UNIT II
LOGIC GATES

Gates

- Let’s examine the processing of the following six types of gates
  - NOT
  - AND
  - OR
  - XOR
  - NAND
  - NOR
- Typically, logic diagrams are black and white, and the gates are distinguished only by their shape

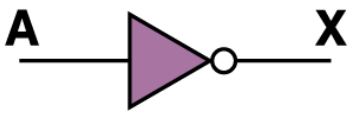
NOT Gate

- A NOT gate accepts one input value and produces one output value

Boolean Expression

$X = A'$

Logic Diagram Symbol



Truth Table

A	X
0	1
1	0

Figure 4.1 **Various representations of a NOT gate**

NOT Gate

- By definition, if the input value for a NOT gate is 0, the output value is 1, and if the input value is 1, the output is 0
- A NOT gate is sometimes referred to as an *inverter* because it inverts the input value

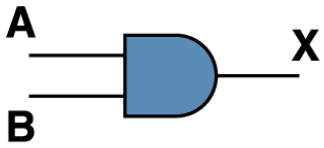
AND Gate

- An AND gate accepts two input signals
- If the two input values for an AND gate are both 1, the output is 1; otherwise, the output is 0

Boolean Expression

$$X = A \cdot B$$

Logic Diagram Symbol



Truth Table

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

Figure 4.2 **Various representations of an AND gate**

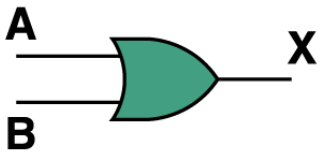
OR Gate

- If the two input values are both 0, the output value is 0; otherwise, the output is 1

Boolean Expression

$$X = A + B$$

Logic Diagram Symbol



Truth Table

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

Figure 4.3 **Various representations of a OR gate**

XOR Gate

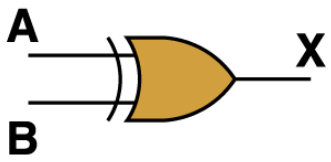
- XOR, or *exclusive* OR, gate
  - An XOR gate produces 0 if its two inputs are the same, and a 1 otherwise
  - Note the difference between the XOR gate and the OR gate; they differ only in one input situation
  - When both input signals are 1, the OR gate produces a 1 and the XOR produces a 0

XOR Gate

Boolean Expression

$$X = A \oplus B$$

Logic Diagram Symbol



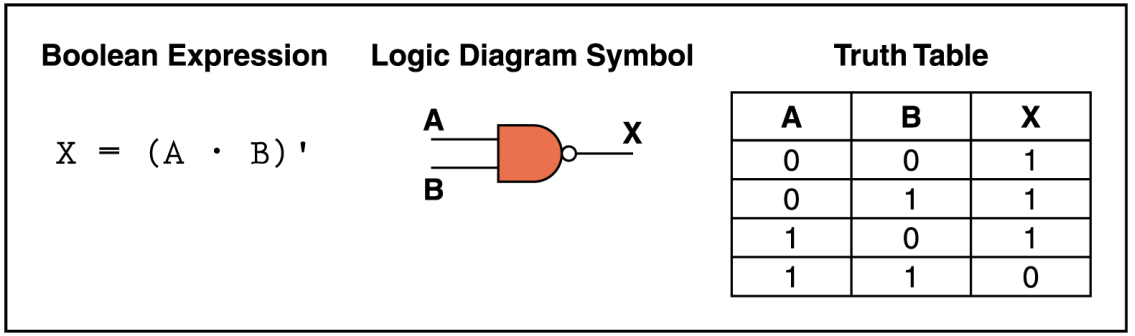
Truth Table

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

Figure 4.4 **Various representations of an XOR gate**

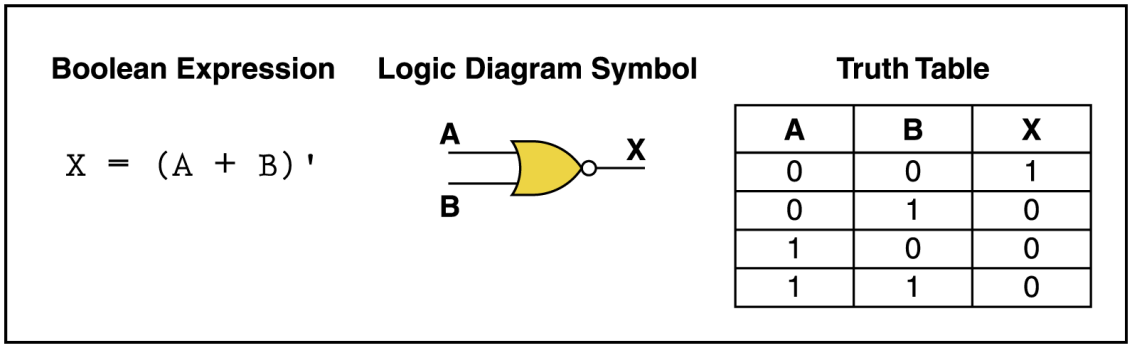
NAND and NOR Gates

- The NAND and NOR gates are essentially the opposite of the AND and OR gates, respectively



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

Figure 4.5 Various representations of a NAND gate



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

Figure 4.6 Various representations of a NOR gate

Boolean Theorems (Single-Variable)

- x* 0 =0
- x* 1 =x
- x*x=x
- x*x’=0
- x+0=x
- x+1=1
- x+x=x
- x+x’=1

Boolean theorems (multivariable)

- X Y=YX
- X+(Y+Z)=(X+Y)+Z
- X(YZ)=(XY)Z
- X(Y+Z)=XY+XZ
- (X')'=X

- We can use Boolean identities to simplify the function:

  as follows:

$(X + Y)(X + \overline{Y})\,\overline{(X\overline{Z})}$	Idempotent Law (Rewriting)
$(X + Y)(X + \overline{Y})(\overline{X} + Z)$	DeMorgan's Law
$(XX + X\overline{Y} + XY + Y\overline{Y})(\overline{X} + Z)$	Distributive Law
$((X + Y\overline{Y}) + X(Y + \overline{Y}))(\overline{X} + Z)$	Commutative & Distributive Laws
$((X + 0) + X(1))(\overline{X} + Z)$	Inverse Law
$X(\overline{X} + Z)$	Idempotent Law
$X\overline{X} + XZ$	Distributive Law
$0 + XZ$	Inverse Law
XZ	Idempotent Law

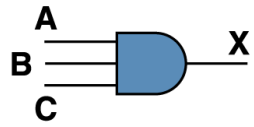
Gates with More Inputs

- Gates can be designed to accept three or more input values
- A three-input AND gate, for example, produces an output of 1 only if all input values are 1

Boolean Expression

$X = A \cdot B \cdot C$

Logic Diagram Symbol



Truth Table

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

Figure 4.7 Various representations of a three-input AND gate

Prove the following identities using truth table

Perfect Induction Method

(X + Y) (X + Z) = X + YZ

X	Y	Z	X+Y	X+Z	YZ	(X+Y)(X+Z)	X+YZ
0	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	1	0	1	1
1	0	0	1	1	0	1	1
1	0	1	1	1	0	1	1
1	1	0	1	1	1	1	1
1	1	1	1	1	1	1	1

R.H.S. = L.H.S.

$X + \overline{X}\,Y = X + Y$

X	Y	$\overline{X}$	$\overline{X}\,Y$	$X + \overline{X}\,Y$	X + Y
0	0	1	0	0	0
0	1	1	1	1	1
1	0	0	0	1	1
1	1	0	0	1	1

L.H.S. =R.H.S.

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

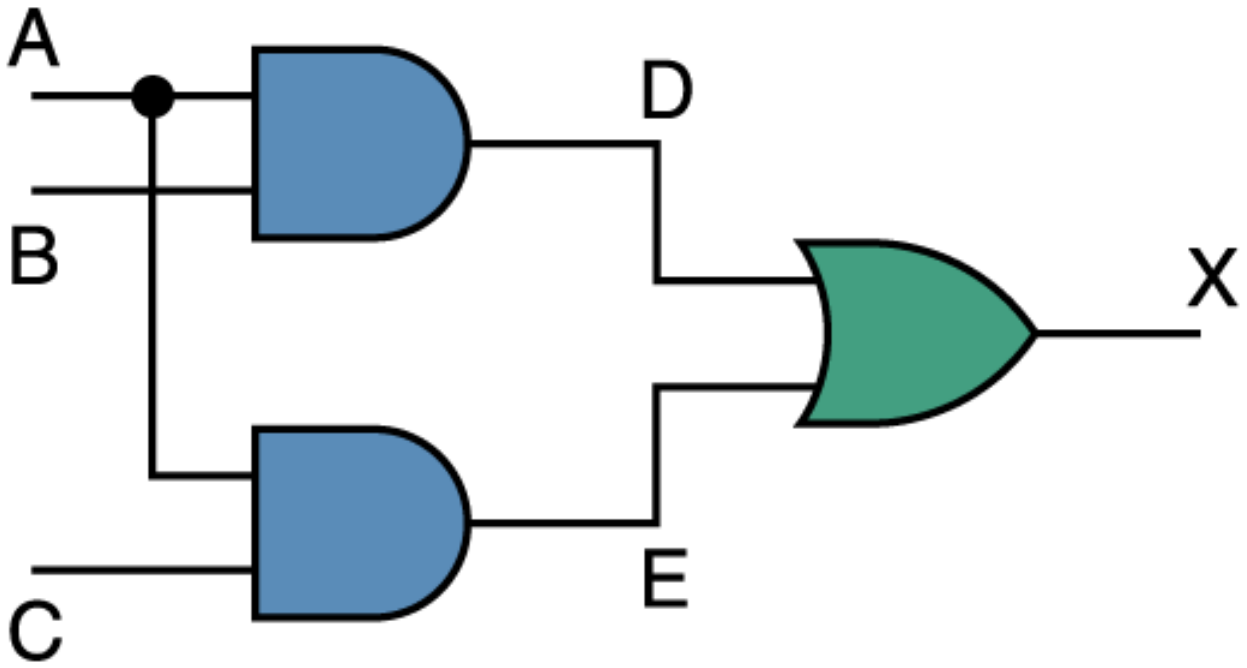
De Morgan's Theorems:

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

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

Combinational Circuits

- Gates are combined into circuits by using the output of one gate as the input



Combinational Circuits

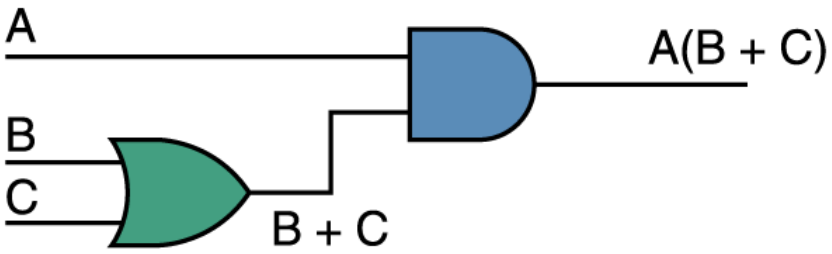
A	B	C	D	E	X
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	0	0	0
1	0	1	0	1	1
1	1	0	1	0	1
1	1	1	1	1	1

