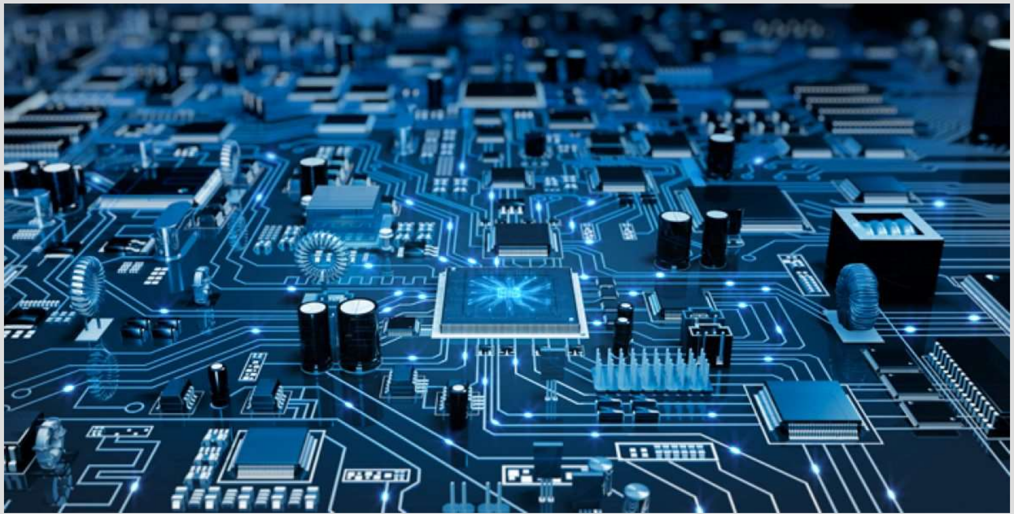
Digital System Design



Boolean Algebra and Logic Simplification

In this Chapter…

Objectives:

- Laws and Rules of Boolean Algebra
- Standard Forms of Boolean Expression
- Simplification by using algebra rules
- Karnaugh map and logic simplification

Boolean Algebra

- In 1854， George Boole， ***An Investigation of the Laws of Thought， on which are Founded the Mathematical Theories of Logic and Probabilities***
  - The design of the following treatise is to **investigate the fundamental laws of those operations of the mind** by which reasoning is performed;…to collect from the various elements of truth brought to view in the course of these inquiries **some probable intimations concerning the nature and constitution of the human mind**.
  - “the operations of the mind are in a certain real sense subject to laws, and that a science of the mind is therefore possible.”

- In 1938，Claude E. Shannon, ***A Symbolic Analysis of Relay and Switching Circuits***, Master thesis
  - Application of Boolean Algebra in Digital Design and Analysis

Boolean Addition

In Boolean algebra, a **variable** is a symbol used to represent an action, a condition, or data. A single variable can only have a value of 1 or 0.

The **complement** represents the inverse of a variable and is indicated with an overbar. Thus, the complement of A is $\overline{A}$.

A **literal** is a variable or its complement.

Addition is equivalent to the OR operation. The sum term is 1 if one or more if the literals are 1. The sum term is zero only if each literal is 0.

Example Determine the values of A, B, and C that make the sum term of the expression $\overline{A} + B + \overline{C} = 0$?

Solution Each literal must = 0; therefore $A = 1$, $B = 0$ and $C = 1$.

Boolean Multiplication

In Boolean algebra, multiplication is equivalent to the AND operation. The product of literals forms a product term. The product term will be 1 only if all of the literals are 1.

Example What are the values of the A, B and C if the product term of $A \cdot \overline{B} \cdot \overline{C} = 1$?

Solution Each literal must = 1; therefore $A = 1$, $B = 0$ and $C = 0$.

Commutative Laws

The **commutative laws** are applied to addition and multiplication. For addition, the commutative law states

In terms of the result, the order in which variables are ORed makes no difference.

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

For multiplication, the commutative law states

In terms of the result, the order in which variables are ANDed makes no difference.

$$\boxed{AB = BA}$$

Associative Laws

The **associative laws** are also applied to addition and multiplication. For addition, the associative law states

When ORing more than two variables, the result is the same regardless of the grouping of the variables.

$$\boxed{A + (B + C) = (A + B) + C}$$

For multiplication, the associative law states

When ANDing more than two variables, the result is the same regardless of the grouping of the variables.

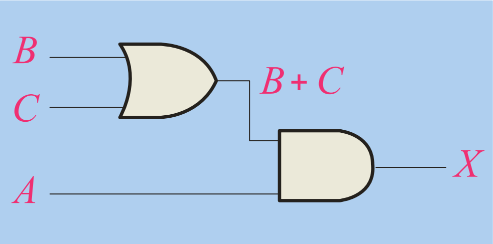
$$\boxed{A(BC) = (AB)C}$$

Distributive Law

The **distributive law** is the factoring law. A common variable can be factored from an expression just as in ordinary algebra. That is

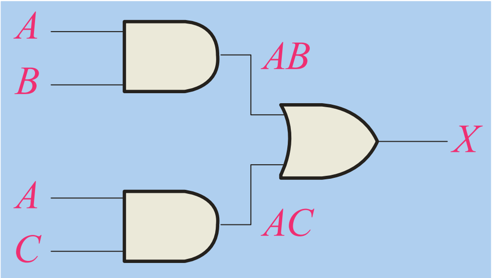
$$\boxed{AB + AC = A(B + C)}$$

The distributive law can be illustrated with equivalent circuits:



$A(B + C)$

≡



$AB + AC$

Rules of Boolean Algebra

1. $A + 0 = A$

2. $A + 1 = 1$

3. $A \cdot 0 = 0$

4. $A \cdot 1 = A$

5. $A + A = A$

6. $A + \overline{A} = 1$

7. $A \cdot A = A$

8. $A \cdot \overline{A} = 0$

9. $\overline{\overline{A}} = A$

10. $A + AB = A$

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

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

13. Consensus Theorem

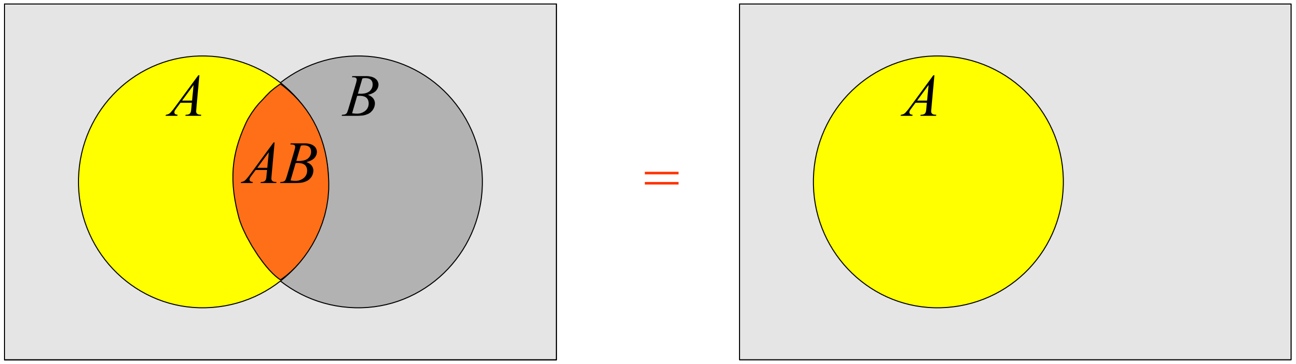
$$AB+\overline{A}C+BC = AB+\overline{A}C$$

Rules of Boolean Algebra

Rules of Boolean algebra can be illustrated with *Venn* diagrams. The variable A is shown as an area.

The rule $A + AB = A$ can be illustrated easily with a diagram. Add an overlapping area to represent the variable B.

The overlap region between A and B represents AB.



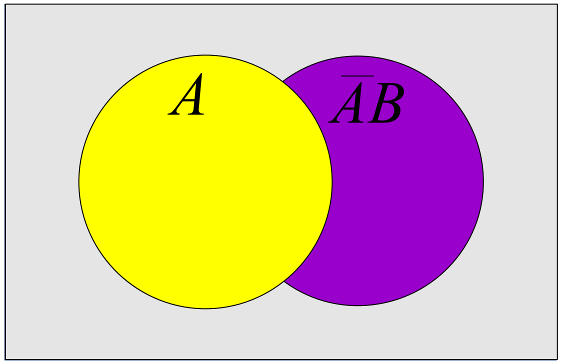
The diagram visually shows that $A + AB = A$. Other rules can be illustrated with the diagrams as well.

Rules of Boolean Algebra

Example Illustrate the rule $A + \overline{A}B = A + B$ with a Venn diagram.

Solution

This time, $\overline{A}$ is represented by the blue area and B again by the red circle. The intersection represents $\overline{A}B$. Notice that $A + \overline{A}B = A + B$



Rules of Boolean Algebra

Rule 12, which states that $(A + B)(A + C) = A + BC$, can be proven by applying earlier rules as follows:

$$
\begin{aligned}
(A + B)(A + C) &= AA + AC + AB + BC \\
&= A + AC + AB + BC \\
&= A(1 + C + B) + BC \\
&= A \cdot 1 + BC \\
&= A + BC
\end{aligned}
$$

This rule is a little more complicated, but it can also be shown with a Venn diagram, as given on the following slide…

Three areas represent the variables A, B, and C.

