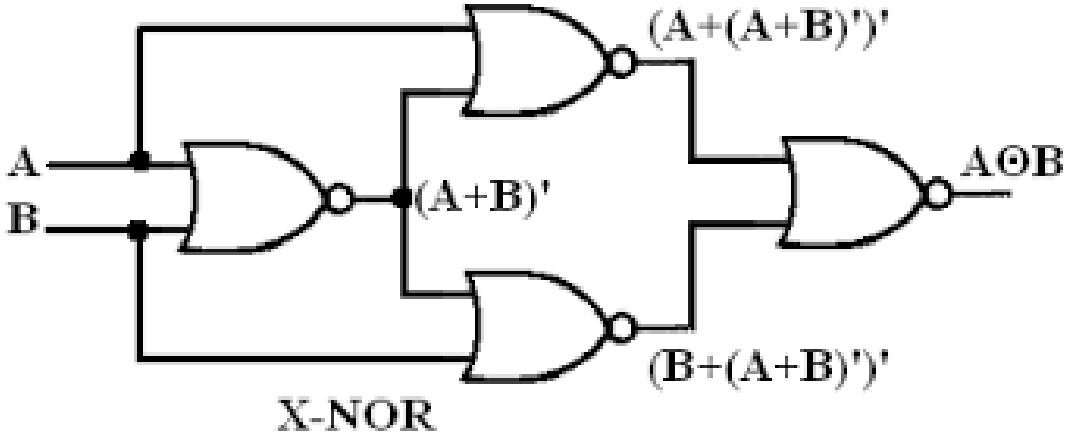
NAND and NOR as Universal Gates



NAND and NOR as Universal Gates



XOR Gate using NAND Gates



X-NOR

QUICK QUIZ (POLL QUESTION)

Minimum number of two input NAND gates required to implement XOR function is:

A. 3
B. 4
C. 5
D. 6

DIGITAL ELECTRONICS: ECE 213

Topic: Canonical and Standard Forms

UNIT II: Combinational Logic System

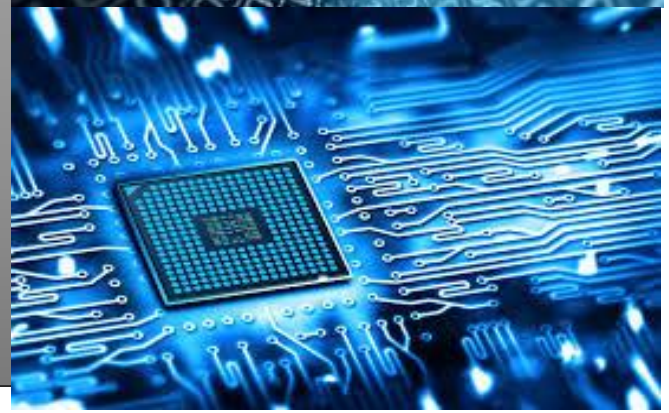
Lecture No.: 10

Prepared By: Irfan Ahmad Pindoo

Assistant Professor

VLSI Design, ECE

School of Computer Science and Engineering



Simplification using Boolean Algebra

Practice Question:

1. Simplify the following Boolean expression:

$$\overline{\overline{AB} + \bar{A} + AB}$$

Simplification using Boolean Algebra

Practice Question:

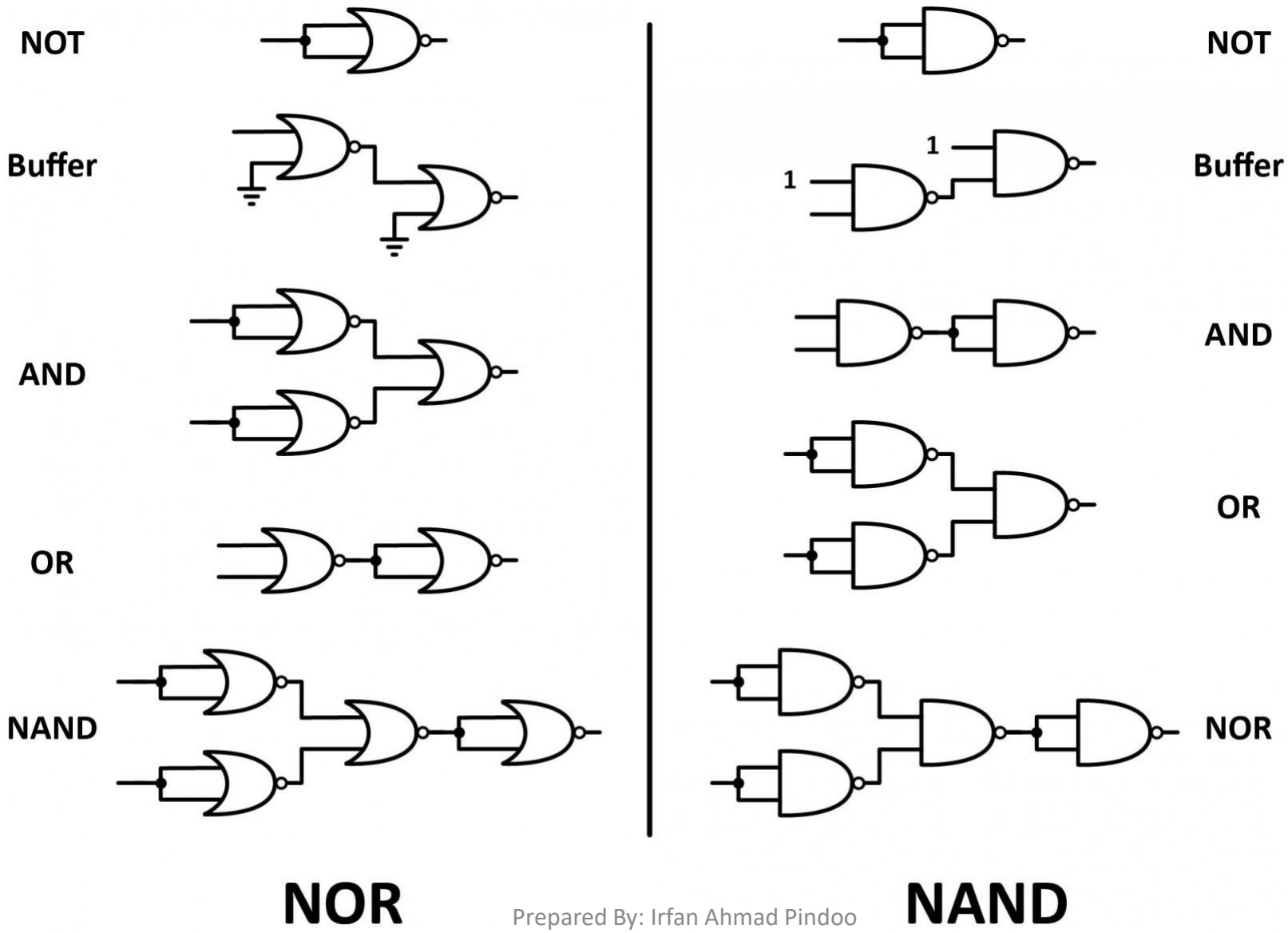
1. Simplify the following Boolean expression:

$$A\left[B + \bar{C}\left(\overline{AB + A\bar{C}}\right)\right]$$

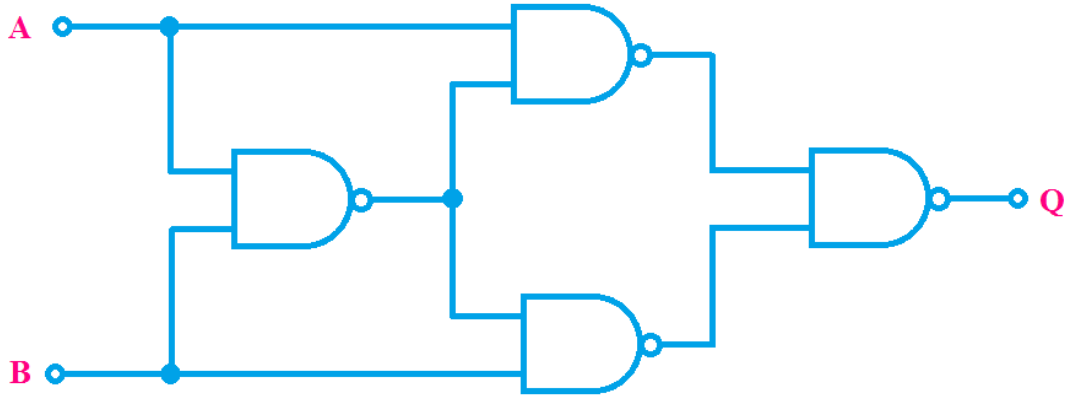
Advantages of Boolean Algebra

1. To obtain simpler expression
2. To reduce the number of gates in the circuit
3. To reduce the number of inputs in the circuit
4. To reduce complexity
5. To reduce cost.

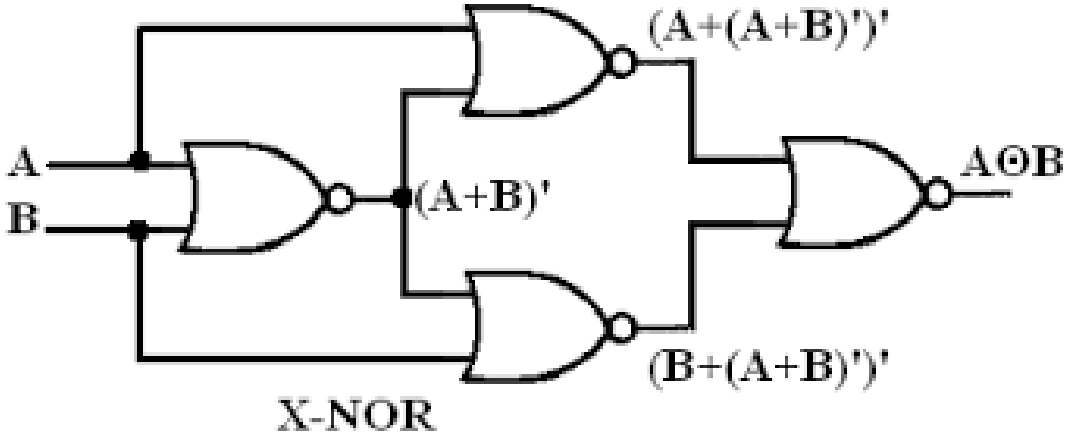
NAND and NOR as Universal Gates



NAND and NOR as Universal Gates



XOR Gate using NAND Gates



X-NOR

QUICK QUIZ (POLL QUESTION)

Minimum number of two input NAND gates required to implement XOR function is:

A. 3
B. 4
C. 5
D. 6

DUAL of the function

Basic Rules of Boolean Algebra

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

DeMorgan's Theorem

$$\overline{(AB)} = (\overline{A} + \overline{B}) \qquad \overline{(A + B)} = (\overline{A}\ \overline{B})$$

In a dual function:

1. AND operator of a given function is changed to OR operator and vice-versa.
2. A constant 1 (or true) of a given function is changed to a constant 0 (or false) and vice-versa.
3. Don’t change literals (No complements)

DUAL of the function

RULES:

1. Interchange AND and OR logic operators
2. Interchange 1's and 0's.
3. Interchange literals as well.

Example:

$$If\ f = \bar{x}y\bar{z} + \bar{x}\bar{y}z$$

Then, it dual is $f = (\bar{x} + y + \bar{z}).(\bar{x} + \bar{y} + z)$

Complements of the function

The complement of a function f is f’ an is obtained by interchanging 0’s and 1’s in the value of f.