- Because there are three inputs to this circuit, eight rows are required to describe all possible input combinations
- This same circuit using Boolean algebra:

(AB + AC)

Now let's go the other way; let's take a Boolean expression and draw

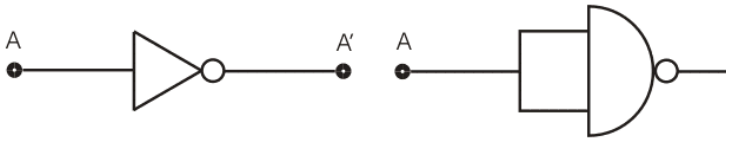
- Consider the following Boolean expression: $A(B + C)$



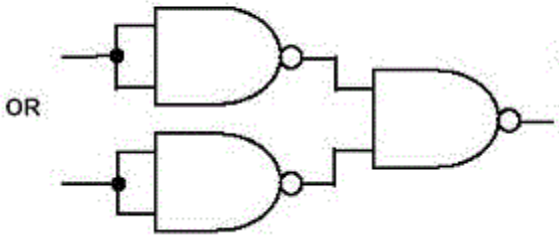
A	B	C	B + C	A(B+C)
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	1	0
1	0	0	0	0
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

- Now compare the final result column in this truth table to the truth table for the previous example
  - They are identical

NAND as NOT

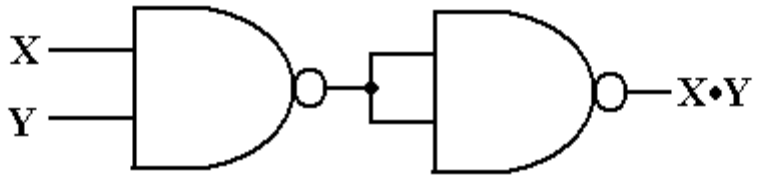


NAND as OR

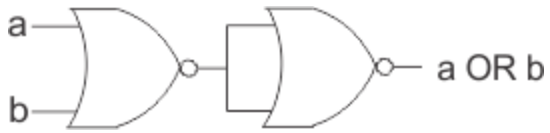


A	B	Output
0	0	0
0	1	1
1	0	1
1	1	1

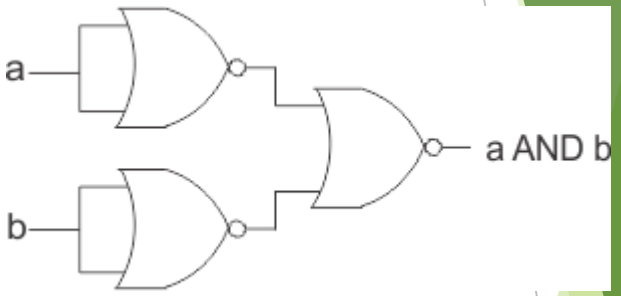
NAND as AND



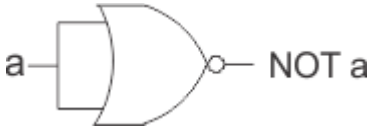
NOR as OR



NOR as AND



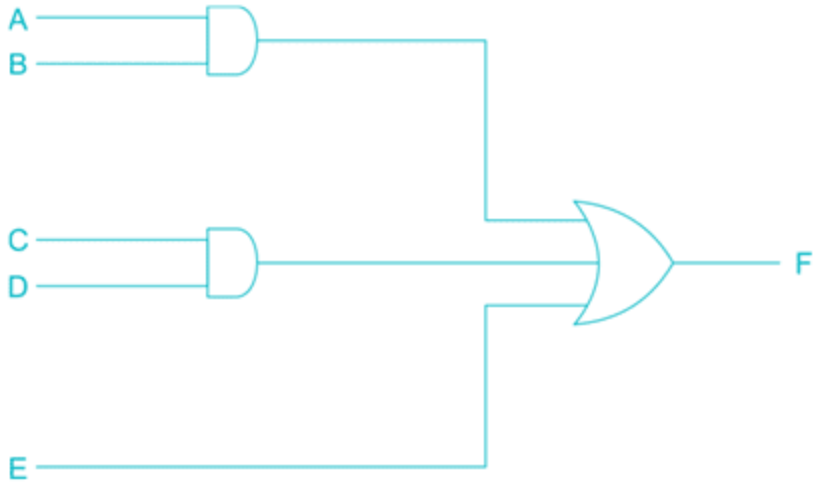
NOR as NOT



Realization of Logic Gates Using Universal Gates

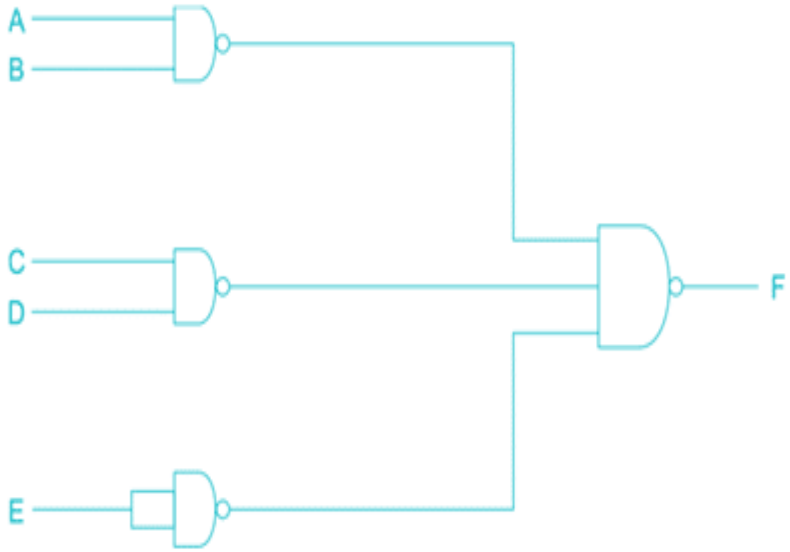
$$F = AB + CD + E$$

Logic representation of F:

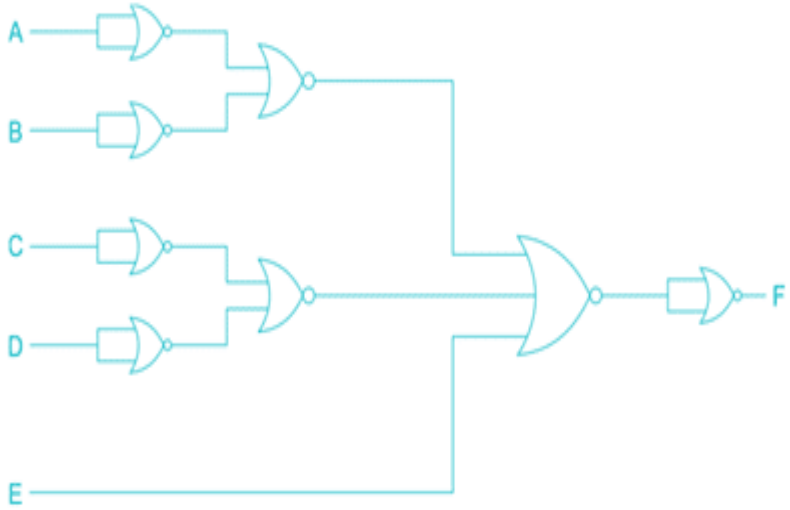


F = AB + CD + E

NAND implementation



NOR implementation



Standard Forms of Boolean Expressions

- All Boolean expressions, regardless of their form, can be converted into either of two standard forms:
  - The sum-of-products (SOP) form
  - The product-of-sums (POS) form

Sum-of-Products (SOP)

Canonical Forms-Minterms and Maxterms

x	y	z	minterm	designation	maxterm	designation
0	0	0	$\bar{x}\,\bar{y}\,\bar{z}$	m_0	$x+y+z$	M_0
0	0	1	$\bar{x}\,\bar{y}\,z$	m_1	$x+y+\bar{z}$	M_1
0	1	0	$\bar{x}\,y\,\bar{z}$	m_2	$x+\bar{y}+z$	M_2
0	1	1	$\bar{x}\,y\,z$	m_3	$x+\bar{y}+\bar{z}$	M_3
1	0	0	$x\,\bar{y}\,\bar{z}$	m_4	$\bar{x}+y+z$	M_4
1	0	1	$x\,\bar{y}\,z$	m_5	$\bar{x}+y+\bar{z}$	M_5
1	1	0	$x\,y\,\bar{z}$	m_6	$\bar{x}+\bar{y}+z$	M_6
1	1	1	$x\,y\,z$	m_7	$\bar{x}+\bar{y}+\bar{z}$	M_7
			(AND terms)		(OR terms)	

The Sum-of-Products (SOP) Form

- **An SOP expression → when two or more product terms are summed by Boolean addition.**
  - **Examples:**

$$AB + ABC$$

$$ABC + CDE + \overline{B}C\overline{D}$$

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

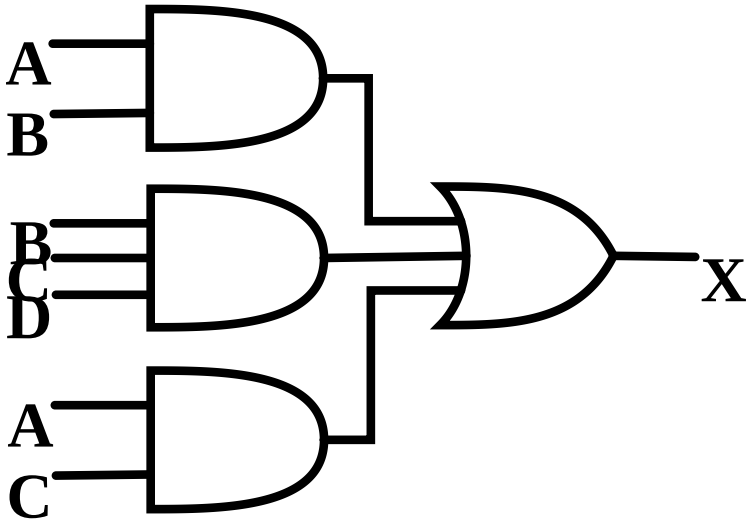
  - **Also:**

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

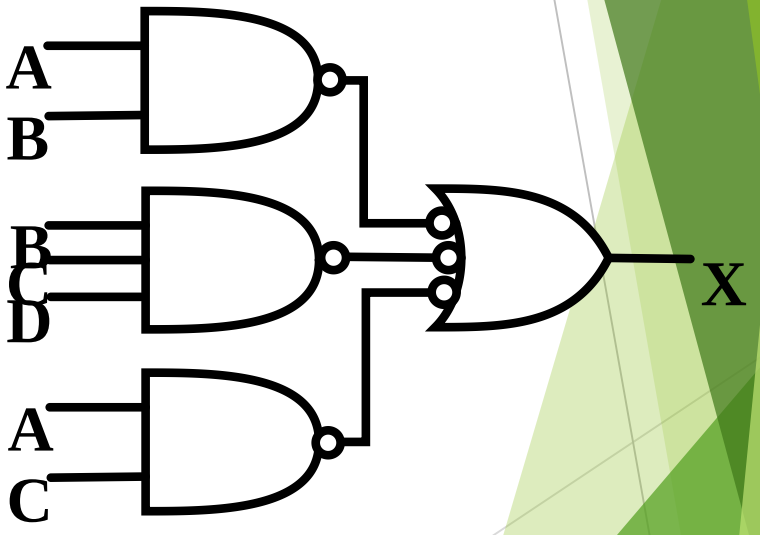
Implementation of an SOP

$X=AB+BCD+AC$

- **AND/OR implementation**



- **NAND/NAND implementation**



Product-of-Sums (POS)