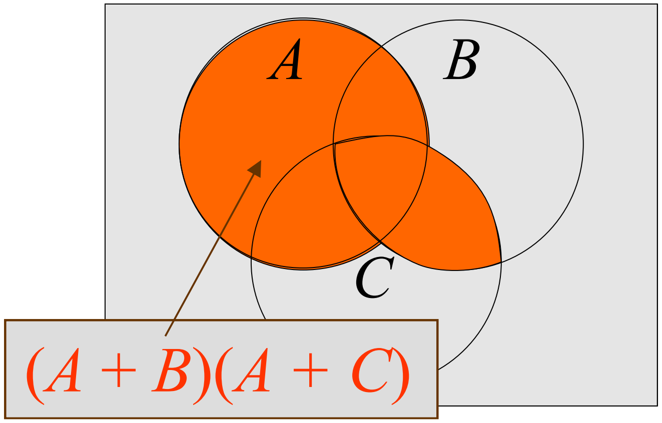
The area representing $A + B$ is shown in yellow.

The area representing $A + C$ is shown in red.

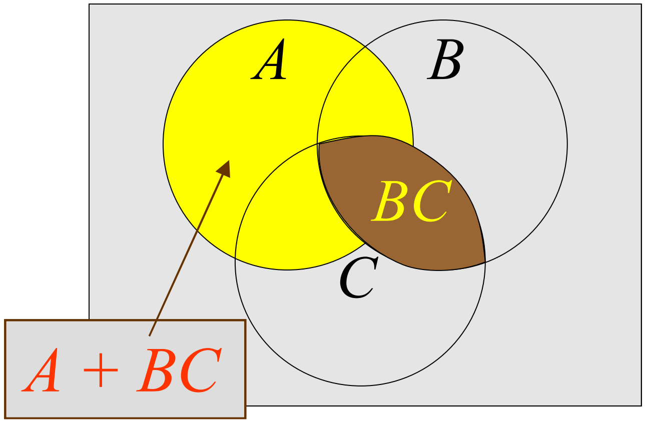
The overlap of red and yellow is shown in orange.

The overlapping area between B and C represents BC.

ORing with A gives the same area as before.



=



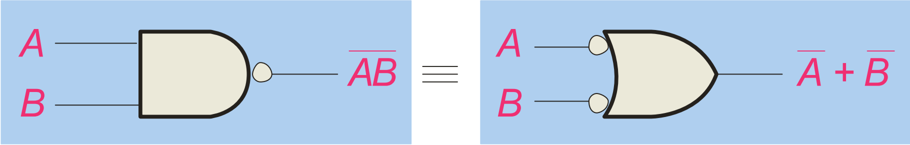
DeMorgan’ s Theorem

DeMorgan’ s 1st Theorem

The complement of a product of variables is equal to the sum of the complemented variables.

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

Applying DeMorgan’ s first theorem to gates:



NAND

Negative-OR

Inputs		Output	
A	B	$\overline{AB}$	$\overline{A} + \overline{B}$
0	0	1	1
0	1	1	1
1	0	1	1
1	1	0	0

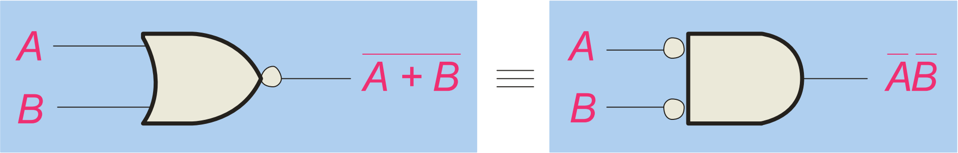
DeMorgan’ s Theorem

DeMorgan’ s 2nd Theorem

The complement of a sum of variables is equal to the product of the complemented variables.

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

Applying DeMorgan’ s second theorem to gates:



NOR

Negative-AND

Inputs		Output	
A	B	$\overline{A + B}$	$\overline{A}\,\overline{B}$
0	0	1	1
0	1	0	0
1	0	0	0
1	1	0	0

DeMorgan’ s Theorem

Example Apply DeMorgan’ s theorem to remove the overbar covering both terms from the expression $X = \overline{\overline{C} + D}$.

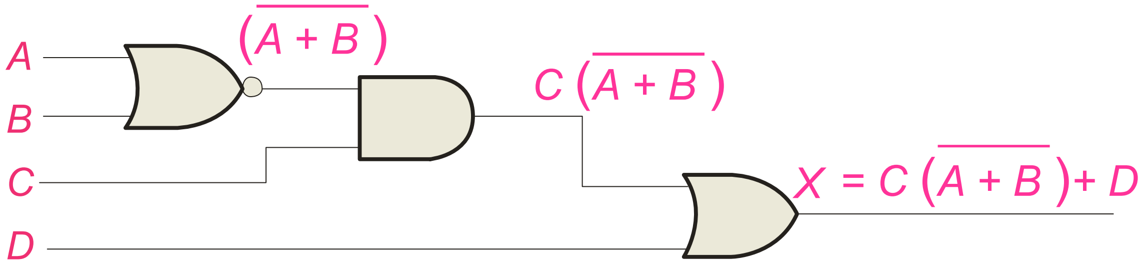
Solution To apply DeMorgan’ s theorem to the expression, you can break the overbar covering both terms and change the sign between the terms. This results in $X = \overline{\overline{C}} \cdot \overline{D}$. Deleting the double bar gives $X = C \cdot \overline{D}$.

Boolean Analysis of Logic Circuits

Combinational logic circuits can be analyzed by writing the expression for each gate and combining the expressions according to the rules for Boolean algebra.

Example Apply Boolean algebra to derive the expression for *X*.

Solution Write the expression for each gate:



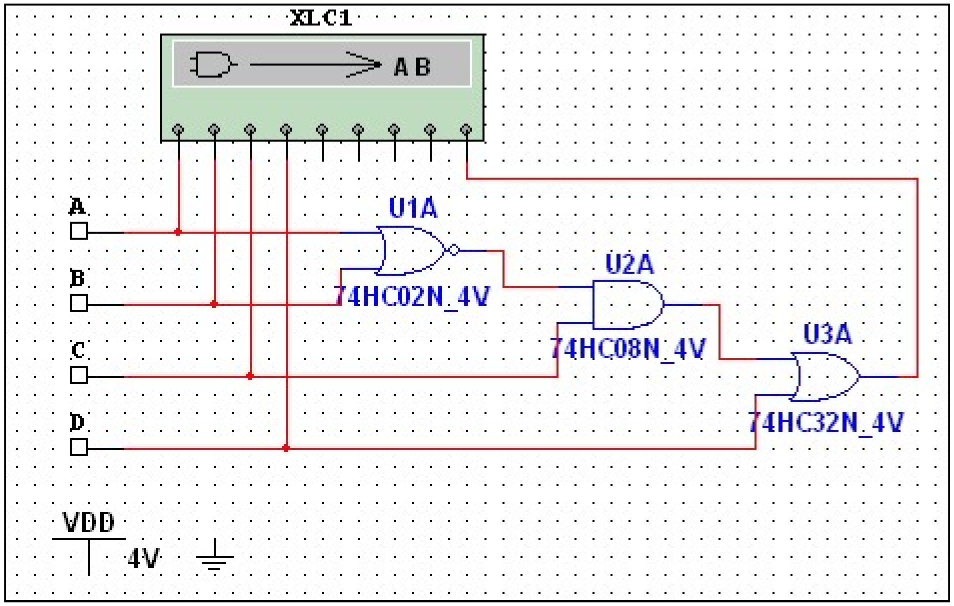
Applying DeMorgan’ s theorem and the distribution law:

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

Boolean Analysis of Logic Circuits

Example Use Multisim to generate the truth table for the circuit in the previous example.

