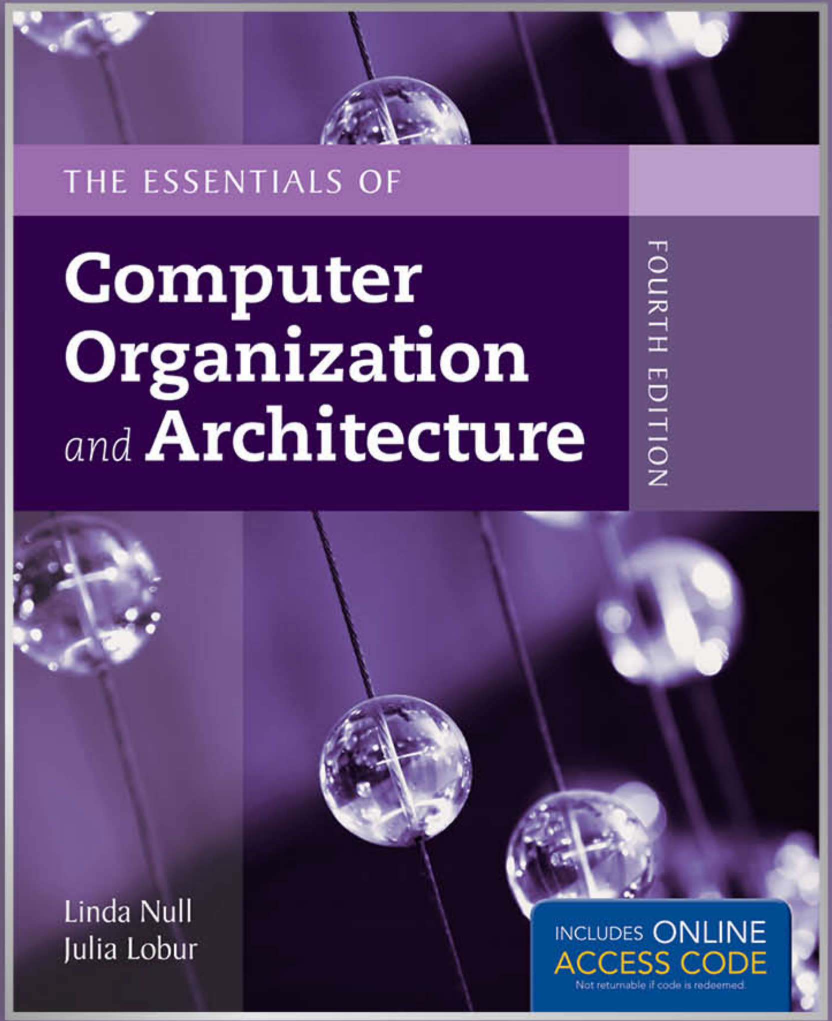
# Chapter 3 Special Section

## Karnaugh Maps



# 3A.1 Introduction

- **Simplification of Boolean functions** leads to **simpler and faster circuits**.
- Simplifying functions **using identity laws** is **time-consuming and error-prone**.
- This section presents **Karnaugh Maps** (Kmaps), an **easy, systematic method for reducing Boolean functions**.

# 3A.2 Description of Kmaps

- A Kmap is a **matrix consisting of rows and columns** that **represent the output values** of a **Boolean function**.
- The **output values** placed in each cell are **derived from the minterms** of a function.
- A **minterm** is a **Boolean product** that contains **all variables exactly once**, either **complemented or not complemented**.

# 3A.2 Description of Kmaps

- The minterms for a function having the inputs *x* and *y* are **x′y′, x′y, xy′**, and **xy**.
- Consider the Boolean function, **F(x,y)= xy + xy′**
- **Minterm table**:

| Minterm | X | Y |
|---|---|---|
| X'Y' | 0 | 0 |
| X'Y | 0 | 1 |
| X Y' | 1 | 0 |
| X Y | 1 | 1 |

# 3A.2 Description of Kmaps

- Minterm table of a function with variables x, y, z:

| Minterm | X | Y | Z |
|---|---|---|---|
| X'Y'Z' | 0 | 0 | 0 |
| X'Y'Z | 0 | 0 | 1 |
| X'Y Z' | 0 | 1 | 0 |
| X'Y Z | 0 | 1 | 1 |
| X Y'Z' | 1 | 0 | 0 |
| X Y'Z | 1 | 0 | 1 |
| X Y Z' | 1 | 1 | 0 |
| X Y Z | 1 | 1 | 1 |

# 3A.2 Deriving Kmaps from Truth Tables

$F(x, y) = (x'y) + (xy') + (xy)$

| X | Y | F(x,y) |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

# 3A.2 Deriving Kmaps from Truth Tables

$$F(x,y,z) = (x'yz') + (x'yz) + (xy'z') + (xyz') + (xyz)$$

| x | y | z | F(x,y,z) |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

# 3A.2 Description of Kmaps

- A Kmap has a **cell for each minterm**.
- This means that it has a **cell for each line of the truth table** of a function.
- **Example:**