The Product-of-Sums (POS) Form

- **When two or more sum terms are multiplied, the result expression is a product-of-sums (POS):**
  - **Examples:**

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

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

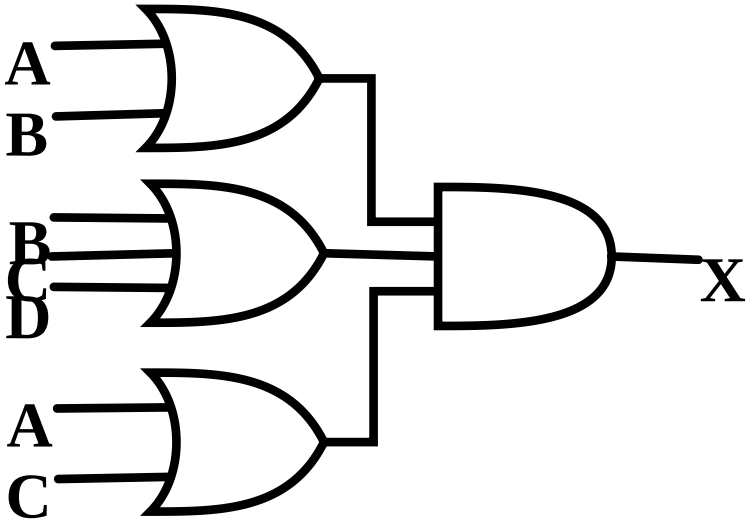
$$(A+B)(A+\overline{B}+C)(\overline{A}+C)$$

  - **Also:** $\overline{A}(\overline{A}+\overline{B}+C)(B+C+\overline{D})$

Implementation of a POS

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

- **OR/AND implementation**



The Standard POS Form

- **A standard POS expression is one in which *all* the variables in the domain appear <u>in each sum term</u> in the expression.**
  - **Example:** $(\overline{A}+\overline{B}+\overline{C}+\overline{D})(A+\overline{B}+C+D)(A+B+\overline{C}+D)$

- **Standard POS expressions are important in:**
  - **Constructing truth tables**
  - **The Karnaugh map simplification method**

Converting SOP Expressions to Truth Table Format (example)

- **Develop a truth table for the standard SOP expression**

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

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

Converting POS Expressions to Truth Table Format (example)

- **Develop a truth table for the standard SOP expression**

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

Inputs			Output	Product Term
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	

The Karnaugh Map

The 2 Variable K-Map

A. SOP: -

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

B. POS: -

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

The 3 Variable K-Map

- **There are 8 cells as shown:**

AB \ C	0	1
00	$\overline{A}\,\overline{B}\,\overline{C}$	$\overline{A}\,\overline{B}C$
01	$\overline{A}B\overline{C}$	$\overline{A}BC$
11	$AB\overline{C}$	ABC
10	$A\overline{B}\,\overline{C}$	$A\overline{B}C$

The 4-Variable K-Map

AB \ CD	00	01	11	10
00	$\bar{A}\bar{B}\bar{C}\bar{D}$	$\bar{A}\bar{B}\bar{C}D$	$\bar{A}\bar{B}CD$	$\bar{A}\bar{B}C\bar{D}$
01	$\bar{A}B\bar{C}\bar{D}$	$\bar{A}B\bar{C}D$	$\bar{A}BCD$	$\bar{A}BC\bar{D}$
11	$AB\bar{C}\bar{D}$	$AB\bar{C}D$	$ABCD$	$ABC\bar{D}$
10	$A\bar{B}\bar{C}\bar{D}$	$A\bar{B}\bar{C}D$	$A\bar{B}CD$	$A\bar{B}C\bar{D}$

Mapping a Nonstandard SOP Expression

- **Map the following SOP expressions on K-maps:**

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

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

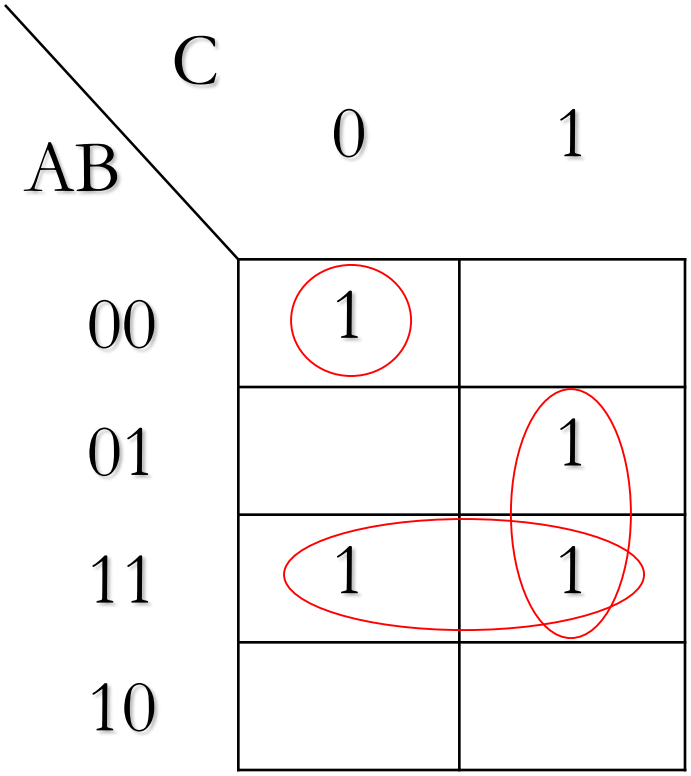
K-Map Simplification of SOP Expressions

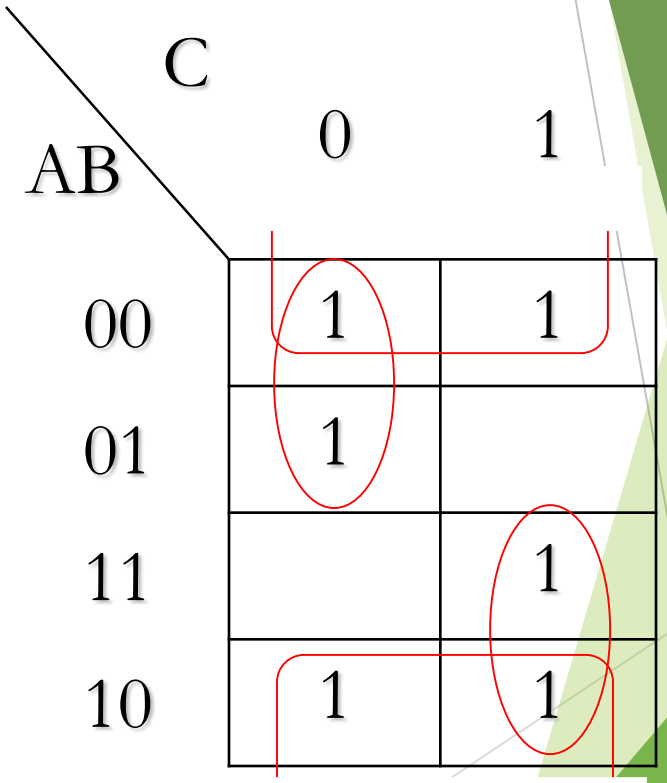
- **After an SOP expression has been mapped, we can do the process of *minimization:***
  - **Grouping the 1s**
  - **Determining the minimum SOP expression from the map**

Grouping the 1s (rules)

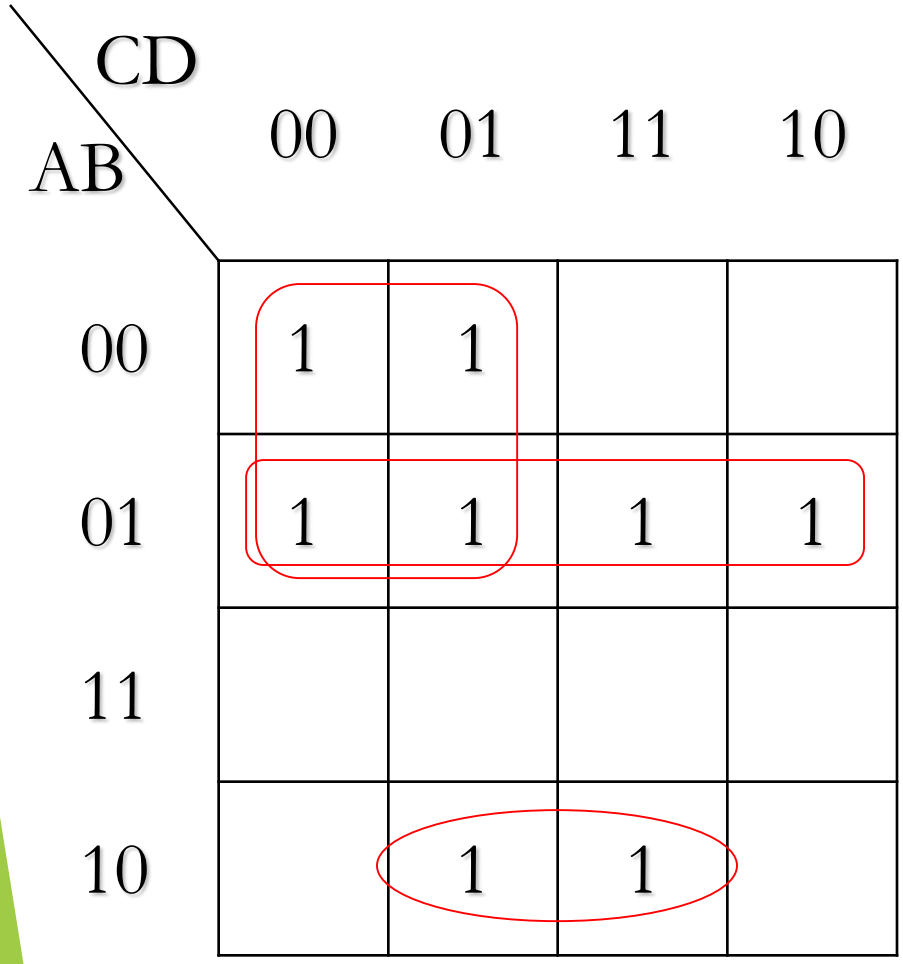
1. A group must contain either 1,2,4,8,or 16 cells (depending on number of variables in the expression)
2. Each cell in a group must be adjacent to one or more cells in that same group, but all cells in the group do not have to be adjacent to each other.
3. Always include the largest possible number of 1s in a group in accordance with rule 1.
4. Each 1 on the map must be included in at least one group. The 1s already in a group can be included in another group as long as the overlapping groups include noncommon 1s.