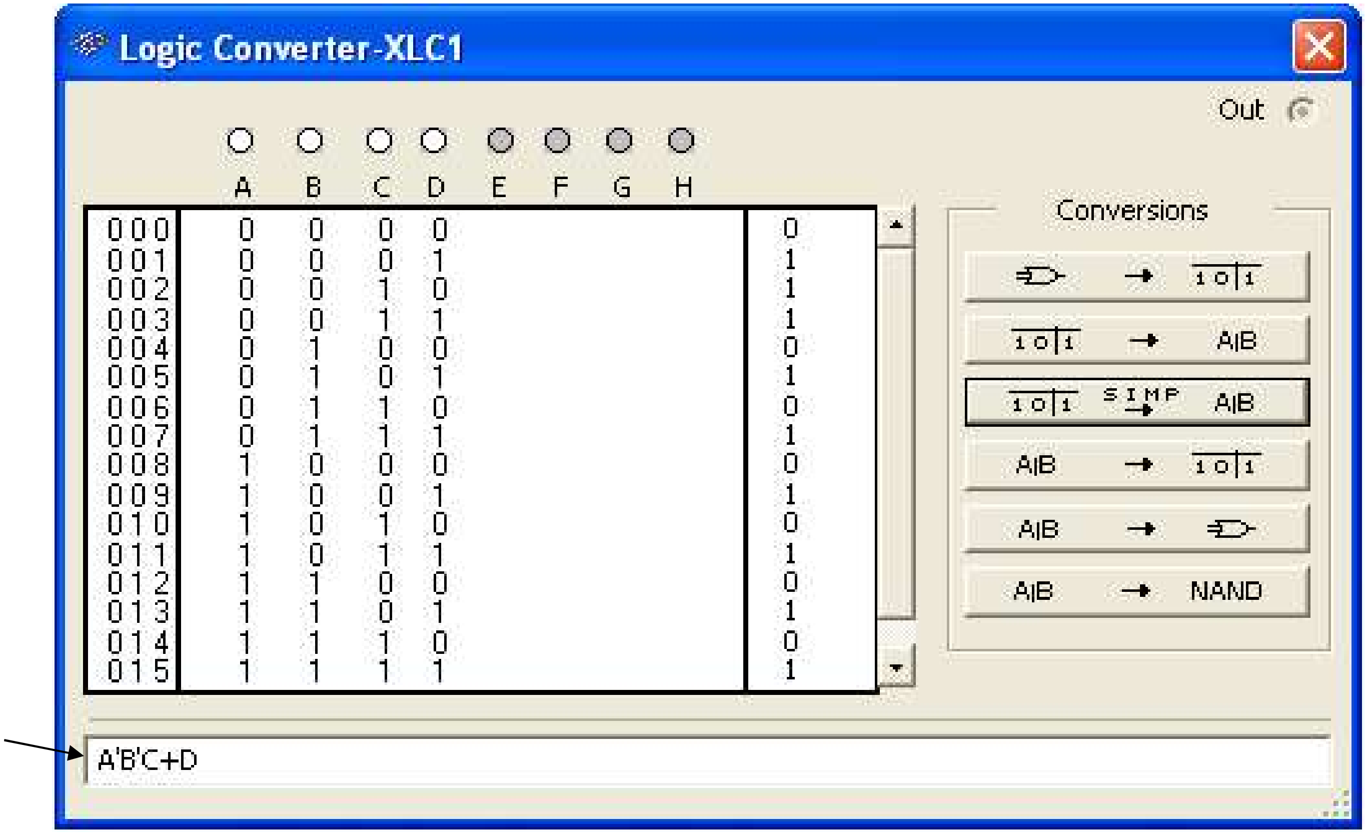
Solution Set up the circuit using the Logic Converter as shown. (Note that the logic converter has no "real-world" counterpart.)



Double-click the Logic Converter top open it. Then click on the conversion bar on the right side to see the truth table for the circuit (see next slide).

Boolean Analysis of Logic Circuits

The simplified logic expression can be viewed by clicking 1 0|1 SIMP→ A|B



	A	B	C	D	
000	0	0	0	0	0
001	0	0	0	1	1
002	0	0	1	0	1
003	0	0	1	1	1
004	0	1	0	0	0
005	0	1	0	1	1
006	0	1	1	0	0
007	0	1	1	1	1
008	1	0	0	0	0
009	1	0	0	1	1
010	1	0	1	0	0
011	1	0	1	1	1
012	1	1	0	0	0
013	1	1	0	1	1
014	1	1	1	0	0
015	1	1	1	1	1

A'B'C+D

Simplified expression

SOP and POS forms

Boolean expressions can be written in the **sum-of-products** form (**SOP**) or in the **product-of-sums** form (**POS**). These forms can simplify the implementation of combinational logic, particularly with PLDs. In both forms, an overbar cannot extend over more than one variable.

An expression is in SOP form when two or more product terms are summed as in the following examples:

$\overline{A}\,\overline{B}\,\overline{C} + A\,B$ $\qquad$ $A\,B\,\overline{C} + \overline{C}\,\overline{D}$ $\qquad$ $C\,D + \overline{E}$

An expression is in POS form when two or more sum terms are multiplied as in the following examples:

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

SOP Standard form

In **SOP standard form**, every variable in the domain must appear in each term. This form is useful for constructing truth tables or for implementing logic in PLDs.

You can expand a nonstandard term to standard form by multiplying the term by a term consisting of the sum of the missing variable and its complement.

Example

Convert $X = \overline{A}\,\overline{B} + A\,B\,C$ to standard form.

Solution

The first term does not include the variable C. Therefore, multiply it by the $(C + \overline{C})$, which = 1:

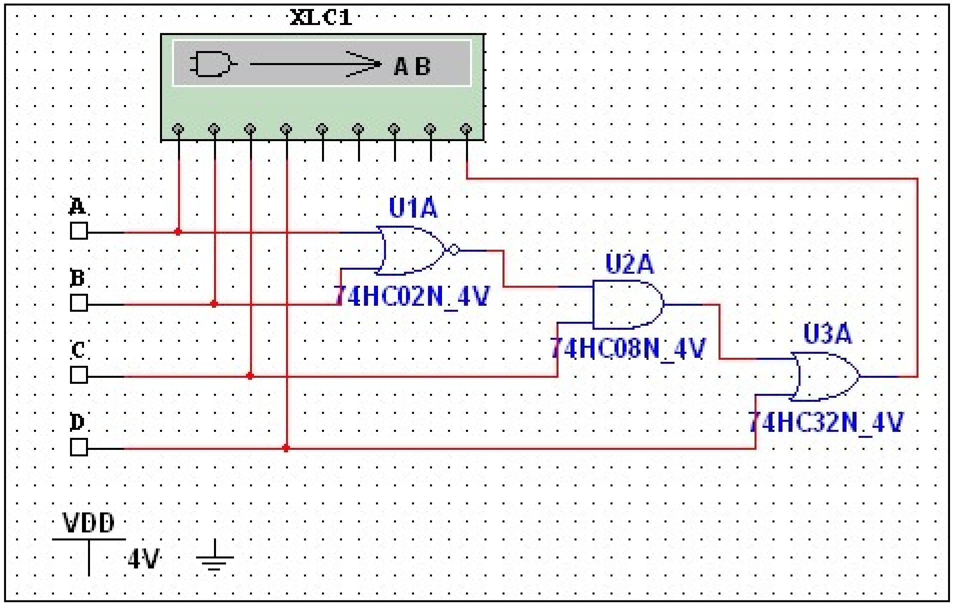
$$X = \overline{A}\,\overline{B}\,(C + \overline{C}) + A\,B\,C$$

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

SOP Standard form

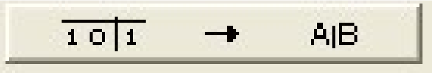
The Logic Converter in Multisim can convert a circuit into standard SOP form.

Example Use Multisim to view the logic for the circuit in standard SOP form.



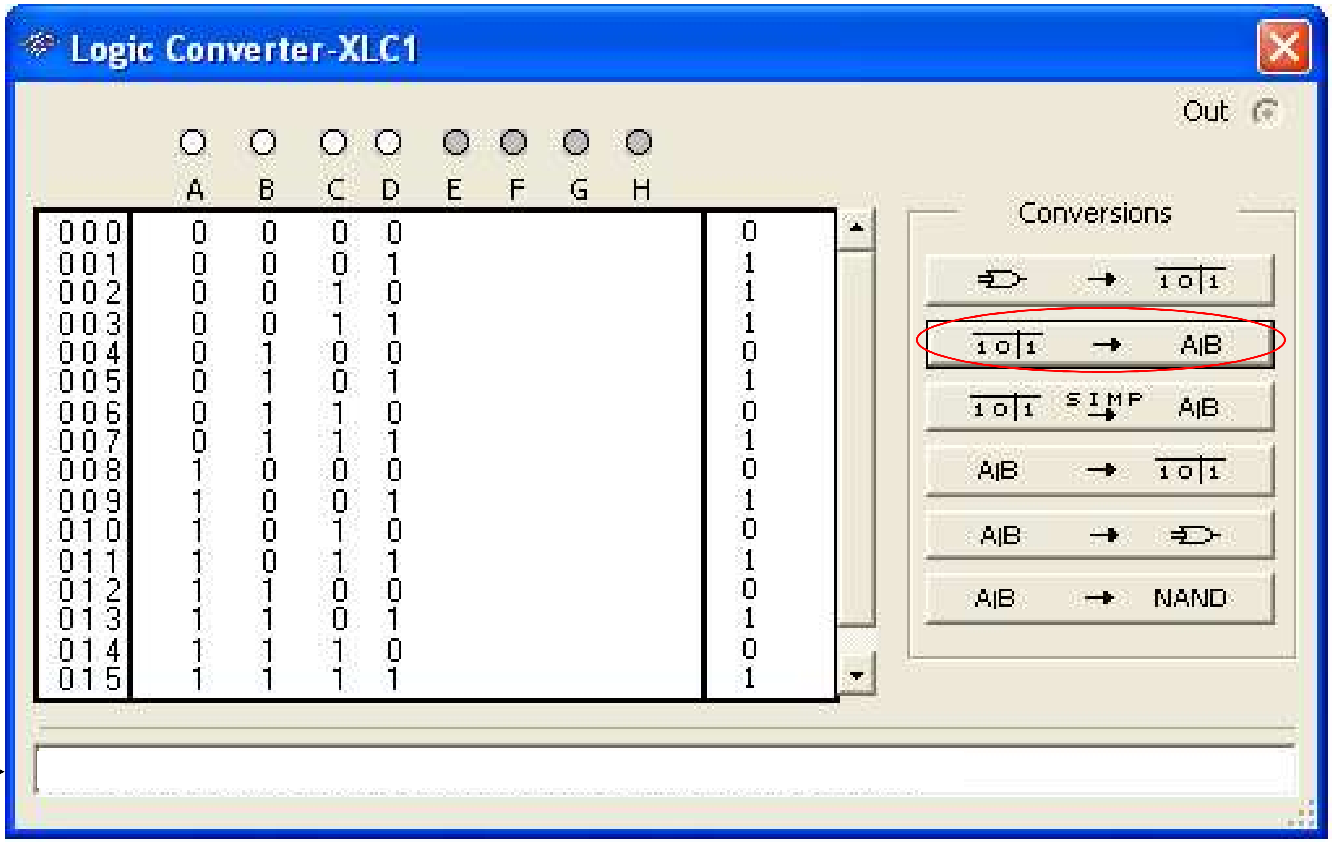
Solution

Click the truth table to logic button on the Logic Converter.



