

---

# EEE 3132 - Digital Electronics

---

## Lecture 3 : Logic Fundamentals: Karnaugh Maps

Instructor: Jerry MUWAMBA

Email: [jerry.muwamba@unza.zm](mailto:jerry.muwamba@unza.zm)

[jerrymuwamba@yahoo.com](mailto:jerrymuwamba@yahoo.com)

October 2015

---

# References

Our main reference text books in this course are

- [1] Smith R. J., Dorf R. C., [Circuits Devices and Systems](#), 5<sup>th</sup> Ed., 2004, John Wiley and Sons Ltd, ISBN 9971-51-172-X.
- [2] Maini Anil K., [Digital Electronics: Principles, Devices and Applications](#), 2007, John Wiley and Sons Ltd, ISBN 978-0-470-03214-5.

However, feel free to use some additional text which you might find relevant to our course.

---

# Introduction

- ✓ In this lecture, we will discuss the **Karnaugh Map technique** other than the application of **laws** and **theorems of Boolean algebra** discussed in the preceding lectures for minimizing a given complex Boolean expression.
- ✓ The **primary objective** of all **simplification procedures** is to obtain an **expression** that has the **minimum number of terms**.
- ✓ **Obtaining an expression** with the **minimum number** of **literals** is usually the **secondary objective**.
- ✓ If there is **more than one possible solution** with the **same number of terms**, the one having the minimum number of literals is the choice.

## Logic circuit Analysis

- ✓ The **analysis** of a **logic ckts** consists in **writing a logic statement** expressing the overall **operation performed** in the operation.
- ✓ This can be done in a **straightforward manner**, by starting at the **input and tracing** through the **circuit**, noting function realized at each output.

---

# Introduction

- ✓ The **resulting expression** can be **simplified or written** in an alternative form using **Boolean algebra**. A **truth table** can then be constructed.

## Logic Circuit Synthesis

- ✓ One **fascinating aspects** of **digital electronics** is the **construction of circuits** that can perform **simple mental processes** at **superhuman speeds**.
- ✓ A **typical digital computer** can perform thousands of additions of 10-numbers per second.
- ✓ The **logic designer** starts with a **logical statement** or **truth table**, converts the logic function into a convenient form, and then realizes the desired function by means of standard or special logic elements.

---

# Simplification Techniques

- ✓ Before we move on to discuss the **Karnaugh map technique**, it would be relevant briefly to describe **sum-of-products** and **product-of-sums** Boolean expressions.
- ✓ The given **Boolean expression** will be in **either of the two forms**, and the **objective** is to find a **minimized expression** in the same or the other form.

## Sum-of-Products Boolean Expressions

- ✓ A **sum-of-products (SOP) expression** contains the **sum of different terms**, with each term **being either a single literal** or product of **more than one literal**.
- ✓ It can be obtained from the **truth table** directly by **considering those input combinations** that produce a **logic '1'** at the **output**. Each such input combination produces a term.
- ✓ **Different terms** are given by the **product** of the **corresponding literals**. The **sum of all terms** gives the expression.

# Boolean Algebra

## ✓ Example 1

Consider the truth table in table 1. Obtain the Boolean expression and show the circuit realization. Hence, use the Boolean laws and theorems to minimize this expression and show the circuit realization after simplification.

**Table 1** 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

✓ Other terms can be explained similarly.

## ✓ Solution

- ✓ Considering the first term, the output is '1' when  $A = 0$ ,  $B = 1$  and  $C = 1$ .
- ✓ This is only possible when  $\bar{A}$ ,  $B$  and  $C$  are ANDed.
- ✓ Also, for the second term, the output is '1' only when  $A$ ,  $\bar{B}$  and  $C$  are ANDed.

# Boolean Algebra

## ✓ Example 1 Solution

- The Boolean expression is thus,

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

- **Assuming** that the complement of each variable is available, as is true in most computers, the straight forward ckt realization is as shown in Figure 1 (a).

