TSN1101
Computer Architecture and Organization

Section A (Digital Logic Design)

Lecture 03

[redacted]

Design of Combinational Logic Circuits

TOPIC COVERAGE IN THE LECTURE - Part 1 of 2

- **Standard forms of Boolean Expression**
  - Sum of Products (SOP)
  - Product of Sums (POS)
- **Conversions**
  - SOP ⇔ POS
  - SOP/POS ⟺ Truth table
- **Simplification using Karnaugh Map**
  - Two, Three, and Four Variables K-Map
  - SOP/POS minimization
  - K- Map with Don't Care Conditions

TOPIC COVERAGE IN THE LECTURE - Part 2 of 2

Design of Combinational Logic Circuits

- From a Boolean Expression/Truth Table/Given Problem statement to a simplified Logic circuit

Universal Gates (NAND & NOR)

- Construction of Other gates using only NAND or using only NOR gates
- Design of Combinational Logic Circuits using only NAND or using only NOR Gates

Design of Combinational Logic Circuits – Part 1 of 2

- **SOP and POS forms, Conversions**
- **Simplification using K-map**

Standard Forms of Boolean Expression

- Sum of Products (SOP)

Sum of Products (SOP) Form

• When two or more product terms are summed by Boolean addition, the resulting expression is called sum-of-products (SOP).

•Example:

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

$$A + AB + C\overline{D} + EF + GK + H\overline{I}$$

• Each of these sum of products expressions consists of 2 or more AND terms (products) that are ORed together.

• Each AND term consists of one or more variable(s) individually appearing in either complemented or non complemented form also known as literal.

Sum of Products (SOP) Form - Examples

• Single overbar cannot extend over more than one variable, although more than one variable in a term can have an overbar.

• Example: Can have the term $\overline{A}\,B\,\overline{C}$ but not $\overline{ABC}$

Problem:

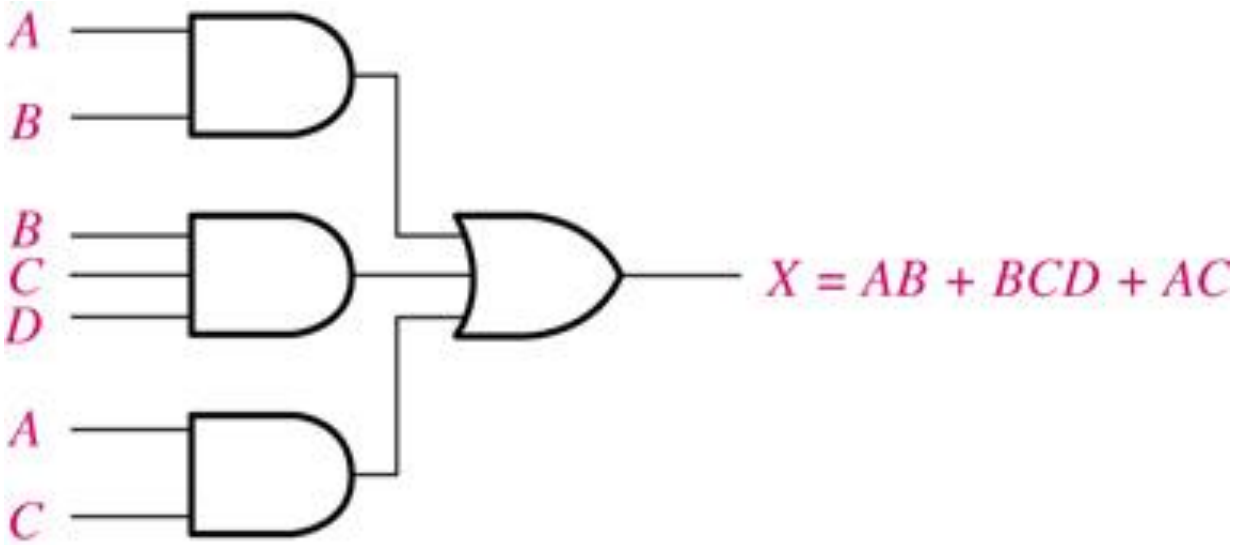
Which of the following expressions is SOP ?

1. $AB + B(CD + EF)$

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

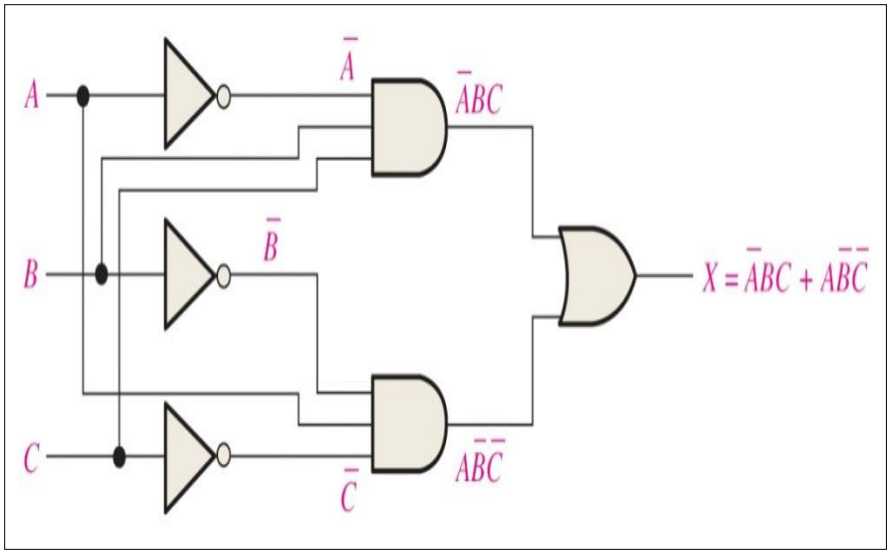
Sum of Products (SOP) Form - Implementation

• A SOP expression can be implemented by AND–OR logic in which the outputs of a number (equal to the number of product terms in the expression) of AND gates feed into the inputs of an OR gate .



Sum of Products (SOP) Form
- Implementation – An example

INPUTS			OUTPUT	
A	*B*	*C*	*X*	PRODUCT TERM
0	0	0	0	
0	0	1	0	
0	1	0	0	
0	1	1	1	$\overline{A}BC$
1	0	0	1	$A\overline{B}\,\overline{C}$
1	0	1	0	
1	1	0	0	
1	1	1	0	



Standard Sum of Products (SOP) Form

• Minterm is a product term in which all variables in the domain must be present either in complemented or uncomplemented form.

•A standard SOP expression is one in which all the variables in the domain appear in each product term in the expression (i.e) all terms in the expression must be minterms.

Example:

$$A\bar{B}CD + \bar{A}\bar{B}C\bar{D} + AB\bar{C}\bar{D}$$

• Standard SOP expression are important in constructing truth tables and in the Karnaugh map (K-map) simplification method.

Standard Forms of Boolean Expression

- Product of Sums(POS)

Product of Sums (POS) Form

• When two or more sum terms are multiplied, the resulting expression is called **product-of-sums (POS)**

•Example:

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

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

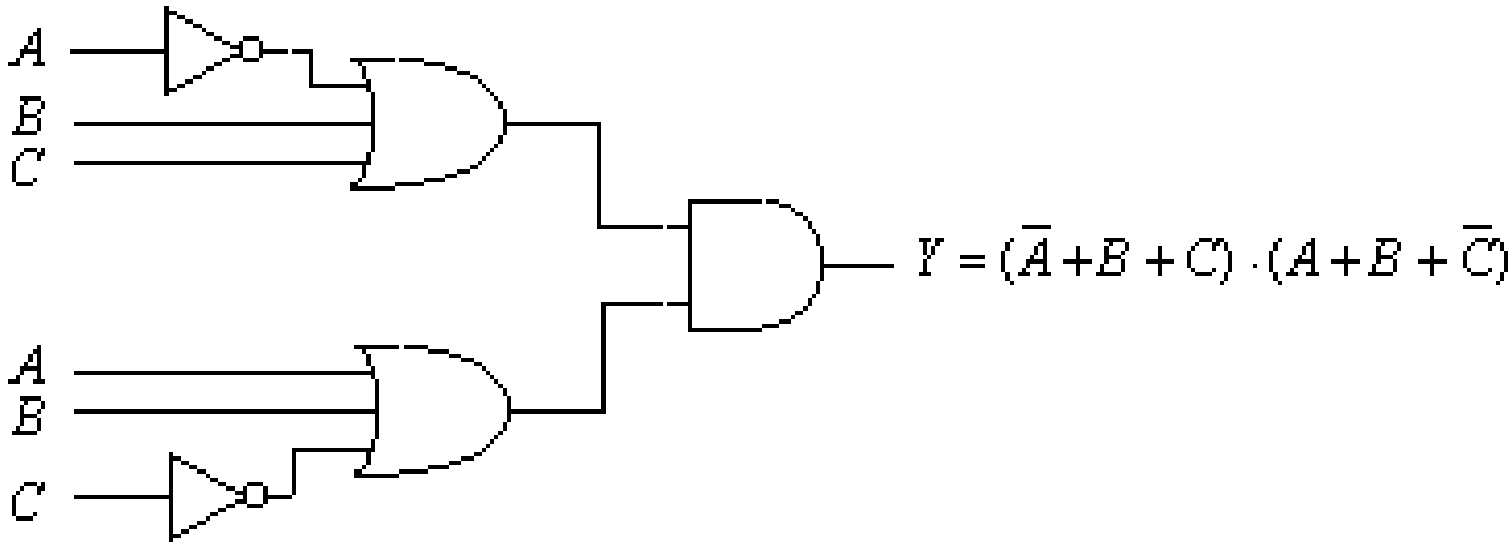
$$(\overline{A}+\overline{B}+\overline{C})(C+\overline{D}+E)(\overline{B}+C+D)$$

• Each of these product of sum expressions consists of 2 or more OR terms (sums) that are ANDed together.

• Each OR term consists of one or more variable(s) individually appearing in either complemented or non complemented form.

Product of Sums (POS) Form - Implementation

• A POS expression can be implemented by OR–AND logic in which the outputs of a number (equal to the number of sum terms in the expression) of OR gates feed into the inputs of an AND gate .



Implementation of the POS expression

Standard Product of Sums (POS) Form

Maxterm is a sum term in which all variables in the domain must be present either in complemented or uncomplemented form.

A standard POS expression is one in which all the variables in the domain appear in each sum term in the expression. (i.e) all terms in the expression must be maxterms.

For example:

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

• Any nonstandard POS expression can be converted to the standard form using Boolean algebra/truth table.

Conversions

-SOP expression to Truth Table
-SOP expression to POS expression

-POS expression to Truth Table
-POS expression to SOP expression

-Truth Table to SOP expression
-Truth Table to POS expression

CONVERSIONS
- SOP expression to Truth Table

Step 1: List all possible combinations of binary values of the variables in the expression.

Step 2: Convert the SOP expression to standard form if it is not already.

Step 3: Place a 1 in the output column for each binary value.

CONVERSIONS- SOP expression to Truth Table - Example

Example: Develop a truth table for the standard SOP expression, $\overline{A}\,\overline{B}\,C + A\,\overline{B}\,\overline{C} + A\,B\,C$

Solution: There are three variables in the domain, so there are eight possible combinations of binary value of the variables as listed left in the three columns of table below. The binary values that make the product terms, in the expression equal to 1 are $\overline{A}\overline{B}C : 001; A\overline{B}\overline{C} : 100; ABC : 111$

INPUTS			OUTPUT	PRODUCT TERM
A	B	C	X	
0	0	0	0	
0	0	1	1	$\overline{A}\overline{B}C$
0	1	0	0	
0	1	1	0	
1	0	0	1	$A\overline{B}\overline{C}$
1	0	1	0	
1	1	0	0	
1	1	1	1	ABC

For each these binary values, a 1 is placed in the output column as shown in the table.

For each of the remaining binary combinations, a 0 is placed in the output column.

CONVERSIONS
- SOP expression to POS expression

• The binary values of the product terms in a given standard SOP expression are not present in the equivalent standard POS expression

Therefore, to convert from standard SOP to standard POS, the following steps are taken:

Step 1: Evaluate each term in the SOP expression. That is, determine the binary numbers that represent the product terms.

Step 2: Determine all of the binary numbers not included in the evaluation in step 1.

Step 3: Write the equivalent sum term for each binary from step 2 and express it in POS form.

CONVERSIONS- SOP expression to POS expression - Example

Example: Convert the given SOP expression to POS expression $\overline{A}\,\overline{B}\,C + A\,\overline{B}\,\overline{C} + A\,B\,C$

Solution: There are three variables in the domain, so there are eight possible combinations of binary value of the variables as listed left in the three columns of table below. The binary values that make the product terms, in the expression equal to 1 are $\overline{A}\,\overline{B}C : 001; A\overline{B}\,\overline{C} : 100; ABC : 111$

Inputs			Output	Product Term	Sum Term
A	B	C	X		
0	0	0	0		A + B + C
0	0	1	1	$\overline{A}\,\overline{B}C$	
0	1	0	0		A + $\overline{B}$ + C
0	1	1	0		A + $\overline{B}$ + $\overline{C}$
1	0	0	1	$A\overline{B}\,\overline{C}$	
1	0	1	0		$\overline{A}$ + B + $\overline{C}$
1	1	0	0		$\overline{A}$ + $\overline{B}$ + C
1	1	1	1	ABC	

For each these binary values, a 1 is placed in the output column as shown in the table.
For each of the remaining binary combinations, a 0 is placed in the output column.

POS expression from the Truth Table:

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

CONVERSIONS
- POS expression to Truth Table

Step 1: List all the possible combinations of binary values of the variables.

Step 2: Convert the POS expression to standard form if it is not already.

Step 3: Place a 0 in the output column for each binary value that makes the expression a 0 and place a 1 for all the remaining binary values.

CONVERSIONS- POS expression to Truth Table - Example

Example: Determine the truth table for the following standard POS expression:

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

Solution: There are three variables in the domain and the eight possible binary values are listed in the left three columns in the table below. The binary values that make the sum terms in the expression equal to 0 are

$$A+B+C:000;\ A+\overline{B}+C:010,\ A+\overline{B}+\overline{C}:011;\ \overline{A}+B+\overline{C}:101;\ \overline{A}+\overline{B}+C:110$$

For each of the remaining binary combinations, a 1 is placed in the output column.

INPUTS			OUTPUT	SUM TERMS
A	**B**	**C**	**X**	
0	0	0	0	$(A+B+C)$
0	0	1	1	
0	1	0	0	$(A+\overline{B}+C)$
0	1	1	0	$(A+\overline{B}+\overline{C})$
1	0	0	1	
1	0	1	0	$(\overline{A}+B+\overline{C})$
1	1	0	0	$(\overline{A}+\overline{B}+C)$
1	1	1	1	

CONVERSIONS
- POS expression to SOP expression

• The binary values of the sum terms in a given standard POS expression are not present in the equivalent standard SOP expression

Therefore, to convert from standard POS to standard SOP, the following steps are taken:

Step 1: Evaluate each term in the POS expression. That is, determine the binary numbers that represent the sum terms.

Step 2: Determine all of the binary numbers not included in the evaluation in step 1.

Step 3: Write the equivalent product term for each binary from step 2 and express it in SOP form.