See next slide…

SOP Standard form



	A	B	C	D	E	F	G	H	
000	0	0	0	0					0
001	0	0	0	1					1
002	0	0	1	0					1
003	0	0	1	1					1
004	0	1	0	0					0
005	0	1	0	1					1
006	0	1	1	0					0
007	0	1	1	1					1
008	1	0	0	0					0
009	1	0	0	1					1
010	1	0	1	0					0
011	1	0	1	1					1
012	1	1	0	0					0
013	1	1	0	1					1
014	1	1	1	0					0
015	1	1	1	1					1

SOP
Standard
form →

POS Standard form

In **POS standard form**, every variable in the domain must appear in each sum term of the expression.

You can expand a nonstandard POS expression to standard form by adding the product of the missing variable and its complement and applying rule 12, which states that $(A + B)(A + C) = A + BC$.

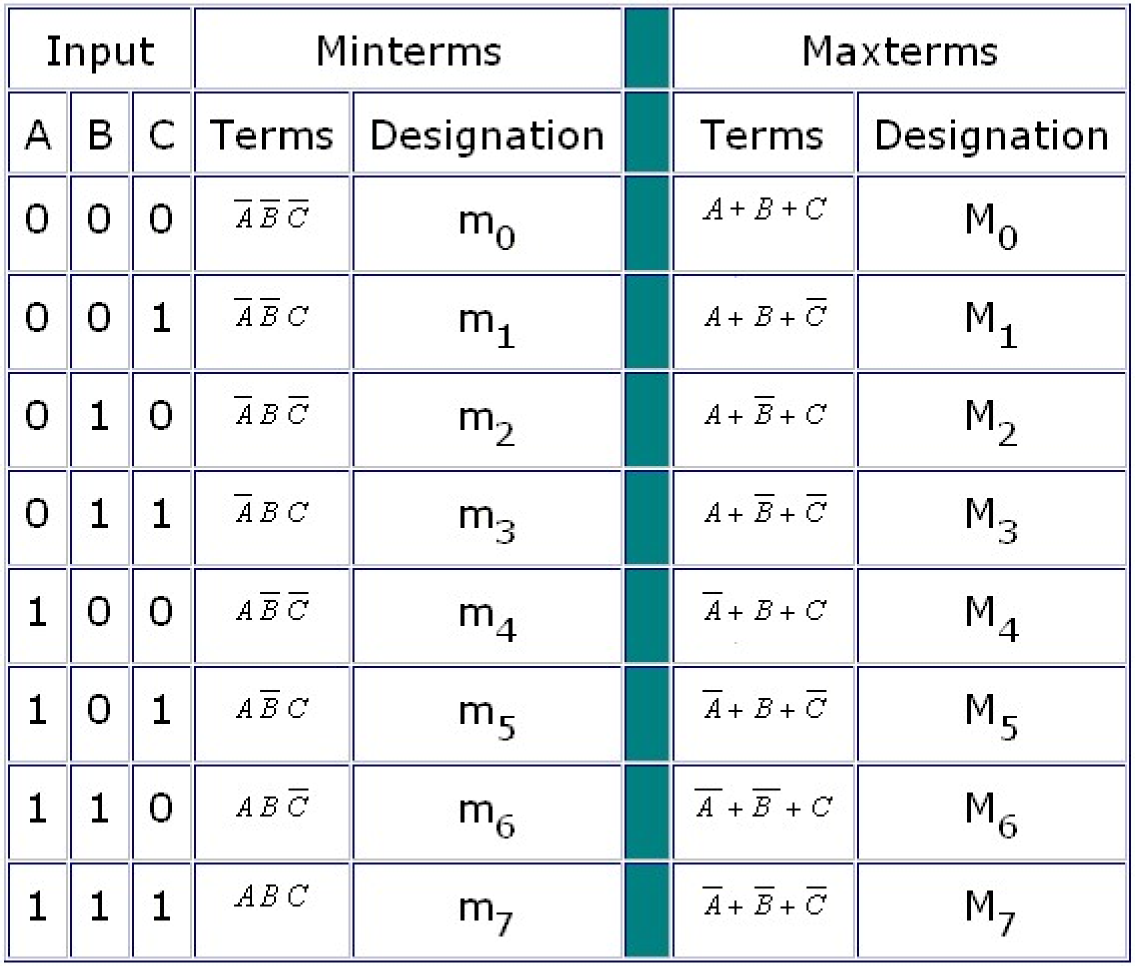
Example Convert $X = (\overline{A} + \overline{B})(A + B + C)$ to standard form.

Solution The first sum term does not include the variable C. Therefore, add $C\,\overline{C}$ and expand the result by rule 12.

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

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

Minterms and Maxterms



Input			Minterms		Maxterms	
A	B	C	Terms	Designation	Terms	Designation
0	0	0	$\bar{A}\bar{B}\bar{C}$	m_0	$A+B+C$	M_0
0	0	1	$\bar{A}\bar{B}C$	m_1	$A+B+\bar{C}$	M_1
0	1	0	$\bar{A}B\bar{C}$	m_2	$A+\bar{B}+C$	M_2
0	1	1	$\bar{A}BC$	m_3	$A+\bar{B}+\bar{C}$	M_3
1	0	0	$A\bar{B}\bar{C}$	m_4	$\bar{A}+B+C$	M_4
1	0	1	$A\bar{B}C$	m_5	$\bar{A}+B+\bar{C}$	M_5
1	1	0	$AB\bar{C}$	m_6	$\bar{A}+\bar{B}+C$	M_6
1	1	1	ABC	m_7	$\bar{A}+\bar{B}+\bar{C}$	M_7

NOTE:

$$m_i = \overline{M_i}$$

Minterms and Maxterms

- Using $\boldsymbol{m_i}$ to represent minterms.

  Sum of minterms expression:

$$f = \sum_i m_i$$

- Using $\boldsymbol{M_i}$ to represent maxterms.

  Product of maxterms expression:

$$f = \prod_i M_i$$

Minterm to Maxterm

$$\begin{tikzcd}
f = \sum_{i} m_i \arrow[r, "{\sum m_i = 1}", "{m_i = \overline{M}_i}"'] & \bar{f} = \sum_{j \neq i} m_j = \sum_{j \neq i} \overline{M}_j \arrow[d] \\
 & f = \prod_{j \neq i} M_j
\end{tikzcd}$$

Theorem

$$\boldsymbol{f = \sum_{i} m_i = \prod_{k \neq i} M_k}$$

Proof:

$$\boldsymbol{f = \sum m_i = \overline{\overline{\sum m_i}} = \overline{\prod \overline{m_i}} = \overline{\prod M_i} = \prod_{k \neq i} M_k}$$

Example

- Convert the following function to

(a) Sum of minterms

(b) Product of maxterms

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

Converting SOP form to Truth Table

Steps:

1. List *all possible combinations* of binary values of the variables in the expression.
2. Convert the SOP form to its standard form.
3. Place a **1** in the output column for each binary value that make the standard SOP expression a 1 and place **0** for all the remaining binary values.