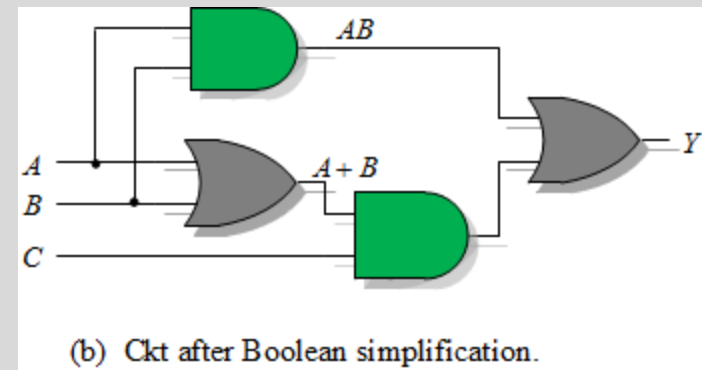
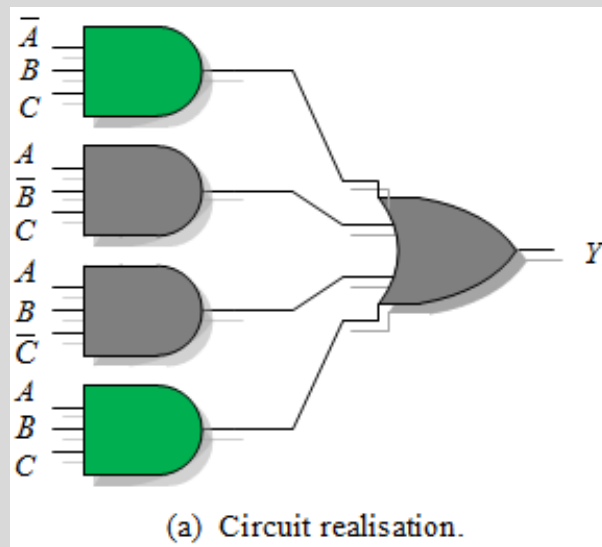


Figure 1

# Boolean Algebra

## ✓ Example 1 Solution

- To simplify the Boolean expression, we first expand it, i.e.,

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

since  $ABC + ABC = ABC$ , by the **Idempotent theorem**.

- By the **distributive theorem**, we factor the above expression to yield

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

- Recall that  $\bar{C} + C = 1$ , and  $\bar{A}B + A\bar{B} + AB = A + B$ , the function becomes

$$Y = C(A + B) + AB$$

- This function requires only four logic elements as shown in Figure 1 (b).

---

# Simplification Techniques

## Product-of-Sums Boolean Expressions

- ✓ A **product-of-sums (POS) expression** contains the **product of different terms**, with each term being either a **single literal** or a **sum of more than one literal**.
- ✓ It can be obtained from the **truth table** by considering those **input combinations** that produce a **logic '0'** at the **output**. Each such **input combination** gives a **term**, and the **product** of all such **terms** gives the expression.
- ✓ **Different terms** are obtained by taking the sum of the corresponding literals.
- ✓ Here, **'0'** and **'1'** respectively mean the **uncomplemented** and **complemented** variables, **unlike** sum-of-products expressions where **'0'** and **'1'** respectively mean **complemented** and **uncomplemented** variables.
- ✓ To illustrate this, consider the **truth table in Table 1** of example 1.

# Boolean Algebra

## ✓ Example 2

- ❑ Consider the truth table in table 1. Obtain the product-of-sums Boolean expression. Hence, show that it equals its sum-of-products dual.

### Solution

- ❑ Each term in this case is a sum of literals implemented using an OR operation.
- ❑ Now, an OR gate produces a logic '0' only when all its inputs are in the logic '0' state.
- ❑ Thus, the first term corresponding to the first row of the table will be  $A + B + C$ . All the other terms are obtained in a similar manner.
- ❑ Therefore, the expression is

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

# Boolean Algebra

## ✓ Example 2 Solution

- ❑ Transforming the given product-of-sums expression into an equivalent sum-of-products is a straightforward process.
- ❑ We simply find the dual of the expression.
- ❑ **Dual of a Boolean Expression**
- ❑ The dual of a Boolean expression is obtained by replacing all ‘ $\cdot$ ’ operations with ‘ $+$ ’ operations, all ‘ $+$ ’ operations with ‘ $\cdot$ ’ operations, all 0s with 1s and all 1s with 0s and leaving all literals unchanged.
- ❑ Thus, dual of  $Y = (A + B + C) \cdot (A + B + \bar{C}) \cdot (A + \bar{B} + C) \cdot (\bar{A} + B + C)$  is

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

# Simplification Techniques

## Expanded Forms of Boolean Expressions

- ✓ Are **useful not only** in analyzing Boolean expressions **but in the application** of **minimizing techniques** such as the Karnaugh mapping method for simplifying Boolean expressions.
- ✓ The **expanded form**, sum-of-products (SOP) or product-of-sums (POS), **is obtained** by **including all possible** combinations of **missing variables**.
- ✓ To illustrate this, consider the following **sum-of-products expression**:

$$A \cdot \bar{B} + B \cdot \bar{C} + A \cdot B \cdot \bar{C} + \bar{A} \cdot C$$

- ✓ It is a three-variable expression. Expanded versions of different miniterms can be written as follows:
  - ❑  $A \cdot \bar{B} = A \cdot \bar{B} \cdot (C + \bar{C}) = A \cdot \bar{B} \cdot C + A \cdot \bar{B} \cdot \bar{C}$  .
  - ❑  $B \cdot \bar{C} = B \cdot \bar{C} \cdot (A + \bar{A}) = B \cdot \bar{C} \cdot A + B \cdot \bar{C} \cdot \bar{A}$  .
  - ❑  $A \cdot B \cdot \bar{C}$  is a complete term and has no missing variable .
  - ❑  $\bar{A} \cdot C = \bar{A} \cdot C \cdot (B + \bar{B}) = \bar{A} \cdot C \cdot B + \bar{A} \cdot C \cdot \bar{B}$  .