RULES:

1. Interchange AND and OR logic operators
2. Interchange 1’s and 0’s.
3. Interchange literals as well.

Example:

$$If\ f = \bar{x}y\bar{z} + \bar{x}\bar{y}z$$

Then, it dual is $f = (\bar{x} + y + \bar{z}).(\bar{x} + \bar{y} + z)$

And its complement $\bar{f} = (x + \bar{y} + z).(x + y + \bar{z})$

Simplification using Boolean Algebra

Practice Question:

Find the dual and the complement of the function:

$$f = x(\bar{y}\bar{z} + yz)$$

Simplification using Boolean Algebra

Practice Question:

Find the dual and the complement of the function:

$$f = (x + y)'.(\bar{x} + \bar{y})$$

Canonical and Standard Forms

Canonical Form

- Boolean expression in which each term contains all the literals in complemented or un-complemented form.
- Example:

$$f = A'B + AB'$$

$$f = A'B'C + A'BC' + AB'C' + ABC$$

QUICK QUIZ (POLL QUESTION)

A variable on its own or in its complemented form is known as a _____

a) Product Term
b) Literal
c) Sum Term
d) Word

QUICK QUIZ (POLL QUESTION)

There are _____ Minterms for 3 variables (a, b, c).

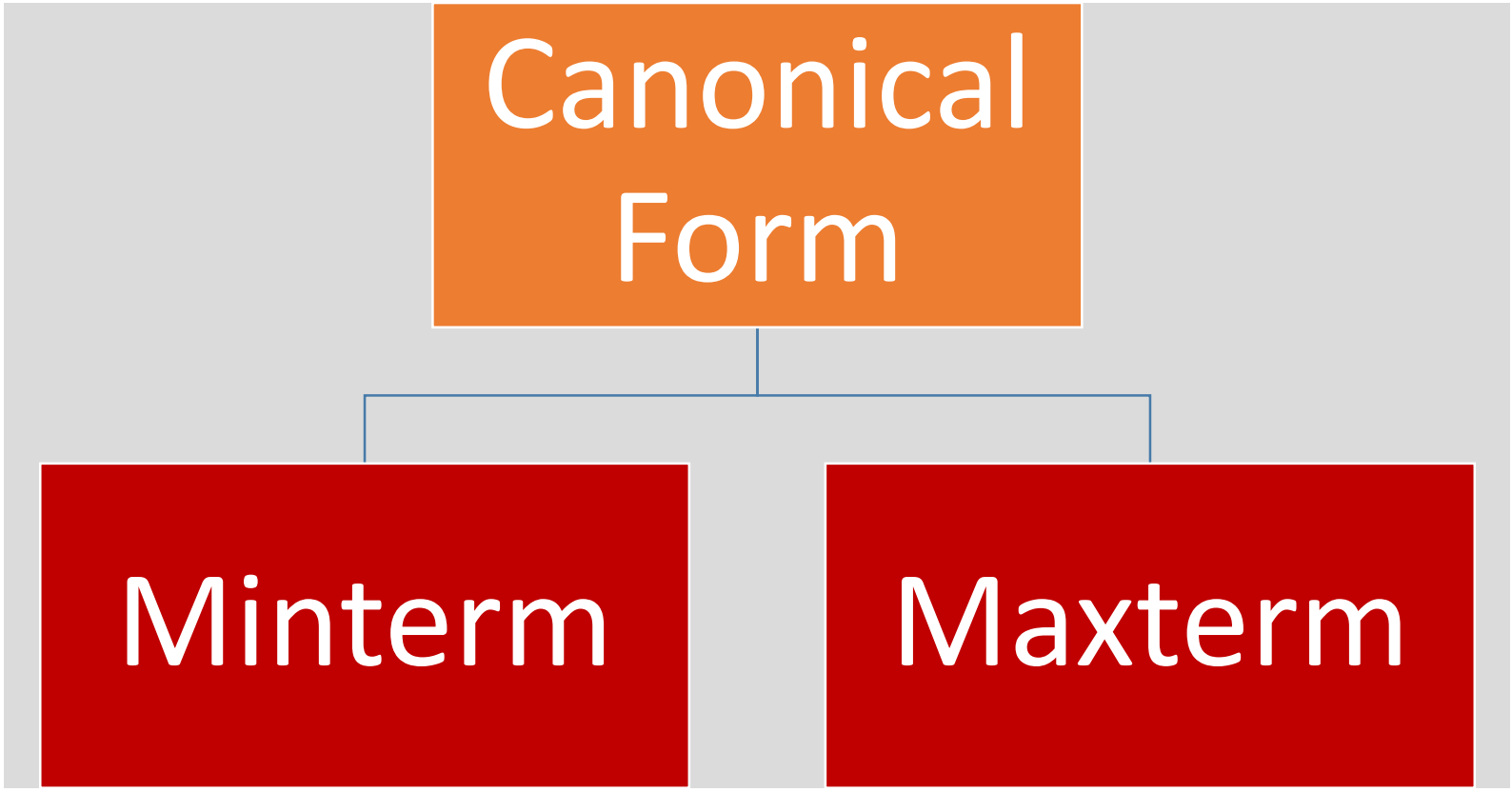
a) 0

b) 2

c) 8

d) 1

Canonical and Standard Forms



Canonical and Standard Forms

Canonical Form

- A binary variable may appear either in its normal form (x) or in its complement form (x').
- Now consider two binary variables x and y combined with an AND operation. Since each variable may appear in either form, there are four possible combinations: $x'y', x'y, xy', and\ xy$.
- Each of these four AND terms is called a *minterm*, or a *standard product*.
- Each minterm is obtained from an AND term of the n variables, with each variable being primed if the corresponding bit of the binary number is a 0 and unprimed if a 1.
- In a similar manner, n variables can be combined to form 2^n minterms.

Canonical and Standard Forms

Canonical Form

- In a similar fashion, n variables forming an OR term, with each variable being primed or unprimed, provide 2^n possible combinations, called maxterms, or standard sums.
- Any 2^n maxterms for n variables may be determined similar to minterms.
- It is important to note that

1. Each maxterm is obtained from an OR term of the n variables, with each variable being unprimed if the corresponding bit is a 0 and primed if a 1, and
2. Each maxterm is the complement of its corresponding minterm and **vice versa**

Canonical and Standard Forms

Minterms and Maxterms for Three Binary Variables

			Minterms		Maxterms	
x	y	z	Term	Designation	Term	Designation
0	0	0	$x'y'z'$	m_0	$x + y + z$	M_0
0	0	1	$x'y'z$	m_1	$x + y + z'$	M_1
0	1	0	$x'yz'$	m_2	$x + y' + z$	M_2
0	1	1	$x'yz$	m_3	$x + y' + z'$	M_3
1	0	0	$xy'z'$	m_4	$x' + y + z$	M_4
1	0	1	$xy'z$	m_5	$x' + y + z'$	M_5
1	1	0	xyz'	m_6	$x' + y' + z$	M_6
1	1	1	xyz	m_7	$x' + y' + z'$	M_7

Canonical and Standard Forms

Example:

Functions of Three Variables

x	y	z	Function f_1	Function f_2
0	0	0	0	0
0	0	1	1	0
0	1	0	0	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$$f_1 = x'y'z + xy'z' + xyz = m_1 + m_4 + m_7$$

$$f_2 = x'yz + xy'z + xyz' + xyz = m_3 + m_5 + m_6 + m_7$$

A Boolean function can be expressed algebraically from a given truth table by forming a minterm for each combination of the variables that produces a 1 in the function and then taking the OR of all those terms.

Canonical and Standard Forms

Example:

Functions of Three Variables

x	y	z	Function f_1	Function f_2
0	0	0	0	0
0	0	1	1	0
0	1	0	0	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$$f_1' = x'y'z' + x'yz' + x'yz + xy'z + xyz'$$

If we take the complement of f_1', we obtain the function f_1:

$$\begin{aligned} f_1 &= (x + y + z)(x + y' + z)(x' + y + z')(x' + y' + z) \\ &= M_0 \cdot M_2 \cdot M_3 \cdot M_5 \cdot M_6 \end{aligned}$$

Canonical and Standard Forms

Example:

Functions of Three Variables

x	y	z	Function f_1	Function f_2
0	0	0	0	0
0	0	1	1	0
0	1	0	0	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$$f_1' = x'y'z' + x'yz' + x'yz + xy'z + xyz'$$

If we take the complement of f_1', we obtain the function f_1:

$$\begin{aligned} f_1 &= (x + y + z)(x + y' + z)(x' + y + z')(x' + y' + z) \\ &= M_0 \cdot M_2 \cdot M_3 \cdot M_5 \cdot M_6 \end{aligned}$$

Similarly, it is possible to read the expression for f_2 from the table:

$$\begin{aligned} f_2 &= (x + y + z)(x + y + z')(x + y' + z)(x' + y + z) \\ &= M_0 M_1 M_2 M_4 \end{aligned}$$

These examples demonstrate a second property of Boolean algebra:
Any Boolean function can be expressed as a product of maxterms (with "product" meaning the ANDing of terms).

Canonical and Standard Forms

Example:

Truth Table for F = xy + x′z

x	y	z	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

$$f_1' = x'y'z' + x'yz' + x'yz + xy'z + xyz'$$

If we take the complement of f_1', we obtain the function f_1:

$$\begin{aligned} f_1 &= (x + y + z)(x + y' + z)(x' + y + z')(x' + y' + z) \\ &= M_0 \cdot M_2 \cdot M_3 \cdot M_5 \cdot M_6 \end{aligned}$$

Similarly, it is possible to read the expression for f_2 from the table:

$$\begin{aligned} f_2 &= (x + y + z)(x + y + z')(x + y' + z)(x' + y + z) \\ &= M_0 M_1 M_2 M_4 \end{aligned}$$

These examples demonstrate a second property of Boolean algebra:
Any Boolean function can be expressed as a product of maxterms (with "product" meaning the ANDing of terms).

Canonical and Standard Forms

Example:

- Express the Boolean function $f = A + B'C$ as a *sum of minterms*?

Canonical and Standard Forms

Example:

- Express the Boolean function $f = xy + x'z$ as a *product of maxterms*?

QUICK QUIZ (POLL QUESTION)

The canonical sum of product form of the function y(A,B) = A + B is _____

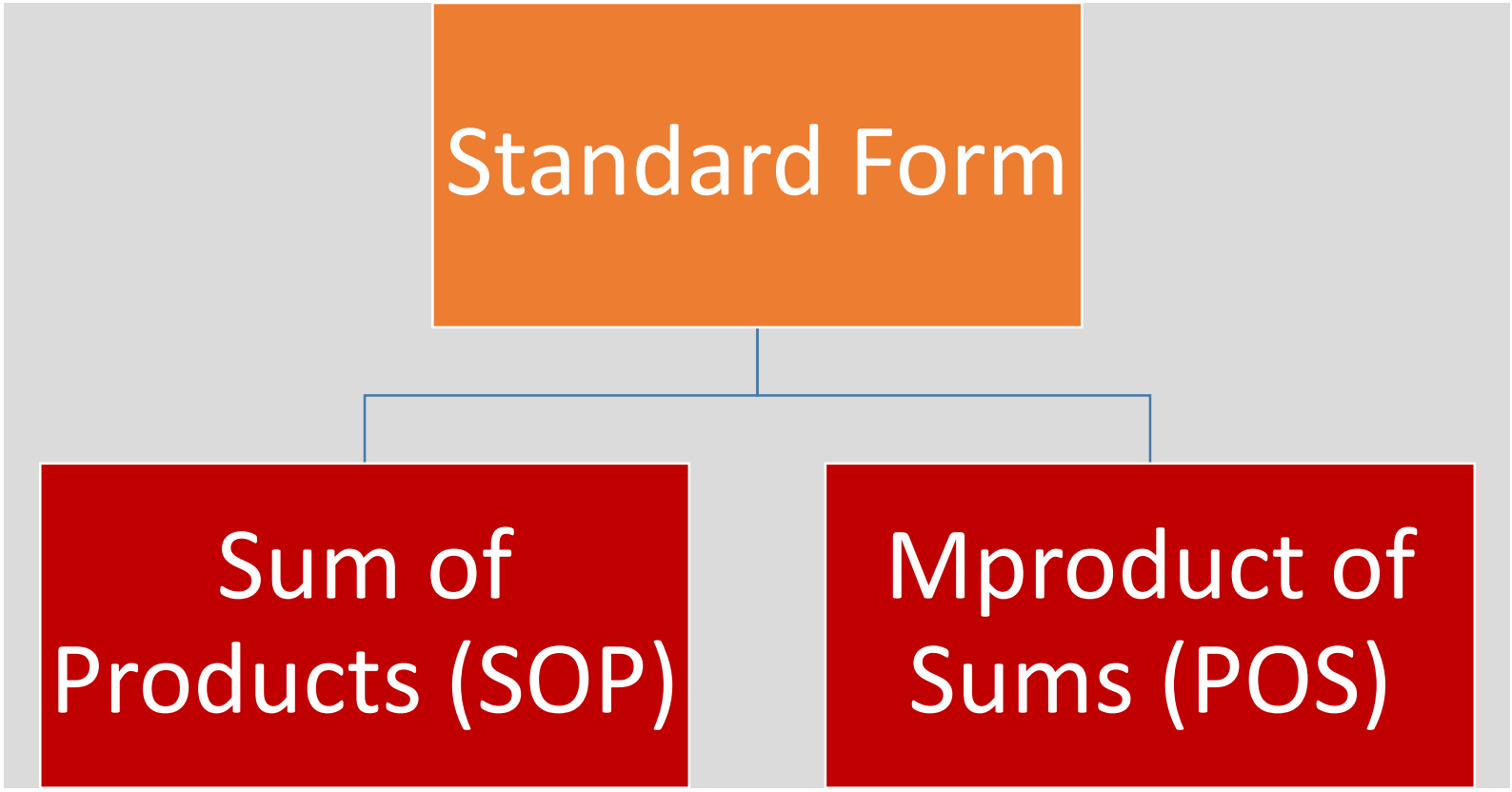
a) AB + BB + A'A

b) AB + AB' + A'B

c) BA + BA' + A'B'

d) AB' + A'B + A'B'