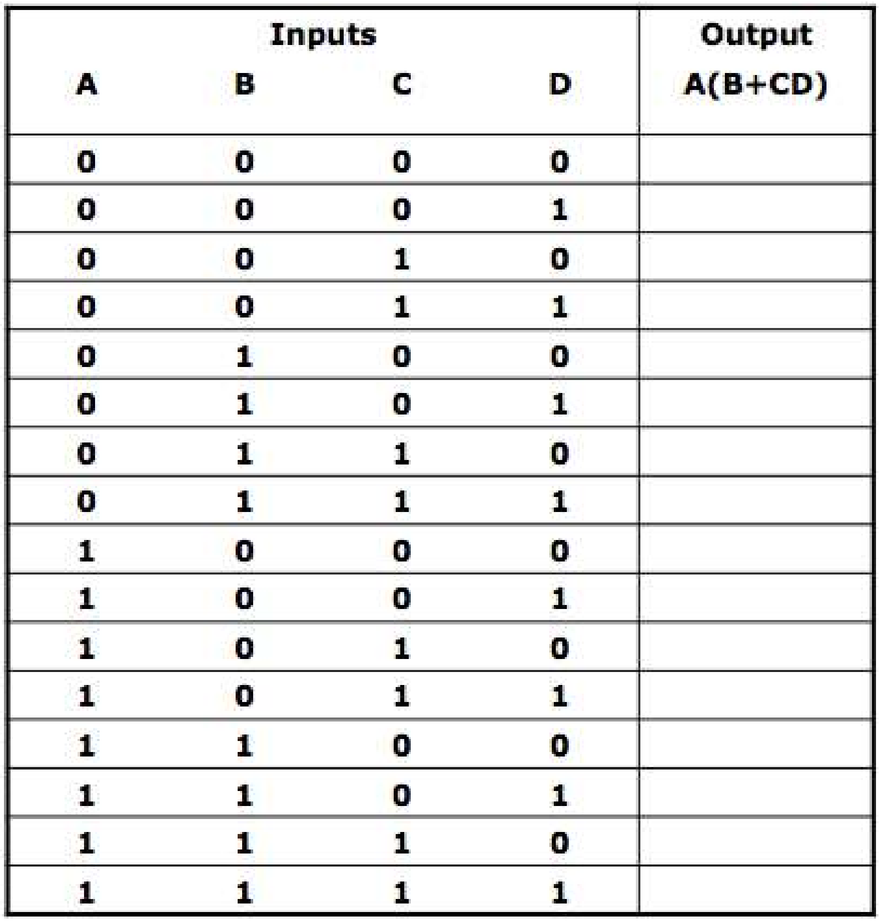
Converting SOP form to Truth Table

Example

X=A(B+CD)

Solution

Step1: List the combination of input variable values.



Inputs				Output
A	B	C	D	A(B+CD)
0	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	0	0	0	
1	0	0	1	
1	0	1	0	
1	0	1	1	
1	1	0	0	
1	1	0	1	
1	1	1	0	
1	1	1	1	

Step2: Convert to standard SOP form.

$$A(B+CD)=AB+ACD$$

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

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

$$=\underline{ABCD+ABC\overline{D}+AB\overline{C}D+AB\overline{C}\overline{D}}+\underline{ABCD+A\overline{B}CD}$$

$$=\underset{\substack{\downarrow\\1111}}{ABCD}+\underset{\substack{\downarrow\\1110}}{ABC\overline{D}}+\underset{\substack{\downarrow\\1101}}{AB\overline{C}D}+\underset{\substack{\downarrow\\1100}}{AB\overline{C}\overline{D}}+\underset{\substack{\downarrow\\1011}}{A\overline{B}CD}$$

Step3: Place 1s and 0s in truth table.

	Inputs				Output
	A	B	C	D	A(B+CD)
	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	0
	1	0	0	0	0
	1	0	0	1	0
	1	0	1	0	0
→	1	0	1	1	1
→	1	1	0	0	1
→	1	1	0	1	1
→	1	1	1	0	1
→	1	1	1	1	1

Converting POS form to Truth Table

Steps:

1. List all possible combinations of binary values of the variables in the expression.
2. Convert the POS form to its standard form.
3. Place a **0** in the output column for each binary value that make the standard POS expression a 0 and place **1** for all the remaining binary values.

Converting POS form to Truth Table

Example Determine the truth table for following expression:

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

Solution

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

⬇ ⬇ ⬇ ⬇ ⬇

000 010 011 101 110

And the Truth Table on the next page…

Converting POS form to Truth Table

Example Determine the truth table for following expression:

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

Solution

Input			Output
A	B	C	x
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Converting Truth Table to SOP/POS standard form

Example

Determine the SOP/POS of x:

Input			Output
A	B	C	x
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Solution

SOP std. form:

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

$$= m_3 + m_4 + m_6 + m_7$$

$$= \sum(3, 4, 6, 7)$$

POS std. form:

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

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

$$= M_0 \cdot M_1 \cdot M_2 \cdot M_5$$

$$= \prod(0,1,2,5)$$

Converting Truth Table to SOP/POS standard form

Example

Determine the SOP/POS of x:

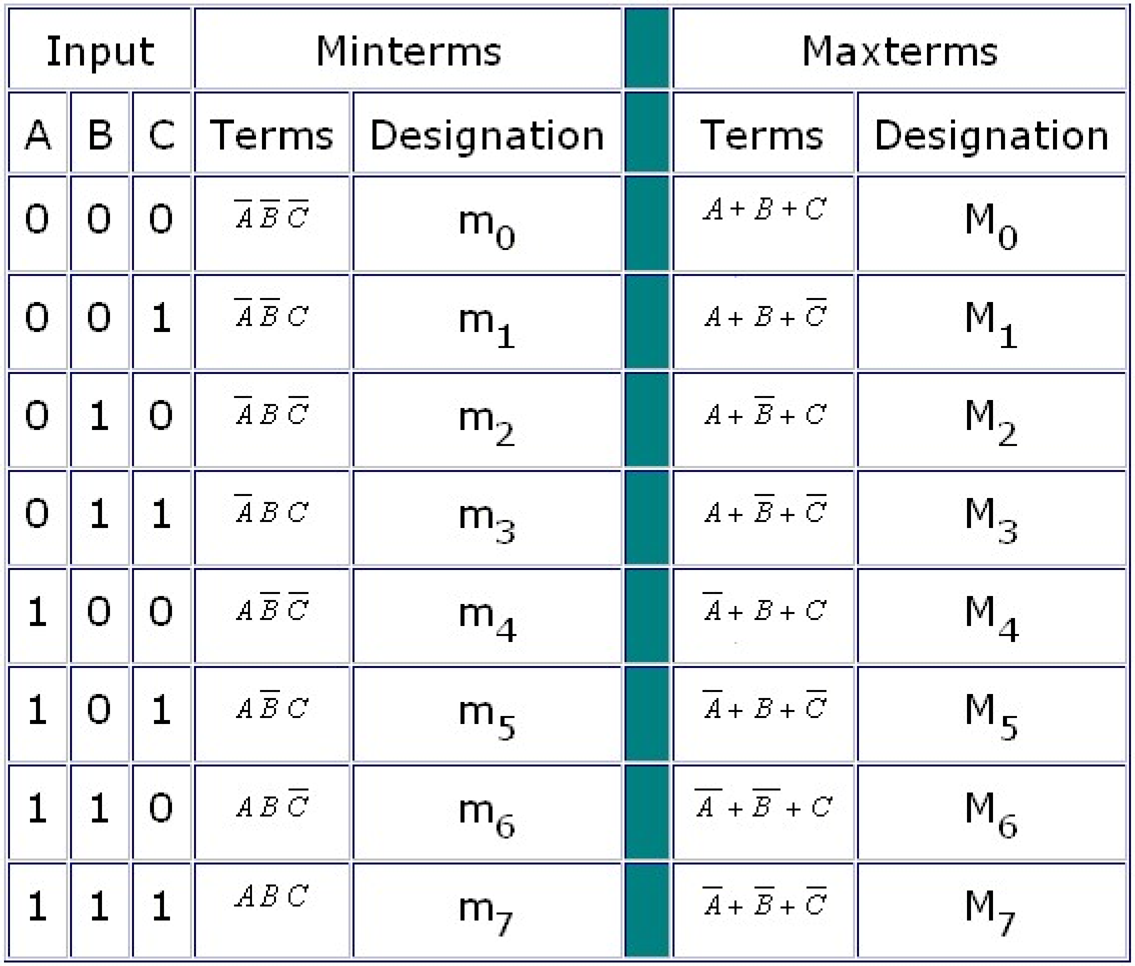
Input			Output
A	B	C	x
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Solution

SOP std. form:

POS std. form:

Minterms and Maxterms



Input			Minterms			Maxterms	
A	B	C	Terms	Designation		Terms	Designation
0	0	0	$\bar{A}\bar{B}\bar{C}$	m_0		$A+B+C$	M_0
0	0	1	$\bar{A}\bar{B}C$	m_1		$A+B+\bar{C}$	M_1
0	1	0	$\bar{A}B\bar{C}$	m_2		$A+\bar{B}+C$	M_2
0	1	1	$\bar{A}BC$	m_3		$A+\bar{B}+\bar{C}$	M_3
1	0	0	$A\bar{B}\bar{C}$	m_4		$\bar{A}+B+C$	M_4
1	0	1	$A\bar{B}C$	m_5		$\bar{A}+B+\bar{C}$	M_5
1	1	0	$AB\bar{C}$	m_6		$\bar{A}+\bar{B}+C$	M_6
1	1	1	ABC	m_7		$\bar{A}+\bar{B}+\bar{C}$	M_7

Logic Expressions and Truth Table

Example

Determine the Truth Table of X according to the std. SOP:

$$\underset{(1)}{ABC} + \underset{(2)}{\overline{A}\,\overline{B}C} + \underset{(3)}{\overline{A}B\overline{C}}$$

