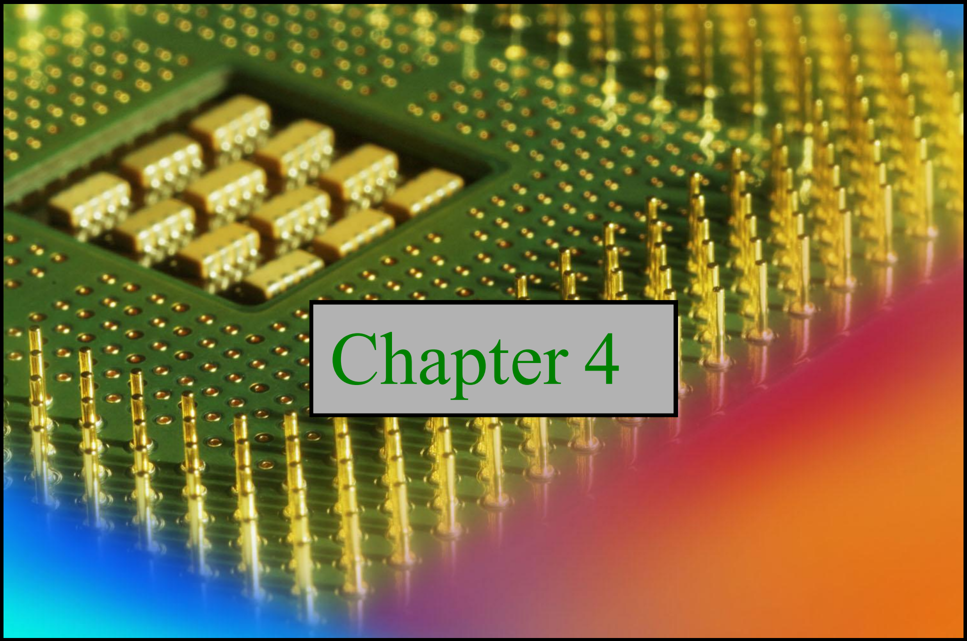
Digital Fundamentals

Tenth Edition

Floyd



Chapter 4

Summary

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 $C = 1$.

Summary

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 $C = 0$.

Summary

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.

$$\boxed{AB = BA}$$

Summary

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.

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

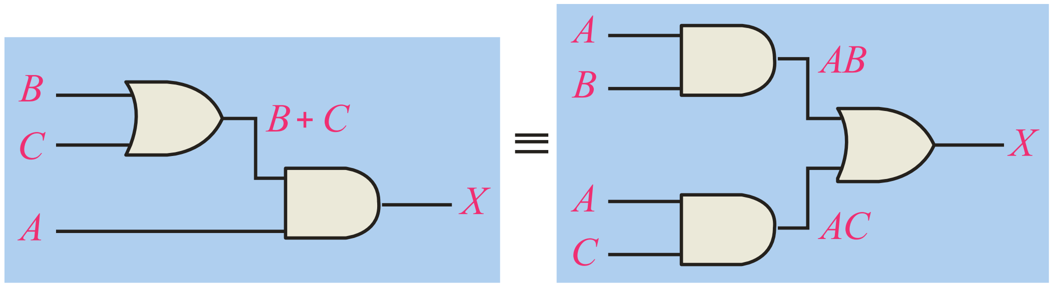
Summary

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:



$A(B+C)$ $\quad$ $AB + AC$

Summary

Rules of Boolean Algebra

1. $A + 0 = A$
2. $A + 1 = 1$
3. $A \cdot 0 = 0$
4. $A \cdot 1 = 1$
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. $(A + B)(A + C) = A + BC$

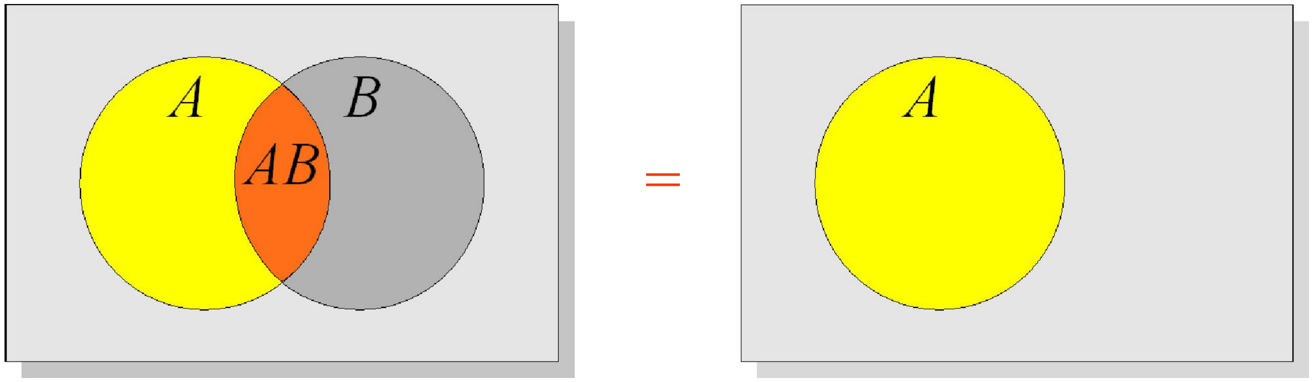
Summary

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.