CONVERSIONS- POS expression to SOP expression - Example

Example: Convert the given POS expression to SOP expression:

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

Solution: There are three variables in the domain and the eight possible binary values are listed in the left three columns in the table below. The binary values that make the sum terms in the expression equal to 0 are

$$A+B+C:000;\ A+\overline{B}+C:010,\ A+\overline{B}+\overline{C}:011;\ \overline{A}+B+\overline{C}:101;\ \overline{A}+\overline{B}+C:110$$

For each of the remaining binary combinations, a 1 is placed in the output column.

Inputs			Output	Sum Term	Product Term
A	**B**	**C**	**X**		
0	0	0	0	$(A+B+C)$	
0	0	1	**1**		$(\overline{A}.\overline{B}.C)$
0	1	0	0	$(A+\overline{B}+C)$	
0	1	1	0	$(A+\overline{B}+\overline{C})$	
1	0	**0**	**1**		$(A.\overline{B}.\overline{C})$
1	0	1	0	$(\overline{A}+B+\overline{C})$	
1	1	0	0	$(\overline{A}+\overline{B}+C)$	
1	1	1	**1**		$(A.B.C)$

SOP expression from the Truth Table:

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

CONVERSIONS
- Truth Table to SOP expression

Step 1: List the binary values of the input variables for which output is 1.

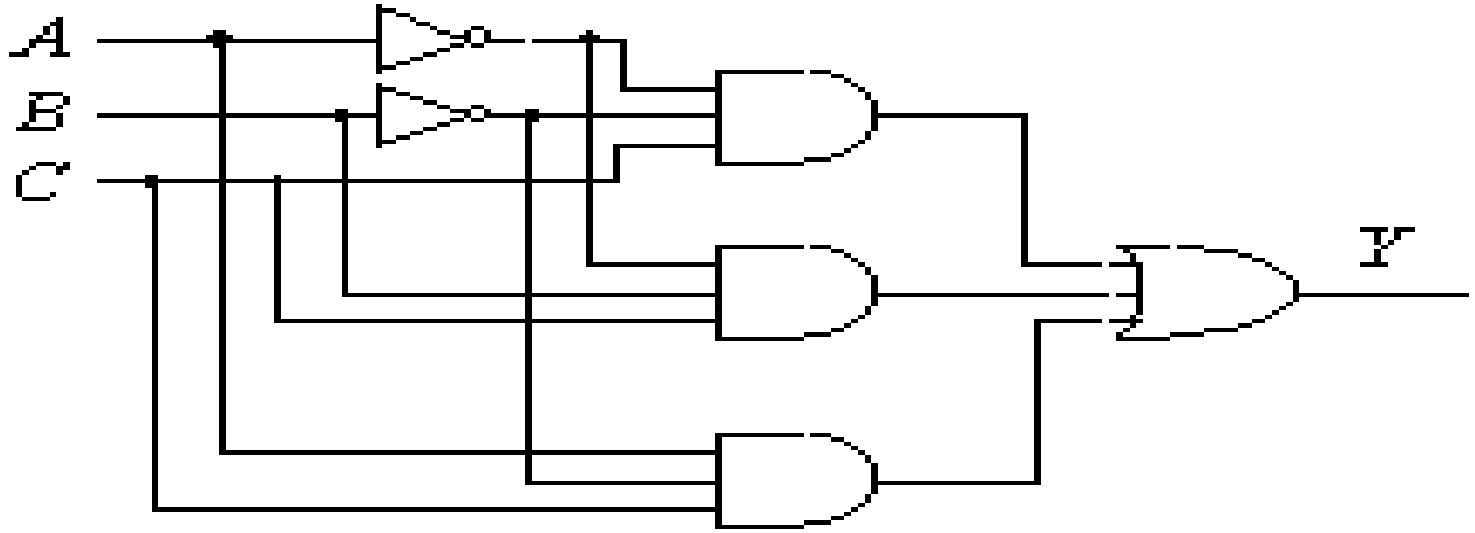
Step 2: Convert the binary values to the corresponding product term by replacing each 1 with the corresponding product term.

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

A	B	C	Y	
0	0	0	0	
0	0	1	1	$\overline{A}\,\overline{B}\,C$
0	1	0	0	
0	1	1	1	$\overline{A}BC$
1	0	0	0	
1	0	1	1	$A\overline{B}C$
1	1	0	0	
1	1	1	0	

Truth Table

CONVERSIONS -Truth Table to SOP expression - Implementation



Equivalent AND-OR circuit

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

CONVERSIONS -Truth Table to SOP expression - Problem

Write a sum of product Boolean expression and draw their logic diagram for given truth table.

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

CONVERSIONS
- Truth Table to POS expression

Step 1: List the binary values of the input variables for which output is 0.

Step 2: Convert the binary values to the corresponding sum term by replacing each 1 with the corresponding variable complement and each 0 with the corresponding variable.

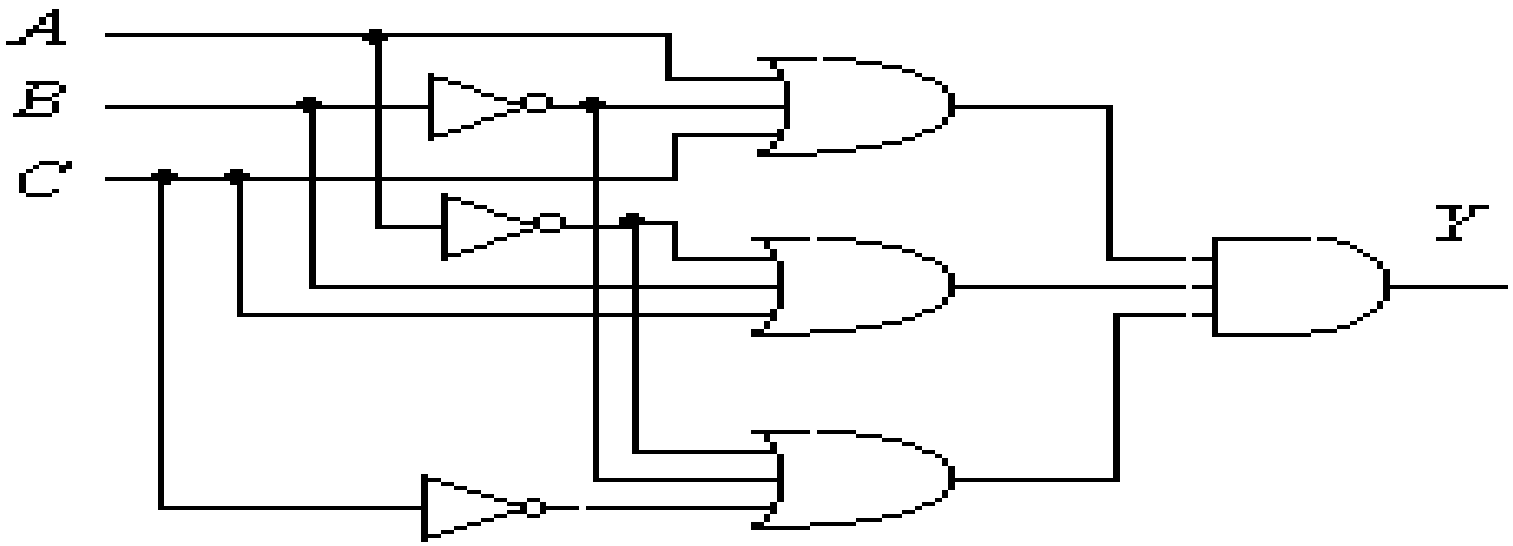
A	B	C	Y	
0	0	0	1	
0	0	1	1	
0	1	0	0	$A+\overline{B}+C$
0	1	1	1	
1	0	0	0	$\overline{A}+B+C$
1	0	1	1	
1	1	0	1	
1	1	1	0	$\overline{A}+\overline{B}+\overline{C}$

Truth Table

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

CONVERSIONS -Truth Table to POS expression - Implementation

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



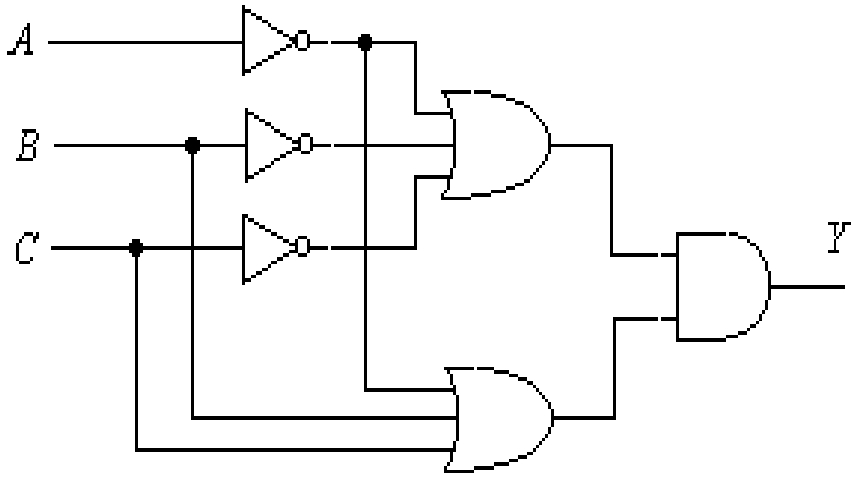
Equivalent OR-AND circuit

CONVERSIONS -Truth Table to POS expression - Problem

Problem:

Write a product of sum Boolean expression and draw their logic diagram for the given truth table.

Solution:



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

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

Review - Terms

A variable is a symbol used to represent an action, a condition, or data. Each variable has a value of 1 or 0.

The complement represents the inverse of a variable; indicated by an over-bar or prime symbol. Thus, the complement of *A* is $\overline{A}$ or A'

A literal is a variable or its complement.

When two or more product terms are summed by Boolean addition, the resulting expression is called sum-of-products (SOP).

When two or more sum terms are multiplied, the resulting expression is called **product-of-sums (POS)**

Minterm is a product term in which all variables in the domain must be present either in complemented or uncomplemented form.
A standard SOP expression is one in which all the variables in the domain appear in each product term in the expression (i.e) all terms in the expression must be minterms.

Maxterm is a sum term in which all variables in the domain must be present either in complemented or uncomplemented form.
A standard POS expression is one in which all the variables in the domain appear in each sum term in the expression. (i.e) all terms in the expression must be maxterms.

Simplification of Boolean Expression

- Karnaugh Map method

KARNAUGH MAP
- Introduction

- Provides a systematic method for simplifying Boolean expression. Hence it is used to minimize the number of logic gates that are required in a digital circuit.
- Presents all possible values of input variables and the resulting output of each value.

- Karnaugh map is an array of cells in which each cell represents a binary value of the input variables.

If two variables are used, then a 2X2 map is used

If three variables are used, then a 2X4 or 4X2 map is used

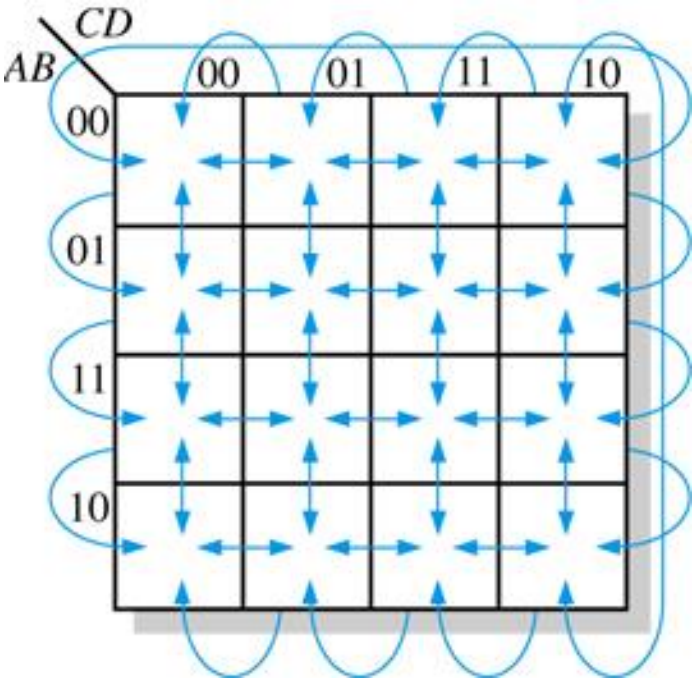
If four variables are used, then a 4X4 map is used

If five variables are used, then a 8X4 map is used

Number of cells in the map = total number of input combinations

KARNAUGH MAP
- Properties

Adjacent cells on a Karnaugh map are those that differ by only one variable. Arrows point between adjacent cells.



•When going from one cell of the map to an adjacent cell will only require one variable to change.

•When moving on the map only go right or left, up or down, never go on a diagonal.

• Also the map folds around its self so the going from a cell on the top right to one on the bottom right only changes one variable.

Two variable Karnaugh Map
- Introduction

A	B	Product term
0	0	$\overline{A}\,\overline{B}$
0	1	$\overline{A}B$
1	0	$A\overline{B}$
1	1	AB

	$\overline{B}$	B
$\overline{A}$	$\overline{A}\,\overline{B}$	$\overline{A}B$
A	$A\overline{B}$	AB

Mapping Two variable Karnaugh Map - An Example

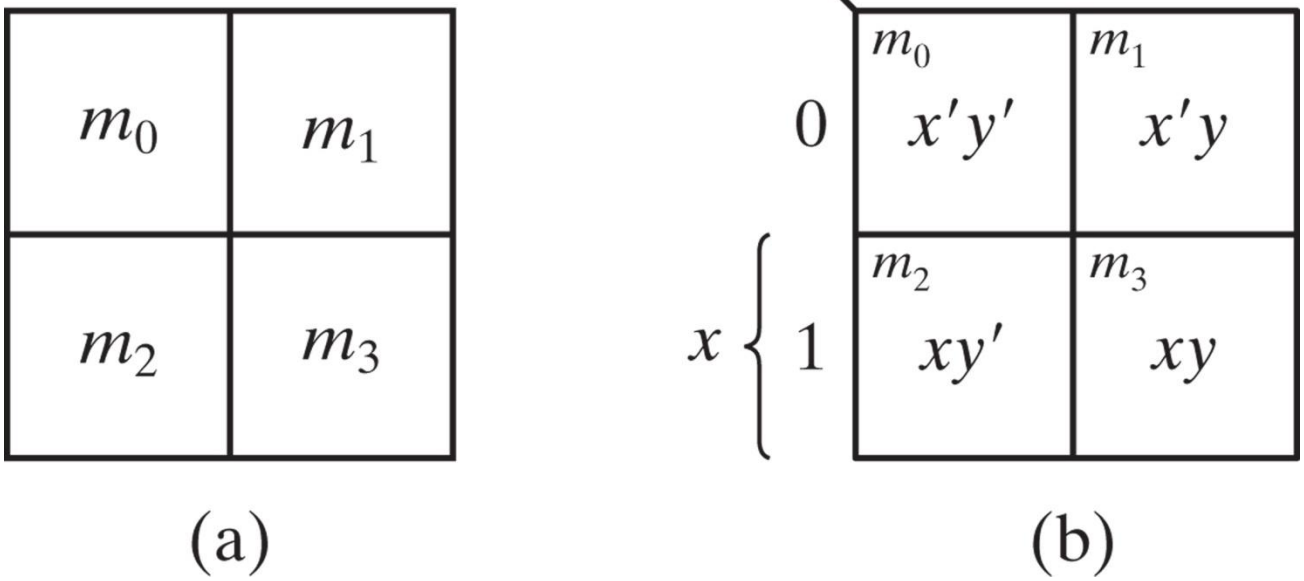
Map the following SOP expression of two variables