Solution

Input			Output	
A	B	C		
0	0	0	0	
0	0	1	1	(2)
0	1	0	1	(3)
0	1	1	0	
1	0	0	0	
1	0	1	0	
1	1	0	0	
1	1	1	1	(1)

Logic Expressions and Truth Table

Example

Determine the Truth Table of X according to the std. SOP:

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

(1) (2) (3) (4)

Solution

Input			Output	
A	B	C		
0	0	0	1	(2)
0	0	1	0	
0	1	0	1	(3)
0	1	1	0	
1	0	0	0	
1	0	1	1	(1)
1	1	0	0	
1	1	1	1	(4)

Logic Expressions and Truth Table

Example

Determine the Truth Table of X according to the POS:

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

Solution

Input			Output
A	B	C	
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

Logic Expressions and Truth Table

Example

Determine the Truth Table of X according to the std. POS:

$$X = \underset{(1)}{(A+B+C)}\underset{(2)}{(\overline{A}+B+\overline{C})}$$

Solution

Input			Output	
A	B	C		
0	0	0	0	(1)
0	0	1	1	
0	1	0	1	
0	1	1	1	
1	0	0	1	
1	0	1	0	(2)
1	1	0	1	
1	1	1	1	

Logic Expressions and Truth Table

Example

Determine the std. SOP of X according to the Truth Table:

Input			Output X	
A	B	C		
0	0	0	1	(1)
0	0	1	0	
0	1	0	1	(2)
0	1	1	0	
1	0	0	0	
1	0	1	1	(3)
1	1	0	0	
1	1	1	1	(4)

Solution

$$\underset{(1)}{\overline{A}\,\overline{B}\,\overline{C}} + \underset{(2)}{\overline{A}B\overline{C}} + \underset{(3)}{A\overline{B}C} + \underset{(4)}{ABC}$$

Logic Expressions and Truth Table

Example

Determine the std. POS of X according to the Truth Table:

Input			Output X	
A	B	C		
0	0	0	1	
0	0	1	0	(1)
0	1	0	1	
0	1	1	0	(2)
1	0	0	0	(3)
1	0	1	1	
1	1	0	0	(4)
1	1	1	1	

Solution

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

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

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

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

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

Logic Simplification

- Why simplification?
  - In software design, simpler logic means less program branches and simpler codes
  - In hardware design, simpler logic means less gates, lower cost and less energy consumption
- Ways to simplify logic functions
  - With rules in Boolean Algebra
  - With Karnaugh map

Logic Simplification

Example

Simplify the logic function below.

AB+A(B+C)+B(B+C)

Solution

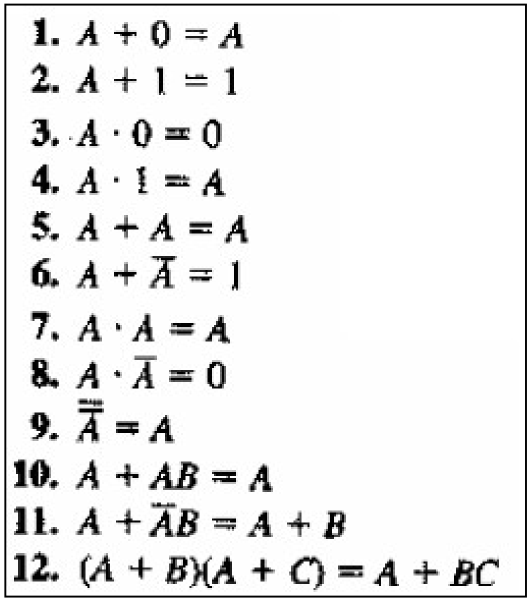
AB+A(B+C)+B(B+C)

=AB+AB+AC+BB+BC

=AB+AC+B+BC

=B(A+1+C)+AC

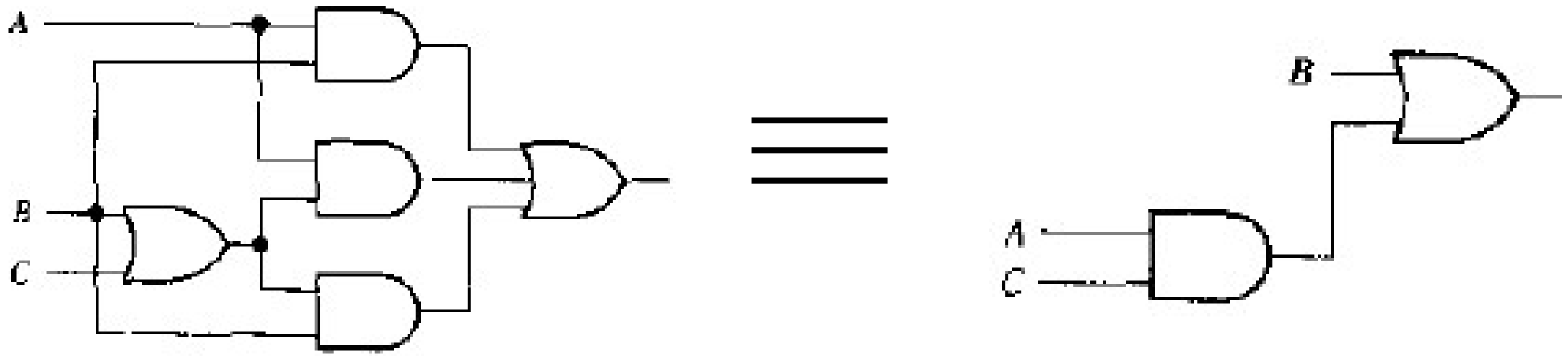
=B+AC



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

Logic Simplification

- AB+A(B+C)+B(B+C)=B+AC



Logic Simplification

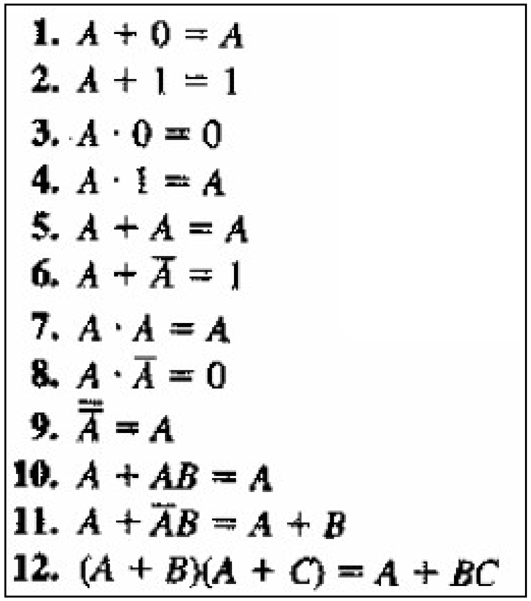
Example

Simplify the logic function below.