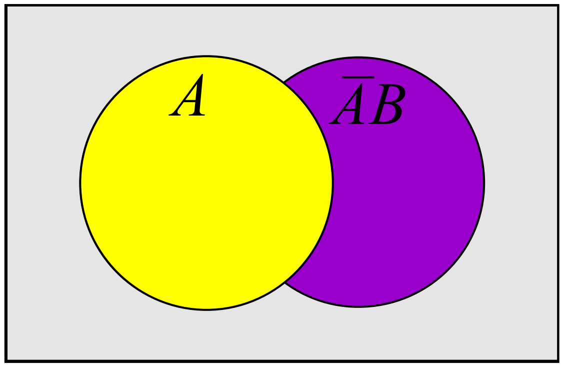
Summary

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



Summary

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…

Summary

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

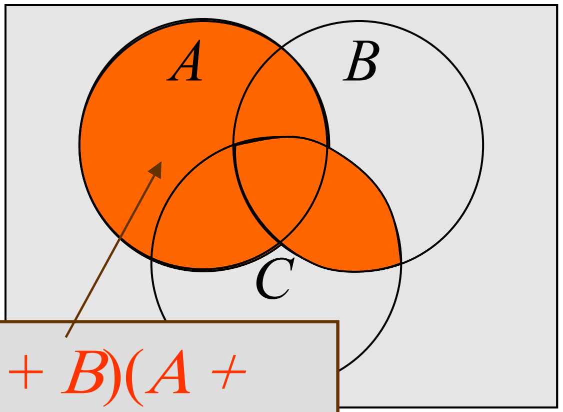
The area representing $A + B$ is shown in yellow.

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

The overlap of red and yellow is shown in orange.

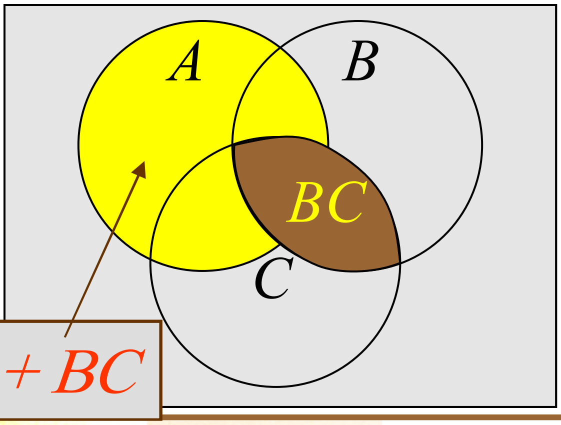
The overlapping area between B and C represents BC.

ORing with A gives the same area as before.



$(A + B)(A + C)$

$=$



$A + BC$

Summary

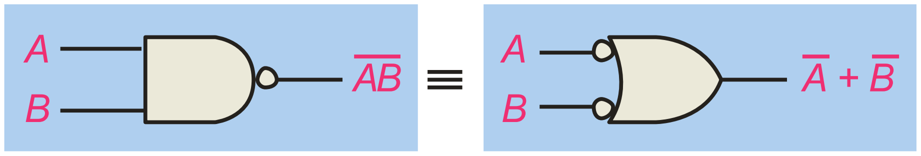
DeMorgan's Theorem

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

Summary

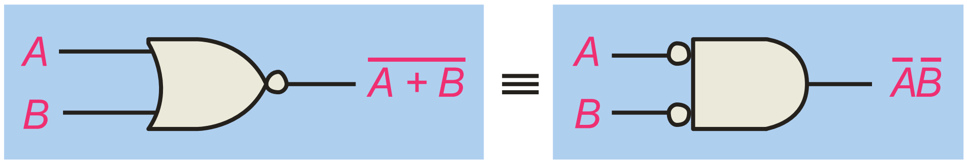
DeMorgan's Theorem

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

Summary

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 $X = C \cdot \overline{D}$.