# Simplification Techniques

- ✓ The expanded sum-of-products expression is therefore given by

$$\begin{aligned} & A \cdot \bar{B} \cdot C + A \cdot \bar{B} \cdot \bar{C} + A \cdot B \cdot \bar{C} + \bar{A} \cdot B \cdot \bar{C} + A \cdot B \cdot C + \bar{A} \cdot B \cdot C + \bar{A} \cdot \bar{B} \cdot C \\ & = A \cdot \bar{B} \cdot C + A \cdot \bar{B} \cdot \bar{C} + A \cdot B \cdot \bar{C} + \bar{A} \cdot B \cdot \bar{C} + \bar{A} \cdot B \cdot C + \bar{A} \cdot \bar{B} \cdot C \end{aligned}$$

- ✓ As another illustration, consider the product-of-sums expression

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

- ✓ It is four-variable expression.  $\bar{A} + B$  in this case expands to

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

- ✓ The expanded product-of-sums expression is therefore given by

$$\begin{aligned} & (\bar{A} + B + C + D) \cdot (\bar{A} + B + C + \bar{D}) \cdot (\bar{A} + B + \bar{C} + D) \cdot (\bar{A} + B + \bar{C} + \bar{D}) \cdot (\bar{A} + B + \bar{C} + \bar{D}) \\ & = (\bar{A} + B + C + D) \cdot (\bar{A} + B + C + \bar{D}) \cdot (\bar{A} + B + \bar{C} + D) \cdot (\bar{A} + B + \bar{C} + \bar{D}) \end{aligned}$$

# Simplification Techniques

## Canonical Form of Boolean Expressions

- ✓ An **expanded form** of Boolean expression, where each term contains all **Boolean variables** in their **true or complemented** form, is also known as the **canonical form** of the expression
- ✓ For instance,  $f(A, B, C) = \bar{A} \cdot \bar{B} \cdot \bar{C} + \bar{A} \cdot \bar{B} \cdot C + A \cdot B \cdot C$  is a **Boolean function** of **three variables** expressed in **canonical form**.
- ✓ This function after simplification reduces to  $f(A, B, C) = \bar{A} \cdot \bar{B} + A \cdot B \cdot C$  and loses its canonical form.

## $\Sigma$ and $\Pi$ Nomenclature

- ✓ Let us consider the Boolean function:  $f(A, B, C) = \bar{A} \cdot \bar{B} \cdot \bar{C} + \bar{A} \cdot \bar{B} \cdot C + A \cdot B \cdot C$
- ✓ Using the  $\Sigma$  notation it is  $f(A, B, C) = \sum 0, 1, 7$ .
- ✓ Similarly, for the function:  $f(A, B, C) = (\bar{A} + \bar{B} + \bar{C}) \cdot (\bar{A} + \bar{B} + C) \cdot (A + B + C)$
- ✓ Since the sum terms denote binary numbers, 111, 110, and 000, this yields
- ✓ Using the  $\Pi$  notation it is  $f(A, B, C) = \prod 0, 6, 7$ .

---

# Karnaugh Map Method

- ✓ A **Karnaugh map** is a graphical representation of the **logic system**. It can be **drawn directly** from either **minterm** (sum-of-products) or **maxterm** (product-of-sums) Boolean expressions.
- ✓ **Drawing** a Karnaugh map from the **truth table** involves an **additional step** of writing the **minterm** or **maxterm** expression depending upon whether it is desired to have a minimized sum-of-products or a minimized product-of-sums expression.