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

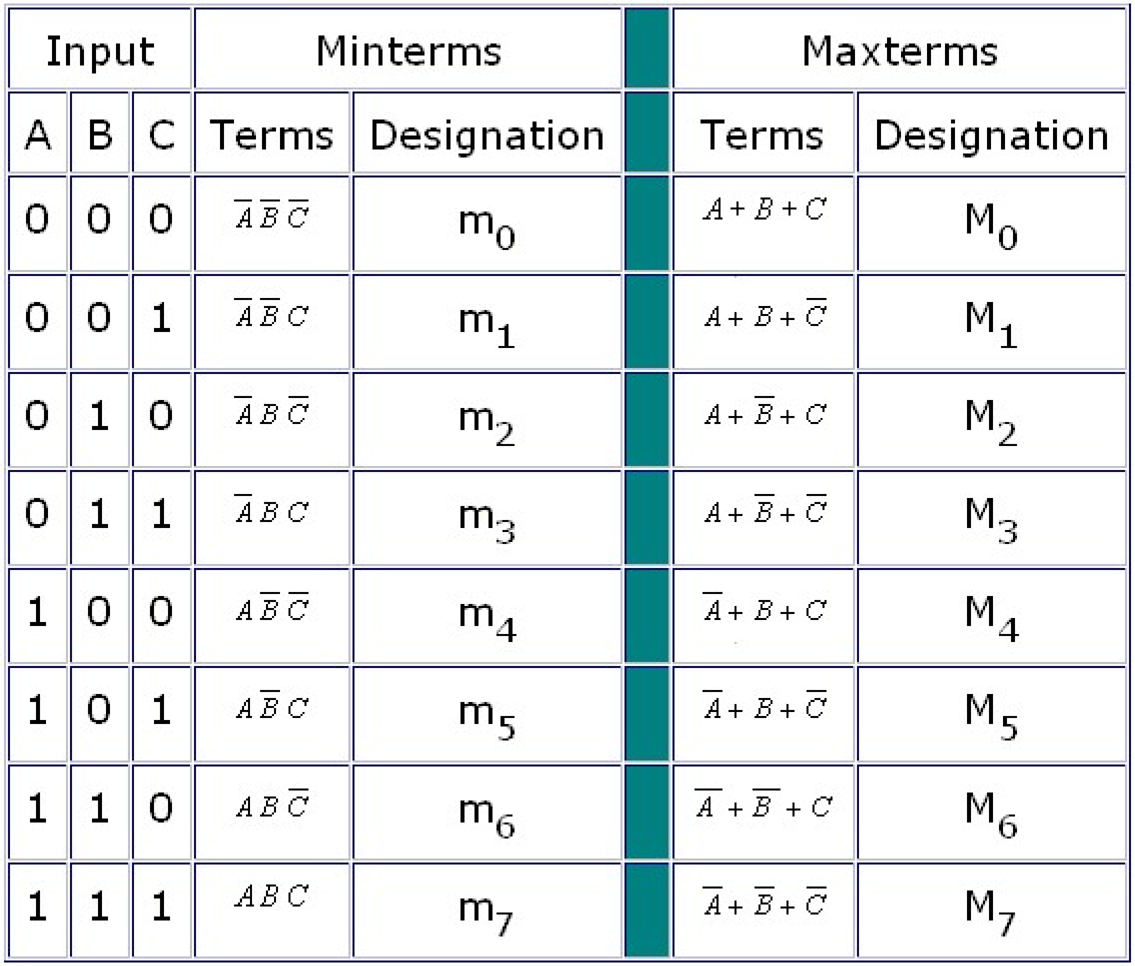
Solution

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



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

Minterms and Maxterms



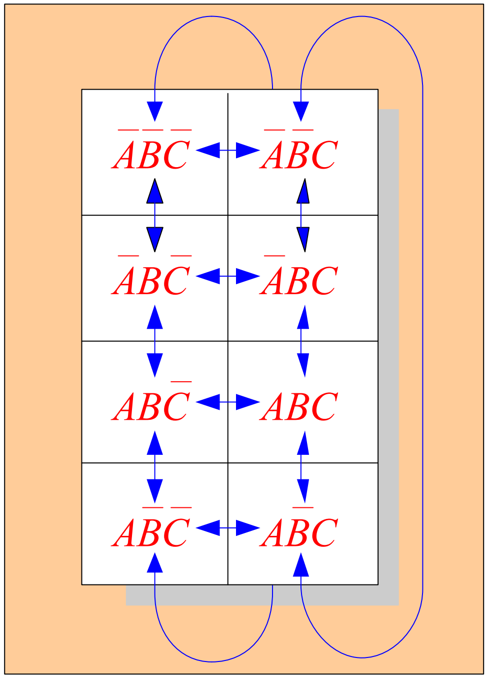
Input			Minterms			Maxterms	
A	B	C	Terms	Designation		Terms	Designation
0	0	0	$\bar{A}\bar{B}\bar{C}$	m_0		$A+B+C$	M_0
0	0	1	$\bar{A}\bar{B}C$	m_1		$A+B+\bar{C}$	M_1
0	1	0	$\bar{A}B\bar{C}$	m_2		$A+\bar{B}+C$	M_2
0	1	1	$\bar{A}BC$	m_3		$A+\bar{B}+\bar{C}$	M_3
1	0	0	$A\bar{B}\bar{C}$	m_4		$\bar{A}+B+C$	M_4
1	0	1	$A\bar{B}C$	m_5		$\bar{A}+B+\bar{C}$	M_5
1	1	0	$AB\bar{C}$	m_6		$\bar{A}+\bar{B}+C$	M_6
1	1	1	ABC	m_7		$\bar{A}+\bar{B}+\bar{C}$	M_7

Karnaugh maps

The Karnaugh map (K-map) is a tool for simplifying combinational logic with 3 or 4 variables. For 3 variables, 8 cells are required (2^3).

The map shown is for three variables labeled *A, B,* and *C*. Each cell represents one possible product term.

Each cell differs from an adjacent cell by only one variable.



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

Karnaugh maps and the truth table

For any logic function *F*(A, B, C), its truth table is:

Input			**Output *F***
A	**B**	**C**	
0	0	0	*F(0,0,0)*
0	0	1	*F(0,0,1)*
0	1	0	*F(0,1,0)*
0	1	1	*F(0,1,1)*
1	0	0	*F(1,0,0)*
1	0	1	*F(1,0,1)*
1	1	0	*F(1,1,0)*
1	1	1	*F(1,1,1)*

Then it has a K-map:

$BC \backslash A$	0	1
00	$F(0,0,0)$	$F(1,0,0)$
01	$F(0,0,1)$	$F(1,0,1)$
11	$F(0,1,1)$	$F(1,1,1)$
10	$F(0,1,0)$	$F(1,1,0)$

Karnaugh maps and the truth table

For a logic function *F*(A, B, C) with truth table:

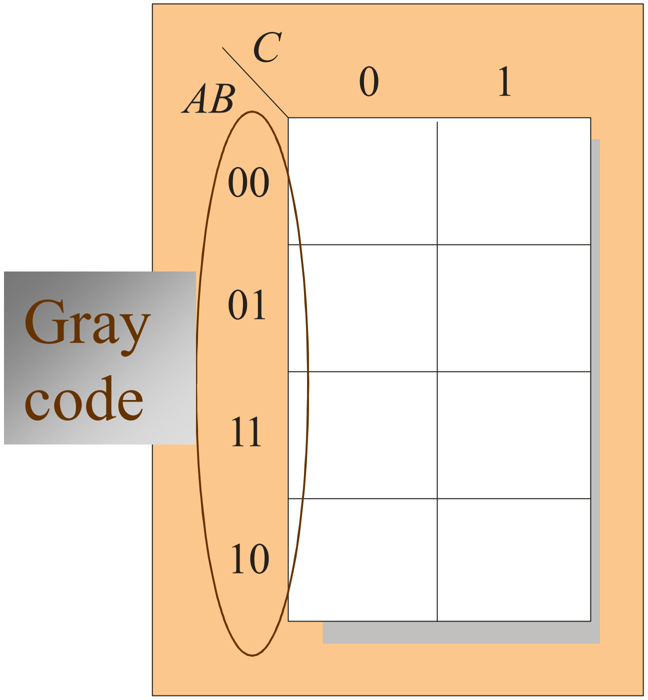
Input			Output *F*
A	**B**	**C**	
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Then it has a K-map:

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

Karnaugh maps

Cells are usually labeled using 0’ s and 1’ s to represent the variable and its complement.



Gray code

The numbers are entered in gray code, to force adjacent cells to be different by only one variable.

Ones are read as the true variable and zeros are read as the complemented variable.

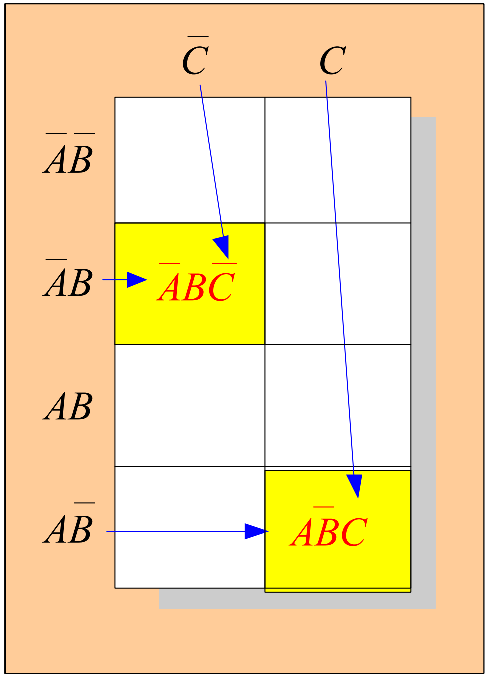
Karnaugh maps

Alternatively, cells can be labeled with the variable letters. This makes it simple to read, but it takes more time preparing the map.

Example Read the terms for the yellow cells.

Solution

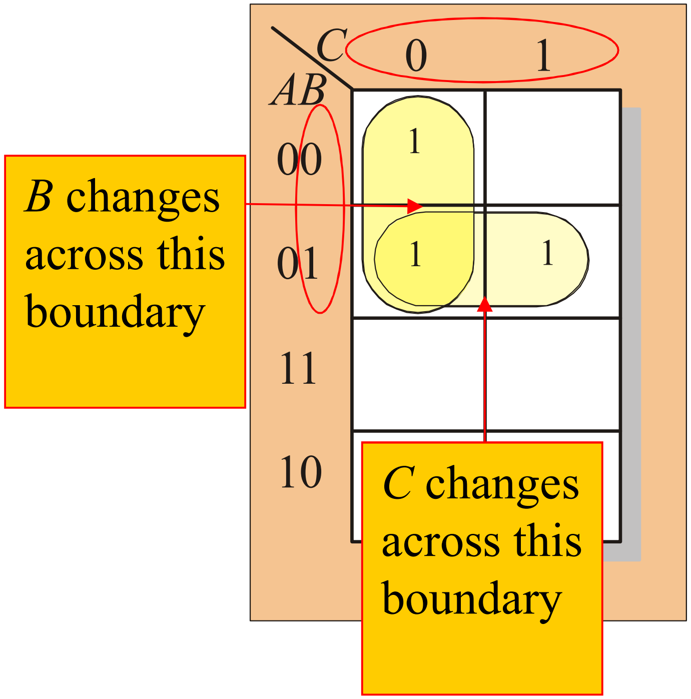
The cells are $\overline{A}B\overline{C}$ and $A\overline{B}C$.



Karnaugh maps

K-maps can simplify combinational logic by grouping cells and eliminating variables that change.