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

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

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

Two variable Karnaugh Map
- Another way of representation



Three variable Karnaugh Map
- Introduction

A	B	C	Product terms
0	0	0	$\bar{A}\bar{B}\bar{C}$
0	0	1	$\bar{A}\bar{B}C$
0	1	0	$\bar{A}B\bar{C}$
0	1	1	$\bar{A}BC$
1	0	0	$A\bar{B}\bar{C}$
1	0	1	$A\bar{B}C$
1	1	0	$AB\bar{C}$
1	1	1	ABC

	$\bar{C}$	C
$\bar{A}\bar{B}$	$\bar{A}\bar{B}\bar{C}$	$\bar{A}\bar{B}C$
$\bar{A}B$	$\bar{A}B\bar{C}$	$\bar{A}BC$
AB	$AB\bar{C}$	ABC
$A\bar{B}$	$A\bar{B}\bar{C}$	$A\bar{B}C$

Mapping Three variable Karnaugh Map - An Example

Map the SOP expression Y = $\overline{A}\,\overline{B}C + \overline{A}B\overline{C} + \overline{A}BC + A\overline{B}C + ABC$

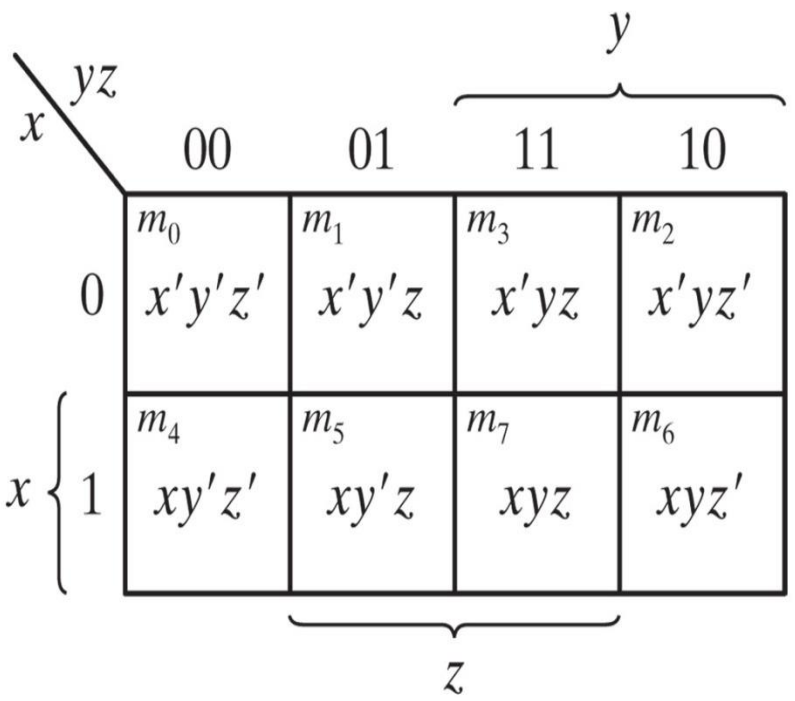
A	B	C	Y	
0	0	0	0	
0	0	1	1	$\overline{A}\,\overline{B}C$
0	1	0	1	$\overline{A}B\overline{C}$
0	1	1	1	$\overline{A}BC$
1	0	0	0	
1	0	1	1	$A\overline{B}C$
1	1	0	0	
1	1	1	1	ABC

	$\overline{C}$	C
$\overline{A}\,\overline{B}$	0	1
$\overline{A}B$	1	1
AB	0	1
$A\overline{B}$	0	1

Three variable Karnaugh Map
- Another way of representation

m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6

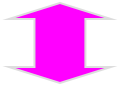
(a)



(b)

Four variable Karnaugh Map
- Introduction

A	B	C	D	Product Terms	A	B	C	D	Product Terms
0	0	0	0	$\overline{A}\,\overline{B}\overline{C}\,\overline{D}$	1	0	0	0	$A\,\overline{B}\overline{C}\,\overline{D}$
0	0	0	1	$\overline{A}\,\overline{B}\,\overline{C}\,D$	1	0	0	1	$A\,\overline{B}\,\overline{C}\,D$
0	0	1	0	$\overline{A}\overline{B}C\overline{D}$	1	0	1	0	$A\,\overline{B}C\overline{D}$
0	0	1	1	$\overline{A}\overline{B}CD$	1	0	1	1	$A\,\overline{B}CD$
0	1	0	0	$\overline{A}\,B\overline{C}\,\overline{D}$	1	1	0	0	$A\,B\overline{C}\,\overline{D}$
0	1	0	1	$\overline{A}\,B\overline{C}\,D$	1	1	0	1	$A\,B\overline{C}\,D$
0	1	1	0	$\overline{A}\,BC\overline{D}$	1	1	1	0	$A\,BC\overline{D}$
0	1	1	1	$\overline{A}\,BCD$	1	1	1	1	$A\,BCD$



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

Mapping Four variable Karnaugh Map - An Example

Map the SOP expression Y =

$$\overline{A}B\overline{C}\overline{D} + A\overline{B}\overline{C}\overline{D} + \overline{A}\overline{B}\overline{C}D + \overline{A}B\overline{C}D + AB\overline{C}D$$
$$+ A\overline{B}\overline{C}D + \overline{A}BCD + A\overline{B}CD + ABCD$$

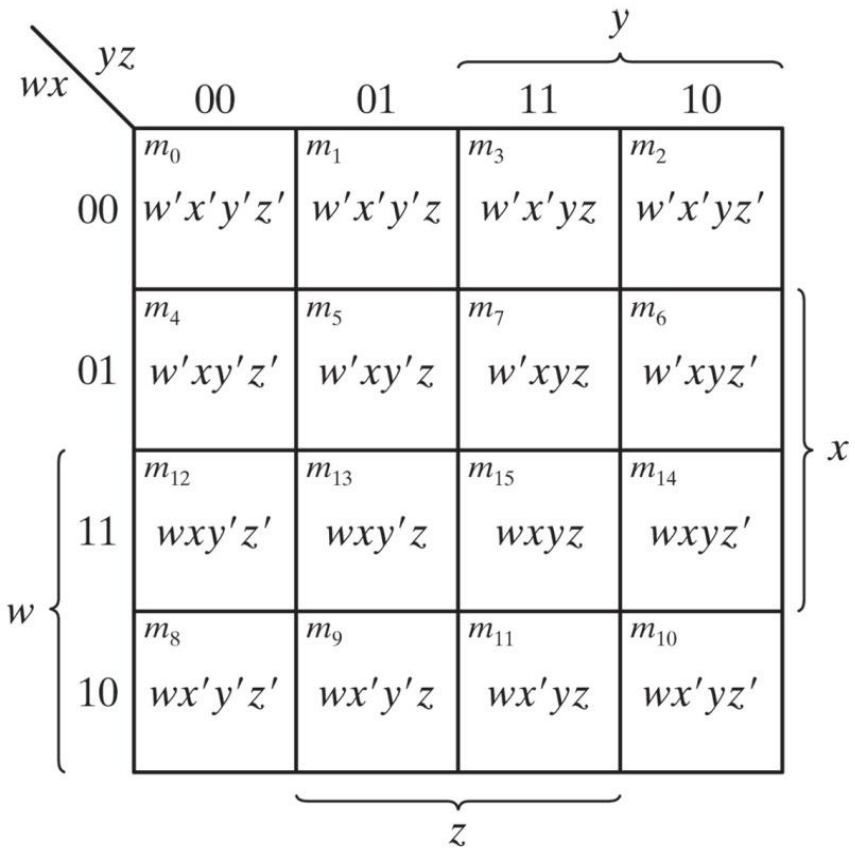
A	B	C	D	Y
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	1

	$\overline{C}\overline{D}$	$\overline{C}D$	CD	$C\overline{D}$
$\overline{A}\overline{B}$	0	1	0	0
$\overline{A}B$	1	1	1	0
AB	0	1	1	0
$A\overline{B}$	1	1	1	0

Four variable Karnaugh Map
- Another way of representation

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)

Karnaugh Map
- Mapping a nonstandard SOP expression

• A Boolean expression must first be in standard form before you use a Karnaugh map.

• Numerical expansion is probably the most efficient approach to convert the nonstandard SOP expression to the standard SOP expression

Karnaugh Map - Mapping a nonstandard SOP expression -An example

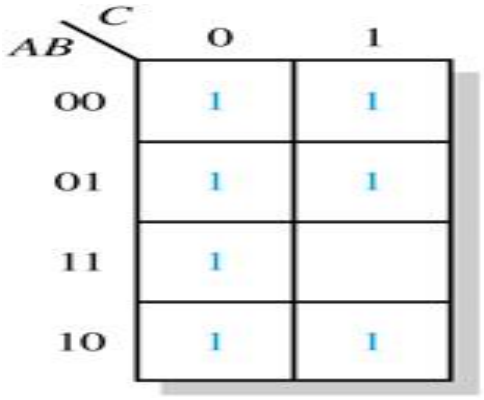
Example: Map the following expression on a Karnaugh map: $\overline{A} + A\overline{B} + AB\overline{C}$

Solution: Expand the terms numerically as follows:

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

000	100	110
001	101	
010		
011		

Each of the resulting binary values is mapped by placing a 1 in the appropriate cell of a 3 variables Karnaugh map



Karnaugh Map
- Looping

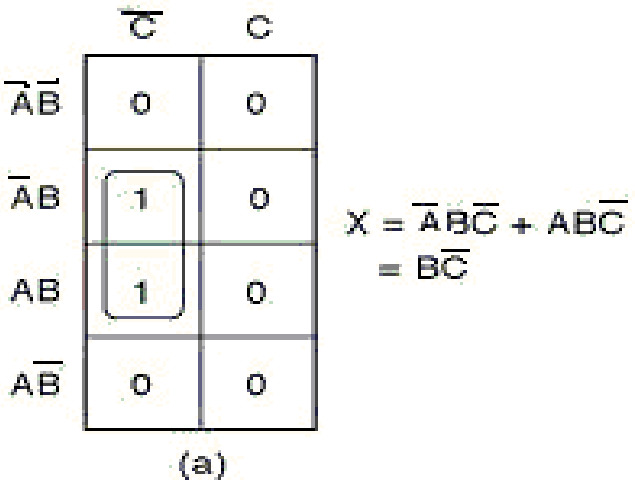
• The expression for output x can be simplified by properly combining those squares in the K- map which contain 1s.
•The process for combining these 1s is called looping

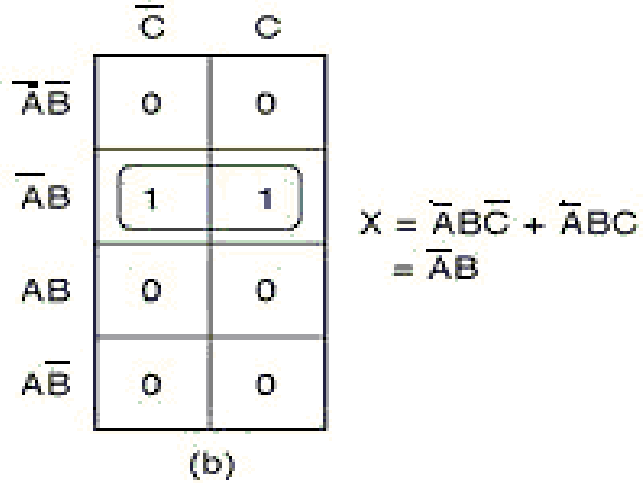
• Looping groups of two - Pairs

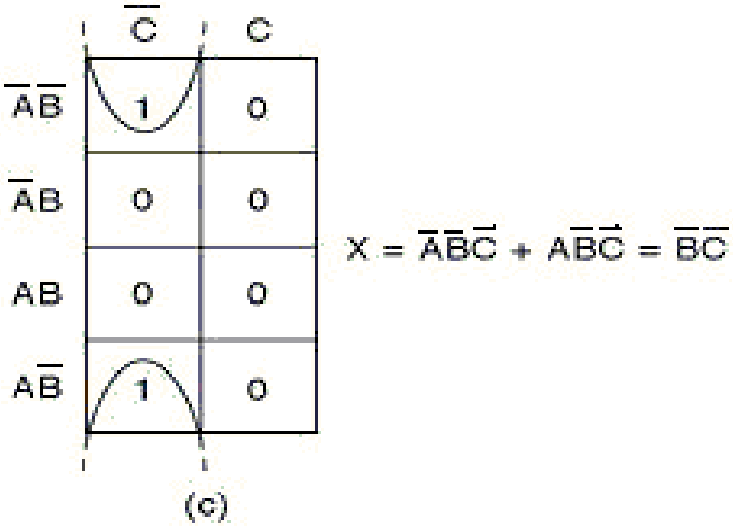
• Looping groups of four - Quads

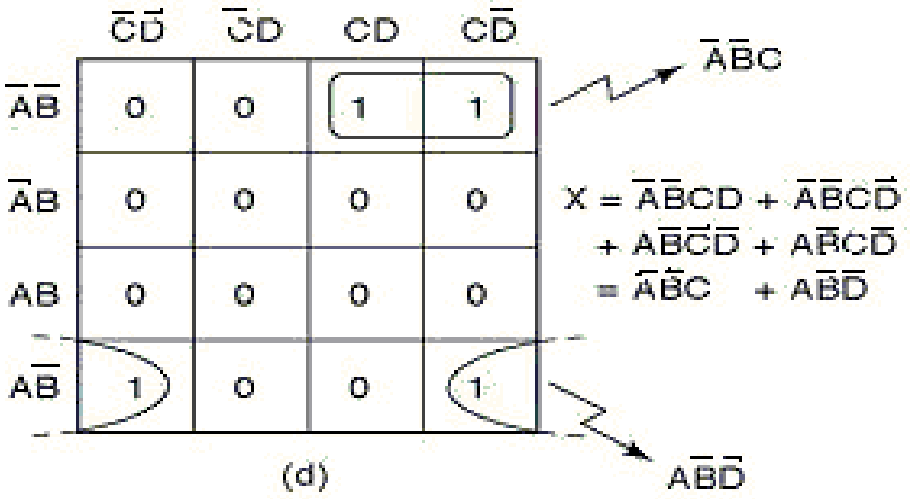
• Looping groups of eight - Octets

Karnaugh Map
- Looping groups of two (pairs)