## Construction of a Karnaugh Map

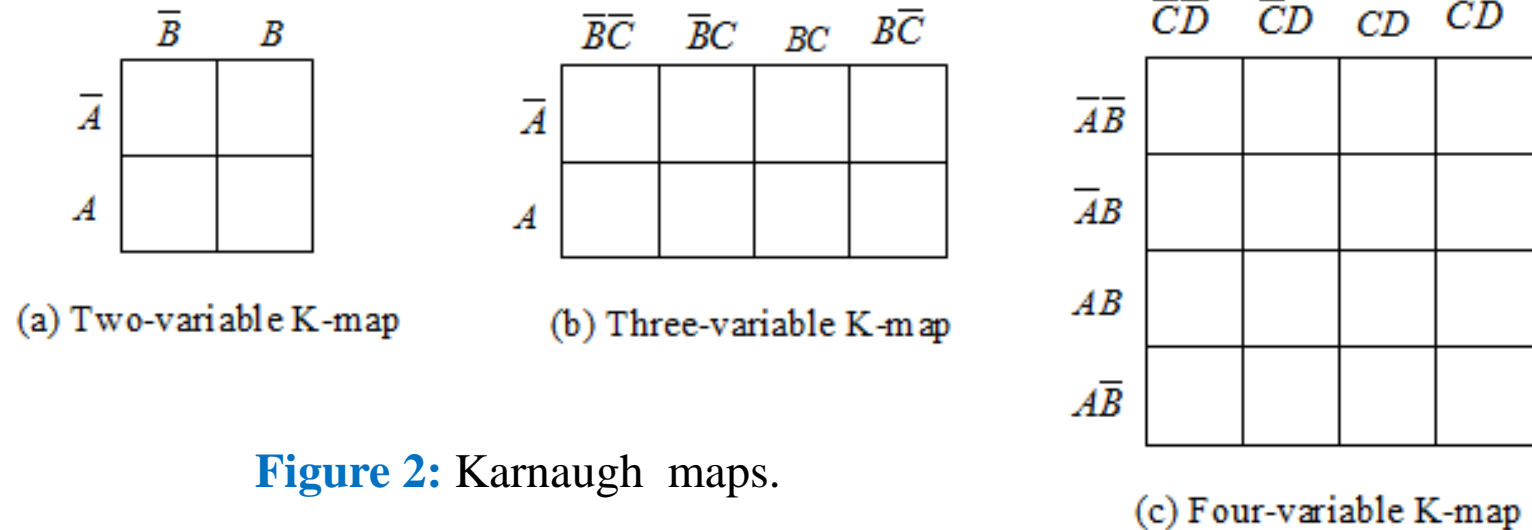
- ✓ An  **$n$ -variable Karnaugh map** has  $2^n$  squares, and each possible input is allotted a square.
- ✓ In the case of a **minterm Karnaugh map**, '1' is placed in **all those squares** for which the **output** is '1', and '0' is placed in all those squares for which the output is '0'. **0s** are **omitted** for **simplicity**.
- ✓ An '**X**' is placed in **squares corresponding** to '**don't cares**' conditions.

---

# Karnaugh Map Method

- ✓ In the case of a **maxterm Karnaugh map**, a '1' is placed in **all those squares** for which the **output** is '0', and a '0' is placed for input entries corresponding to a '1' output. Again, **0s** are **omitted** for **simplicity**, and an 'X' is placed in **squares corresponding** to 'don't care' conditions.
- ✓ **It is worth noting** that, **extreme rows** and **extreme columns** are considered adjacent.
- ✓ The **minterms** are **ordered** according to **Gray code**, i.e., only one variable changes between adjacent squares.
- ✓ The **commonly** used **designation styles** for two-, three- and four-variable minterm Karnaugh maps are given in Figure 2.
- ✓ **Having drawn** the **Karnaugh map**, the next step is to **form groups** as per the following guidelines:
  1. **Each square** containing a '1' must be **considered at least once**, although it can be considered as **often as desired**.
  2. **The objective** is to account for all marked squares in the minimum number of groups.

# Karnaugh Map Method



**Figure 2:** Karnaugh maps.

3. The number of squares in a group must always be a power of 2, i.e., groups can have 1, 2, 4, 8, 16, ... squares.
4. Each group should be as large as possible, which means that a square should not be accounted for by itself if it can be accounted for by a group of two squares and so forth.
5. ‘Don’t care’ entries can be used in accounting for all of 1-squares to make optimum groups. Not all ‘don’t cares’ need to be accounted for though.

# Karnaugh Map Method

- ✓ Having accounted for groups with all 1s, the minimum ‘SOP’ or ‘POS’ expressions can be written directly from the Karnaugh map.

## ✓ Example 3

- Given in table 3 is the truth table of the Boolean function of a two-input OR gate. Write the minterm and maxterm Boolean expressions, hence create the minterm Karnaugh map and maxterm Karnaugh map.

Table 3: truth table.

$A$	$B$	$Y$
0	0	0
0	1	1
1	0	1
1	1	1

## ✓ Solution

	$\bar{B}$	$B$
$\bar{A}$		1
$A$	1	1

Sum-of-products K-map

	$\bar{B}$	$B$
$\bar{A}$		
$A$		1

Product-of-sums K-map

$$Y = \bar{A} \cdot B + A \cdot \bar{B} + A \cdot B, \text{ (minterm or SOP)}$$

$$Y = A + B, \text{ (maxterm or POS)}$$

# Karnaugh Map Method

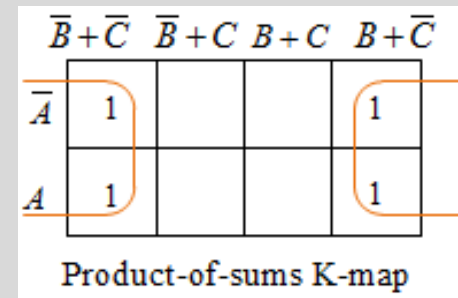
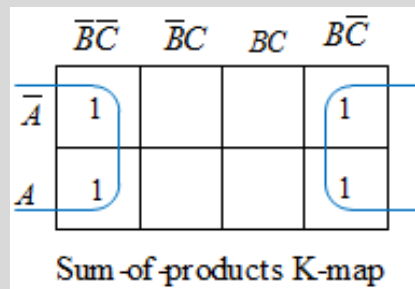
## ✓ Example 4