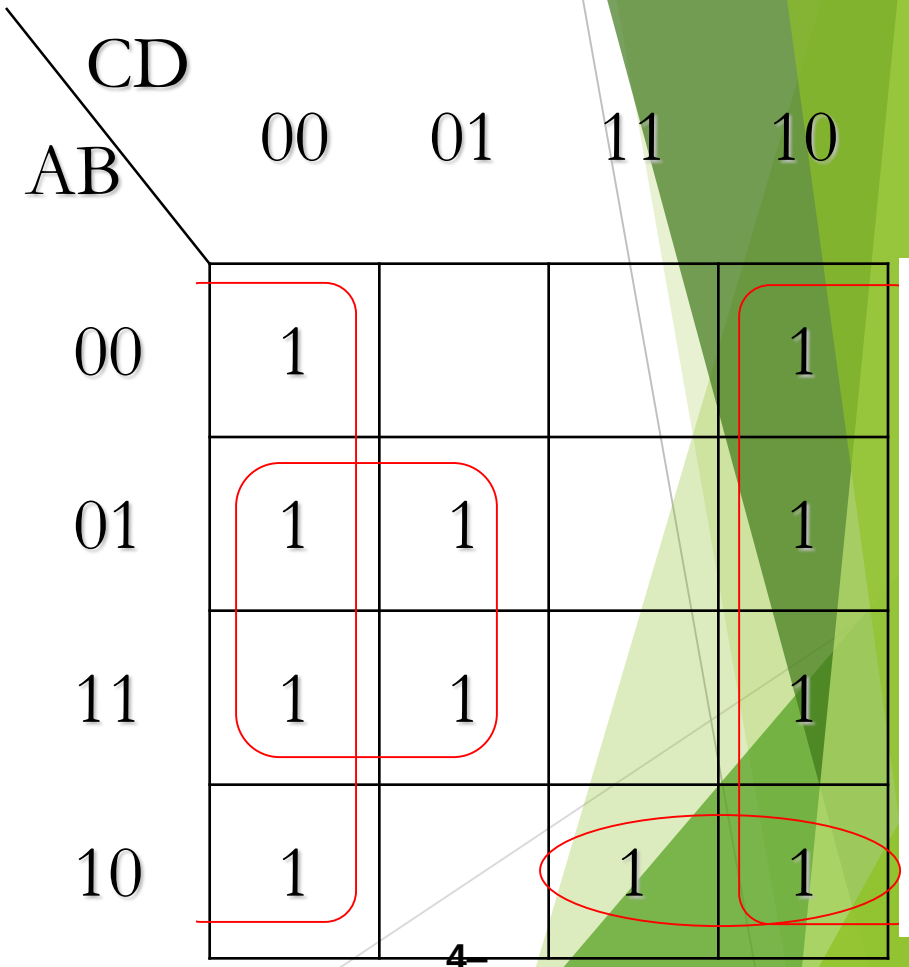
Grouping the 1s (example)





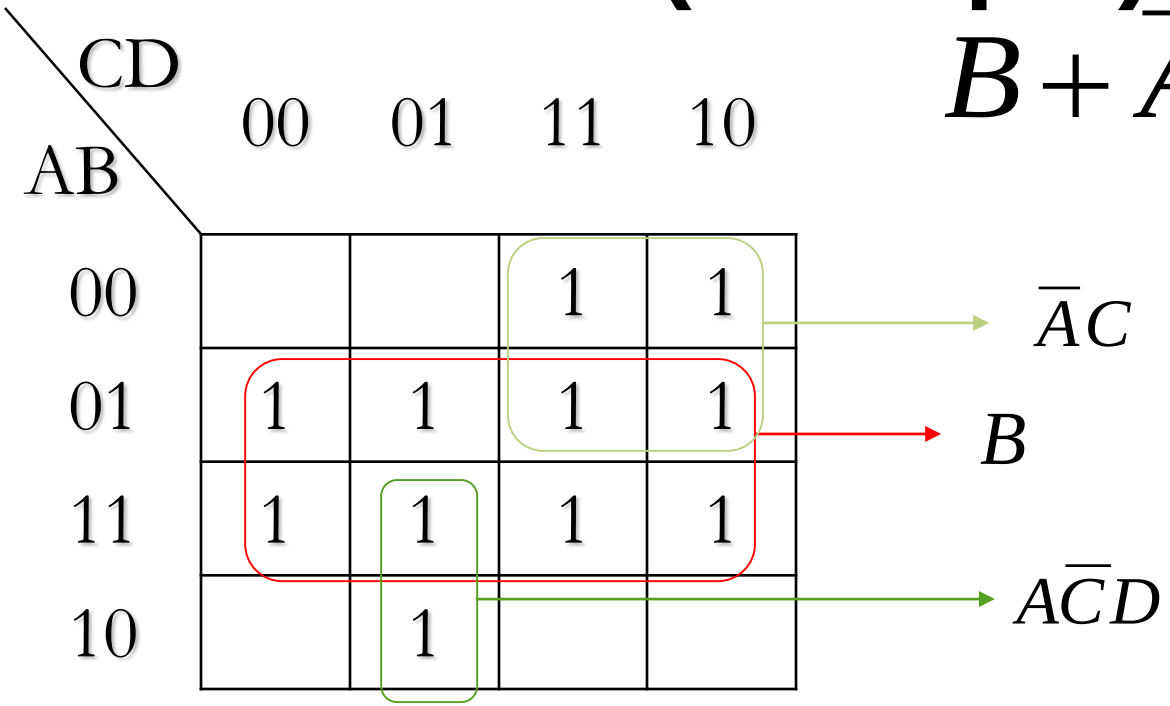
Grouping the 1s (example)



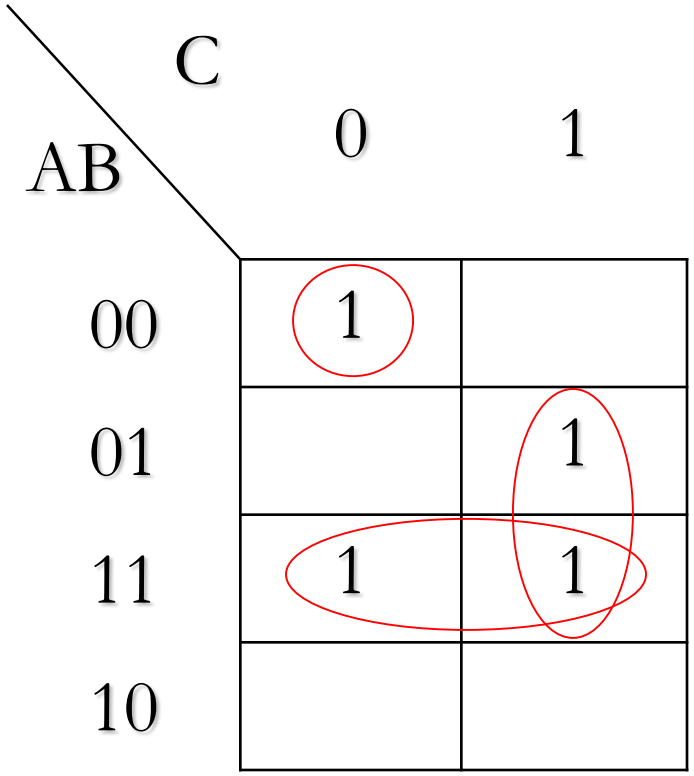


Determining the Minimum SOP Expression from the Map (example)

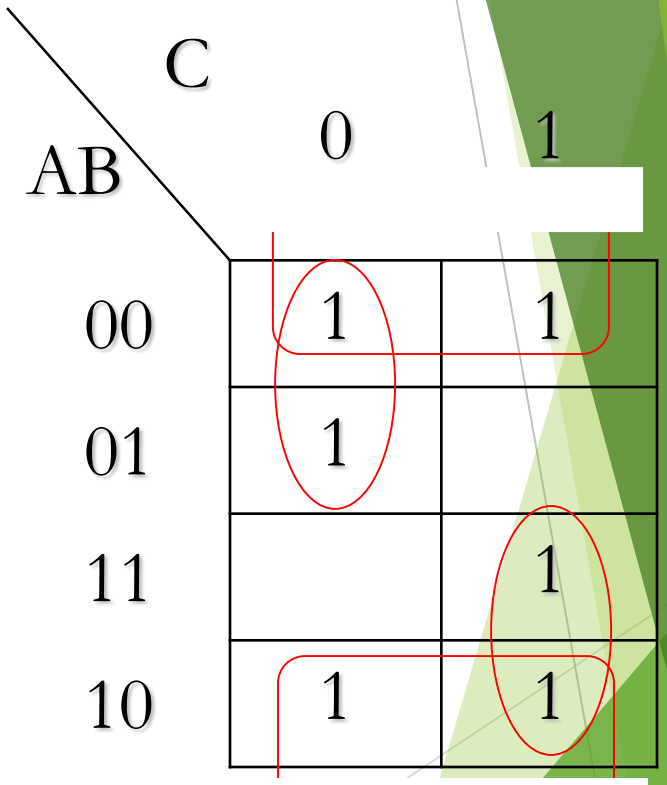
$$B + \overline{A}C + A\overline{C}D$$



Determining the Minimum SOP Expression from the Map (exercises)