Canonical and Standard Forms



Canonical and Standard Forms

Standard Form

- Another way to express Boolean functions is in standard form.
- In this configuration, the terms that form the function may contain one, two, or any number of literals.
- There are two types of standard forms:

1. sum of products
2. products of sums

Canonical and Standard Forms

Standard Form

- The **sum of products** is a Boolean expression containing AND terms, called product terms, with one or more literals each. The sum denotes the ORing of these terms. An example of a function expressed as a sum of products is:

$$F_1 = y' + xy + x'yz'$$

- The expression has thr :h one, two, and three literals. Their sum is, in effect, an OR operation
- The logic diagram of a sum-of-products expression consists of a *group of AND gates followed by a single OR gate.*

Canonical and Standard Forms

Standard Form

- A product of sums is a Boolean expression containing OR terms, called sum terms.
- Each term may have any number of literals. The product denotes the ANDing of these terms. An example of a function expressed as a product of sums is:

$$F_2 = x(y' + z)(x' + y + z')$$

- This expression has three sum terms, with one, two, and three literals. The product is an AND operation.
- The gate structure of the product-of-sums expression consists of a *group of OR gates for the sum terms followed by an AND gate*

QUICK QUIZ (POLL QUESTION)

The logical sum of two or more logical product terms is called _____

a) SOP

b) POS

c) OR operation

d) NAND operation

QUICK QUIZ (POLL QUESTION)

The expression Y=AB+BC+AC shows the _____ operation.

a) EX-OR

b) SOP

c) POS

d) NOR

QUICK QUIZ (POLL QUESTION)

The expression Y=(A+B)(B+C)(C+A) shows the _____ operation.

a) AND
b) POS
c) SOP
d) NAND

DIGITAL ELECTRONICS: ECE 213

Topic: Canonical and Standard Forms

UNIT II: Combinational Logic System

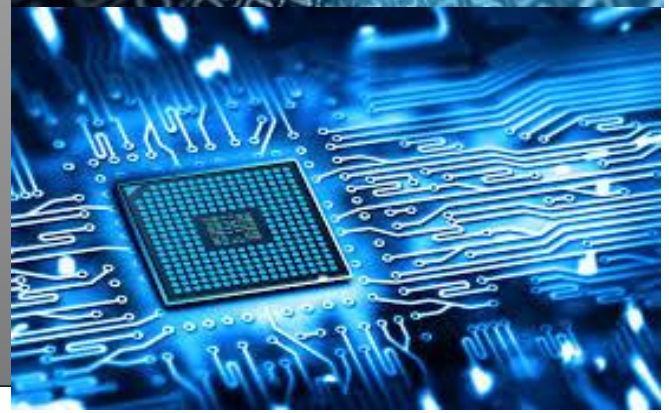
Lecture No.: 11

Prepared By: Irfan Ahmad Pindoo

Assistant Professor

VLSI Design, ECE

School of Computer Science and Engineering



Canonical and Standard Forms

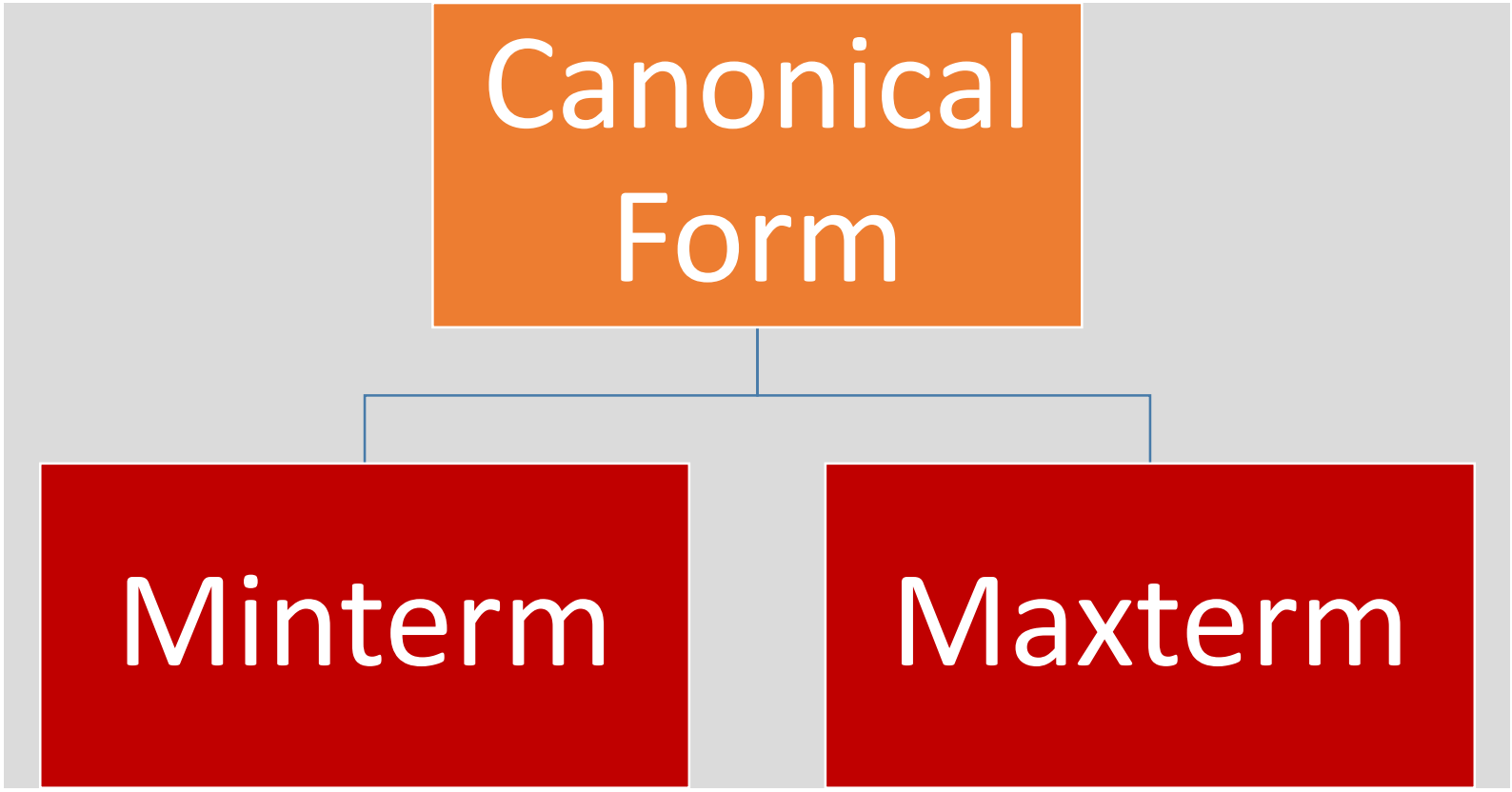
Canonical Form

- Boolean expression in which each term contains all the literals in complemented or un-complemented form.
- Example:

$$f = A'B + AB'$$

$$f = A'B'C + A'BC' + AB'C' + ABC$$

Canonical and Standard Forms



Canonical and Standard Forms

Canonical Form

- A binary variable may appear either in its normal form (x) or in its complement form (x').
- Now consider two binary variables x and y combined with an AND operation. Since each variable may appear in either form, there are four possible combinations: $x'y', x'y, xy', and\ xy$.
- Each of these four AND terms is called a *minterm*, or a *standard product*.
- Each minterm is obtained from an AND term of the n variables, with each variable being primed if the corresponding bit of the binary number is a 0 and unprimed if a 1.
- In a similar manner, n variables can be combined to form 2^n minterms.

Canonical and Standard Forms

Canonical Form

- In a similar fashion, n variables forming an OR term, with each variable being primed or unprimed, provide 2^n possible combinations, called maxterms, or standard sums.
- Any 2^n maxterms for n variables may be determined similar to minterms.
- It is important to note that

1. Each maxterm is obtained from an OR term of the n variables, with each variable being unprimed if the corresponding bit is a 0 and primed if a 1, and
2. Each maxterm is the complement of its corresponding minterm and **vice versa**

Canonical and Standard Forms

Minterms and Maxterms for Three Binary Variables

			Minterms		Maxterms	
x	y	z	Term	Designation	Term	Designation
0	0	0	$x'y'z'$	m_0	$x + y + z$	M_0
0	0	1	$x'y'z$	m_1	$x + y + z'$	M_1
0	1	0	$x'yz'$	m_2	$x + y' + z$	M_2
0	1	1	$x'yz$	m_3	$x + y' + z'$	M_3
1	0	0	$xy'z'$	m_4	$x' + y + z$	M_4
1	0	1	$xy'z$	m_5	$x' + y + z'$	M_5
1	1	0	xyz'	m_6	$x' + y' + z$	M_6
1	1	1	xyz	m_7	$x' + y' + z'$	M_7

Canonical and Standard Forms

Example:

Functions of Three Variables

x	y	z	Function f_1	Function f_2
0	0	0	0	0
0	0	1	1	0
0	1	0	0	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$$f_1 = x'y'z + xy'z' + xyz = m_1 + m_4 + m_7$$

$$f_2 = x'yz + xy'z + xyz' + xyz = m_3 + m_5 + m_6 + m_7$$

A Boolean function can be expressed algebraically from a given truth table by forming a minterm for each combination of the variables that produces a 1 in the function and then taking the OR of all those terms.

Canonical and Standard Forms

Example:

Functions of Three Variables

x	y	z	Function f_1	Function f_2
0	0	0	0	0
0	0	1	1	0
0	1	0	0	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$$f_1' = x'y'z' + x'yz' + x'yz + xy'z + xyz'$$

If we take the complement of f_1', we obtain the function f_1:

$$\begin{aligned} f_1 &= (x + y + z)(x + y' + z)(x' + y + z')(x' + y' + z) \\ &= M_0 \cdot M_2 \cdot M_3 \cdot M_5 \cdot M_6 \end{aligned}$$

Canonical and Standard Forms

Example:

Functions of Three Variables

x	y	z	Function f_1	Function f_2
0	0	0	0	0
0	0	1	1	0
0	1	0	0	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$$f_1' = x'y'z' + x'yz' + x'yz + xy'z + xyz'$$

If we take the complement of f_1', we obtain the function f_1:

$$\begin{aligned} f_1 &= (x + y + z)(x + y' + z)(x' + y + z')(x' + y' + z) \\ &= M_0 \cdot M_2 \cdot M_3 \cdot M_5 \cdot M_6 \end{aligned}$$

Similarly, it is possible to read the expression for f_2 from the table:

$$\begin{aligned} f_2 &= (x + y + z)(x + y + z')(x + y' + z)(x' + y + z) \\ &= M_0 M_1 M_2 M_4 \end{aligned}$$

These examples demonstrate a second property of Boolean algebra:
Any Boolean function can be expressed as a product of maxterms (with "product" meaning the ANDing of terms).