- ❑ Table 4 is the truth table of the three-variable Boolean function, with minterm and maxterm repectively given by  $Y = \bar{A} \cdot \bar{B} \cdot \bar{C} + \bar{A} \cdot B \cdot \bar{C} + A \cdot \bar{B} \cdot \bar{C} + A \cdot B \cdot \bar{C}$  and  $Y = (\bar{A} + \bar{B} + \bar{C}) \cdot (\bar{A} + B + \bar{C}) \cdot (A + \bar{B} + \bar{C}) \cdot (A + B + \bar{C})$
- ❑ Create the minterm Karnaugh map and maxterm Karnaugh map,

**Table 4:** truth table.

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

## ✓ Solution



- ✓ Thus, the simplified SOP and POS expressions are both given by  $Y = \bar{C}$ .

# Karnaugh Map Method

## ✓ Example 5

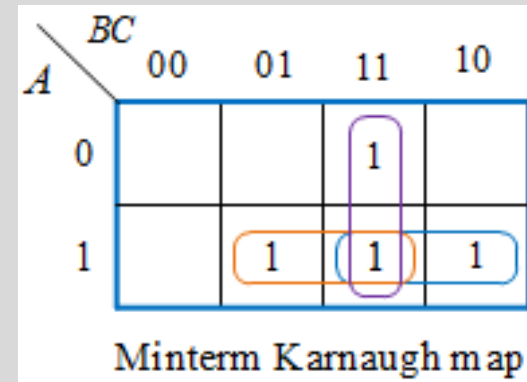
- ❑ The three-variable **Boolean function**  $Y = \overline{A}BC + A\overline{B}C + AB\overline{C} + ABC$  was found for the truth table of example 1. Simplify this expression using the Karnaugh map method.

## ❑ Solution

- ❑ Cover all 1s with maximum grouping.
- ❑ The simplified Boolean equation is one that sums all the terms corresponding to each of the group:

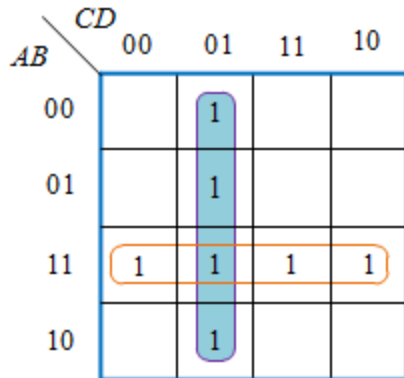
$$Y = AB + AC + BC$$

- ❑ There is no simpler expression for this function.
- ❑ Using DeMorgan's law, the simplified expression can be converted to a **NANDed** product of **NANDs**, i.e.,  $Y = \overline{\overline{AB} \cdot \overline{AC} \cdot \overline{BC}}$