Example Group the 1's on the map and read the minimum logic.



AB \ C	0	1
00	1	
01	1	1
11		
10		

B changes across this boundary

C changes across this boundary

Solution

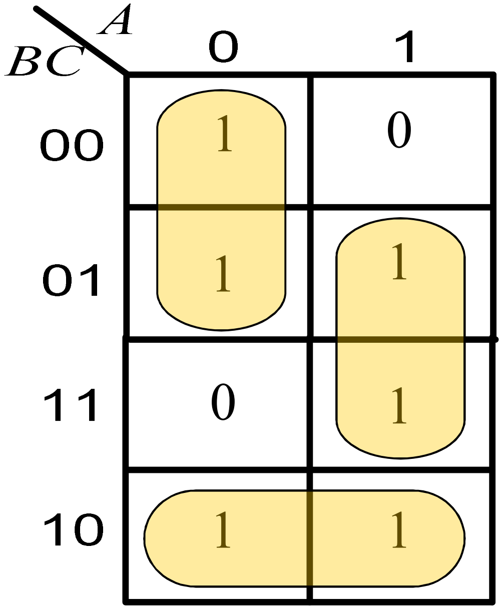
1. Group the 1's into two overlapping groups as indicated.
2. Read each group by eliminating any variable that changes across a boundary.
3. The vertical group is read $\overline{A}\,\overline{C}$.
4. The horizontal group is read $\overline{A}B$.

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

Karnaugh maps

Example

Simplify the logic function with K-map:



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

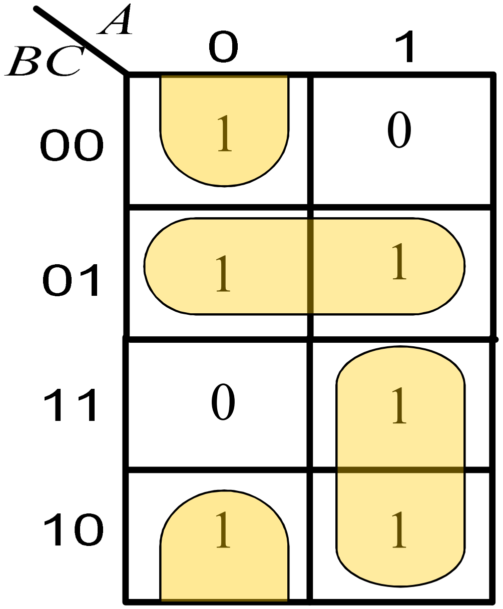
Solution

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

Karnaugh maps

Example

Simplify the logic function with K-map:



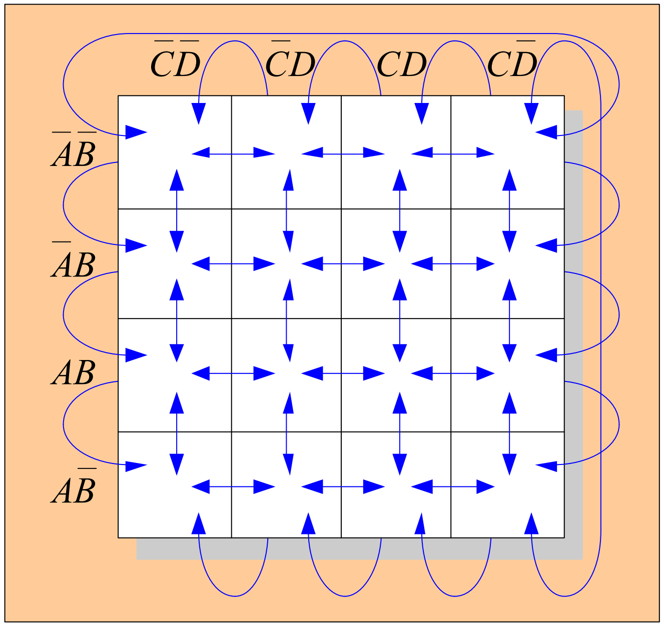
$BC \backslash A$	0	1
00	1	0
01	1	1
11	0	1
10	1	1

Solution

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

Karnaugh maps

A 4-variable map has an adjacent cell on each of its four boundaries as shown.



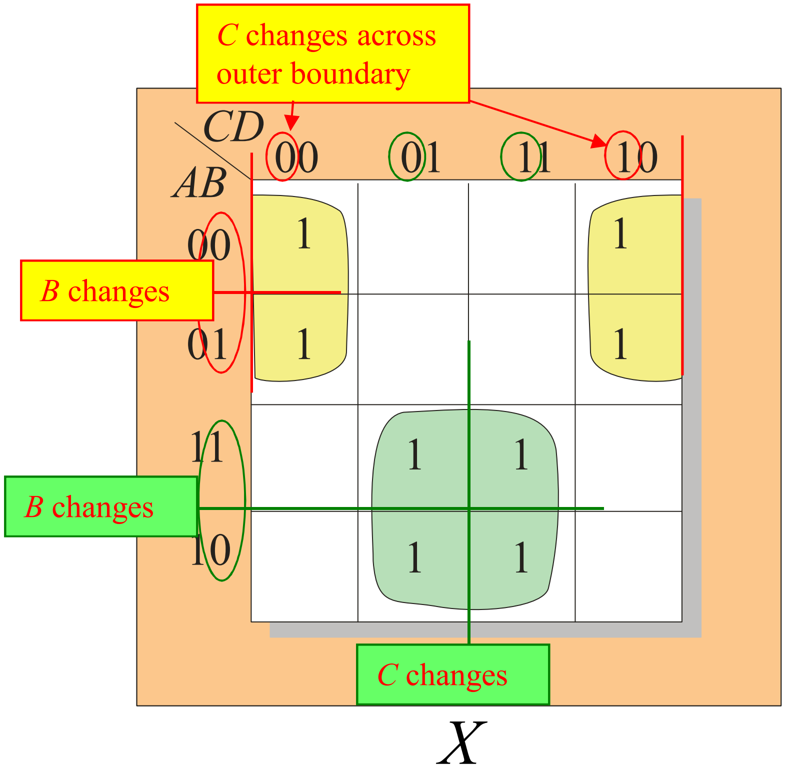
Each cell is different only by one variable from an adjacent cell.

Grouping follows the rules given in the text.

The following slide shows an example of reading a four variable map using binary numbers for the variables…

Karnaugh maps

Example Group the 1’ s on the map and read the minimum logic.



Solution

1. Group the 1’ s into two separate groups as indicated.
2. Read each group by eliminating any variable that changes across a boundary.
3. The upper (yellow) group is read as $\overline{A}\overline{D}$.
4. The lower (green) group is read as AD.

$$X = \overline{A}\overline{D} + AD$$

Simplifying functions with don't care terms

- Don't Care Terms
  - Some input combinations may never appear
  - For example, BCD input never produce "1100"

- How to handle Don't Care Terms
  - May be treated as 1, or 0
  - Often decisions are tend to result in simpler logic

Simplifying functions with don’t care terms

Input				Output
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	1	1	1	X

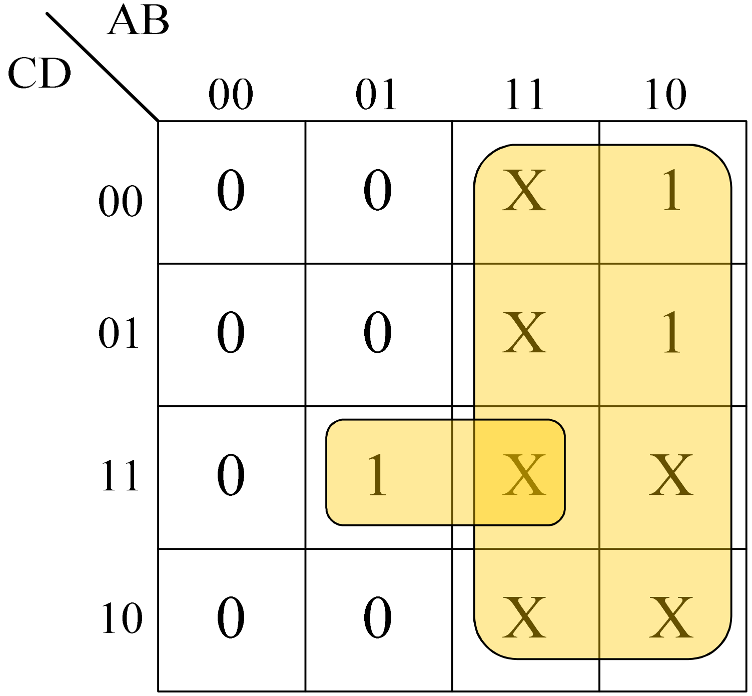
CD \ AB	00	01	11	10
00	0	0	X	1
01	0	0	X	1
11	0	1	X	X
10	0	0	X	X

Treating the ‘X’s as ‘0’s

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

Simplifying functions with don’t care terms

Input				Output
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	1	1	1	X



CD \ AB	00	01	11	10
00	0	0	X	1
01	0	0	X	1
11	0	1	X	X
10	0	0	X	X

Treating the ‘X’ as ‘1’s

$$BCD + A$$