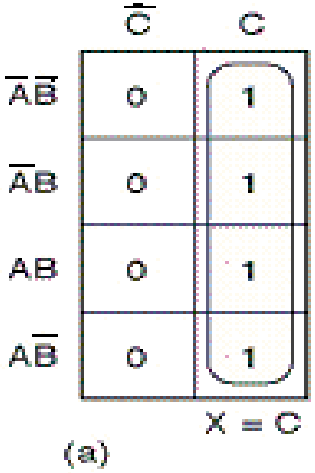


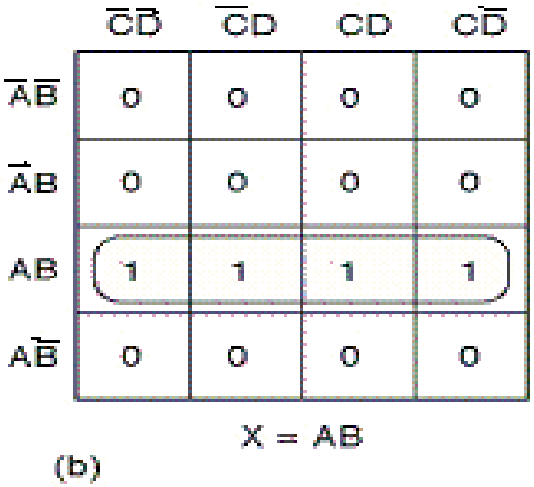


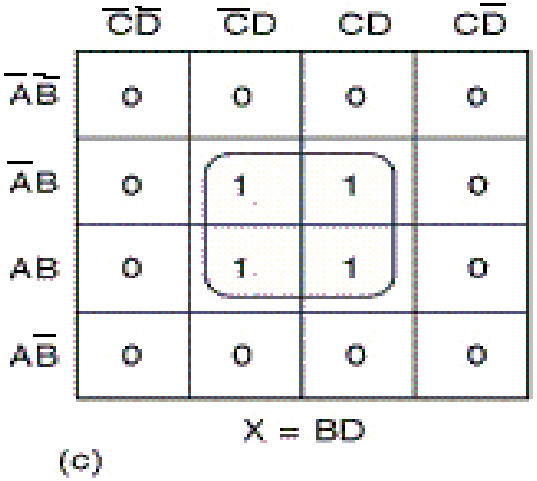


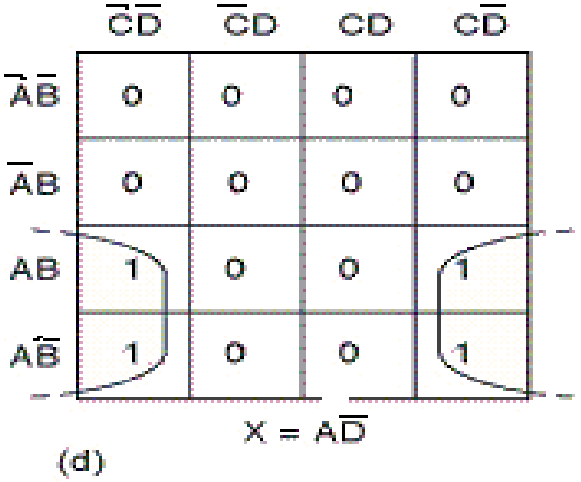


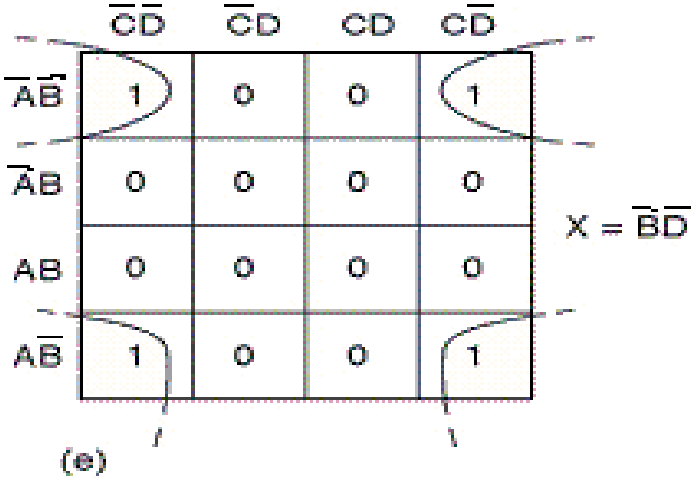
Karnaugh Map
- Looping groups of four (quads)



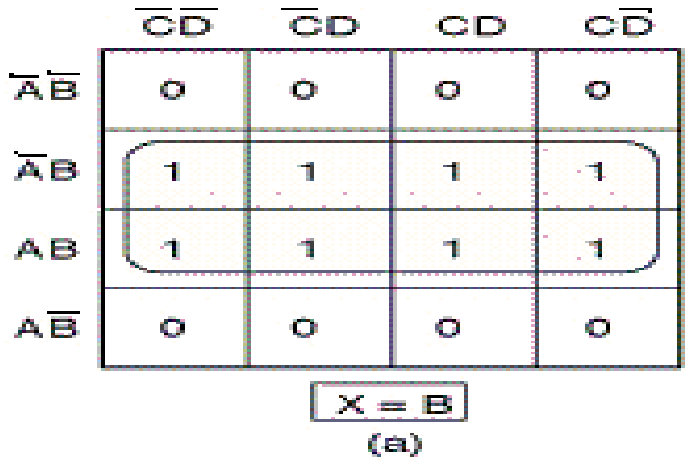


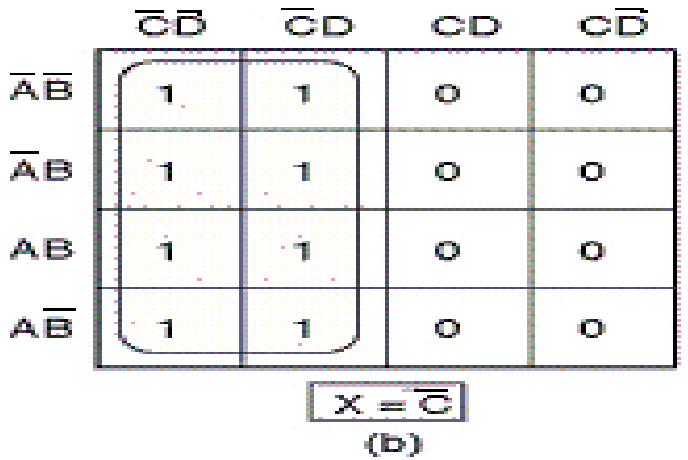


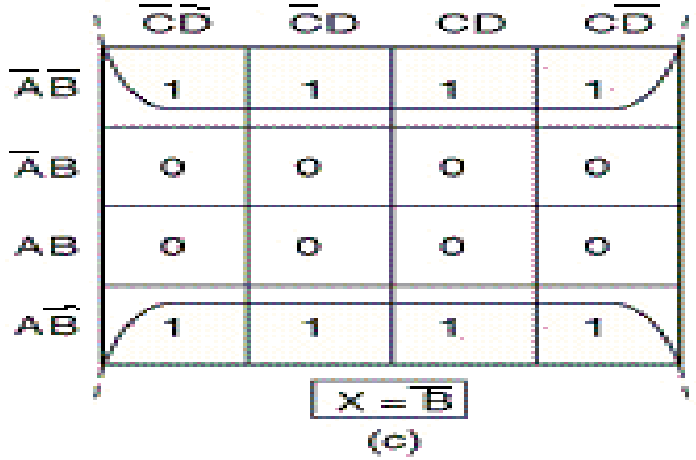


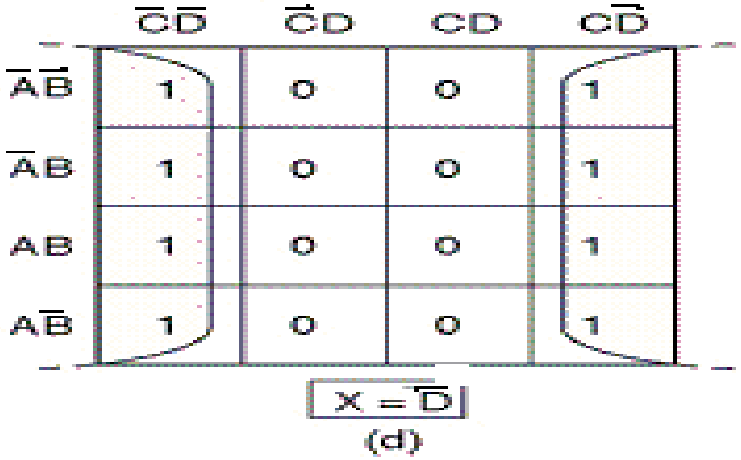


Karnaugh Map
- Looping groups of eight (octets)









Karnaugh Map
- Steps in simplification process

Steps to follow in using the K-map method for simplifying a Boolean expression :

1. Construct the K-map and place 1s in those squares corresponding to the 1s in the truth table. Place 0s in the other squares.

2. Examine the map for adjacent 1s and loop those 1s which are not adjacent to any other 1s. These are called **isolated 1.**

3. Next, look for those 1s which are adjacent to only one other. Loop any pair containing such a 1.

4. **Loop any octet** even if it contains some 1s that already been looped.

5. **Loop any quad** that contains one or more 1s that have not already been looped, making sure to use the minimum number of loops.

6. **Loop any pairs** necessary to include any 1s that have not yet been looped, making sure to use the minimum of loops.

7. Form the OR sum of all the product terms generated by each loop.

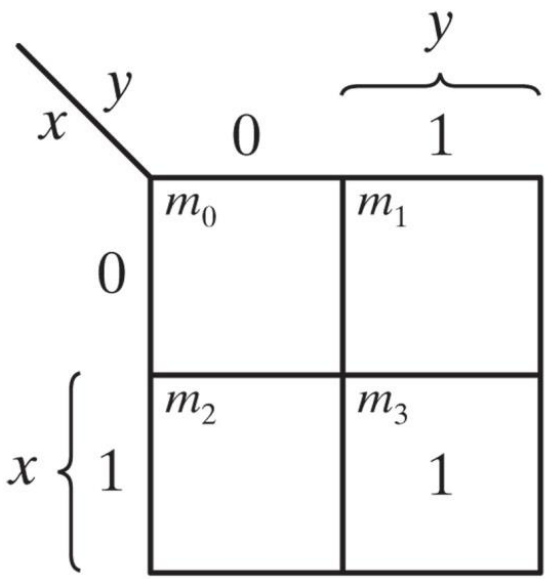
Karnaugh Map
- Simplification process

Note that larger loop of 1s eliminate more variables.

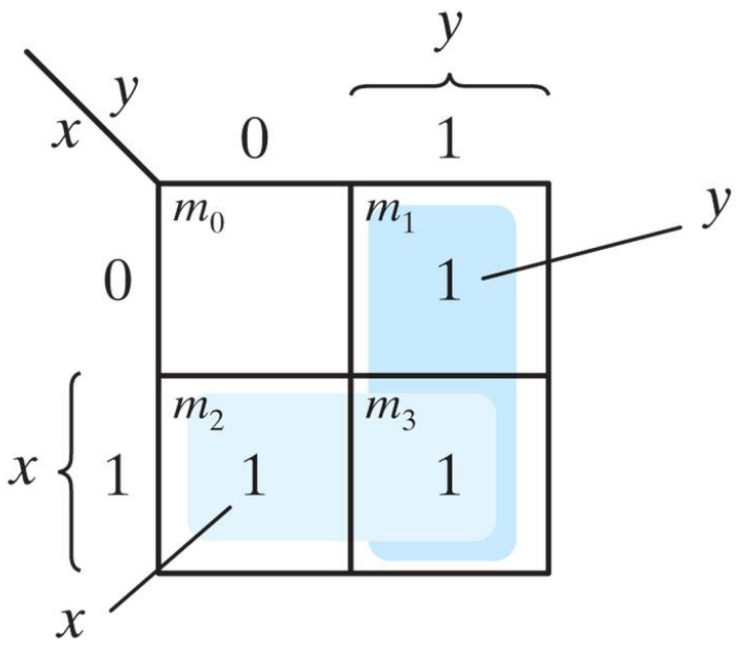
1. loop of 2 eliminates one variable
2. loop of 4 eliminates two variables
3. loop of 8 eliminates three variables

• Looping a pair of adjacent 1s in a K-map eliminates the variable that appears in complemented and uncomplemented form.