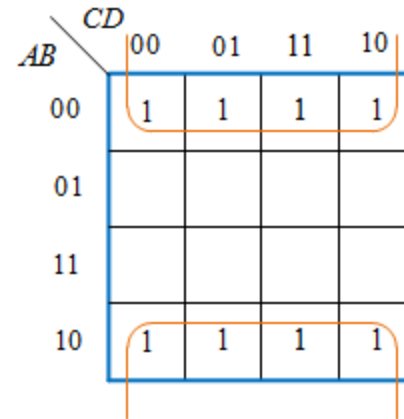


# Karnaugh Map Method

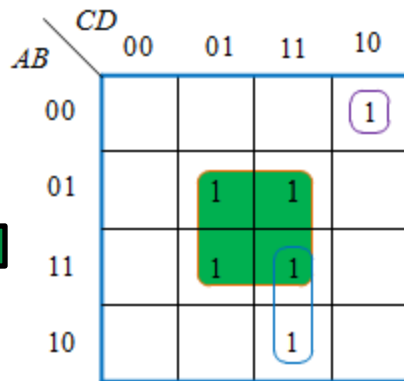
## Further Examples of grouping



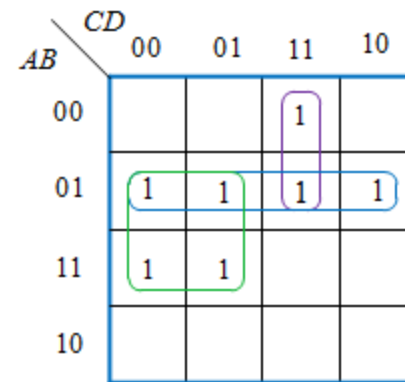
$$Y = AB + \bar{C}D$$



$$Y = \bar{B}$$



$$Y = \bar{A}\bar{B}C\bar{D} + BD + ACD$$



$$Y = \bar{A}B + B\bar{C} + \bar{A}C\bar{D}$$

# Karnaugh Map Method

## ✓ Example 6

- The respective four variable **minterm** and **maxterm** Boolean expressions are

$$Y = \bar{A}\bar{B}\bar{C}\bar{D} + \bar{A}\bar{B}C\bar{D} + \bar{A}B\bar{C}\bar{D} + \bar{A}BCD + A\bar{B}\bar{C}\bar{D} + A\bar{B}C\bar{D} + AB\bar{C}\bar{D} + ABCD$$

and

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

- Draw the respective **minterm** and **maxterm** Karnaugh maps. Hence deduce the minimized expressions from the Karnaugh maps in the two cases.

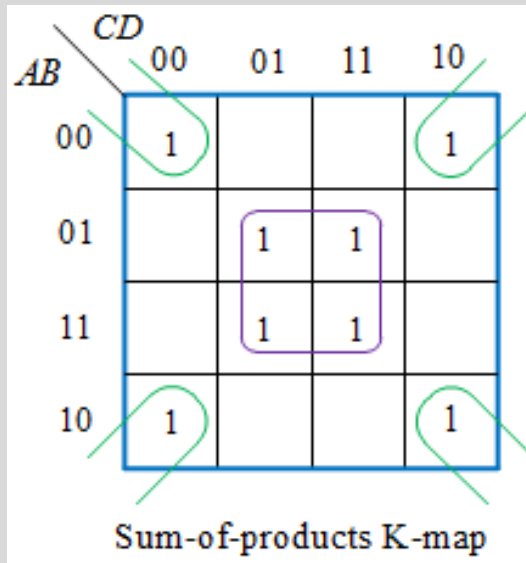
## □ Solution

- The respective **minterm** and **maxterm** Karnaugh maps together with their corresponding **simplified expressions** are as shown below.

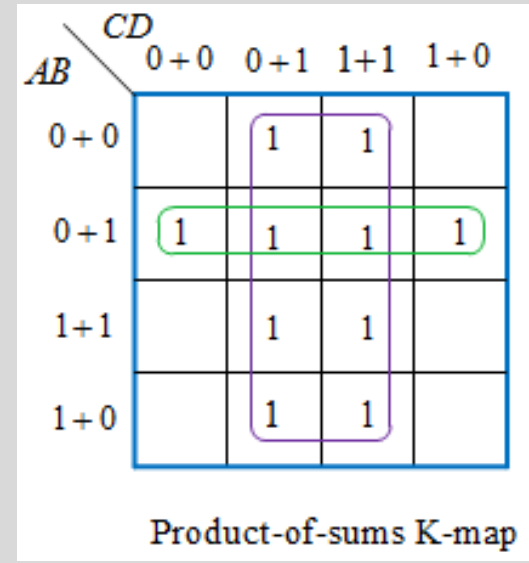
# Karnaugh Map Method

## ✓ Solution to Example 6

- The respective four variable **minterm** and **maxterm** Boolean expressions are