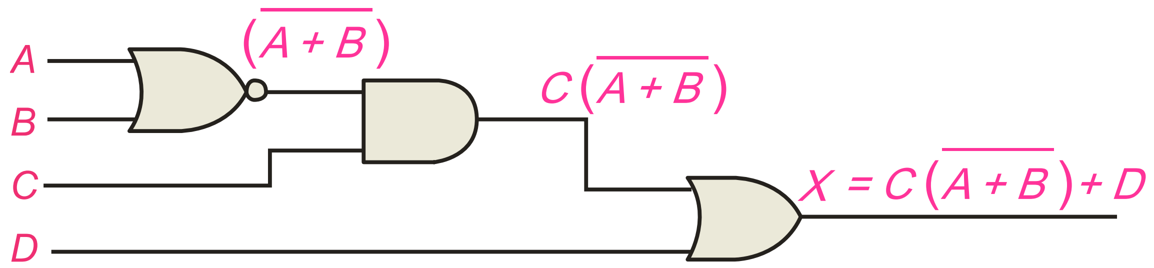
Summary

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:

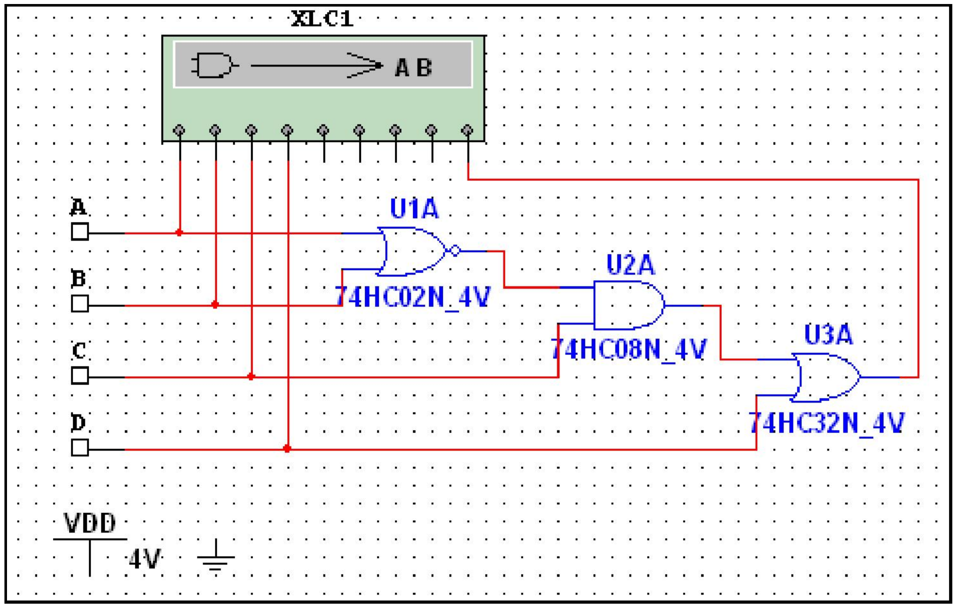
$$X = C(\overline{A}\ \overline{B}) + D = \overline{A}\,\overline{B}\,C + D$$

Summary

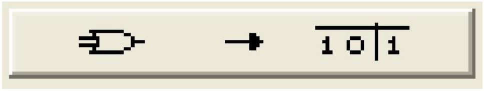
Boolean Analysis of Logic Circuits

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.)