Two variable Karnaugh Map
- Example in simplification process



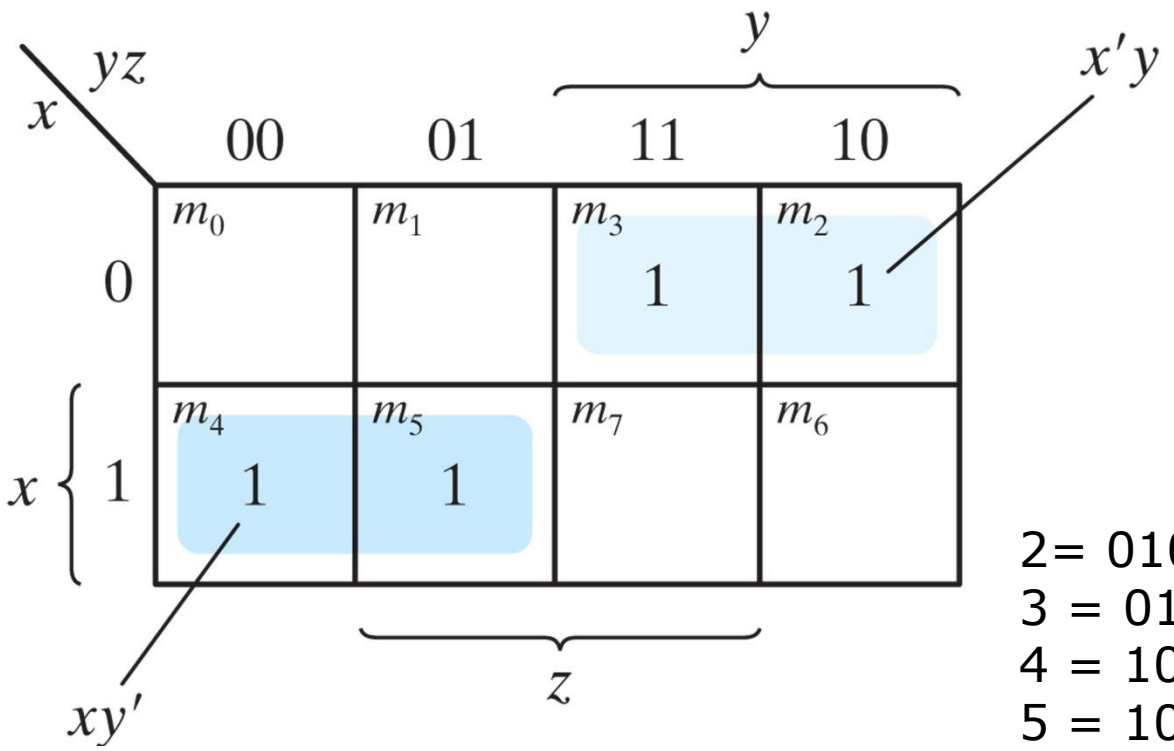
(a) xy



(b) $x + y$

Three variable Karnaugh Map
- Examples in simplification process

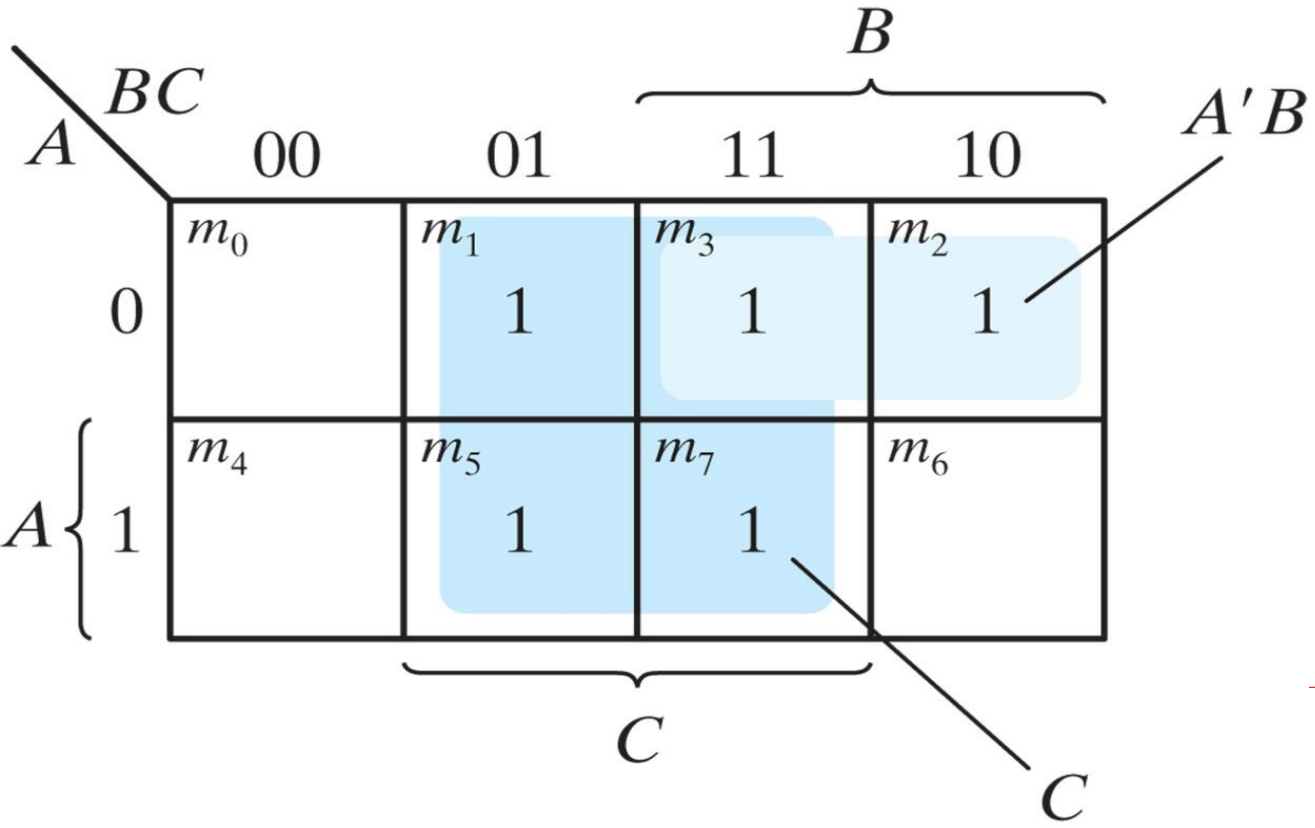
Example 1: $F(x, y, z) = \Sigma m(2, 3, 4, 5) = x'y + xy'$



2= 010
3 = 011
4 = 100
5 = 101

Three variable Karnaugh Map
- Examples in simplification process

Example 2: F = A′C + A′B + AB′C + BC = C + A′B



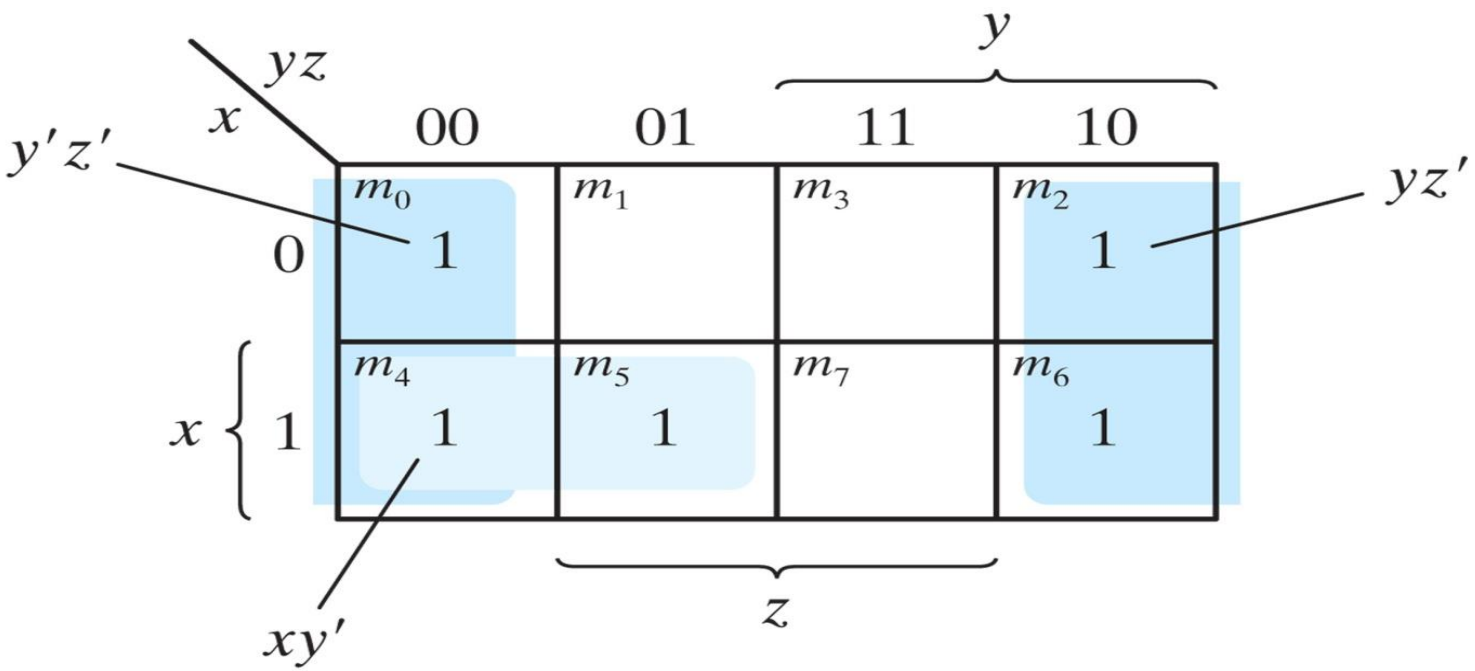
A′C = A′BC+ A′B′C
= 011 + 001

A′B = A′BC + A′BC′
= 011 + 010

BC = A′BC + ABC
= 011 + 111

Three variable Karnaugh Map
- Examples in simplification process

Example 3: $F(x, y, z) = \Sigma m(0, 2, 4, 5, 6) = z' + xy'$



0 = 000
= X′Y′Z′

2 =010
= X′YZ′

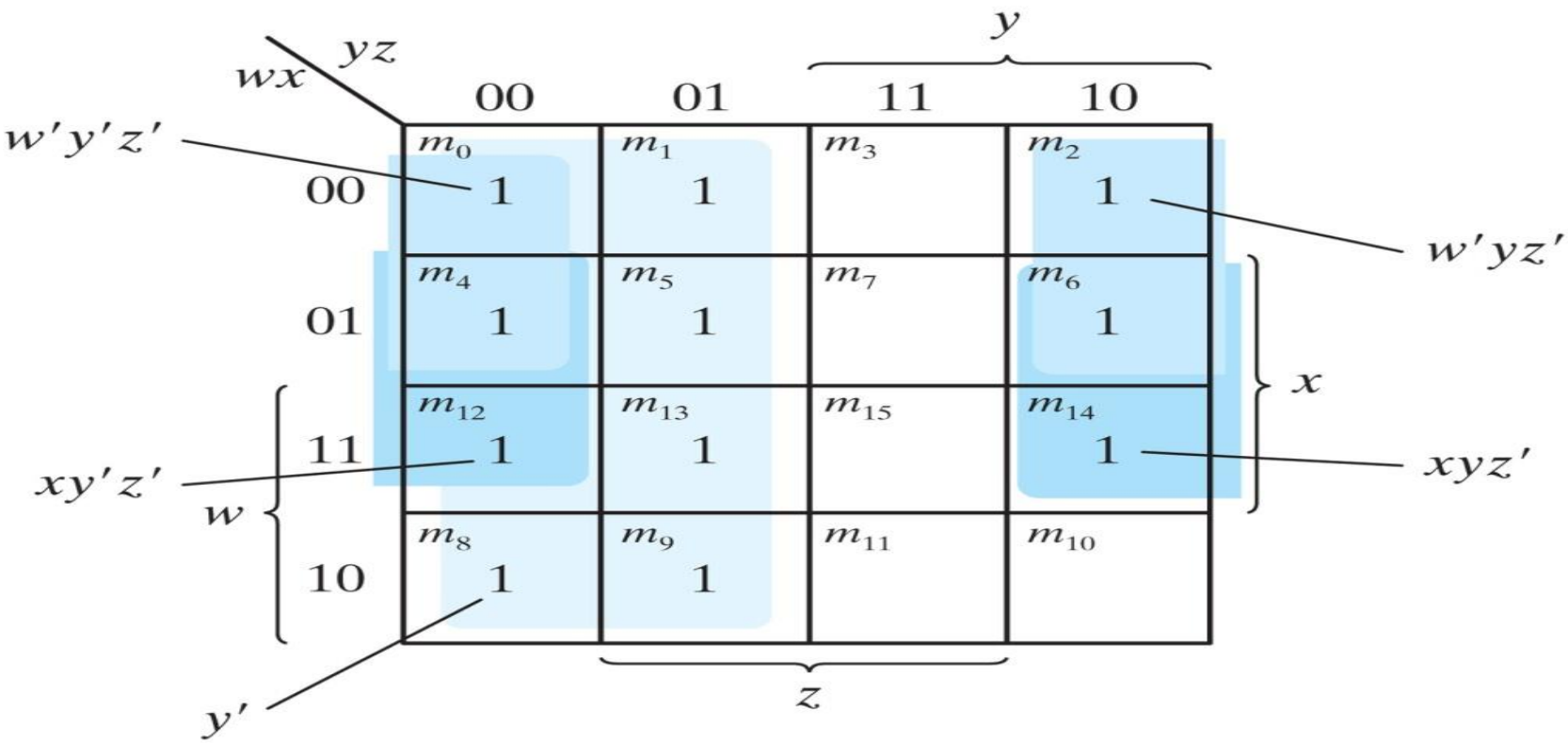
4 = 100
= XY′Z′

5 =101
= XY′Z

6 =110
= XYZ′

Four variable Karnaugh Map
- Examples in simplification process

Example 1: $F(w, x, y, z) = \Sigma m(0, 1, 2, 4, 5, 6, 8, 9, 12, 13, 14) = y' + w'z' + xz'$

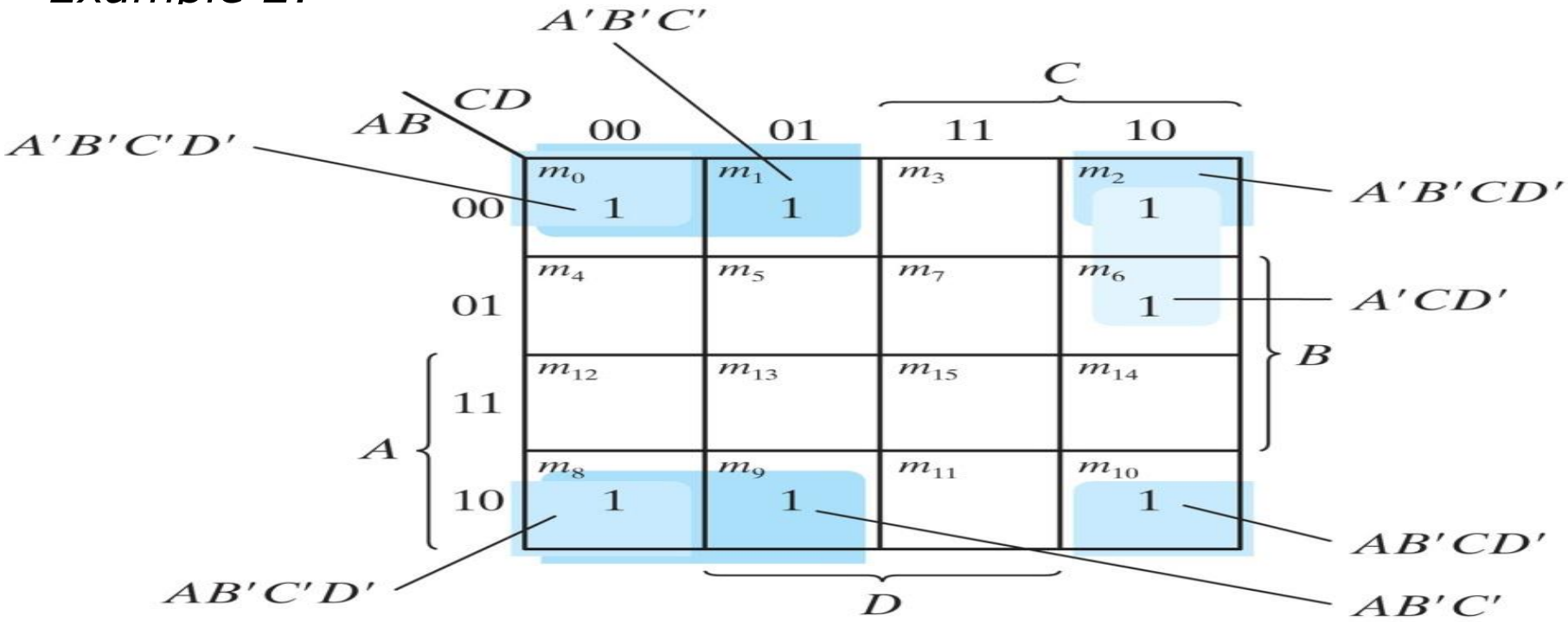


Note: $w'y'z' + w'yz' = w'z'$

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

Four variable Karnaugh Map
- Examples in simplification process

Example 2: $A'B'C' + B'CD' + A'BCD' + AB'C' = B'D' + B'C' + A'CD'$

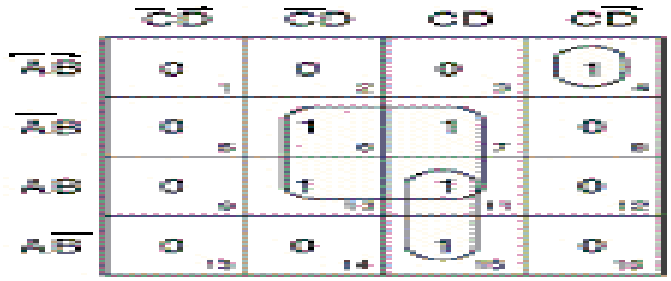


Note: $A'B'C'D' + A'B'CD' = A'B'D'$
$AB'C'D' + AB'CD' = AB'D'$
$A'B'D' + AB'D' = B'D'$
$A'B'C' + AB'C' = B'C'$

Example 2: $A'B'C' + B'CD' + A'BCD' + AB'C' = B'D' + B'C' + A'CD'$

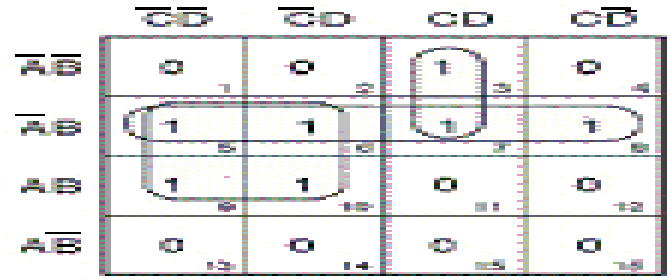
Four variable Karnaugh Map

- More Examples in simplification process



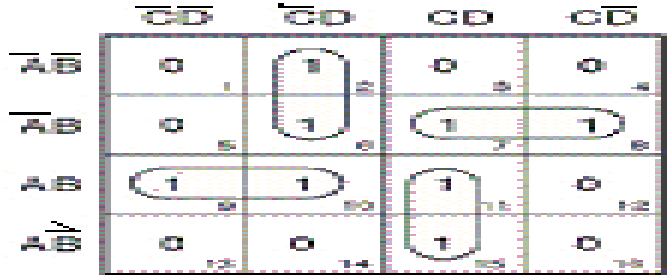
$$X = \underbrace{\overline{A}B\overline{C}\overline{D}}_{\text{loop 4}} + \underbrace{ACD}_{\text{loop 11, 15}} + \underbrace{BD}_{\text{loop 6, 7, 10, 11}}$$