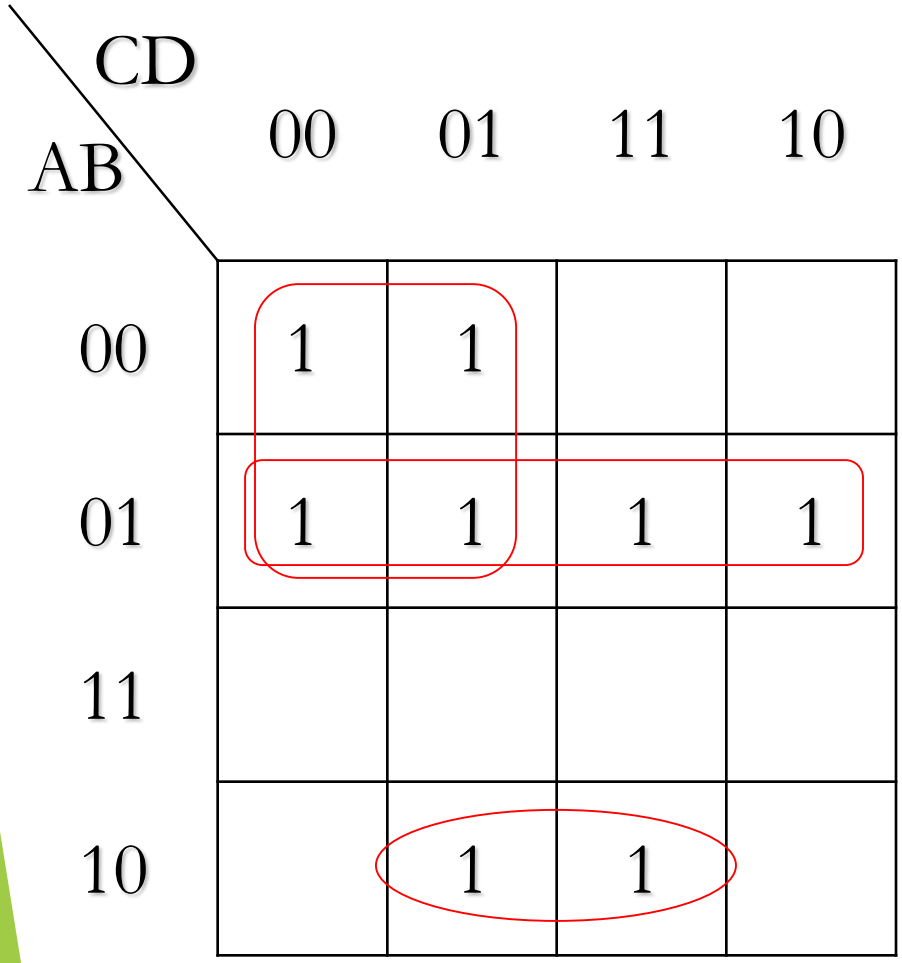


$$AB + BC + \overline{A}\,\overline{B}\,\overline{C}$$

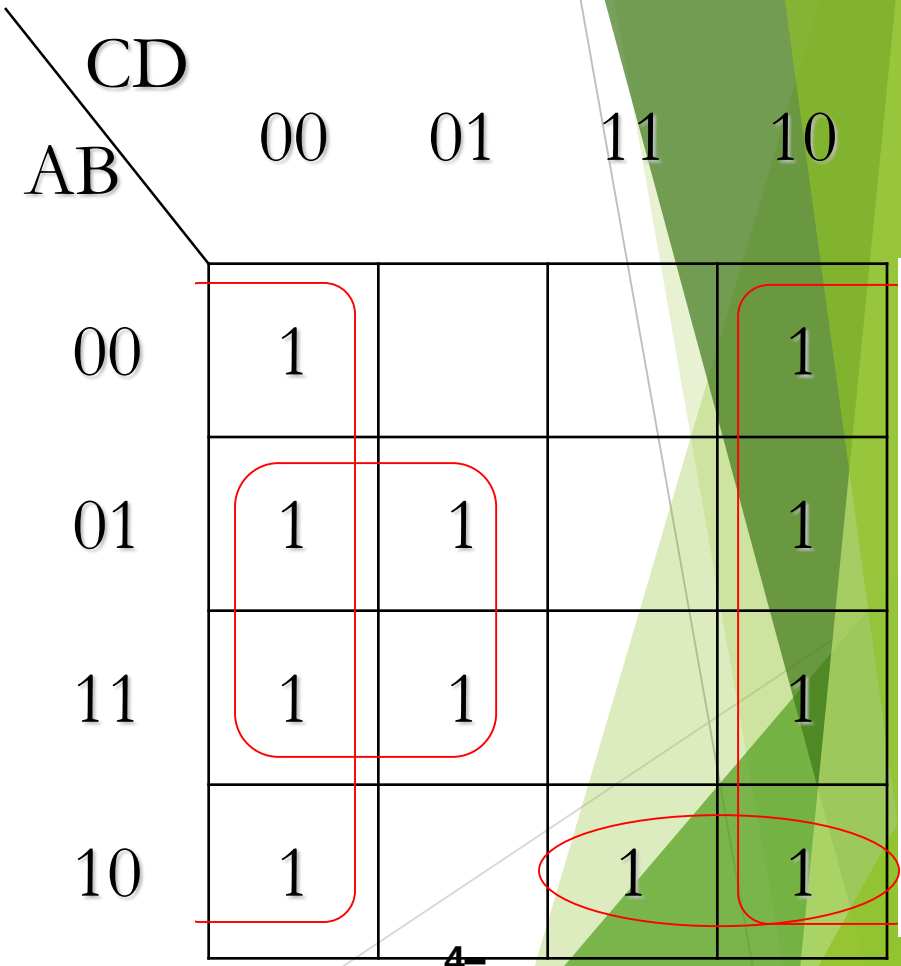


$$\overline{B} + \overline{A}\,\overline{C} + AC$$

Determining the Minimum SOP Expression from the Map (exercises)

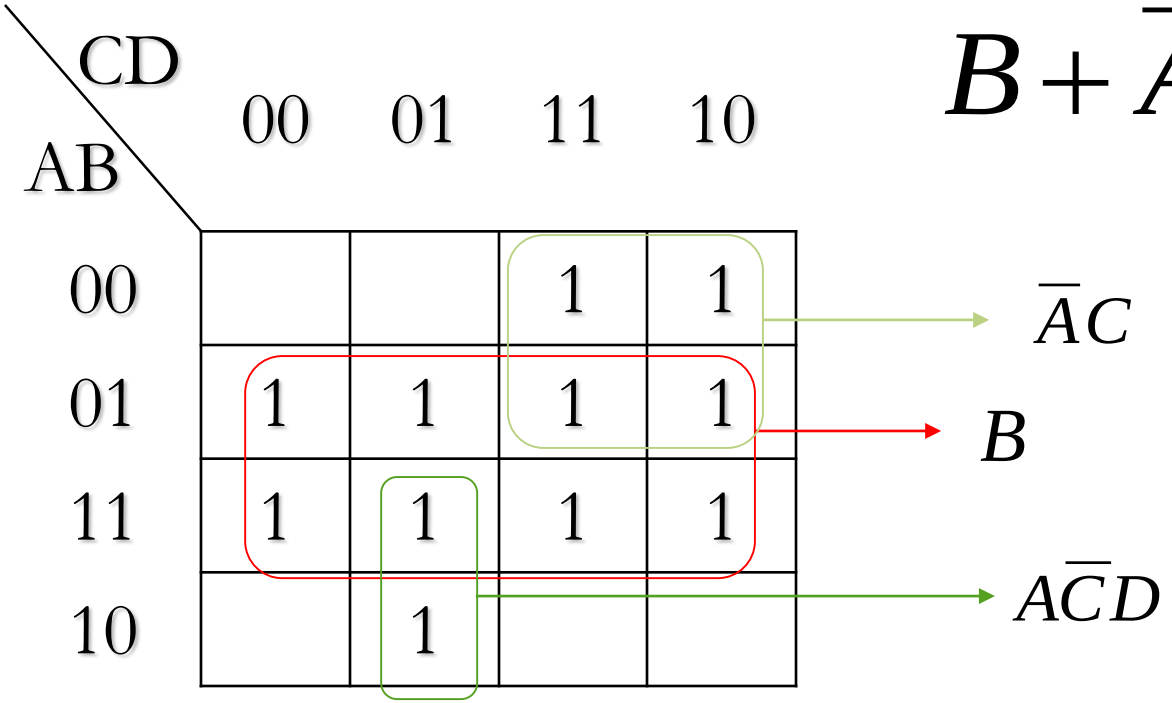


$$\overline{A}B + \overline{A}\overline{C} + A\overline{B}D$$



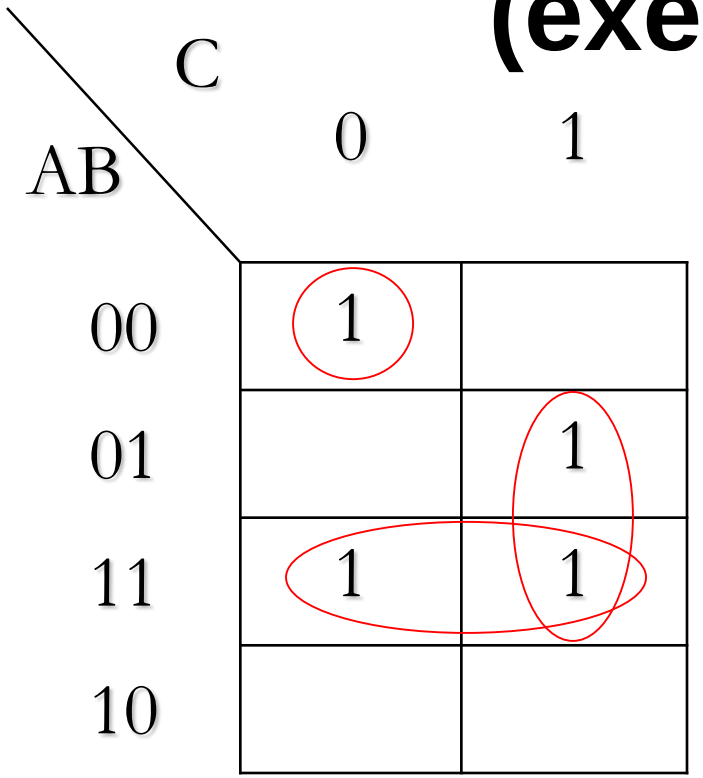
$$\overline{D} + A\overline{B}C + B\overline{C}$$

Determining the Minimum SOP Expression from the Map (example)

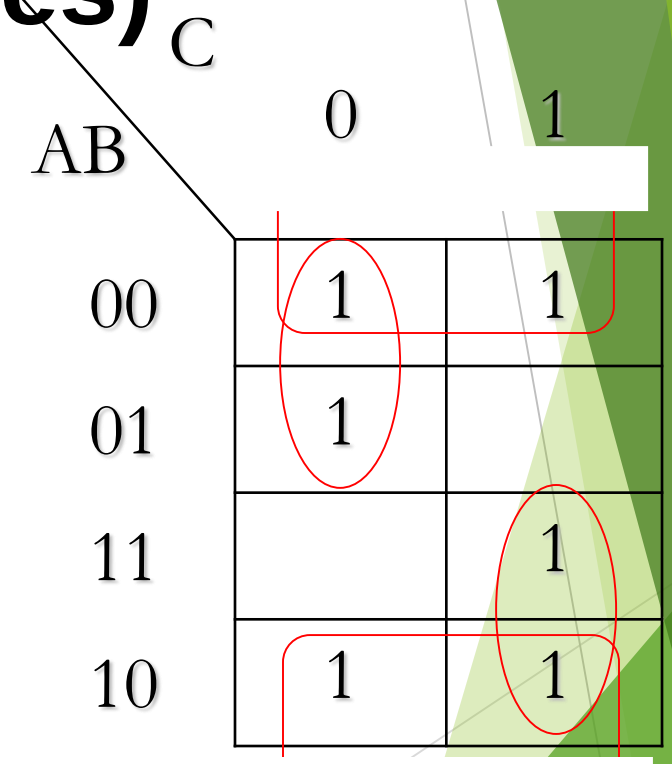


$$B + \overline{A}C + A\overline{C}D$$

Determining the Minimum SOP Expression from the Map (exercises)