Canonical and Standard Forms

Example:

Truth Table for F = xy + x′z

x	y	z	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

$$f_1' = x'y'z' + x'yz' + x'yz + xy'z + xyz'$$

If we take the complement of f_1', we obtain the function f_1:

$$\begin{aligned} f_1 &= (x + y + z)(x + y' + z)(x' + y + z')(x' + y' + z) \\ &= M_0 \cdot M_2 \cdot M_3 \cdot M_5 \cdot M_6 \end{aligned}$$

Similarly, it is possible to read the expression for f_2 from the table:

$$\begin{aligned} f_2 &= (x + y + z)(x + y + z')(x + y' + z)(x' + y + z) \\ &= M_0 M_1 M_2 M_4 \end{aligned}$$

These examples demonstrate a second property of Boolean algebra:
Any Boolean function can be expressed as a product of maxterms (with "product" meaning the ANDing of terms).

Canonical and Standard Forms

Example:

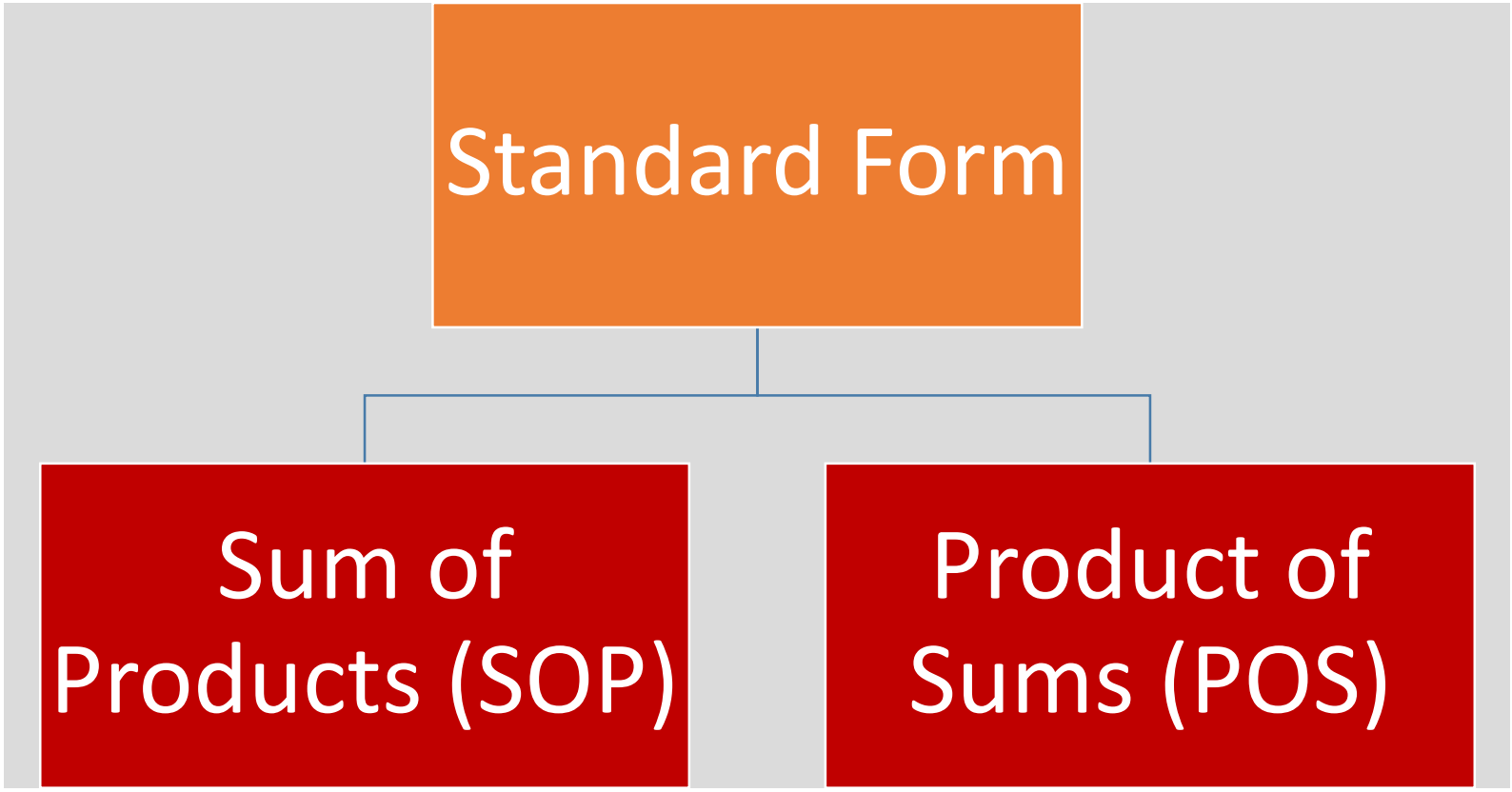
- Express the Boolean function $f = A + B'C$ as a *sum of minterms*?

Canonical and Standard Forms

Example:

- Express the Boolean function $f = xy + x'z$ as a *product of maxterms*?

Canonical and Standard Forms



Canonical and Standard Forms

Standard Form

- Another way to express Boolean functions is in standard form.
- In this configuration, the terms that form the function may contain one, two, or any number of literals.
- There are two types of standard forms:

1. sum of products
2. products of sums

Canonical and Standard Forms

Standard Form

- The **sum of products** is a Boolean expression containing AND terms, called product terms, with one or more literals each. The sum denotes the ORing of these terms. An example of a function expressed as a sum of products is:

$$F_1 = y' + xy + x'yz'$$

- The expression has thr :h one, two, and three literals. Their sum is, in effect, an OR operation
- The logic diagram of a sum-of-products expression consists of a *group of AND gates followed by a single OR gate.*

Canonical and Standard Forms

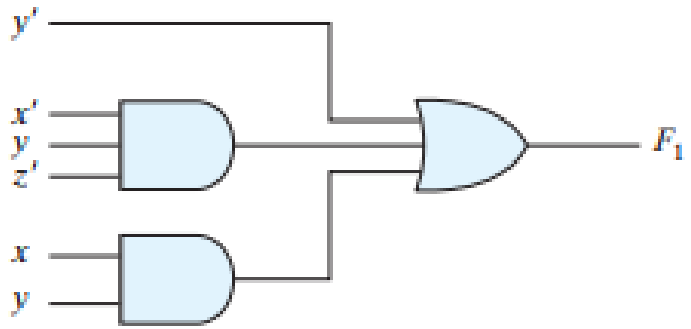
Standard Form

- A **product of sums** is a Boolean expression containing OR terms, called sum terms.
- Each term may have any number of literals. The product denotes the ANDing of these terms. An example of a function expressed as a product of sums is:

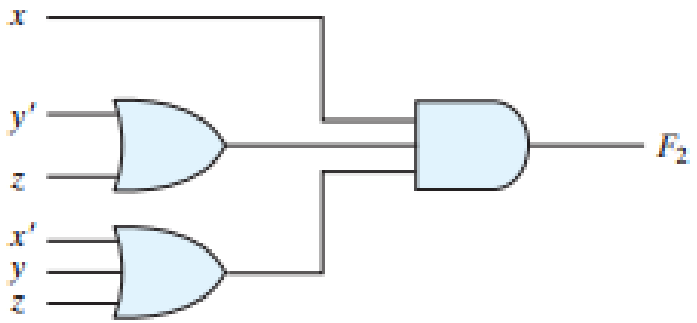
$$F_2 = x(y' + z)(x' + y + z')$$

- This expression has three sum terms, with one, two, and three literals. The product is an AND operation.
- The gate structure of the product-of-sums expression consists of a *group of OR gates for the sum terms followed by an AND gate*

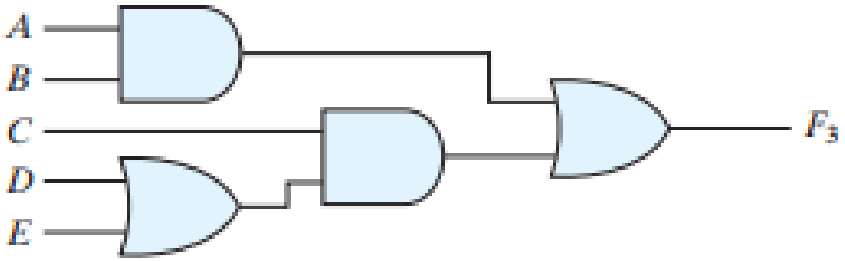
SOP and POS: Implementation Examples



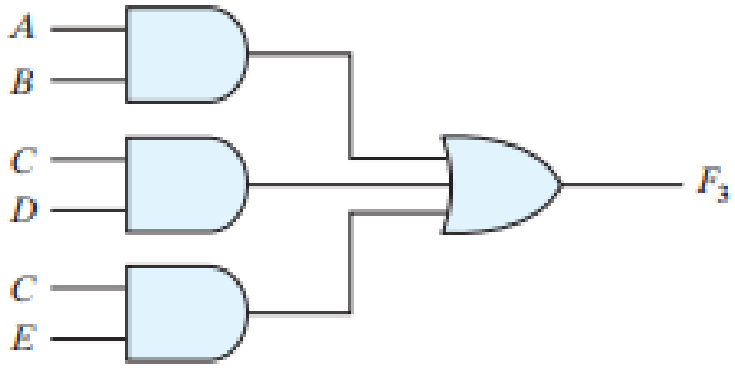
(a) Sum of Products



(b) Product of Sums



(a) $AB + C(D + E)$



(b) $AB + CD + CE$

Limitations of Boolean Algebra Rules

- This procedure of minimization is awkward because it lacks specific rules to predict each succeeding step in the manipulative process.
- Effectiveness entirely depends on the user.
- As the number of input variables increase, its complexity becomes high.
- NO unique solution.

DIGITAL ELECTRONICS: ECE 213

Topic: Karnaugh Maps (Upto 4 Variables)

UNIT II: Combinational Logic System

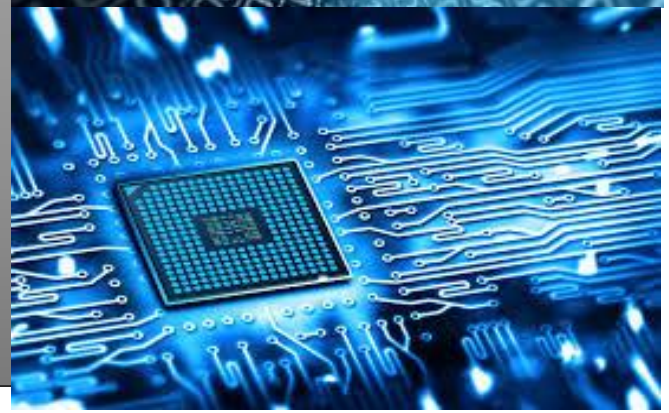
Lecture No.: 12

Prepared By: Irfan Ahmad Pindoo

Assistant Professor

VLSI Design, ECE

School of Computer Science and Engineering



Karnaugh Map Method

- The map method presented here provides a simple, straightforward rule based procedure for minimizing Boolean functions.
- This method may be regarded as a **pictorial** form of a truth table.
- The map method is also known as the **Karnaugh map or K-map.**

Karnaugh Map RULES:

1. Construct K-map and place 1’s or 0’s in the squares according to the truth table.
2. Diagonal groups are not allowed.
3. The number of 1’s or 0’s in a group must be a power of 2
4. The groups must be made as large as possible.
5. Groups can overlap and wrap around the sides of the K-map.