(a)



$$X = \underbrace{\overline{A}B}_{\text{loop 5, 6, 7, 8}} + \underbrace{B\overline{C}}_{\text{loop 5, 6, 9, 10}} + \underbrace{\overline{A}CD}_{\text{loop 3,7}}$$

(b)



$$X = \underbrace{AB\overline{C}}_{9,\ 10} + \underbrace{\overline{A}\overline{C}D}_{2,\ 6} + \underbrace{\overline{A}BC}_{7,\ 8} + \underbrace{ACD}_{11,\ 15}$$

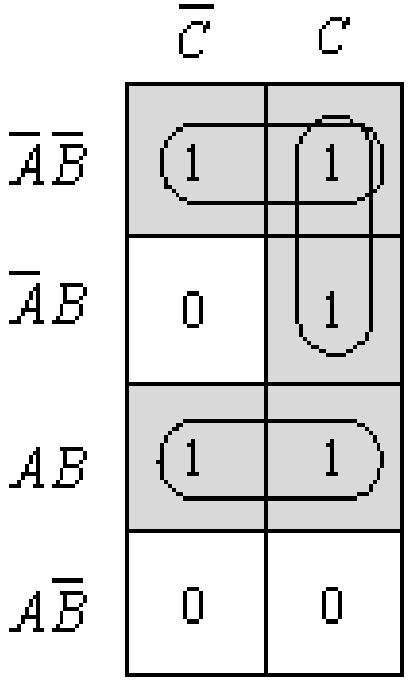
(c)

Karnaugh Map
- SOP minimization Problem

Use a *K*–map to minimize the following SOP Boolean expression

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

Put the 1s for the terms in their respective boxes and apply all possible looping



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

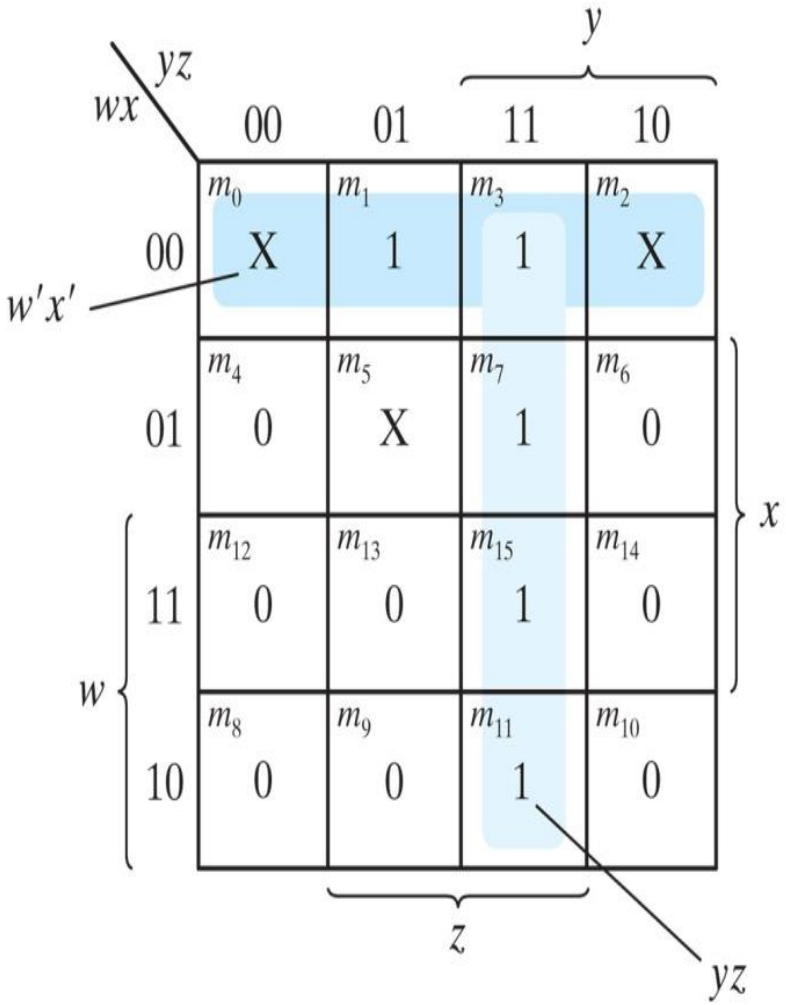
Karnaugh Map
- Don’t Care Conditions

• Some logic circuits can be designed so that there are certain input conditions for which there are no specific output levels, usually because these input conditions will never occur.

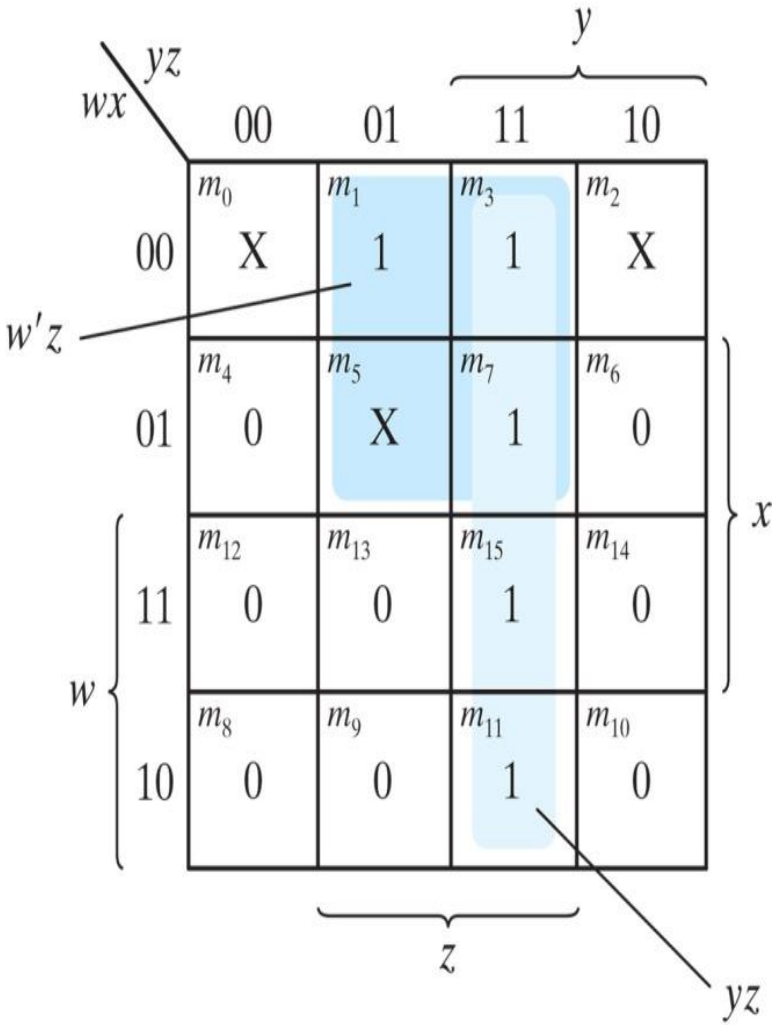
• There will be certain combinations of input level where we “don’t care” whether the output is HIGH or LOW.

• Use the “don’t care” inputs (X) in the looping process if needed.

Karnaugh Map- Don't Care Conditions - Examples



(a) $F = yz + w'x'$



(b) $F = yz + w'z$

Karnaugh Map - Don’t Care Conditions - Problem 1

Design a combinational logic circuit that **controls an elevator** door in a three-storey building. There are 4 input conditions. M is a logic signal that indicates when the elevator is moving (M=1) or stopped (M=0). F1, F2 and F3 are floor indicator signals that are normally LOW, and they go HIGH only when the elevator is positioned at the level of that particular floor.

Karnaugh Map - Don't Care Conditions - Problem 1

This problem refers to the circuit that controls an elevator door in a 3 story building.

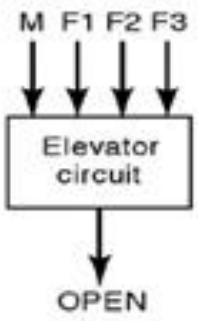
There are 4 inputs:

M: Signal – indicates whether the elevator is moving

M = 1 (Elevator is moving)

M = 0 (Elevator is not moving)

F1, F2, F3: Floor indicators



(a)

M	F1	F2	F3	OPEN
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	X
0	1	0	0	1
0	1	0	1	X
0	1	1	0	X
0	1	1	1	X
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	X
1	1	0	0	0
1	1	0	1	X
1	1	1	0	X
1	1	1	1	X

(b)

	$\bar{F}2\bar{F}3$	$\bar{F}2F3$	$F2F3$	$F2\bar{F}3$
$M\bar{F}1$	0	1	X	1
$\bar{M}\bar{F}1$	1	X	X	X
$\bar{M}F1$	0	X	X	X
$M\bar{F}1$	0	0	X	0

(c)

	$\bar{F}2\bar{F}3$	$\bar{F}2F3$	$F2F3$	$F2\bar{F}3$
$M\bar{F}1$	0	1	1	1
$\bar{M}\bar{F}1$	1	1	1	1
$\bar{M}F1$	0	0	0	0
$M\bar{F}1$	0	0	0	0

$OPEN = \bar{M}(F1 + F2 + F3)$

(d)

Karnaugh Map - Don't Care Conditions - Problem 2

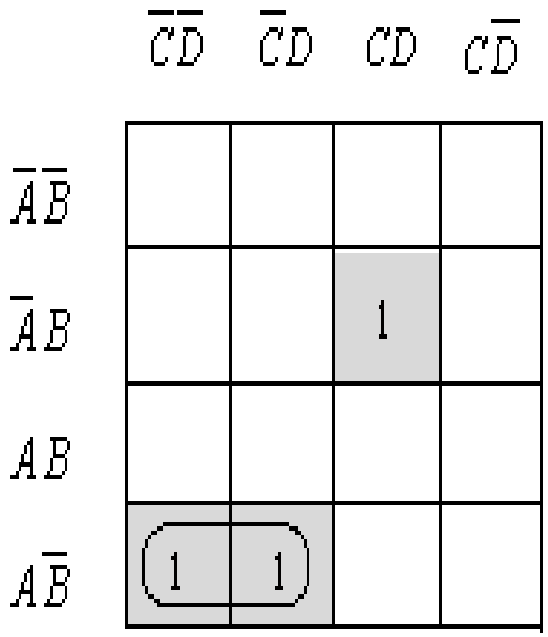
In designing a problem involving BCD code :

- There are six invalid combinations: 1010, 1011, 1100, 1101, 1110 and 1111.
- Since these disallowed states will never occur in an application involving the BCD code, they can be treated as don't care terms with respect to their effect on the output.
- That is for don't care terms either a 1 or a 0 may be assigned to the output; it really does not matter since they will never occur.

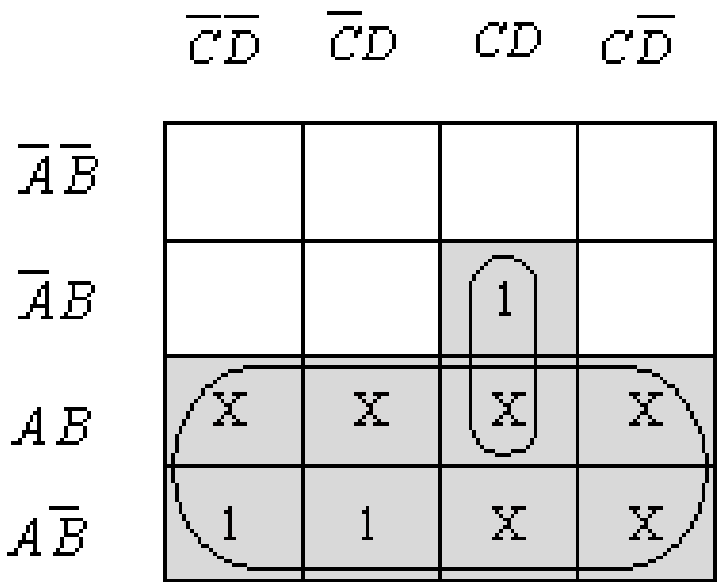
Design a combinational circuit for detecting the decimal digits 7,8, and 9 in 8421 BCD code.

A	B	C	D	Output Y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	X
1	0	1	1	X
1	1	0	0	X
1	1	0	1	X
1	1	1	0	X
1	1	1	1	X

Karnaugh Map - Don’t Care Conditions - Problem 2



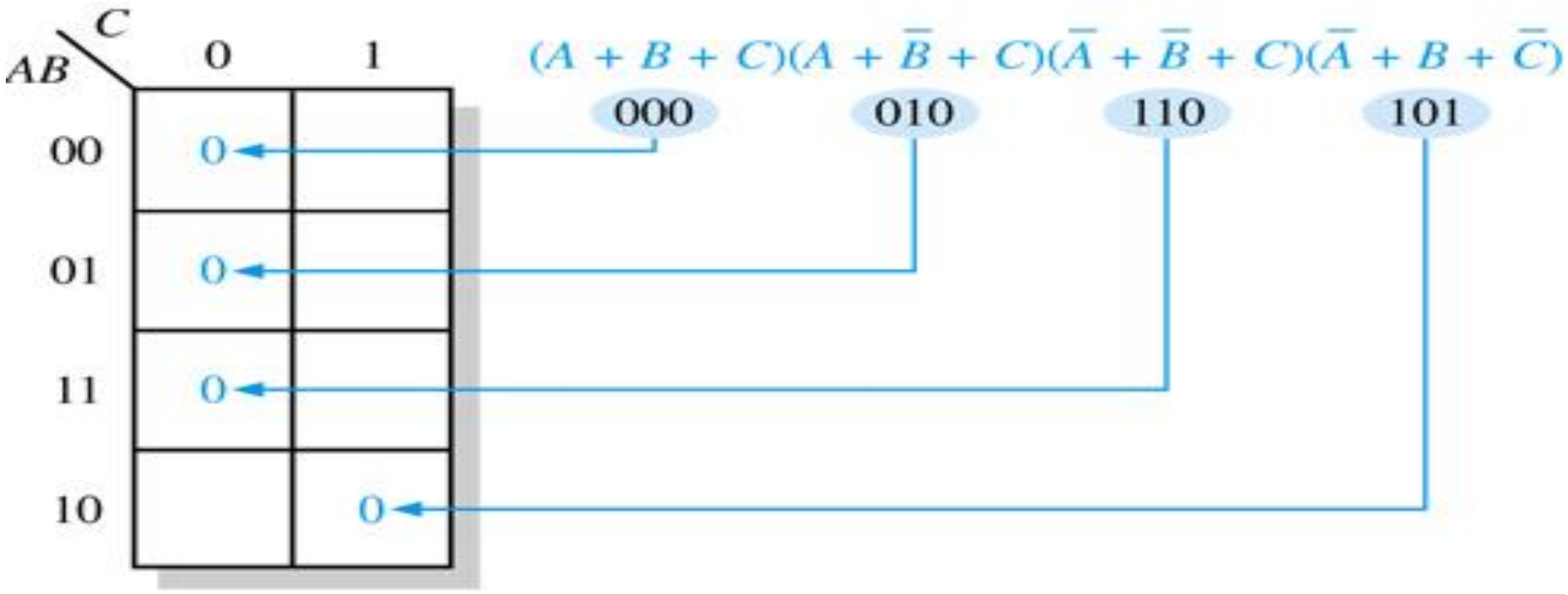
Without don’t cares $Y = A\overline{B}\,\overline{C} + \overline{A}BCD$



With don’t cares $Y = A + BCD$

Karnaugh Map
- Mapping POS expression

- For a POS expression in standard form, a 0 is placed on the Karnaugh map for each sum term in the expression.
- Each 0 is placed in a cell corresponding to the value of a sum term.
- The cells that do not have a 0 are the cells for which the expression is 1.