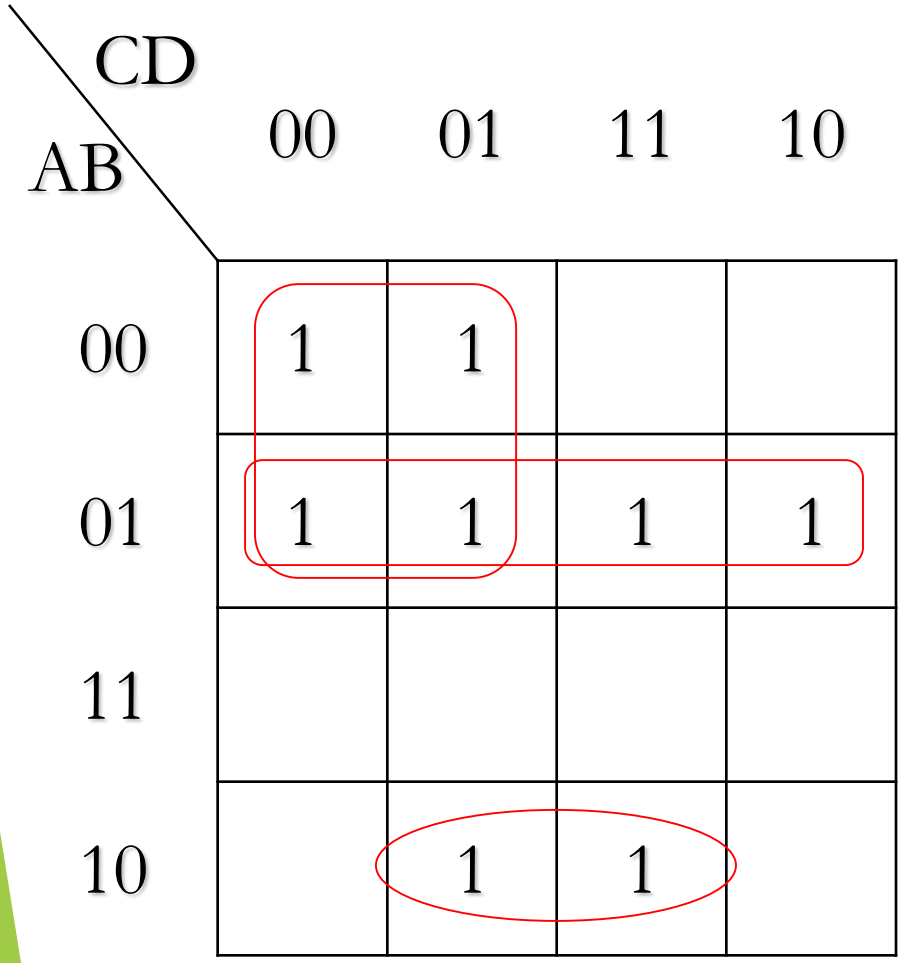


$$AB + BC + \overline{A}\,\overline{B}\,\overline{C}$$

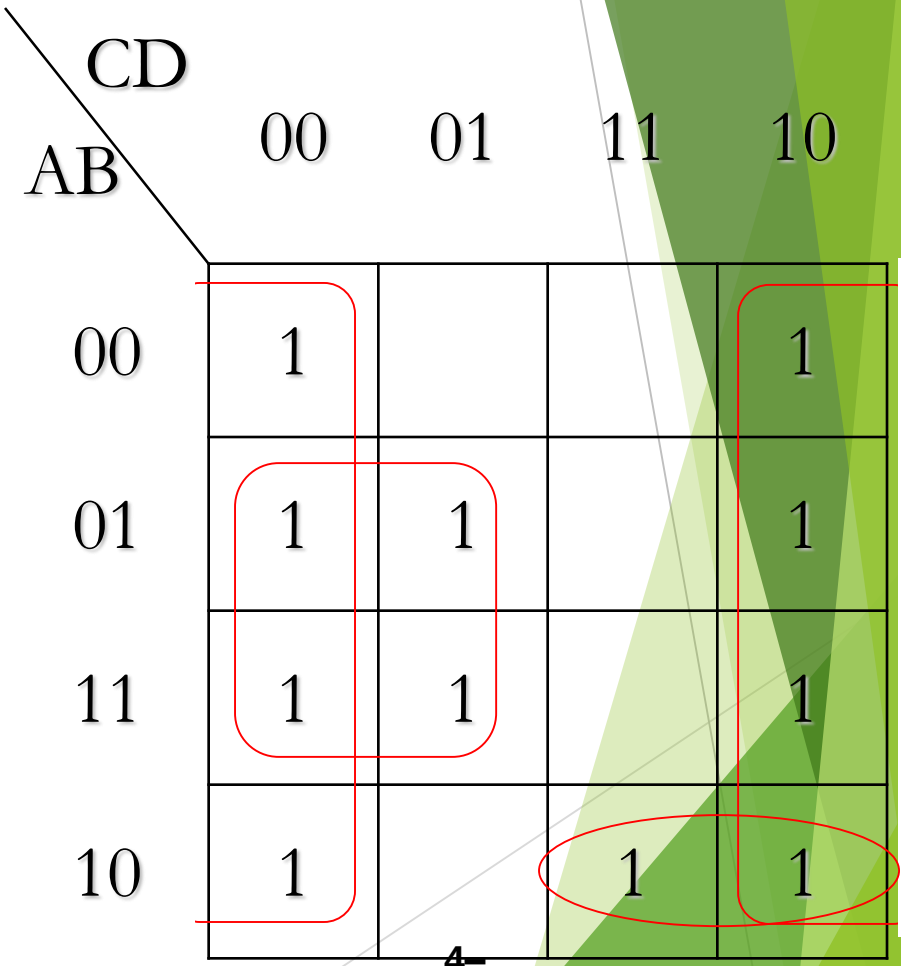


$$\overline{B} + \overline{A}\,\overline{C} + AC$$

Determining the Minimum SOP Expression from the Map (exercises)



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



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

Practicing K-Map (SOP)

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

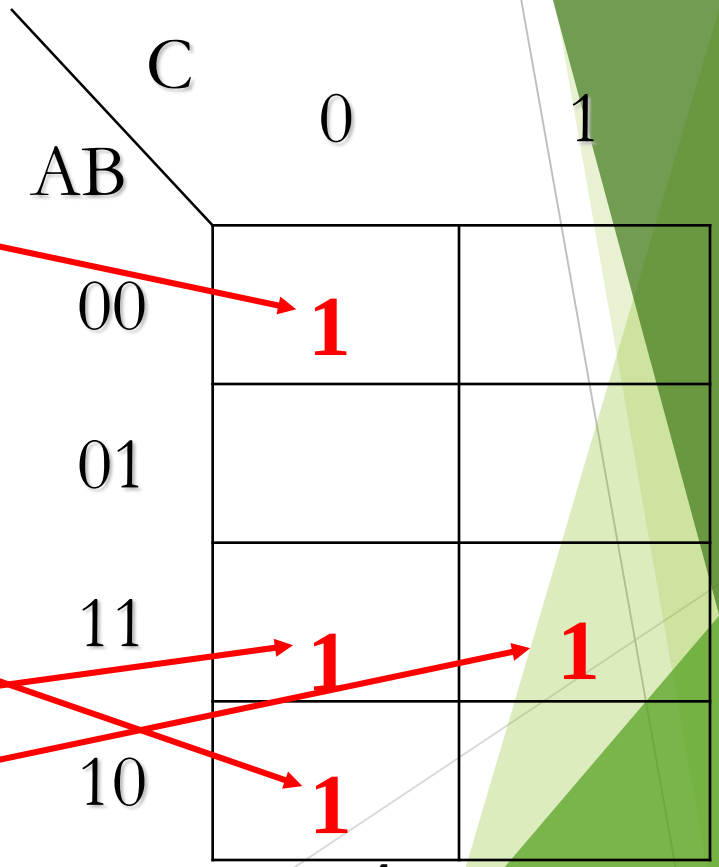
$$\overline{B} + \overline{A}C$$

$$\overline{B}\overline{C}\overline{D} + \overline{A}B\overline{C}\overline{D} + AB\overline{C}\overline{D} + \overline{A}\overline{B}CD + A\overline{B}CD +$$
$$\overline{A}\overline{B}C\overline{D} + \overline{A}BC\overline{D} + ABC\overline{D} + A\overline{B}C\overline{D}$$

$$\overline{D} + \overline{B}C$$

Mapping Directly from a Truth Table

I/P			O/P
A	B	C	X
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

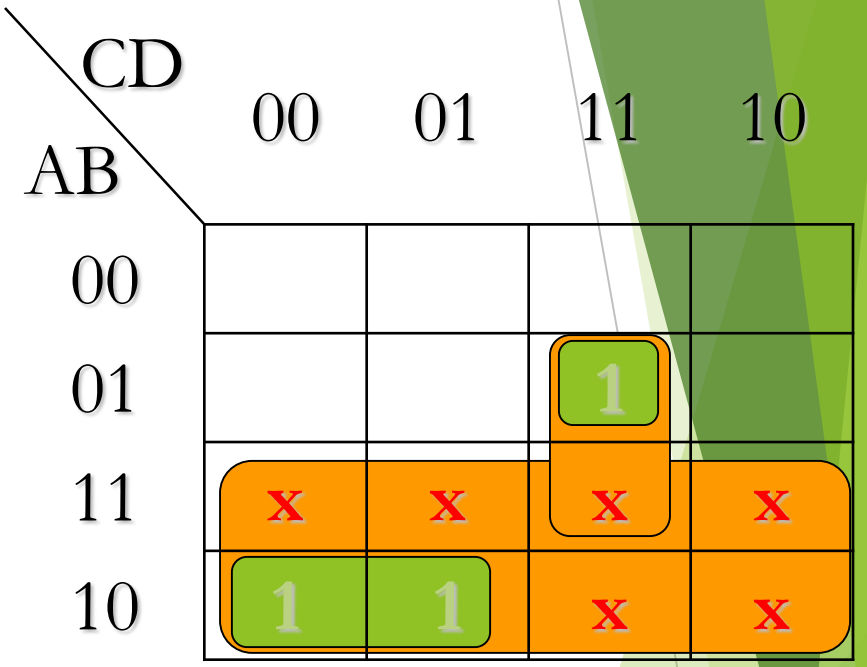


“Don’t Care” Conditions

- **Sometimes a situation arises in which some input variable combinations are not allowed, i.e. BCD code:**
  - **There are six invalid combinations: 1010, 1011, 1100, 1101, 1110, and 1111.**
- **Since these unallowed 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.**
- **The “don’t care” terms can be used to advantage on the K-map.**