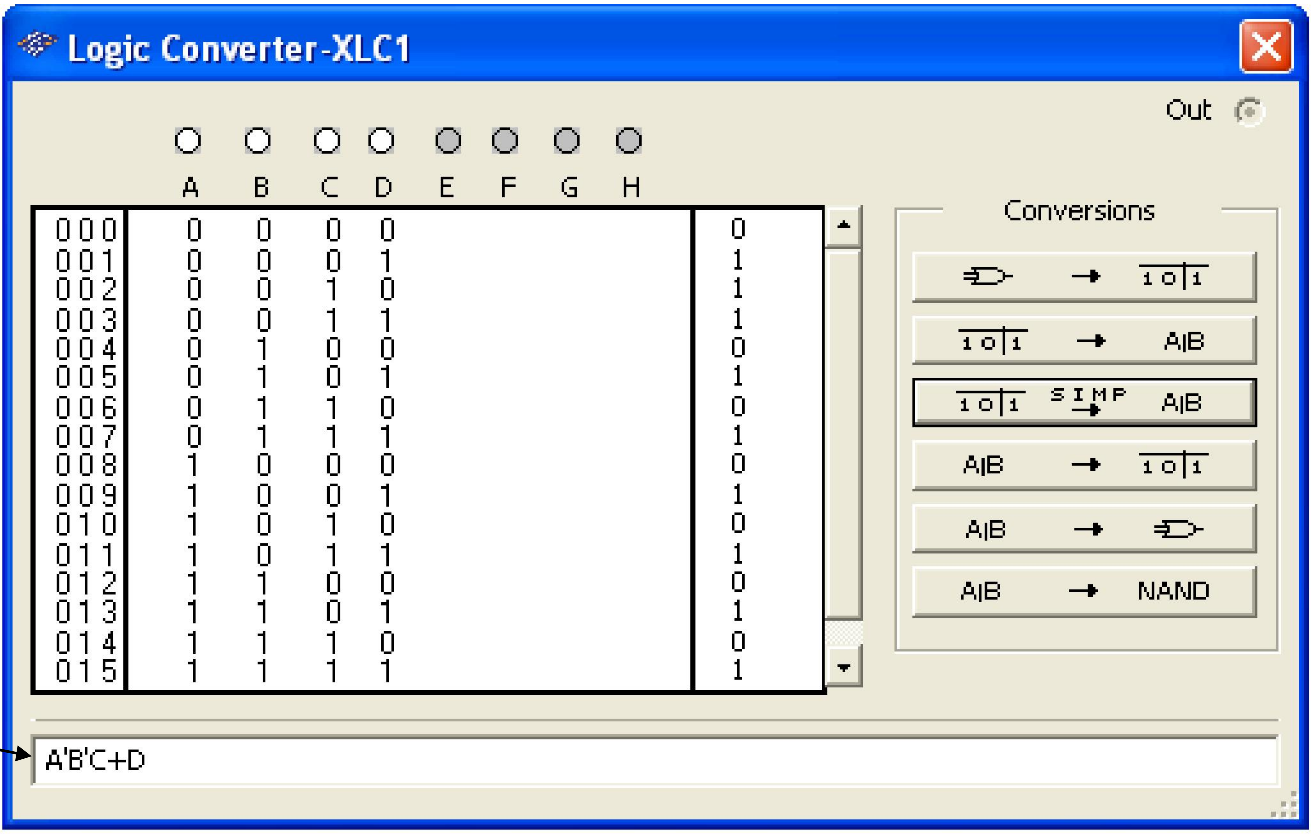
Double-click the Logic Converter top open it. Then click on the conversion bar on the right side to see the truth table for the circuit (see next slide).



Summary

Boolean Analysis of Logic Circuits

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



	A	B	C	D	E	F	G	H	Out
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

Simplified expression → A'B'C+D

Summary

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:

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

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

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

Summary

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:

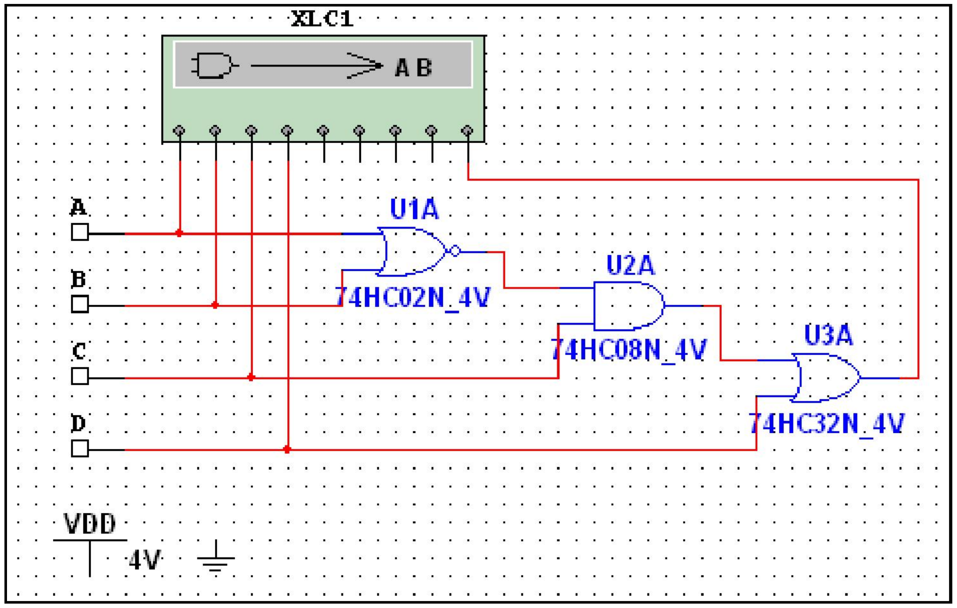
$$X = \overline{A}\,\overline{B}(C + \overline{C}) + A\,B\,C$$
$$= \overline{A}\,\overline{B}\,C + \overline{A}\,\overline{B}\,\overline{C} + A\,B\,C$$

Summary

SOP Standard form

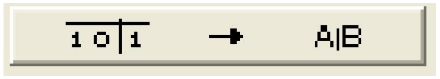
The Logic Converter in Multisim can convert a circuit into standard SOP form.