Karnaugh Map
- Simplifying POS expression

- The process for minimizing a POS expression is basically the same as for an SOP expression except that grouping 0s to produce minimum sum terms instead of grouping 1s to produce minimum product terms.
- The steps for grouping the 0s are the same as those for grouping the 1s.

Karnaugh Map - Simplifying POS expression - An example

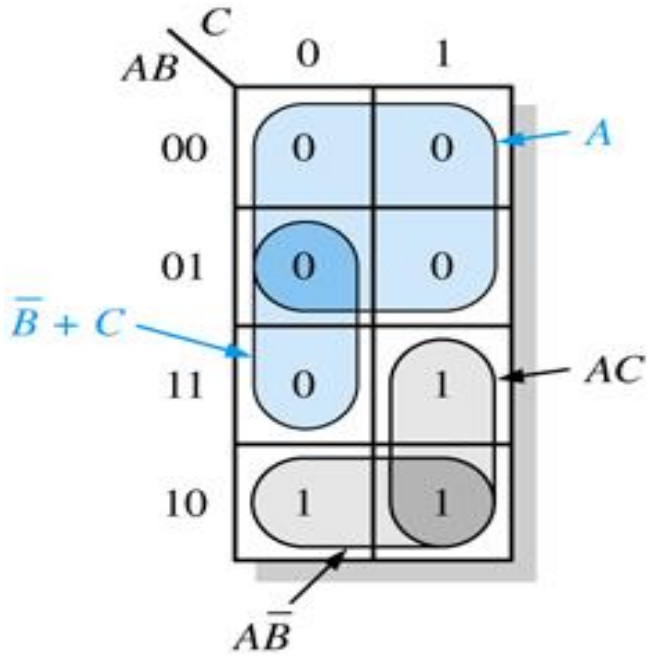
Example: Use a Karnaugh map to minimize the following standard POS expression:

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

Solution: The combination of binary values of the expression are

$$(0+0+0)(0+0+1)(0+1+0)(0+1+1)(1+1+0)$$

The standard POS expression is mapped and the cells are grouped as shown.



The sum term for each blue group is shown in the figure and the resulting minimum POS expression is

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

Grouping the 1s as shown by gray areas yields an SOP expression that is equivalent to grouping the 0s

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

Design of Combinational Logic Circuits – Part 2 of 2

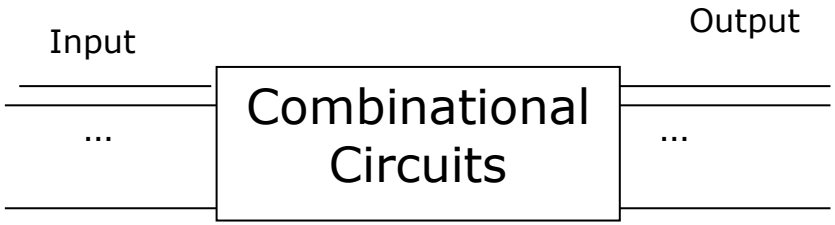
- **Design of Combinational Logic Circuit from the given problem statement/ Boolean expression/Truth Table**
- **Universality of NAND/NOR gates**

Design of Combinational Logic Circuit

Combinational Logic Circuits

• Combinational logic is defined as that class of digital circuits where at any given time, the state of all outputs depends only upon the values of the inputs at that time and not upon any previous states.

• A combinational logic circuit may be regarded as a black box having N input lines and P output lines, each of which carries a digital function which can have only two possible values, commonly denoted as 0 or 1



Combinational Logic Circuit
- From the given boolean expression

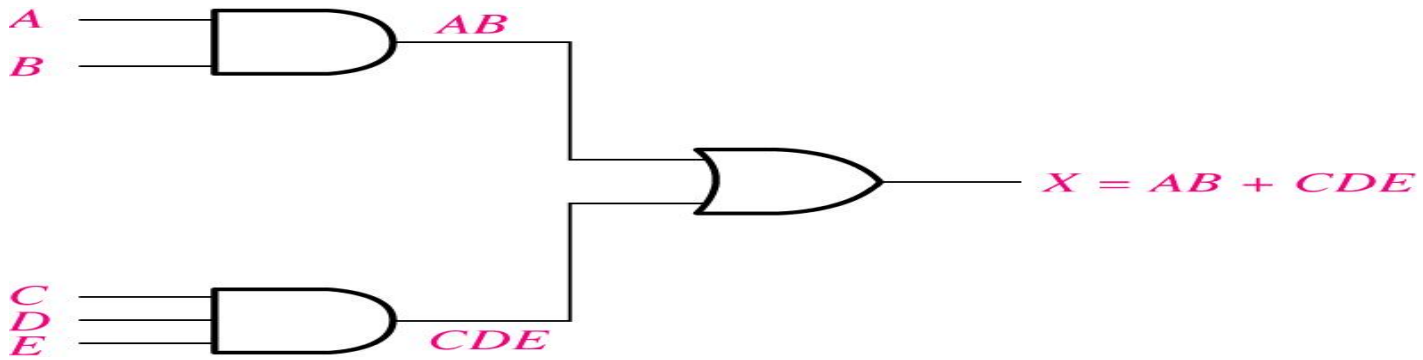
From the following Boolean Expression :

$$X = AB + CDE$$

AND

OR

The AND operations forming the two individual terms, AB and CDE , must be performed before the terms can be ORed.



Logic circuit for X = AB + CDE.

Design of Combinational Logic Circuit using AND-OR gates
- Steps involved

- ✓ Set up the truth table.
- ✓ Write the AND term for each case where the output is a 1.
- ✓ Write the sum of products expression for the output.
- ✓ Simplify the output expression using Boolean algebra rules, laws, theorems / using Karnaugh map method.
- ✓ Implement the circuit for the final expression.

Design of Combinational Logic Circuit
- From the given problem statement

Problem Statement:

Design a three input logic circuit whose output will be high when a majority of the inputs are high (Use AND-OR gates)

Solution:

***Step* 1: Set up truth table**

A	*B*	*C*	*Y*
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

Design of Combinational Logic Circuit
- From the given problem statement

***Step* 2: Write the AND term for each case where the output is 1**

$$\overline{A}BC,\ A\overline{B}C,\ AB\overline{C},\ and\ ABC$$

***Step* 3: Simplify the output expression using Boolean algebra/K-map method**

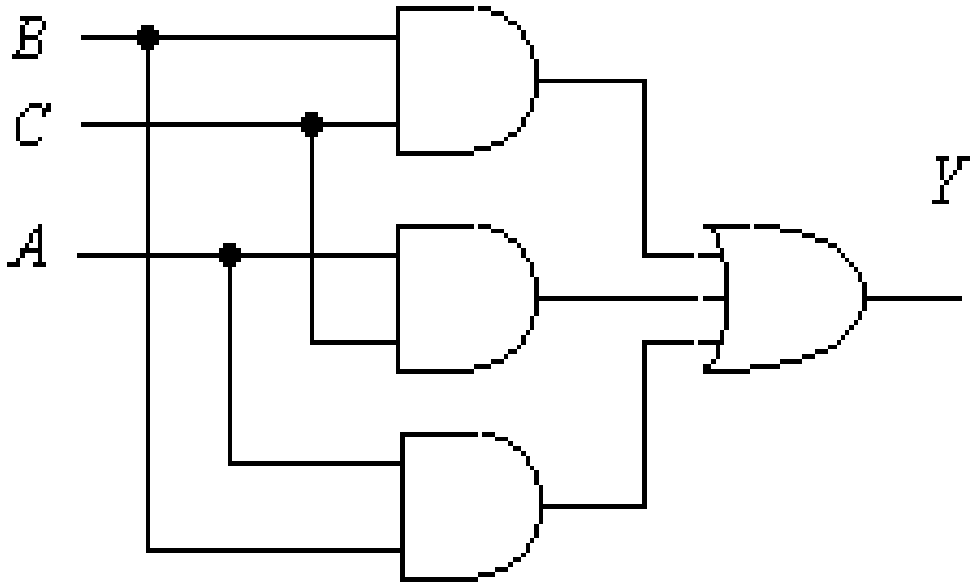
$$Y = \overline{A}BC + ABC + A\overline{B}C + ABC + AB\overline{C} + ABC$$

$$Y = BC(\overline{A} + A) + AC(\overline{B} + B) + AB(\overline{C} + C)$$

$$Y = BC + AC + AB$$

Design of Combinational Logic Circuit
- From the given problem statement

***Step* 4: Implement the expression using AND-OR gates**



$$Y = BC + AC + AB$$

Design of Combinational Logic Circuit
- From the given truth table

Exercise

Design a logic circuit for following truth table

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

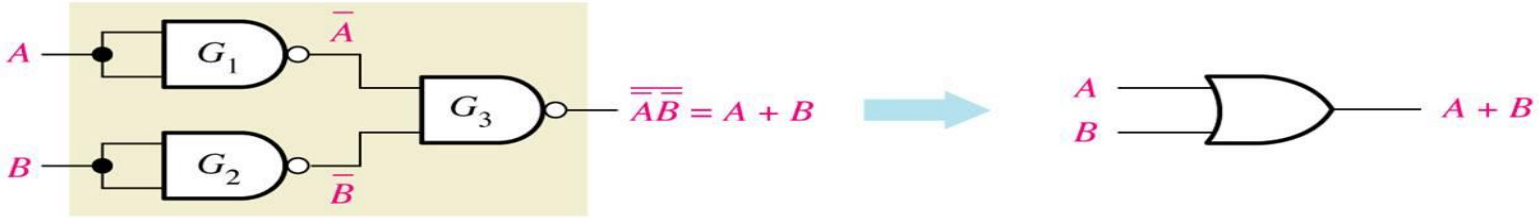
Universality of NAND/NOR Gates

Universality of NAND gates

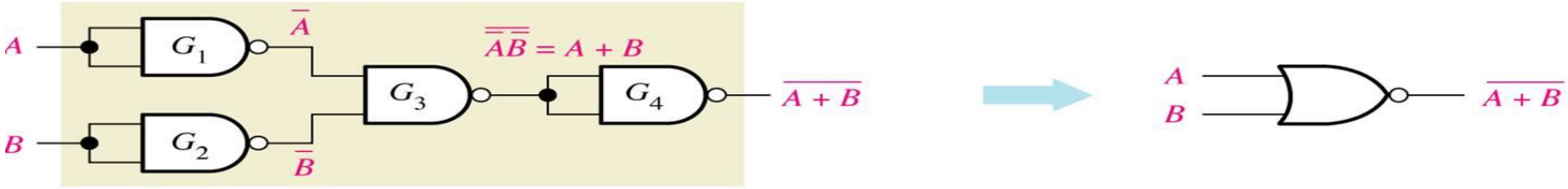
- Construction of NOT, AND, OR and NOR gates using only NAND gates

(a) A NAND gate used as an inverter

(b) Two NAND gates used as an AND gate



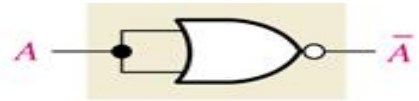
(c) Three NAND gates used as an OR gate



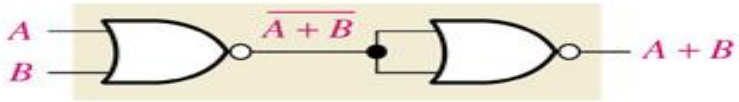
(d) Four NAND gates used as a NOR gate

Universality of NOR gates

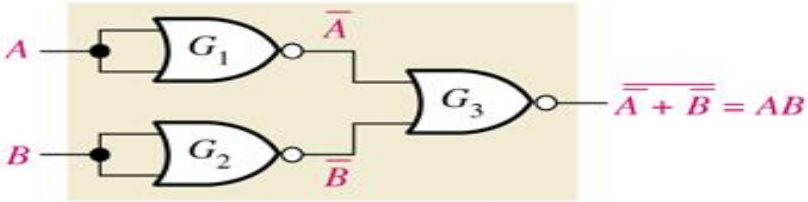
- Construction of NOT, OR, AND and NAND gates using only NAND gates



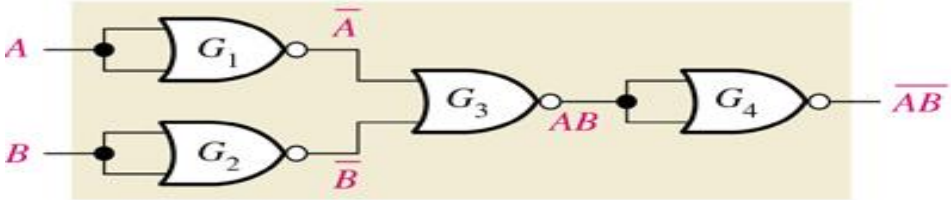
(a) A NOR gate used as an inverter



(b) Two NOR gates used as an OR gate



(c) Three NOR gates used as an AND gate



(d) Four NOR gates used as a NAND gate

Design of Combinational Logic Circuit
- Using only NAND gates

NAND gate can function as either a NAND or a negative-OR because, by DeMorgan's theorem,

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

NAND — Negative-OR

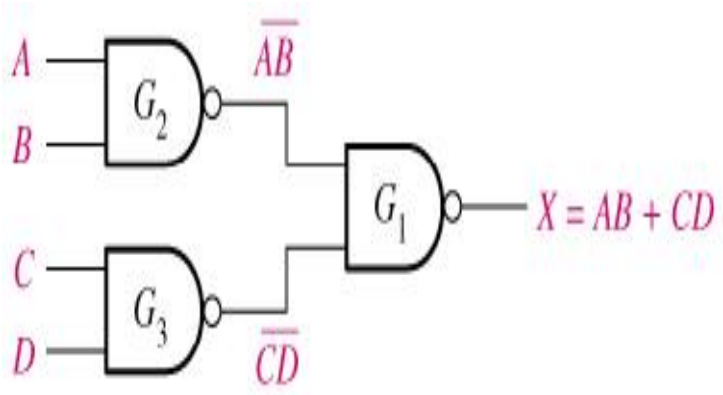
Consider the NAND logic in figure below. The output expression is developed in the following steps:

$$X = \overline{(\overline{AB})(\overline{CD})} = \overline{(\overline{A} + \overline{B})(\overline{C} + \overline{D})}$$