$$Y = \bar{B}\bar{D} + BD$$

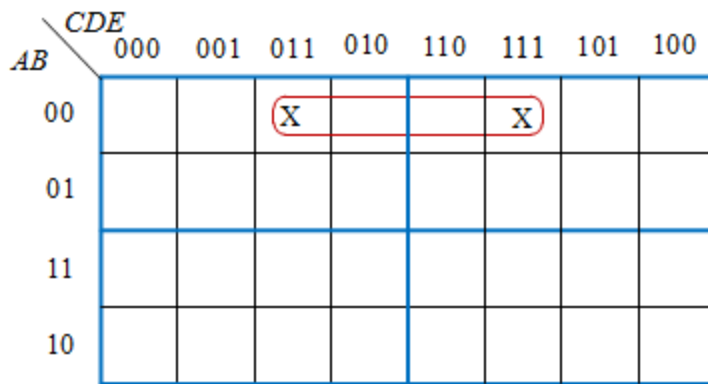


$$Y = \bar{D} \cdot (A + \bar{B})$$

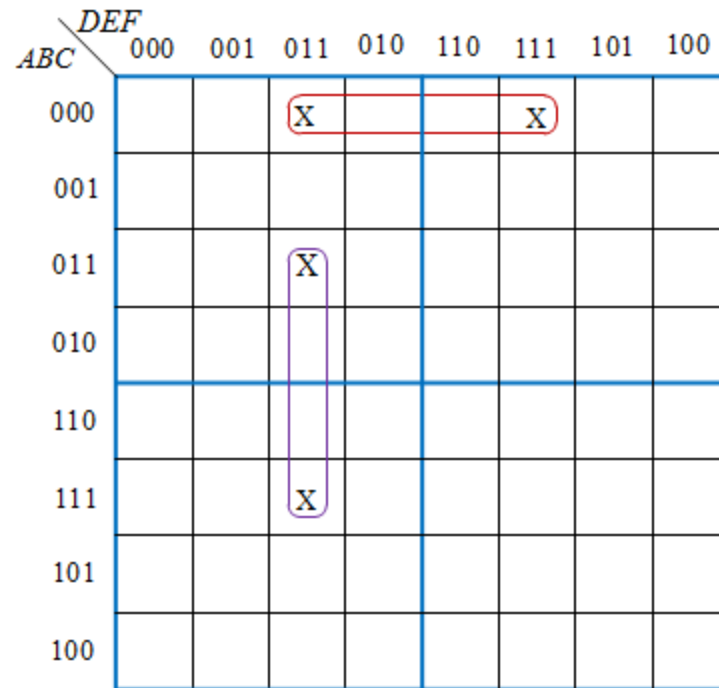
# Karnaugh Map Method

## K-Map for Boolean Expressions with a Large Number of Variables

- ❑ The construction of K-maps for a large number of variables is a complex and cumbersome exercise, although manageable up to six variables.
- ❑ Five- and six-variable representation of Karnaugh maps are shown in Figure 3(a) and (b).



(a) Five-variable K-map



(b) Six-variable K-map

Figure 3: Karnaugh maps.

---

# Karnaugh Map Method

- ❑ It is worth noting that while forming groups in Karnaugh maps involving more than four variables terms equidistant from the central horizontal and central vertical lines are considered adjacent.
- ❑ These lines are shown thicker in Figures 3(a) and (b). Squares marked 'X' in the Figures above are adjacent and therefore can be grouped.
- ❑ In general, an  $n$ -variable Boolean expression can be represented by  $2^{n-4}$  four-variable maps.
- ❑ In such multiple maps, groups are made as before, except that, in addition to adjacencies discussed earlier, corresponding squares in two adjacent maps are also considered adjacent and can therefore be grouped.

# Karnaugh Map Method

## Don't Care Conditions (Optional)

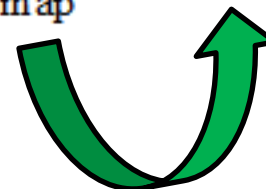
- ❑ In certain cases some of the minterms may never occur or it may not matter what happens if they do.
- ❑ In such cases we fill in the Karnaugh map with  $X$ , meaning don't care.
- ❑ When minimizing an  $X$  can be 0 or 1 – whatever helps best with the minimization.
- ❑ For example