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



Solution

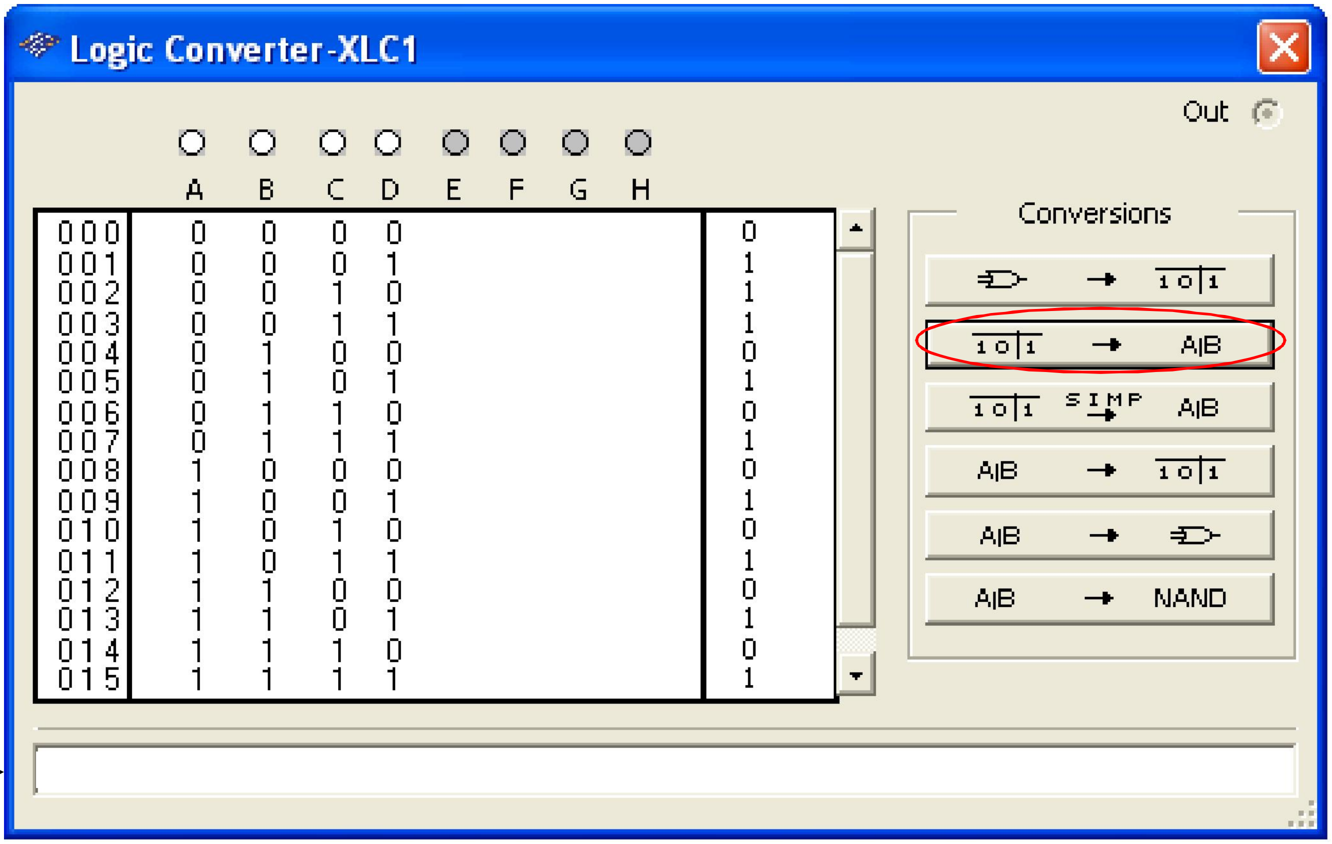
Click the truth table to logic button on the Logic Converter.



See next slide…

Summary

SOP Standard form



	A	B	C	D	Out
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 $\longrightarrow$

Summary

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.

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

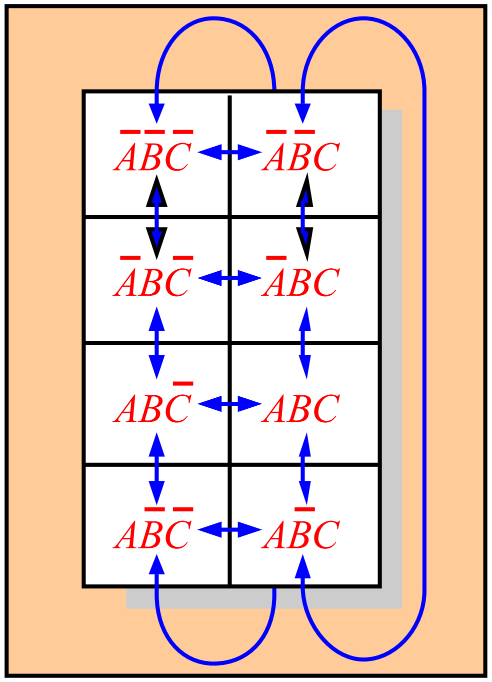
Summary

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.