“Don’t Care” Conditions

INPUTS				O/P
A	B	C	D	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



Without “don’t care”

$$Y = ABC + ABCD$$

With “don’t care”

$$Y = A + BCD$$

K-Map POS Minimization

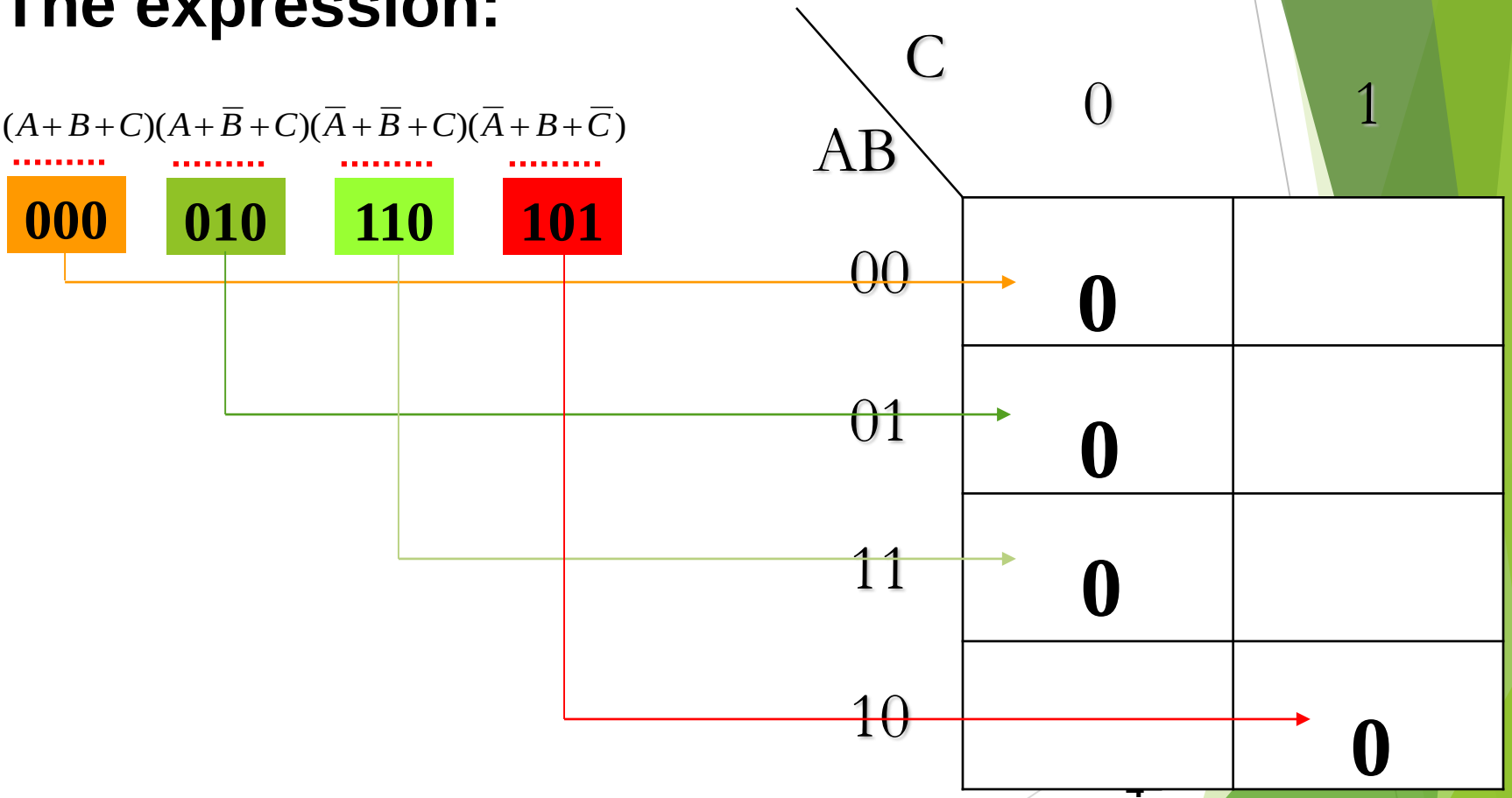
- **The approaches are much the same (as SOP) except that with POS expression, 0s representing the standard sum terms are placed on the K-map instead of 1s.**

Mapping a Standard POS Expression (full example)

The expression:

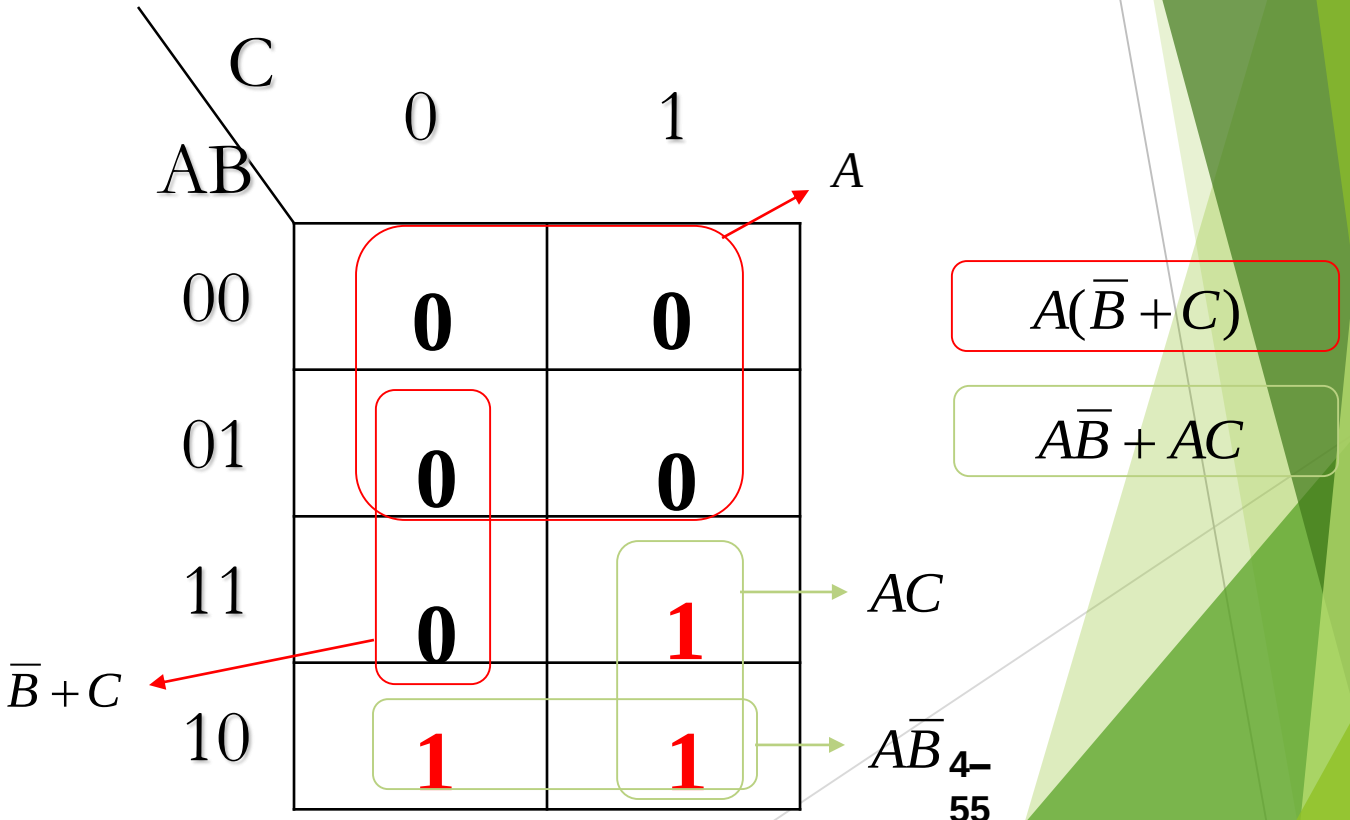
$$(A+B+C)(A+\overline{B}+C)(\overline{A}+\overline{B}+C)(\overline{A}+B+\overline{C})$$

000 **010** **110** **101**



K-map Simplification of POS Expression

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



Shorthand: ∑ and ∏

- $f_1(a,b,c) = \sum m(1,2,4,6)$, where ∑ indicates that this is a sum-of-products form, and m(1,2,4,6) indicates that the minterms to be included are m_1, m_2, m_4, and m_6.
- $f_1(a,b,c) = \prod M(0,3,5,7)$, where ∏ indicates that this is a product-of-sums form, and M(0,3,5,7) indicates that the maxterms to be included are M_0, M_3, M_5, and M_7.
- Since $m_j = M_j'$ for any *j*,
  $\sum m(1,2,4,6) = \prod M(0,3,5,7) = f_1(a,b,c)$

Conversion Between Canonical Forms

- Replace $\sum$ with $\prod$ (or *vice versa*) and replace those *j'*s that appeared in the original form with those that do not.
- Example:

$f_1(a,b,c) = a'b'c + a'bc' + ab'c' + abc'$

$= m_1 + m_2 + m_4 + m_6$

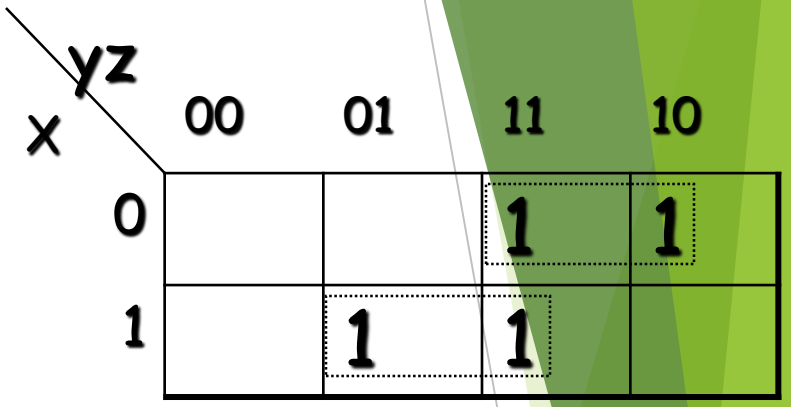
$= \sum(1,2,4,6)$

$= \prod(0,3,5,7)$

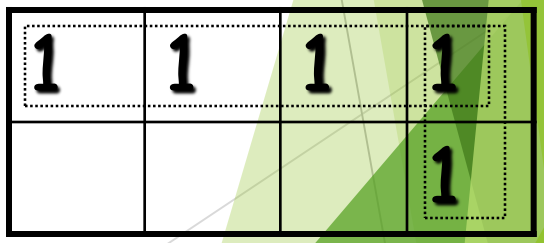
$= (a+b+c)\bullet(a+b'+c')\bullet(a'+b+c')\bullet(a'+b'+c')$