	$BC$			
	00	01	11	10
$A$				
0	0	0	1	X
1	0	0	1	1

Minterm Karnaugh map

$$Y = B$$

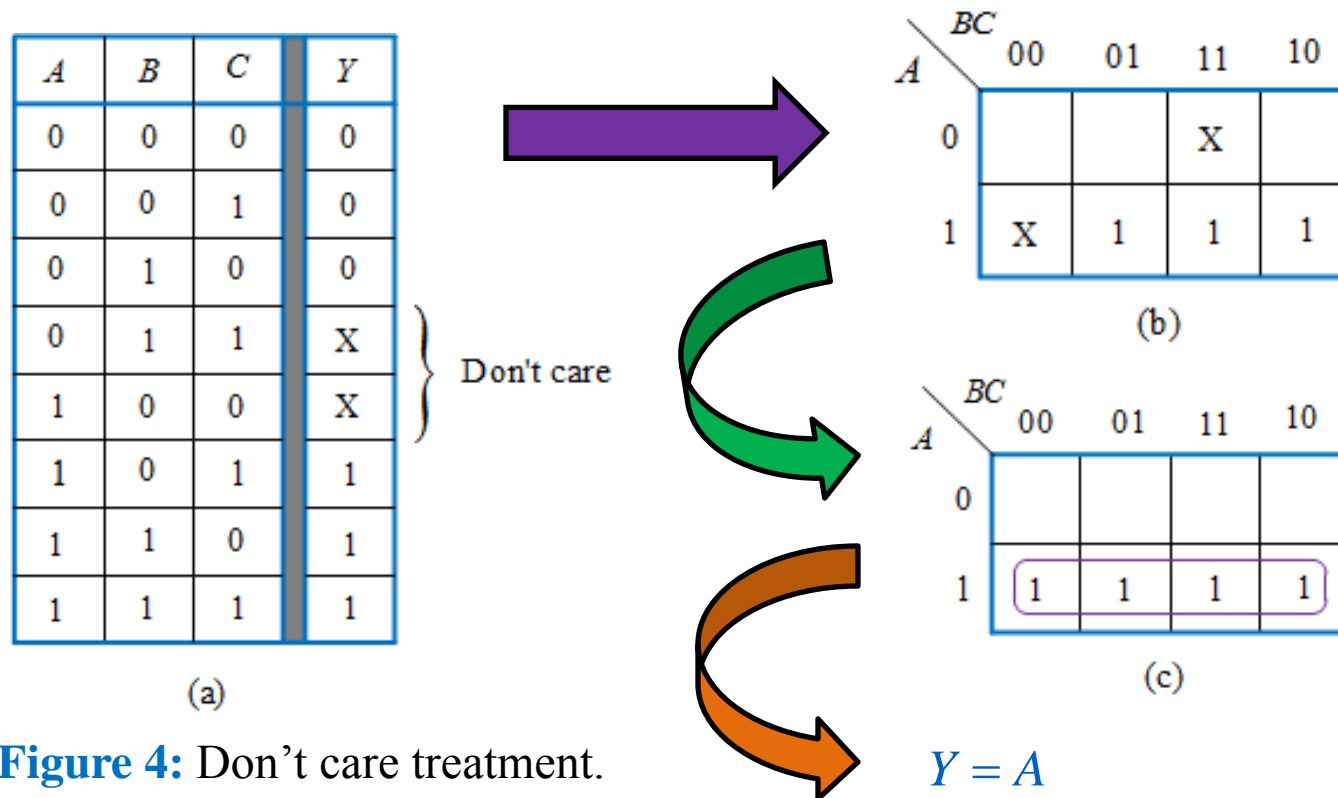
- ❑ Simplifies to  $B$  if  $X$  is assumed 1, i.e.,



# Karnaugh Map Method

## More “Don’t Care” Examples

- Don’t care conditions should be changed to either 0 or 1 to produce K-map looping that yields the simplest expression.



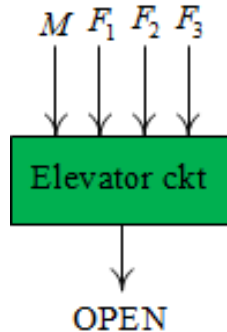
**Figure 4:** Don't care treatment.

$$Y = A$$

# Karnaugh Map Method

## More “Don’t Care” Examples

- Elevator circuit.



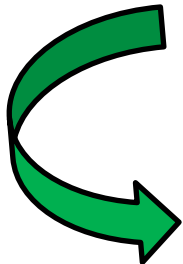
Truth table

$M$	$F_1$	$F_2$	$F_3$	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

$MF_1 \backslash F_2F_3$	00	01	11	10
00		1	X	1
01	1	X	X	X
11		X	X	X
10			X	



$$OPEN = \overline{M}F_1 + \overline{M}F_2 + \overline{M}F_3$$



# Karnaugh Map Method

## ✓ Example 7

- Minimize the Boolean function

$$f(A, B, C) = \sum 0, 1, 3, 5 + \sum_d 2, 7$$

using the mapping method in both minimized sum-of-products and product-of-sums forms. Note that  $d$  denotes the don't cares conditions

## □ Solution

- $f(A, B, C) = \sum 0, 1, 3, 5 + \sum_d 2, 7 = \prod 4, 6 + \prod_d 2, 7$  .

- From given Boolean functions above, we can write SOP and POS Boolean expressions as follows:

$$f(A, B, C) = \bar{A} \cdot \bar{B} \cdot \bar{C} + \bar{A} \cdot \bar{B} \cdot C + \bar{A} \cdot B \cdot C + A \cdot \bar{B} \cdot C$$

and

$$f(A, B, C) = (\bar{A} + B + C) \cdot (\bar{A} + \bar{B} + C)$$

# Karnaugh Map Method

## ✓ Solution to Example 7

- ❑ The 'don't care' input combinations for the SOP Boolean expression are  $\bar{A} \cdot B \cdot \bar{C}$  and  $A \cdot B \cdot C$
- ❑ The 'don't care' input combinations for the POS Boolean expression are  $(A + \bar{B} + C) \cdot (\bar{A} + \bar{B} + \bar{C})$ .
- ❑ The Karnaugh maps are as shown below.

		<i>AB</i>			
		00	01	11	10
<i>C</i>	0	1	X		
	1	1	1	X	1

Minterm Karnaugh map

		<i>AB</i>			
		0+0	0+1	1+1	1+0
<i>C</i>	0	X			
	1	1	1		X

Maxterm Karnaugh map

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

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

---

# End of Lecture 3

**Thank you for your attention!**