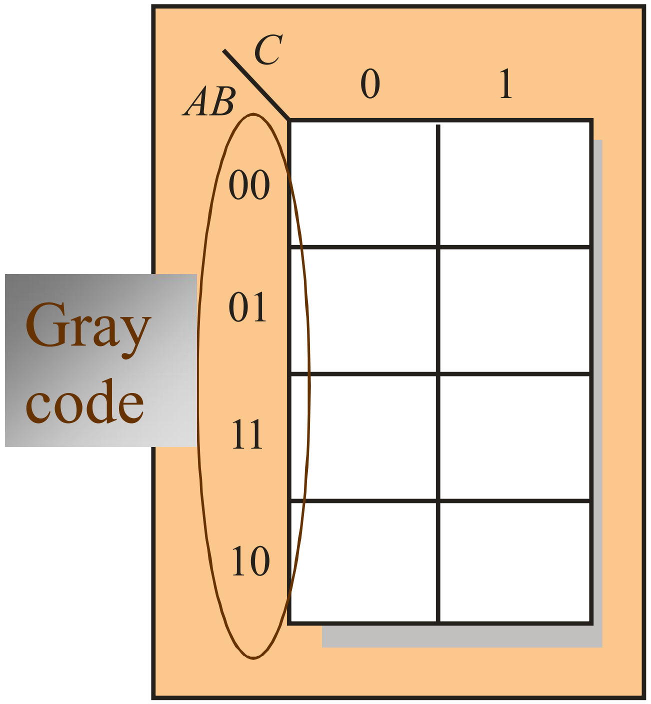
Each cell differs from an adjacent cell by only one variable.



Summary

Karnaugh maps

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



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.

Summary

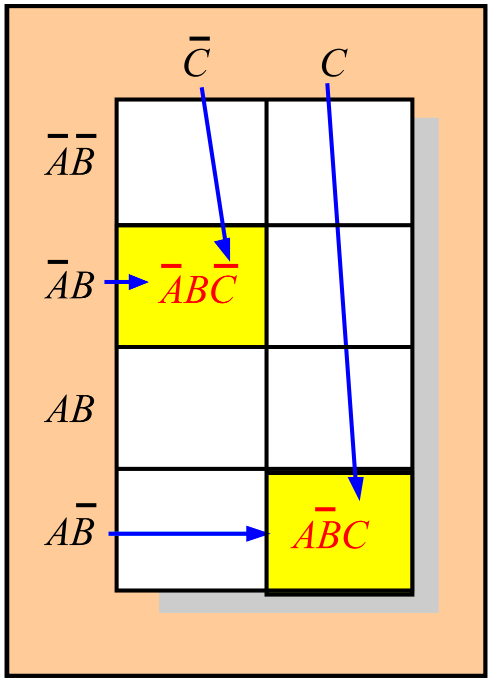
Karnaugh maps

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 $A\overline{B}C$.

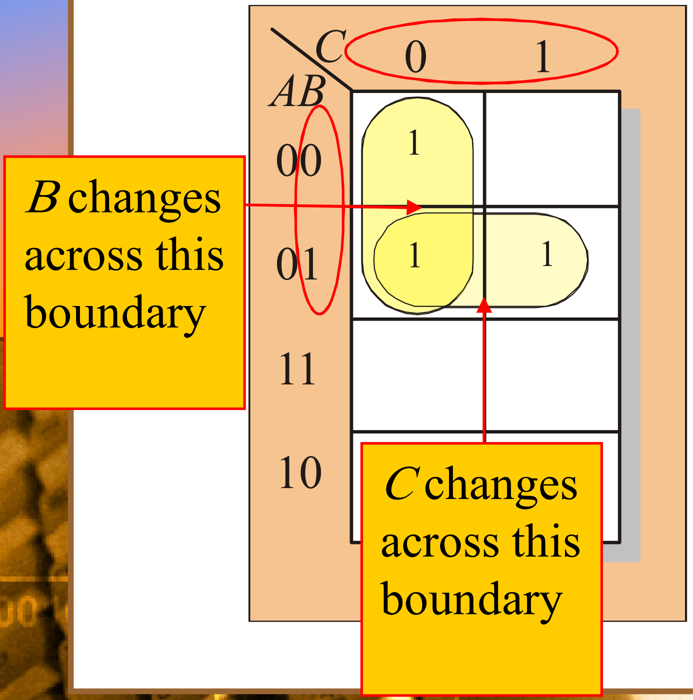


Summary

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.