More Examples

- $f_1(x, y, z) = \sum m(2,3,5,7)$
  - $f_1(x, y, z) = x'y + xz$

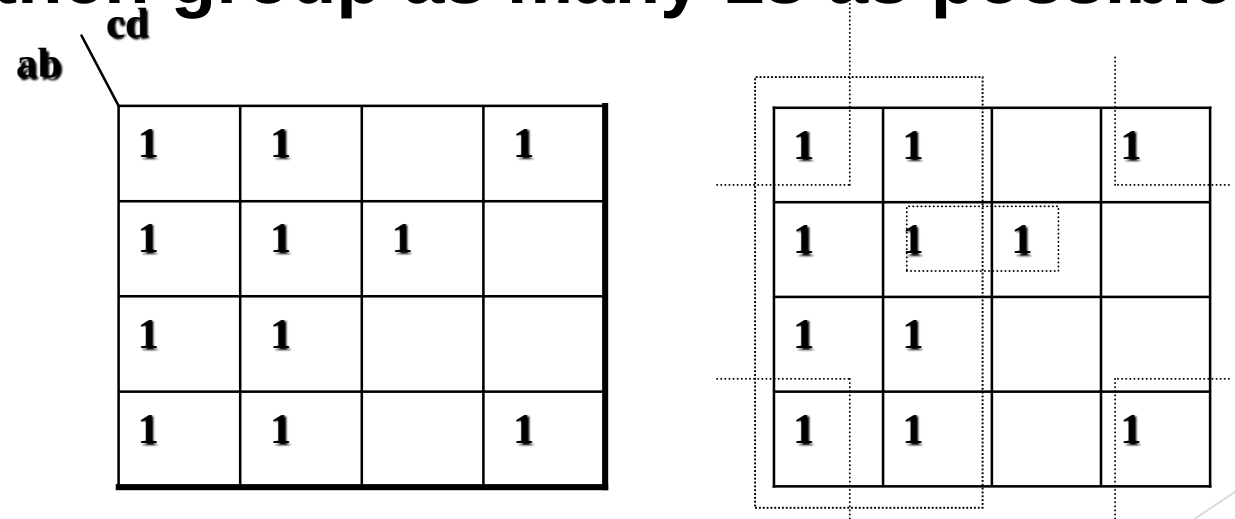


- $f_2(x, y, z) = \sum m(0,1,2,3,6)$
  - $f_2(x, y, z) = x'+yz'$



Example

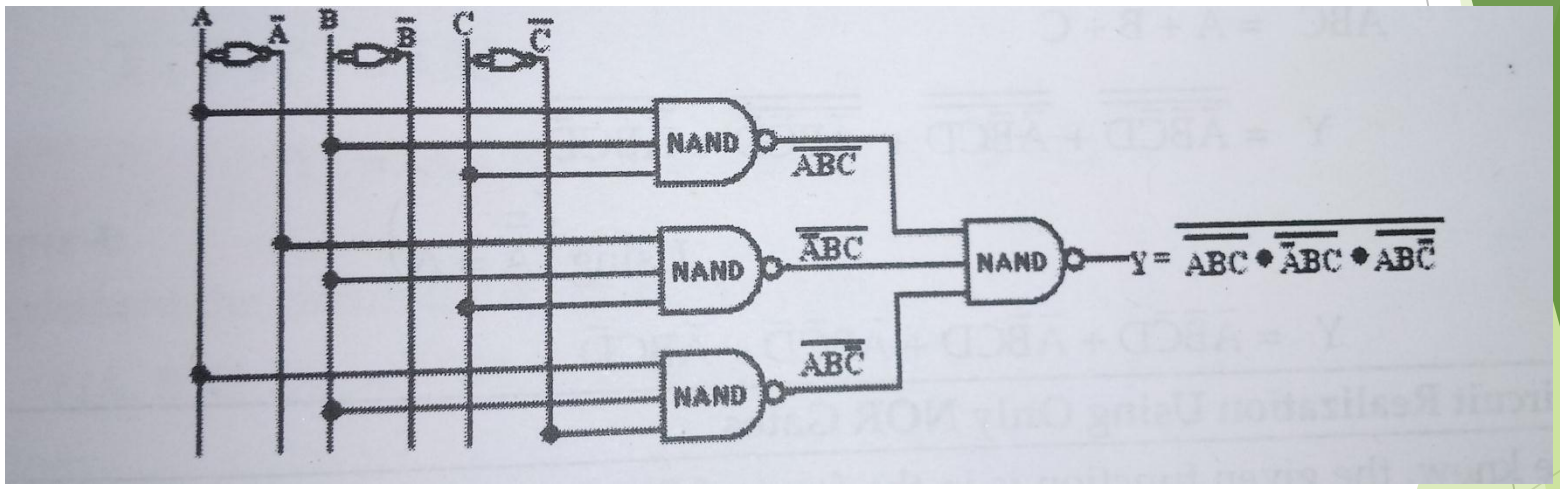
- **Simplify the following Boolean function (A,B,C,D) = ∑m(0,1,2,4,5,7,8,9,10,12,13).**
- **First put the function g( ) into the map, and then group as many 1s as possible.**



$$g(A,B,C,D) = c'+b'd'+a'bd$$

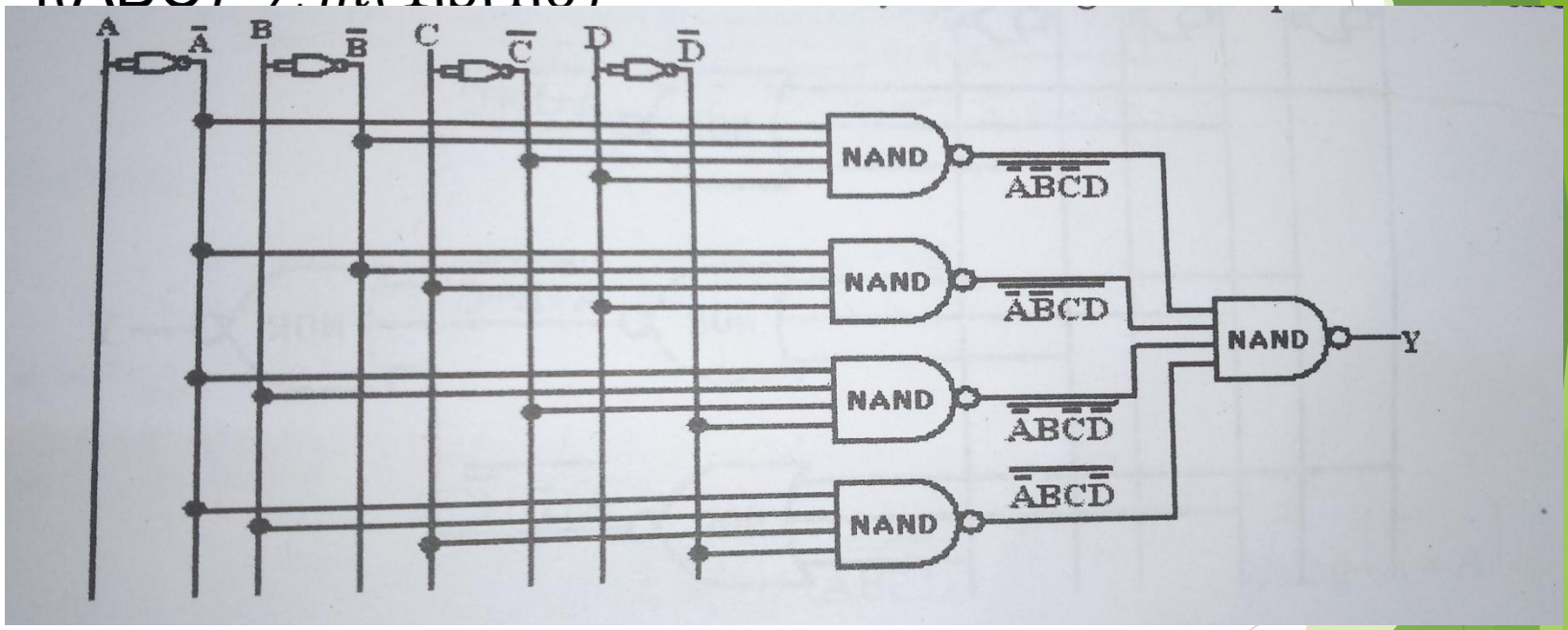
Implement the circuit for the for given minterm(SOP) function (NAND)

- F=ABC+A'BC+ABC'



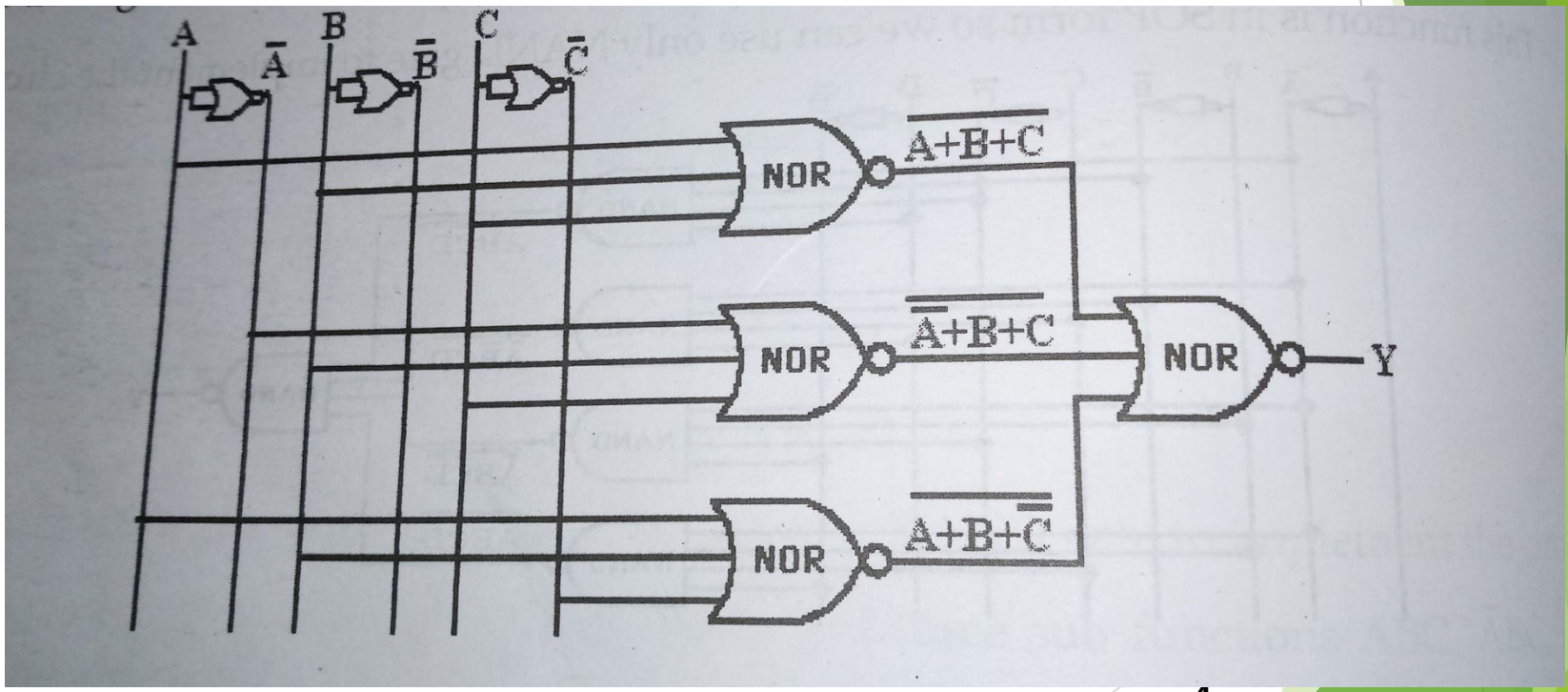
Implement the circuit for the given function

- f(ABC)=$\sum m(1,3,4,6)$



Implement the circuit for the given function

- f(A,B,C) = (A+B+C)(A'+B+C)(A+B+C')



Implement the circuit for the given function