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

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

$$= AB + CD$$



NAND logic for X = AB + CD

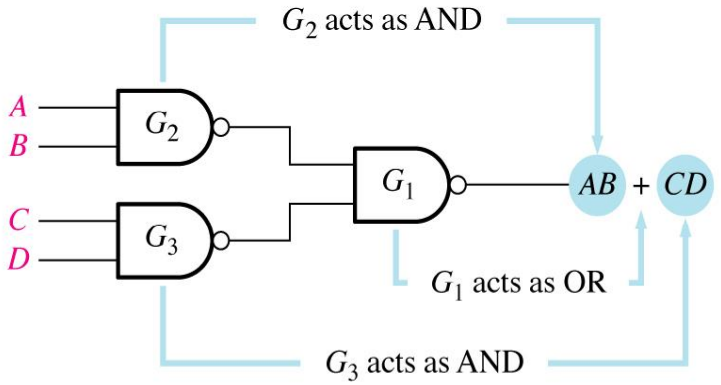
Design of Combinational Logic Circuit - Using only NAND gates

Steps involved:

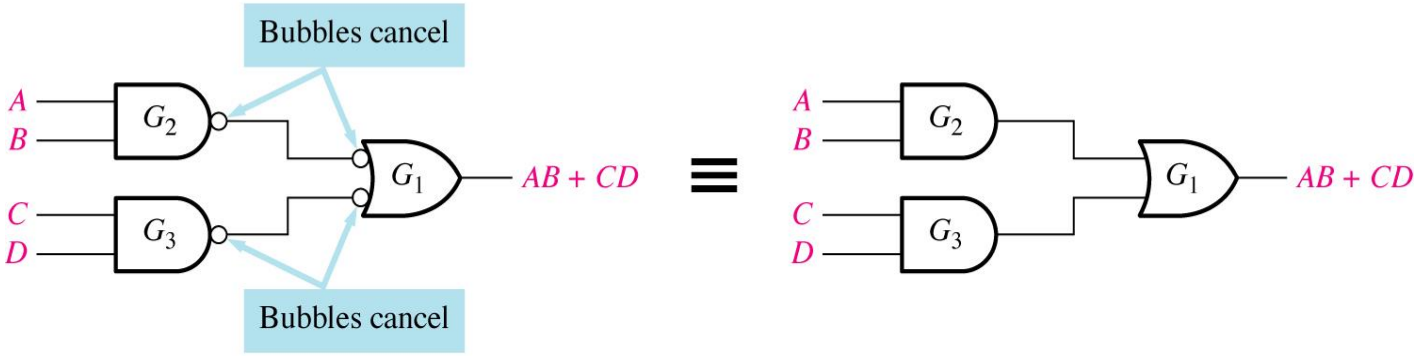
- Simplify the boolean expression in SOP form
- Double negate the expression
- Apply DeMorgan's theorem to the inner negation to get the NAND-NAND form.

Combinational logic circuit using NAND gates - Development of AND-OR equivalent

Development of the AND-OR equivalent of the circuit



(a) Original NAND logic diagram showing effective gate operation relative to the output expression



(b) Equivalent NAND/Negative-OR logic diagram

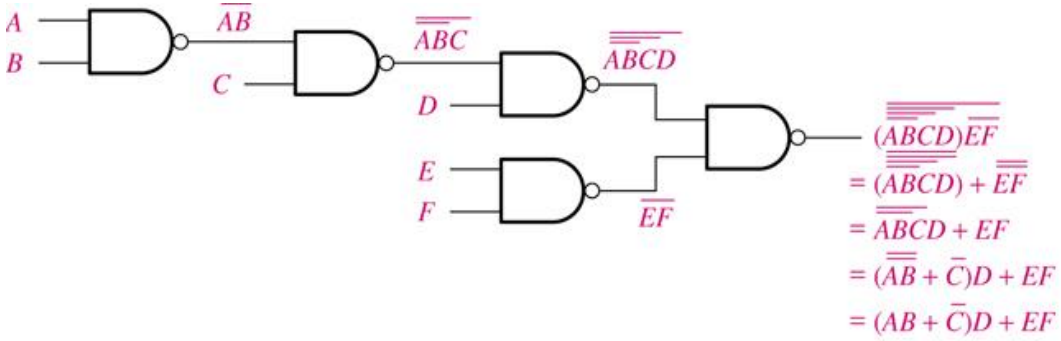
(c) AND-OR equivalent

Simplification of Combinational logic circuit using NAND gates - Using Dual symbols

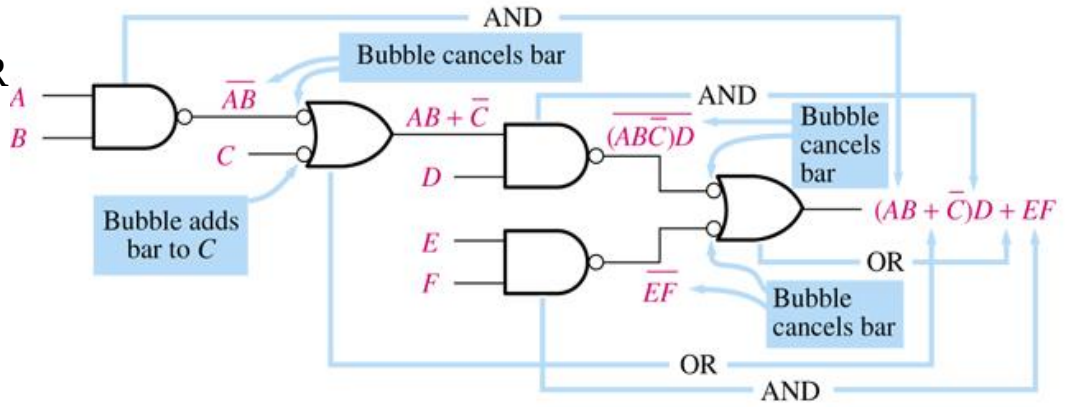
Illustration of the use of the appropriate dual symbols in a NAND logic diagram.

All logic diagrams using NAND gates should be drawn with each gate represented by either NAND symbol or the equivalent negative-OR symbol to reflect the operation or the gate within the logic circuit.

The NAND symbol and the negative-OR symbol are called **dual symbols**.



(a) Several Boolean steps are required to arrive at final output expression.



(b) Output expression can be obtained directly from the function of each gate symbol in the diagram.

Design of Combinational Logic Circuit
- Using only NOR gates

A NOR gate can function as either a NOR or a negative-AND, as shown by DeMorgan's theorem.

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

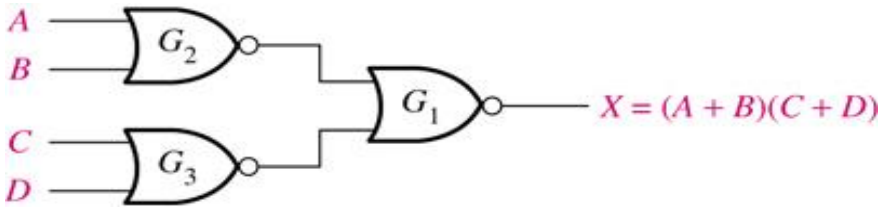
NOR — Negative-AND

Consider the NOR logic in fig. below. The output expression is developed as follows:

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

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

NOR logic for X = (A + B)(C + D)



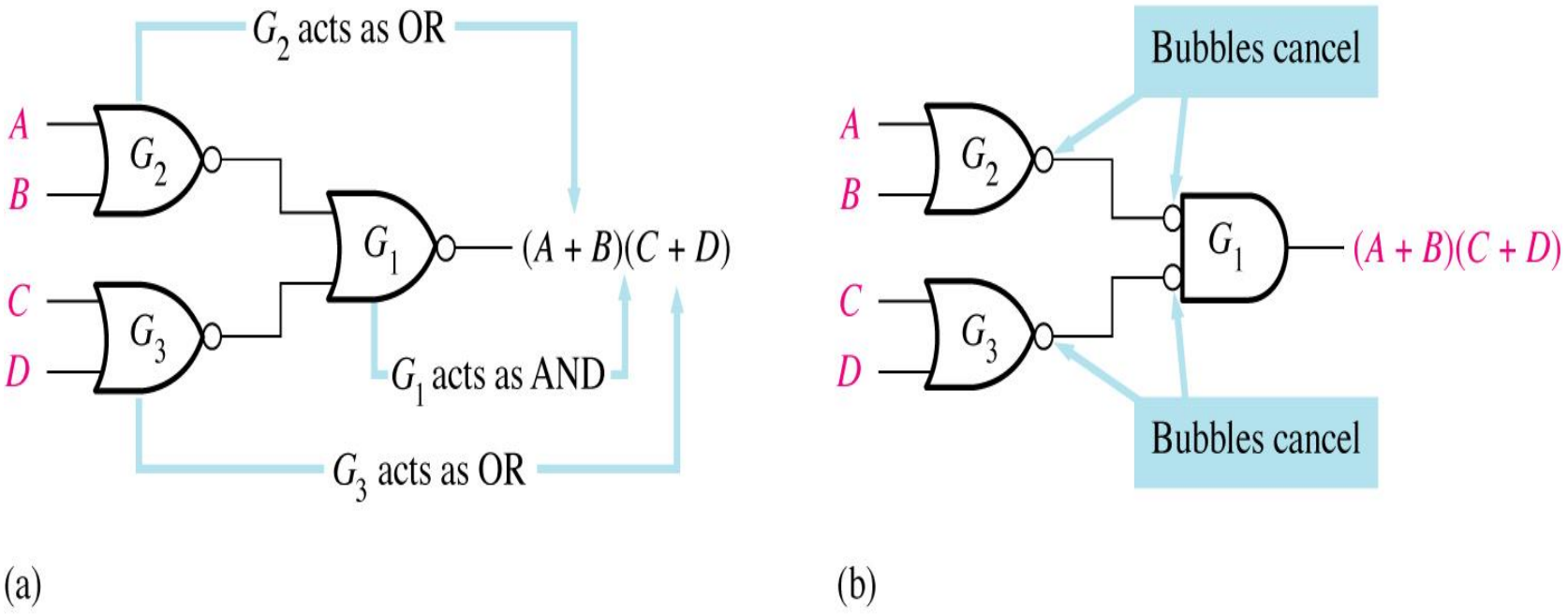
Design of Combinational Logic Circuit
- Using only NOR gates

Steps involved:

- Simplify the boolean expression in POS form
- Double negate the expression
- Apply DeMorgan's theorem to the inner negation to get the NOR-NOR form.

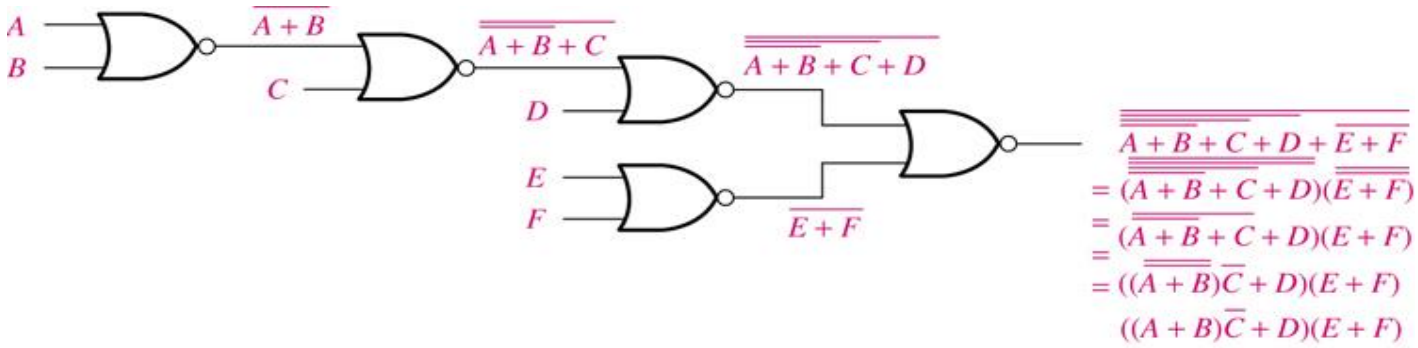
Combinational logic circuit using NOR gates - Development of OR-AND equivalent

The circuit in fig can be redrawn in part (b) with a negative-AND symbol for gate G_1

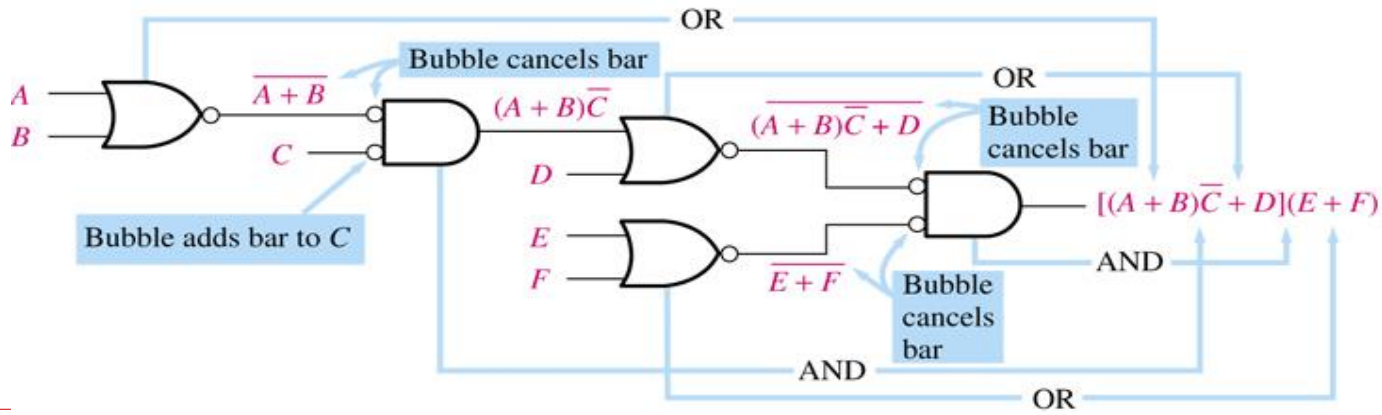


Simplification of Combinational logic circuit using NOR gates
- Using Dual symbols

Illustration of the use of the appropriate dual symbols in a NOR logic diagram.



(a) Final output expression is obtained after several Boolean steps.



(b) Output expression can be obtained directly from the function of each gate symbol in the diagram.

References

Slides adopted from the books

1. Thomas L.Floyd, “Digital Fundamentals,” 11th Edition, Prentice Hall, 2014 (ISBN10:0132737965/ISBN13:9780132737968)

- M.Morris Mano and Michael D. Ciletti, " Digital Design," 5th Edition, Pearson Education International, 2012

- Ronald J.Tocci, Neal S.Widmer, and Gregory L.Moss, "Digital Systems- Principles and Application"- 11th Edition, Pearson Education International, 2011 (ISBN: 9780135103821)