C
AB
0
1
00
01
11
10
1
1
1
B changes across this boundary
C changes across this boundary

Solution

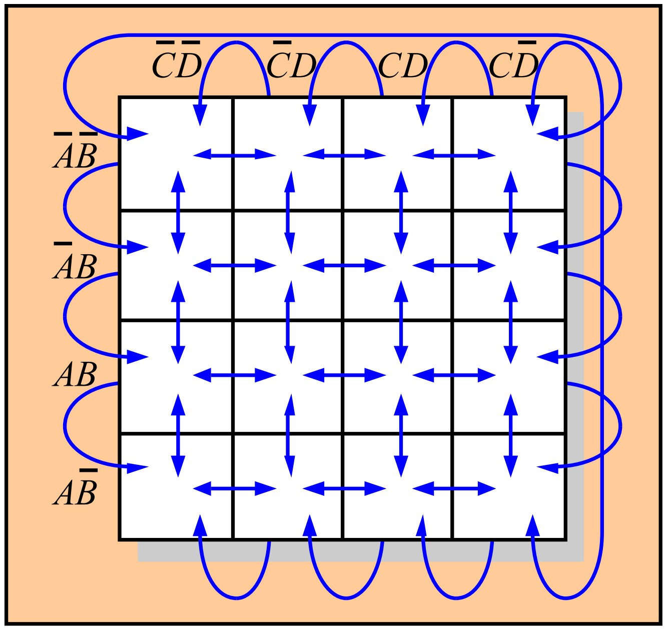
1. Group the 1's into two overlapping groups as indicated.
2. Read each group by eliminating any variable that changes across a boundary.
3. The vertical group is read $\overline{A}\,\overline{C}$.
4. The horizontal group is read $\overline{A}B$.

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

Summary

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.

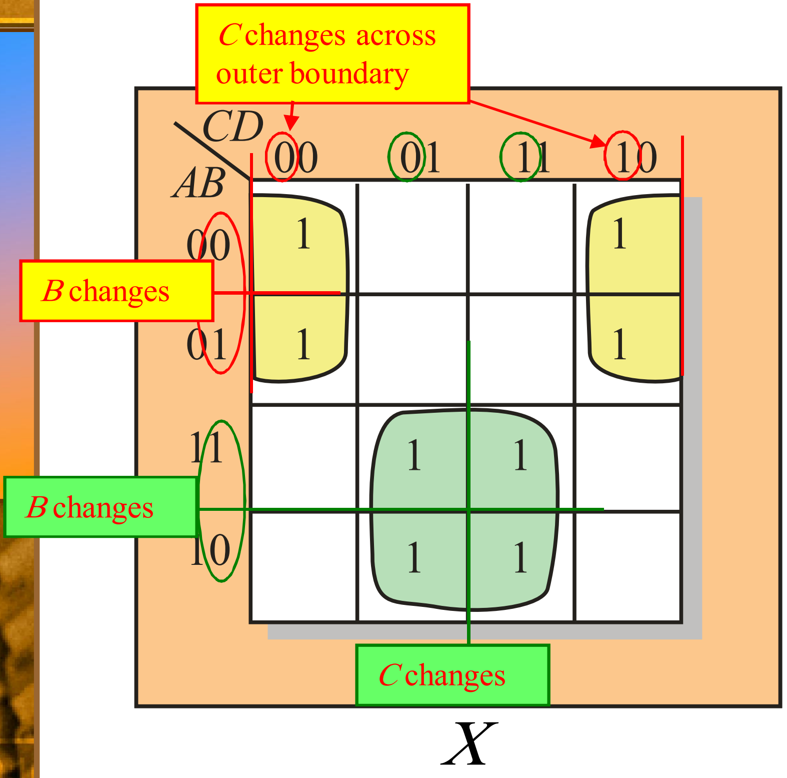
Grouping follows the rules given in the text.

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

Summary

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.

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

Summary

Hardware Description Languages (HDLs)

A Hardware Description Language (HDL) is a tool for implementing a logic design in a PLD. One important language is called VHDL. In VHDL, there are three approaches to describing logic:

1. Structural	Description is like a schematic (components and block diagrams).
2. Dataflow	Description is equations, such as Boolean operations, and registers.
3. Behavioral	Description is specifications over time (state machines, etc.).

Summary

Hardware Description Languages (HDLs)

The *data flow* method for VHDL uses Boolean-type statements. There are two-parts to a basic data flow program: the *entity* and the *architecture*. The entity portion describes the I/O. The architecture portion describes the logic. The following example is a VHDL program showing the two parts. The program is used to detect an invalid BCD code.

```
entity BCDInv is
     port (B,C,D: in bit; X: out bit);
end entity BCDInv
```

```
architecture Invalid of BCDInv
     begin
          X <= (B or C) and D;
end architecture Invalid;
```

Summary

Hardware Description Languages (HDLs)

Another standard HDL is Verilog. In Verilog, the I/O and the logic is described in one unit called a *module*. Verilog uses specific symbols to stand for the Boolean logical operators.