QUIZ QUIZ (POLL)

An n variable K-map can have:

a) n^2

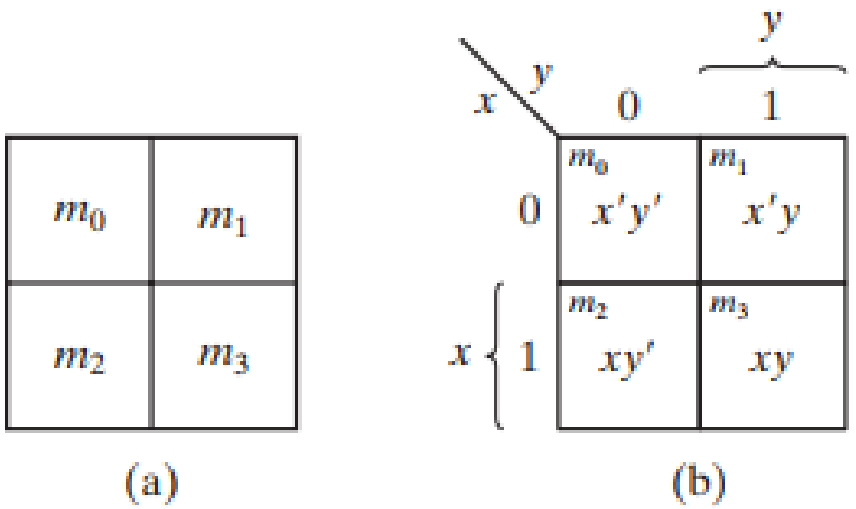
b) n^2-1

c) 2^n

d) 2^n-1

Karnaugh Map Method

- The map is made up of squares, with each square representing one minterm of the function that is to be minimized.
- The simplified expressions produced by the map are always in one of the two standard forms: **sum of products** or **product of sums**.
- This expression produces a circuit diagram with a **minimum number of gates** and the **minimum number of inputs to each gate.**

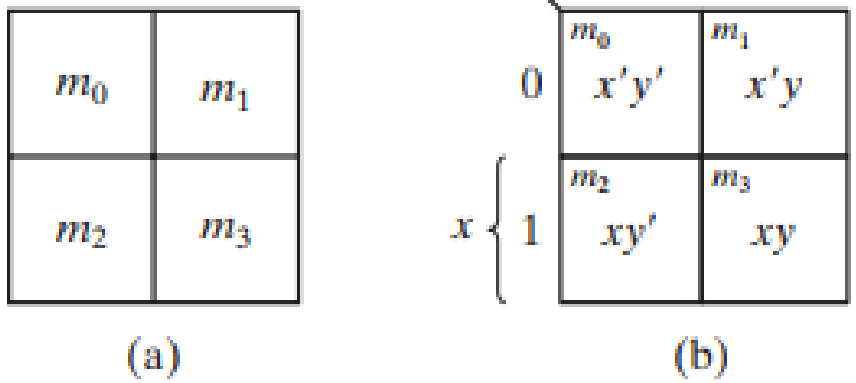


Two Variable Karnaugh Map

- There are four minterms for two variables; hence, the map consists of four squares, one for each minterm.

Example:

$$m_1 + m_2 + m_3 = x'y + xy' + xy = x + y$$

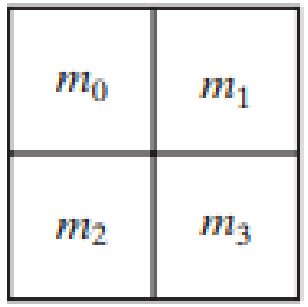


Two Variable Karnaugh Map

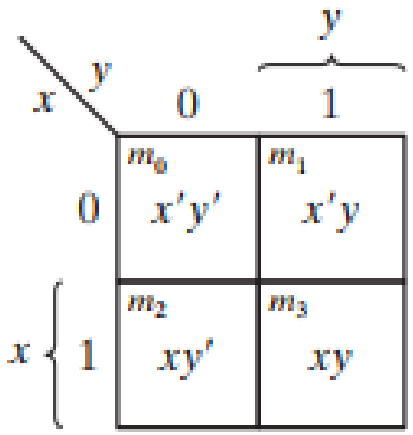
- There are four minterms for two variables; hence, the map consists of four squares, one for each minterm.

Example:

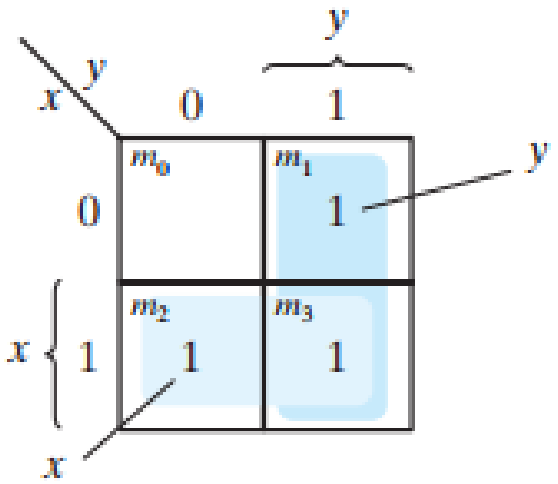
$$m_1 + m_2 + m_3 = x'y + xy' + xy = x + y$$



(a)



(b)



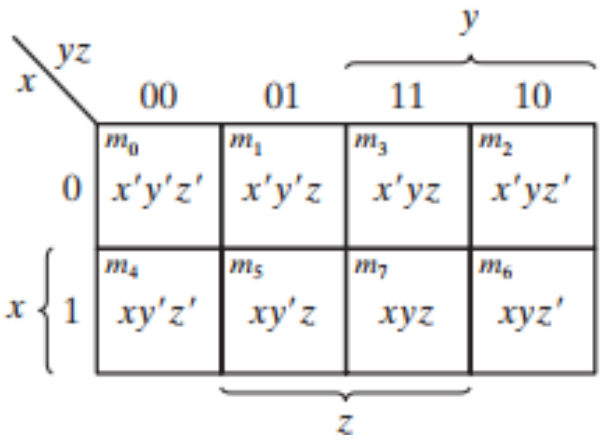
Three Variable Karnaugh Map

- There are eight minterms for three binary variables; therefore, the map consists of eight squares.

NOTE:

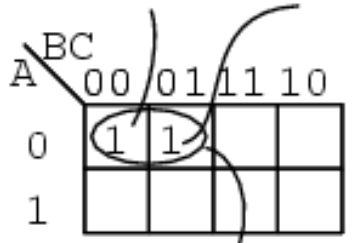
- The minterms are arranged, not in a binary sequence, but in a sequence **similar to the Gray code**.
- The characteristic of this sequence is that **only one bit changes in value from one adjacent column to the next.**

m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6

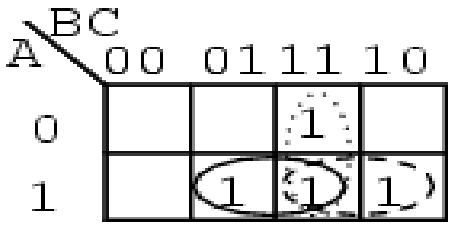


Three Variable Karnaugh Map

Out= $\overline{A}\,\overline{B}\,\overline{C} + \overline{A}\,\overline{B}C$

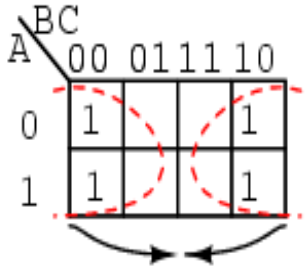


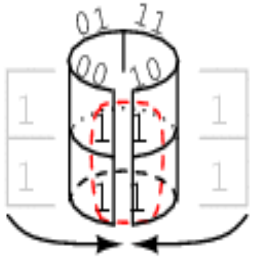
Out= $\overline{A}\,\overline{B}$



Output = AB + BC + AC

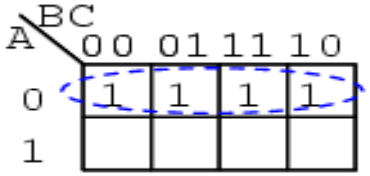
Out= $\overline{A}\,\overline{B}\,\overline{C} + A\,\overline{B}\,\overline{C} + \overline{A}\,B\,\overline{C} + A\,B\,\overline{C}$





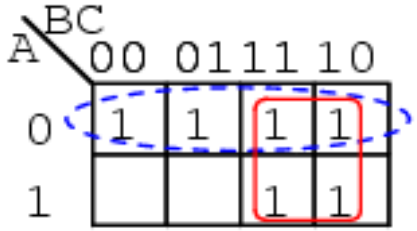
Out= $\overline{C}$

Out= $\overline{A}\,\overline{B}\,\overline{C} + \overline{A}\,\overline{B}C + \overline{A}\,B\,C + \overline{A}\,B\,\overline{C}$



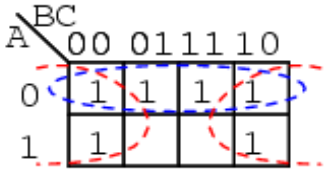
Out= $\overline{A}$

Out= $\overline{A}\,\overline{B}\,\overline{C} + \overline{A}\,\overline{B}C + \overline{A}\,B\,C + \overline{A}\,B\,\overline{C} + ABC + A\,B\,\overline{C}$



Out= $\overline{A} + B$

Out= $\overline{A}\,\overline{B}\,\overline{C} + \overline{A}\,\overline{B}C + \overline{A}\,B\,C + \overline{A}\,B\,\overline{C} + A\,\overline{B}\,\overline{C} + A\,B\,\overline{C}$



Out= $\overline{A} + \overline{C}$

DIGITAL ELECTRONICS: ECE 213

Topic: Karnaugh Maps (Upto 4 Variables)

UNIT II: Combinational Logic System

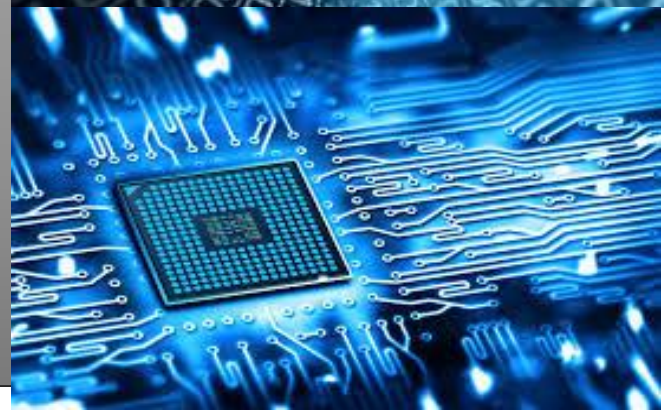
Lecture No.: 13

Prepared By: Irfan Ahmad Pindoo

Assistant Professor

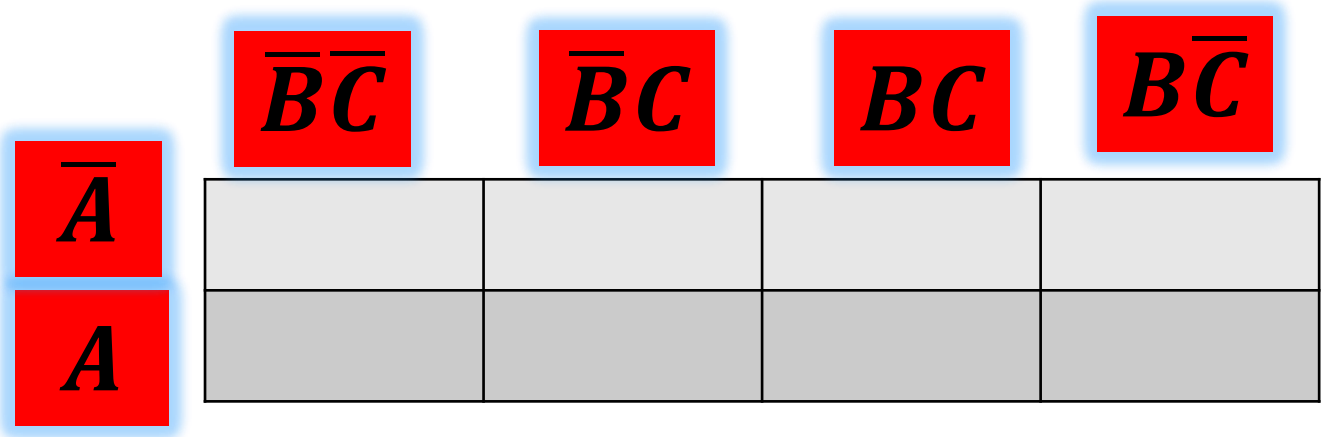
VLSI Design, ECE

School of Computer Science and Engineering

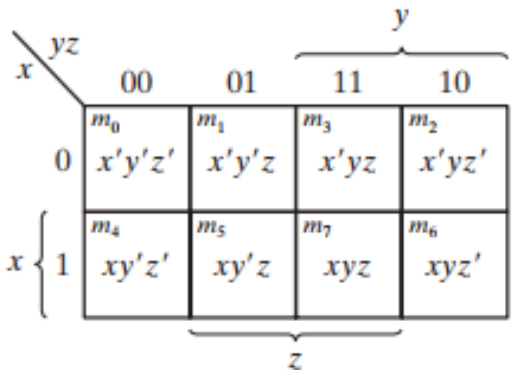


Three Variable Karnaugh Map

Question: Simplify the Boolean function $f(A, B, C) = \sum(0, 2, 4, 5)$ using K-Map?

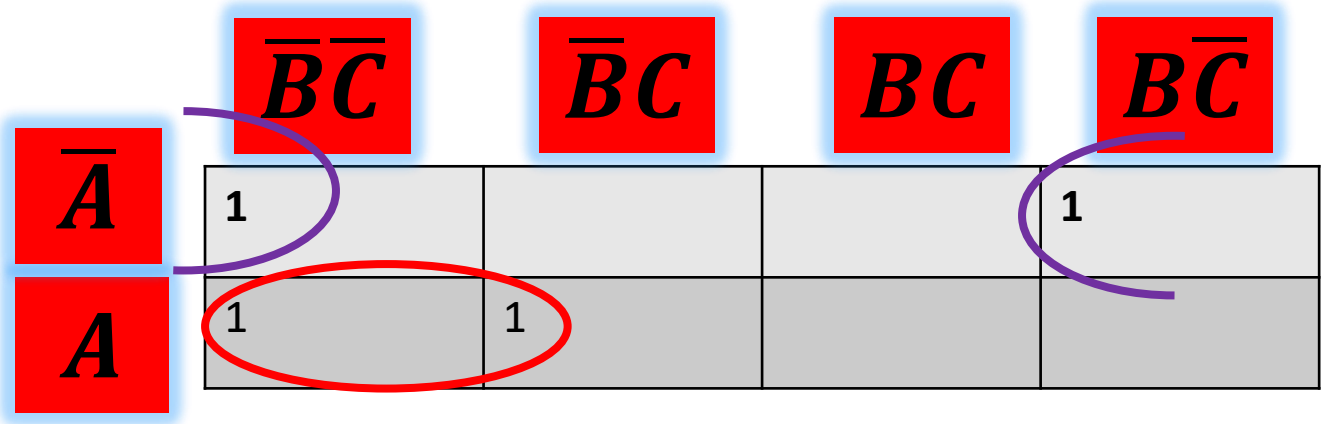


m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6

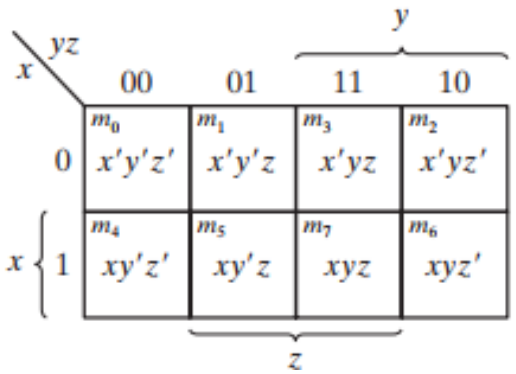


Three Variable Karnaugh Map

Question: Simplify the Boolean function $f(A, B, C) = \sum(0, 2, 4, 5)$ using K-Map?

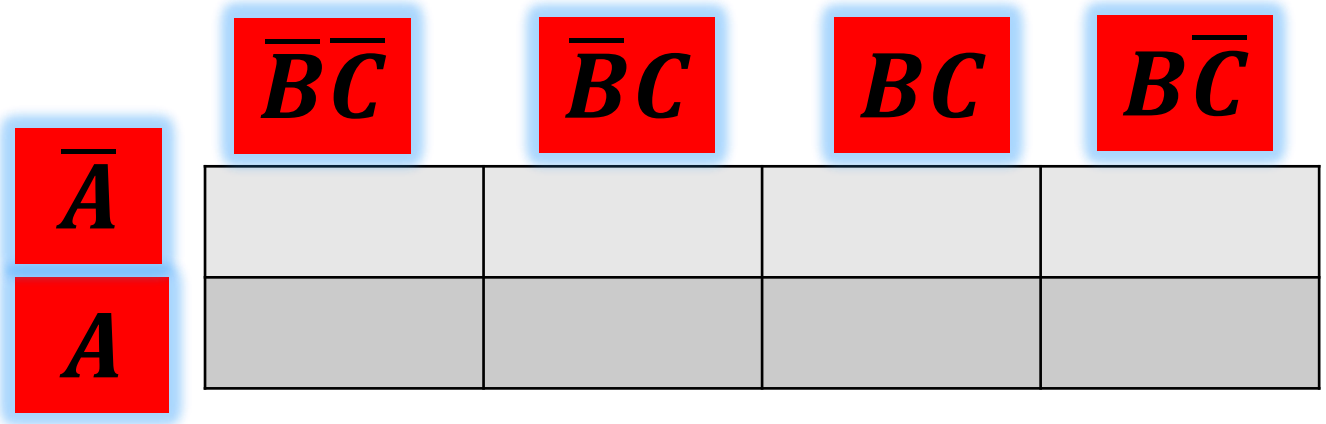


m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6

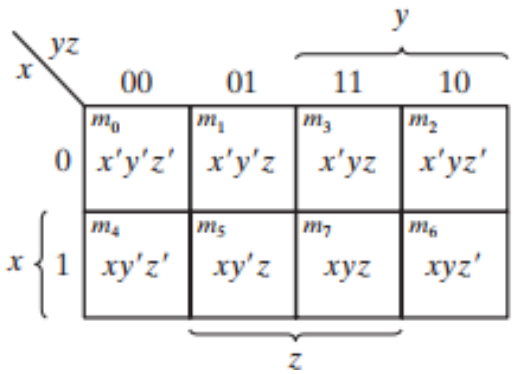


Three Variable Karnaugh Map

Question: Simplify the Boolean function $f(A, B, C) = \sum(0, 2, 4, 5, 6)$ using K-Map?



m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6



QUIZ QUIZ (POLL)

Which of the following combination of cells is NOT possible?

a) 4

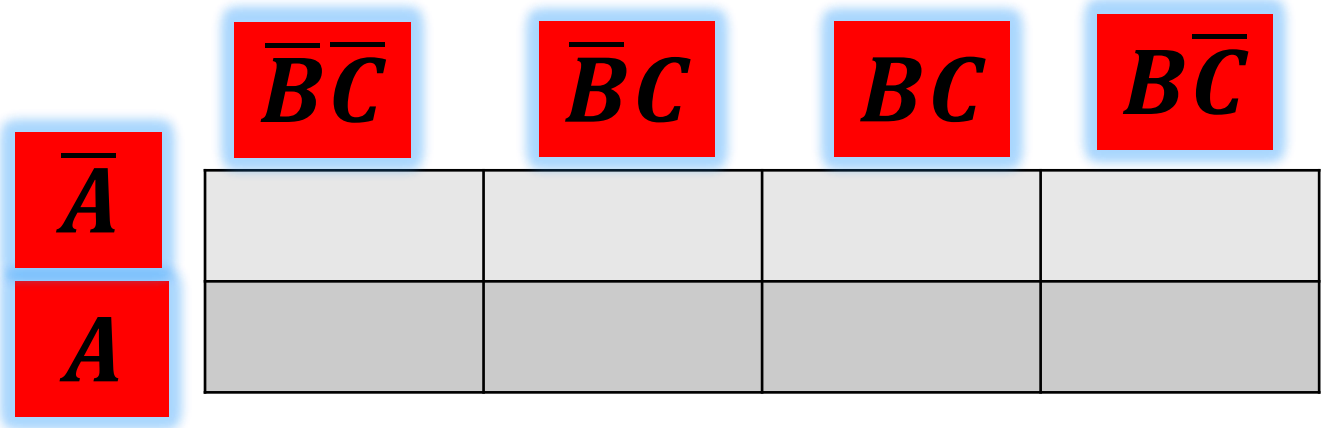
b) 8

c) 12

d) 16

Three Variable Karnaugh Map

Question: Simplify the Boolean function $f(A, B, C) = \sum(1, 2, 4, 7)$ using K-Map?



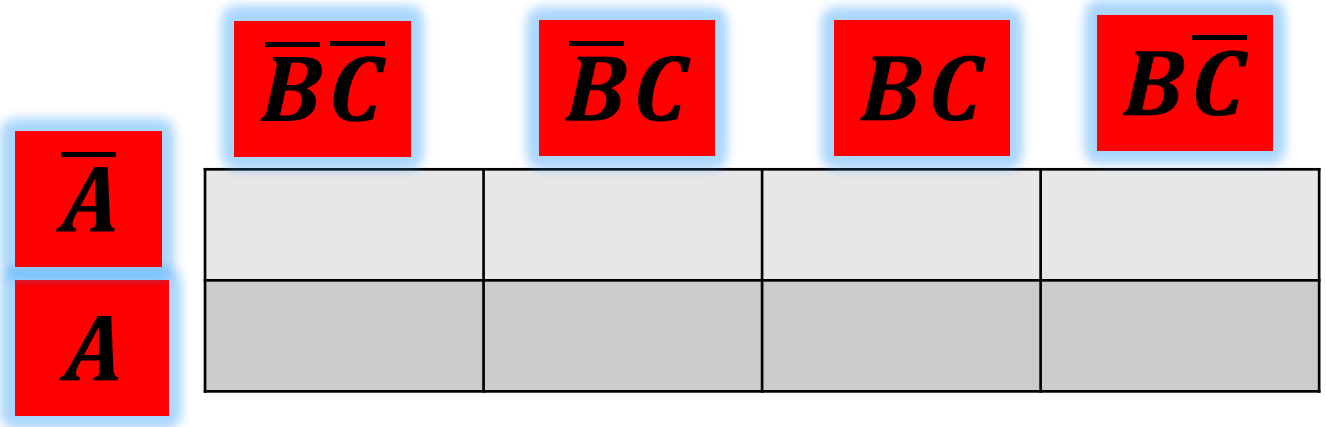
m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6

x \ yz	00	01	11	10
0	m_0 $x'y'z'$	m_1 $x'y'z$	m_3 $x'yz$	m_2 $x'yz'$
1	m_4 $xy'z'$	m_5 $xy'z$	m_7 xyz	m_6 xyz'

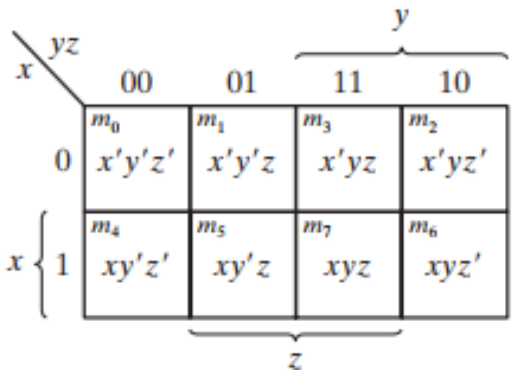
(y spans 11 and 10; z spans 01 and 11)

Three Variable Karnaugh Map

Question: Simplify the Boolean function $f(A, B, C) = \sum(3, 5, 6, 7)$ using K-Map?



m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6



Three Variable Karnaugh Map

Question: Simplify the Boolean function $f(A, B, C) = xy + \overline{x}\overline{y}\overline{z} + \overline{x}y\overline{z}$ using K-Map?