The following is the same program as in the previous slide, written for Verilog:

```
module BCDInv (X, B, C, D);
input B, C, D;
output X;
    assign X = (B | C)&D;
endmodule
```

Selected Key Terms

Variable	A symbol used to represent a logical quantity that can have a value of 1 or 0, usually designated by an italic letter.
Complement	The inverse or opposite of a number. In Boolean algebra, the inverse function, expressed with a bar over the variable.
Sum term	The Boolean sum of two or more literals equivalent to an OR operation.
Product term	The Boolean product of two or more literals equivalent to an AND operation.

Selected Key Terms

Sum-of-products (SOP)	A form of Boolean expression that is basically the ORing of ANDed terms.
Product of sums (POS)	A form of Boolean expression that is basically the ANDing of ORed terms.
Karnaugh map	An arrangement of cells representing combinations of literals in a Boolean expression and used for systematic simplification of the expression.
VHDL	A standard hardware description language. IEEE Std. 1076-1993.

Quiz

1. The associative law for addition is normally written as

a. $A + B = B + A$

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

c. $AB = BA$

d. $A + AB = A$

Quiz

2. The Boolean equation $AB + AC = A(B + C)$ illustrates

a. the distribution law

b. the commutative law

c. the associative law

d. DeMorgan’s theorem

Quiz

3. The Boolean expression $A \cdot 1$ is equal to

   a. A

   b. B

   c. 0

   d. 1

Quiz

4. The Boolean expression $A + 1$ is equal to

   a. A

   b. B

   c. 0

   d. 1

Quiz

5. The Boolean equation $AB + AC = A(B+ C)$ illustrates

a. the distribution law

b. the commutative law

c. the associative law

d. DeMorgan’s theorem

Quiz

6. A Boolean expression that is in standard SOP form is

a. the minimum logic expression

b. contains only one product term

c. has every variable in the domain in every term

d. none of the above

Quiz

7. Adjacent cells on a Karnaugh map differ from each other by

a. one variable

b. two variables

c. three variables

d. answer depends on the size of the map

Quiz

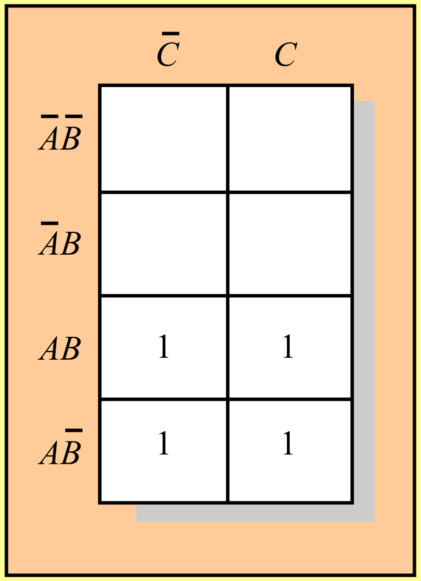
8. The minimum expression that can be read from the Karnaugh map shown is

a. $X = A$

b. $X = \overline{A}$

c. $X = B$

d. $X = \overline{B}$



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

Quiz

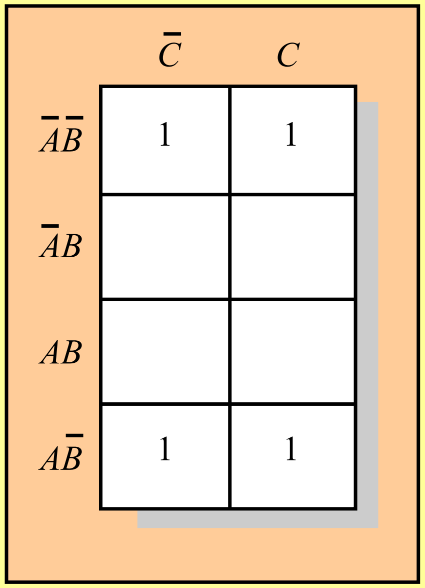
9. The minimum expression that can be read from the Karnaugh map shown is

a. $X = A$

b. $X = \overline{A}$

c. $X = B$

d. $X = \overline{B}$



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

Quiz

10. In VHDL code, the two main parts are called the

a. I/O and the module

b. entity and the architecture

c. port and the module

d. port and the architecture

Quiz

Answers:

1. b	6. c
2. c	7. a
3. a	8. a
4. d	9. d
5. a	10. b