	$\overline{B}\overline{C}$	$\overline{B}C$	BC	$B\overline{C}$
$\overline{A}$				
A				

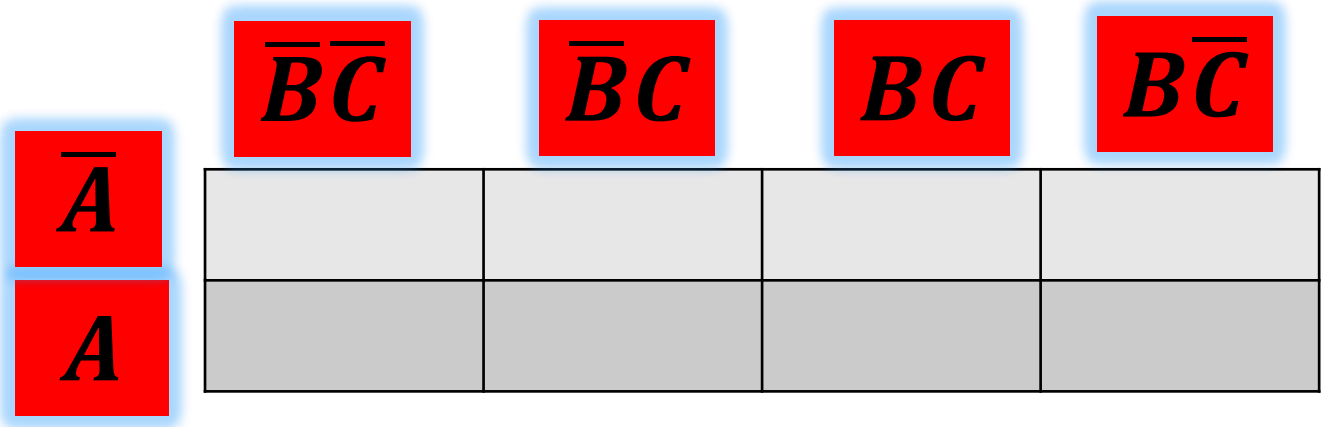
m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6

x \ yz	00	01	11	10
0	m_0 $x'y'z'$	m_1 $x'y'z$	m_3 $x'yz$	m_2 $x'yz'$
1	m_4 $xy'z'$	m_5 $xy'z$	m_7 xyz	m_6 xyz'

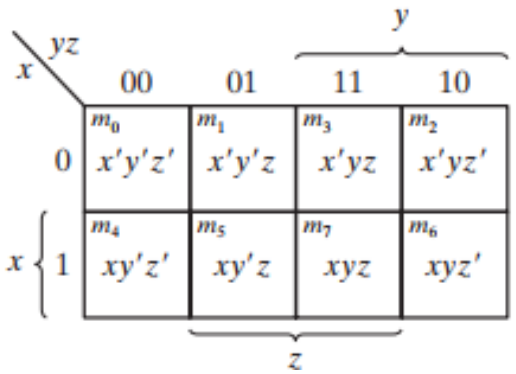
(y spans 11 and 10; z spans 01 and 11)

Three Variable Karnaugh Map: POS Simplification

Question: Simplify the Boolean function $f(A, B, C) = \sum(3, 5, 6, 7)$ into POS form?

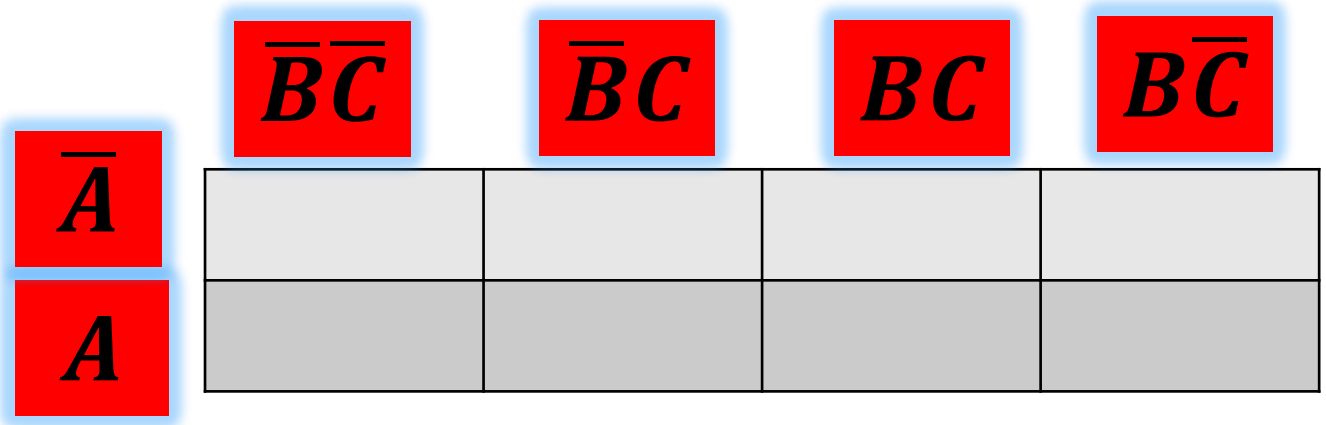


m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6

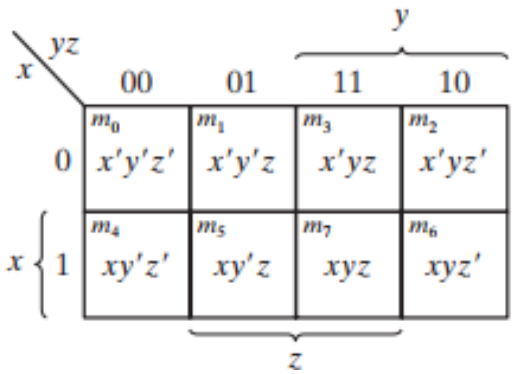


Three Variable Karnaugh Map: POS Simplification

Question: Simplify the Boolean function $f(A, B, C) = \prod(0, 1, 4, 5, 6)$ into POS form?



m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6



DIGITAL ELECTRONICS: ECE 213

Topic: Karnaugh Maps (Upto 4 Variables)

UNIT II: Combinational Logic System

Lecture No.: 14

Prepared By: Irfan Ahmad Pindoo

Assistant Professor

VLSI Design, ECE

School of Computer Science and Engineering

Four Variable Karnaugh Map

- There are sixteen minterms for four binary variables; therefore, the map consists of sixteen squares.

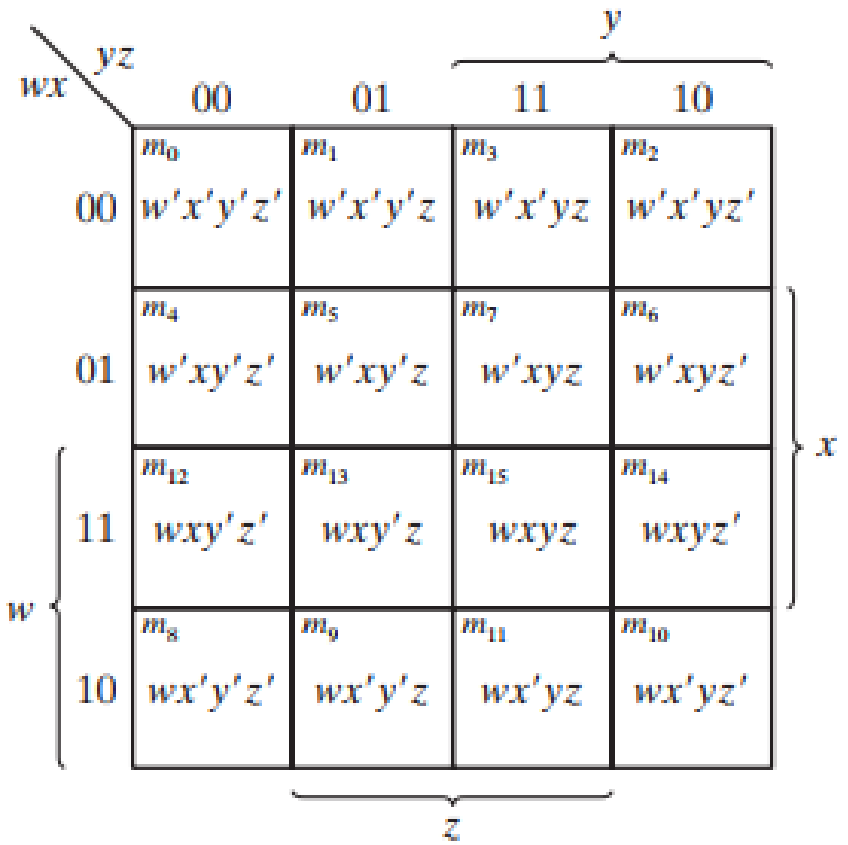
NOTE:

- The minterms are arranged, not in a binary sequence, but in a sequence **similar to the Gray code**.

Four Variable Karnaugh Map

m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6
m_{12}	m_{13}	m_{15}	m_{14}
m_8	m_9	m_{11}	m_{10}

(a)



(b)

Four Variable Karnaugh Map

Question: Simplify the Boolean function:

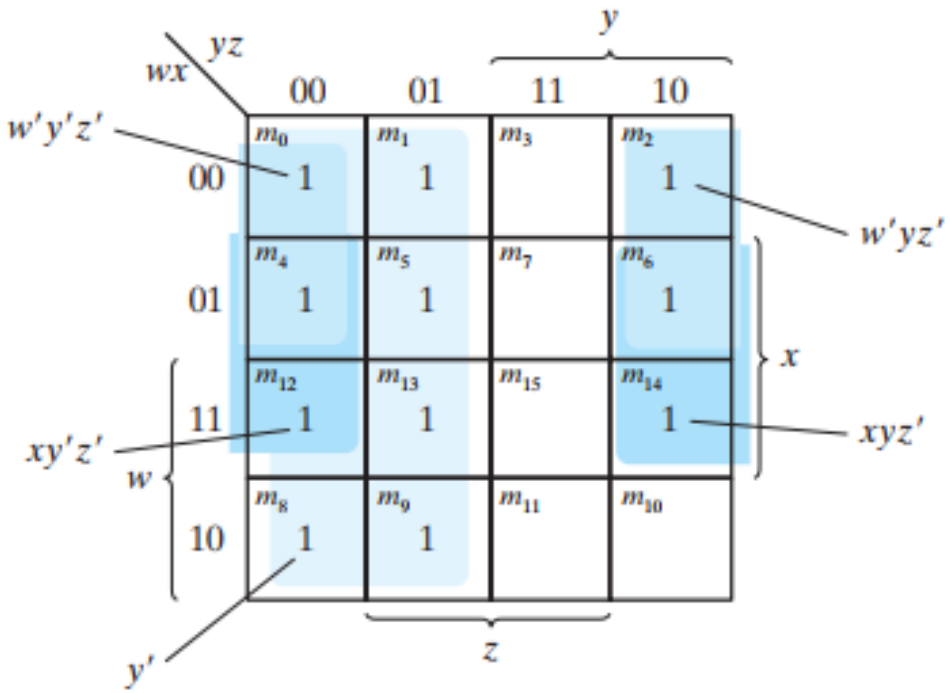
$f(w, x, y, z) = \sum(0, 1, 2, 4, 5, 6, 8, 9, 12, 13, 14)$ **into SOP form?**

	$\bar{y}\bar{z}$	$\bar{y}z$	yz	$y\bar{z}$
$\bar{w}\bar{x}$				
$\bar{w}x$				
wx				
$w\bar{x}$				

Four Variable Karnaugh Map

Question: Simplify the Boolean function:

$f(w, x, y, z) = \sum(0, 1, 2, 4, 5, 6, 8, 9, 12, 13, 14)$ **into SOP form?**



Note: $w'y'z' + w'yz' = w'z'$

$xy'z' + xyz' = xz'$

QUIZ QUIZ (POLL)

Each product term of a group, w’.x.y’ and w.y, represents the _____ in that group.

a) Input
b) POS
c) Sum-of-Minterms
d) Sum of Maxterms

Four Variable Karnaugh Map

Simplify the Boolean function

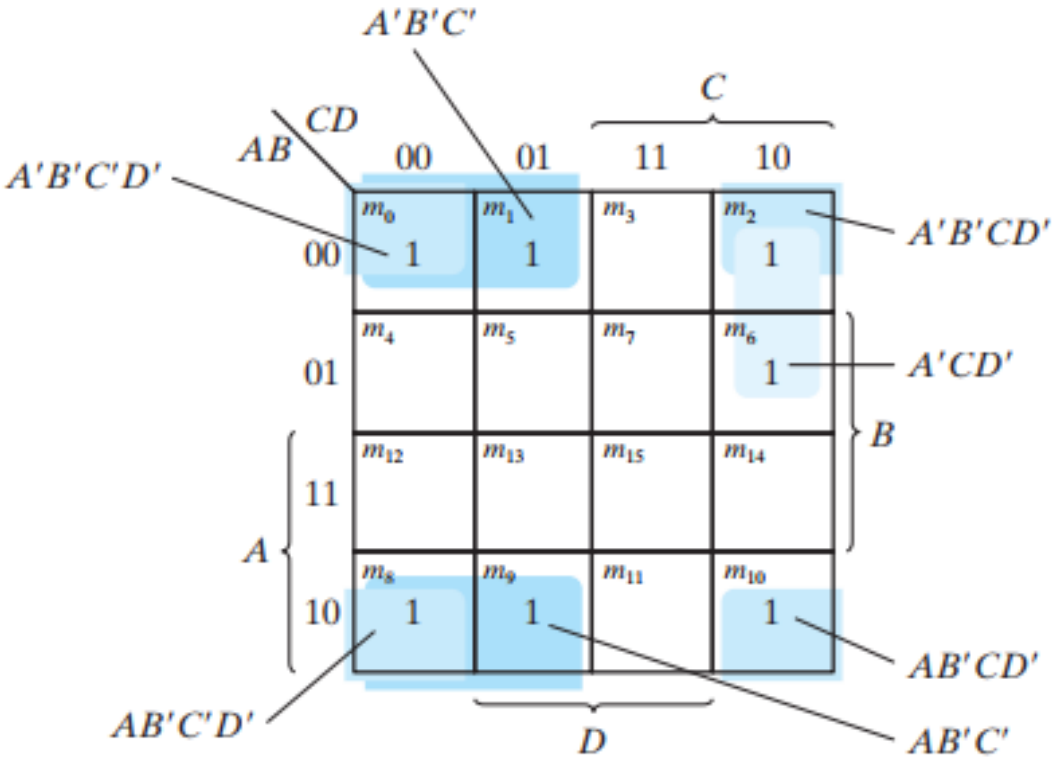
$$F = A'B'C' + B'CD' + A'BCD' + AB'C'$$

	$\overline{C}\,\overline{D}$	$\overline{C}D$	CD	$C\overline{D}$
$\overline{A}\,\overline{B}$				
$\overline{A}B$				
AB				
$A\overline{B}$				

Four Variable Karnaugh Map

Simplify the Boolean function

$$F = A'B'C' + B'CD' + A'BCD' + AB'C'$$



Four Variable Karnaugh Map

Question: Simplify the Boolean function:
$f(w, x, y, z) = \sum(0, 1, 2, 5, 8, 9, 10)$ into:
(1) SOP Form (2) POS form?

	$\overline{C}\,\overline{D}$	$\overline{C}D$	CD	$C\overline{D}$
$\overline{A}\,\overline{B}$				
$\overline{A}B$				
AB				
$A\overline{B}$				

Four Variable Karnaugh Map

Question: Simplify the Boolean function:
$f(w, x, y, z) = \sum(0, 1, 2, 5, 8, 9, 10)$ into:
(1) SOP Form (2) POS form?

	$\overline{C}\,\overline{D}$	$\overline{C}D$	CD	$C\overline{D}$
$\overline{A}\,\overline{B}$				
$\overline{A}B$				
AB				
$A\overline{B}$				

Four Variable Karnaugh Map

Question: Simplify the Boolean function:
$f(w, x, y, z) = \sum(0, 1, 2, 5, 8, 9, 10)$ into:
(1) SOP Form (2) POS form?

	$\overline{C}\,\overline{D}$	$\overline{C}D$	CD	$C\overline{D}$
$\overline{A}\,\overline{B}$				
$\overline{A}B$				
AB				
$A\overline{B}$				

Four Variable Karnaugh Map

Question: Simplify the Boolean function:
$f(w, x, y, z) = \sum(0, 1, 2, 5, 8, 9, 10)$ into:
(1) SOP Form (2) POS form?

	$\overline{C}\,\overline{D}$	$\overline{C}D$	CD	$C\overline{D}$
$\overline{A}\,\overline{B}$				
$\overline{A}B$				
AB				
$A\overline{B}$				

Don’t Care conditions

- In practice, in some applications the function is **not specified** for certain combinations of the variables.
- As an example, the four-bit binary code for the decimal digits has six combinations that are not used and consequently are considered to be unspecified.
- In most applications, we **simply don’t care** what value is assumed by the function for the unspecified minterms.
- For this reason, it is customary to call the **unspecified minterms** of a function **don’t-care conditions**.
- These don’t-care conditions can be used on a map to provide **further simplification of the Boolean expression.**
- Symbol of don’t care: **X**

QUIZ QUIZ (POLL)

Don’t care conditions can be used for simplifying Boolean expressions in _____

a) Registers

b) Terms

c) K-maps

d) Latches

Don’t Care conditions

Qsn: Implement the given function using K-Map:

$$F(A, B, C, D) = \Sigma(2, 4, 10, 12, 14,)$$
$$d(A, B, C, D) = \Sigma(0, 1, 5, 8)$$

	$\overline{C}\overline{D}$	$\overline{C}D$	CD	$C\overline{D}$
$\overline{A}\overline{B}$				
$\overline{A}B$				
AB				
$A\overline{B}$				

Don’t Care conditions

Qsn: Implement the given function using K-Map:

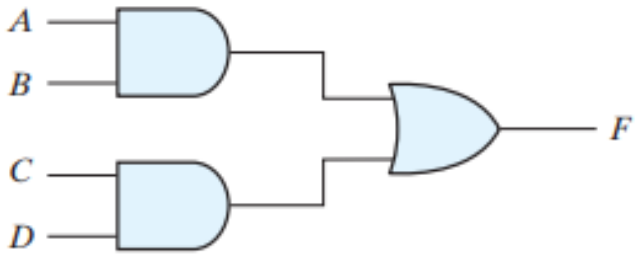
$$F(A, B, C, D) = \Sigma(2, 4, 10, 12, 14,)$$
$$d(A, B, C, D) = \Sigma(0, 1, 5, 8)$$

	$\overline{C}\overline{D}$	$\overline{C}D$	CD	$C\overline{D}$
$\overline{A}\overline{B}$				
$\overline{A}B$				
AB				
$A\overline{B}$				

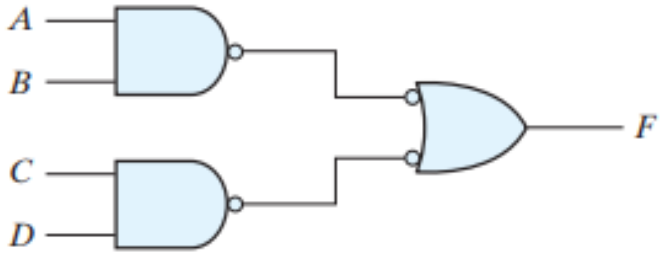
NAND and NOR Implementation

- Digital circuits are frequently constructed with NAND or NOR gates rather than with AND and OR gates. NAND and NOR gates are easier to fabricate with electronic components and are the basic gates used in all IC digital logic families.
- **They are a universal gates because any logic circuit can be implemented with them.**

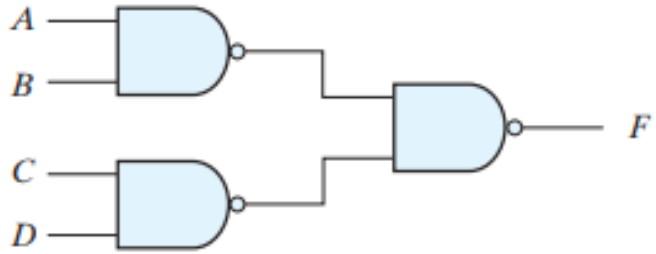
NAND Implementation



(a)

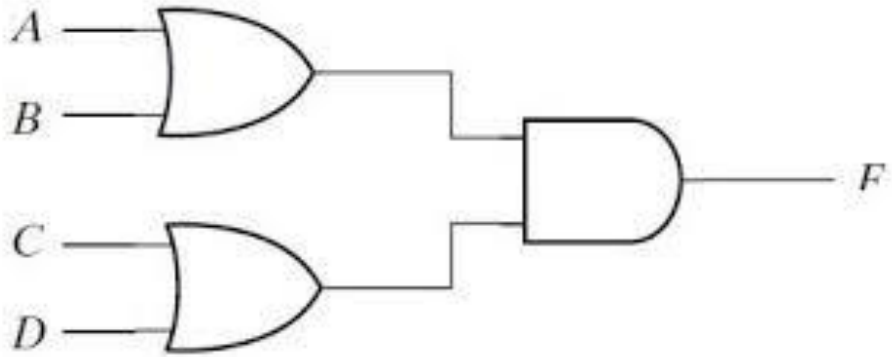


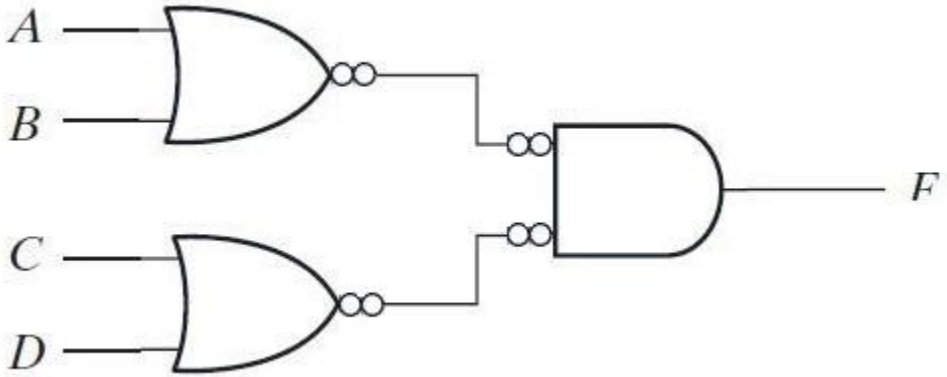
(b)

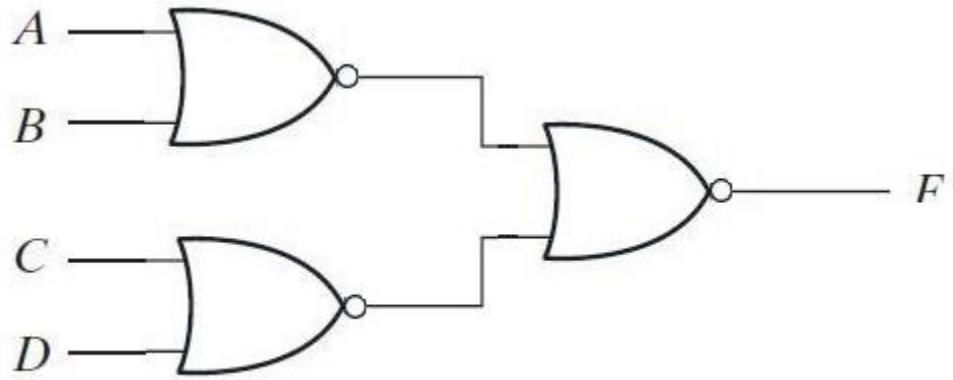


(c)

NOR Implementation







NAND and NOR Implementation

Question: Implement the Boolean function $f(A, B, C) = \sum(1, 2, 3, 4, 5, 7)$ using

(a) NAND gates only.

(b) NOR gates only

(c) SOP form.

(d) POS form.

	$\overline{B}\overline{C}$	$\overline{B}C$	BC	$B\overline{C}$
$\overline{A}$				
A				

Explanation Slide

NAND and NOR Implementation

Question: Implement the Boolean function $f(A, B, C) = \sum(1, 2, 3, 4, 5, 7)$ using

(a) NAND gates only.

(b) NOR gates only

(c) SOP form.

(d) POS form.

	$\overline{B}\overline{C}$	$\overline{B}C$	BC	$B\overline{C}$
$\overline{A}$				
A				

Explanation Slide

QUIZ QUIZ (POLL)

Product-of-Sums expressions can be implemented using _____

a) 2-level OR-AND logic circuits

b) 2-level NOR logic circuits

c) 2-level XOR logic circuits

d) Both 2-level OR-AND and NOR logic circuits