$F(X,Y) = XY$

| X | Y | XY |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

| X \ Y | 0 | 1 |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |

# 3A.3 Kmap Simplification for Two Variables

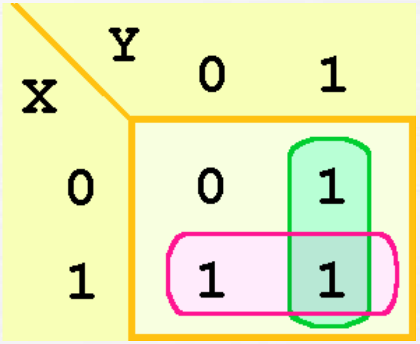
- We can **reduce the function** of the Kmap to its simplest terms **by finding neighbored 1s** that can be collected into groups **that are powers of two**.
- Possible group sizes are 1, 2, 4, 8, 16, …

- In our example, we have two such groups.
  - Can you find them?

| X \ Y | 0 | 1 |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 1 | 1 |

# 3A.3 Kmap Simplification for Two Variables

- We see that **both groups are powers of two** and that the **groups overlap**.
- The next slide gives guidance for selecting Kmap groups.



# 3A.3 Kmap Simplification for Two Variables

The **rules** of Kmap simplification are:

- Groups can contain **1s only**; no 0s.
- Groups can be formed only at **right angles**; diagonal groups are not allowed.
- The **group sizes** must be a **power of 2** – even if it contains a single 1.
- The **groups must** be made **as large as possible**.
- **Groups can overlap** and **wrap around** the sides of the Kmap.

# 3A.3 Kmap Simplification for Three Variables

$$F(X,Y,Z) = X'Y'Z + X'YZ + XY'Z + XYZ$$

– What is the largest group of 1s that is a power of 2?

| X \ YZ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 1 | 0 |

# 3A.3 Kmap Simplification for Three Variables

- This group tells us that changes in the variables *x* and *y* have no influence on the value of the function: They are irrelevant.
- This means that the function,

**F(X,Y,Z)=X′Y′Z + X′YZ + XY′Z + XYZ**

reduces to **F(z) = z**.

| X \ YZ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 1 | 0 |

# 3A.3 Kmap Simplification for Three Variables

- A more complicated Kmap:

$$F(X,Y,Z) = X'Y'Z' + X'Y'Z + X'YZ + X'YZ' + XY'Z' + XYZ'$$

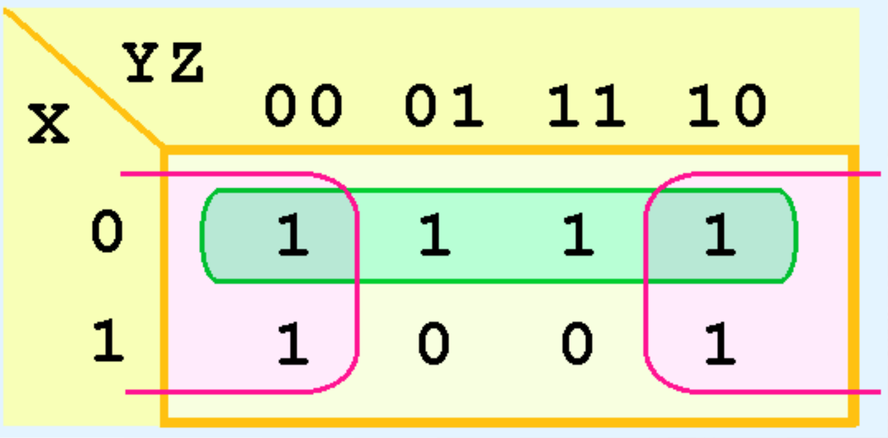
- Its Kmap is shown below. There are (only) two groupings of 1s.
  – Can you find them?

| X \ YZ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 |

# 3A.3 Kmap Simplification for Three Variables

- The **pink group wraps around** the sides of a Kmap.
- This group tells us that the **values of *x* and *y*** are **not relevant** to the term of the function that is represented by the group.
- The **reduced term** of this group is **z'**

What about the green group in the top row?



# 3A.3 Kmap Simplification for Three Variables

- The **green group** tells us that **only the value of *x* is relevant** in that group.
- **x** is complemented in that row, so the **reduced term** of corresponding to this group is **X′** .
- The reduced function the sum of terms of all groups: **F(X,Z) = X′+ Z′**

Recall that we had six minterms in our original function!

| X \ YZ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 |

# 3A.3 Kmap Simplification for Four Variables

- General format of a **Kmap** of a function with **4 variables and 16 minterms**:

| wx \ yz | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | w'x'y'z' | w'x'y'z | w'x y z | w'x'y z' |
| 01 | w'x y'z' | w'x y'z | w'x y z | w'x y z' |
| 11 | w x'y'z' | w x'y'z | w x y z | w x'y z' |
| 10 | w x y'z' | w x y'z | w x y z | w x y z' |

# 3A.3 Kmap Simplification for Four Variables

- Example (only non-zero minterms shown):

$$F(W,X,Y,Z) = W'X'Y'Z' + W'X'Y'Z + W'X'YZ' + W'XYZ' + WX'Y'Z' + WX'Y'Z + WX'YZ'$$

  - Can you identify (only) three groups in this Kmap?

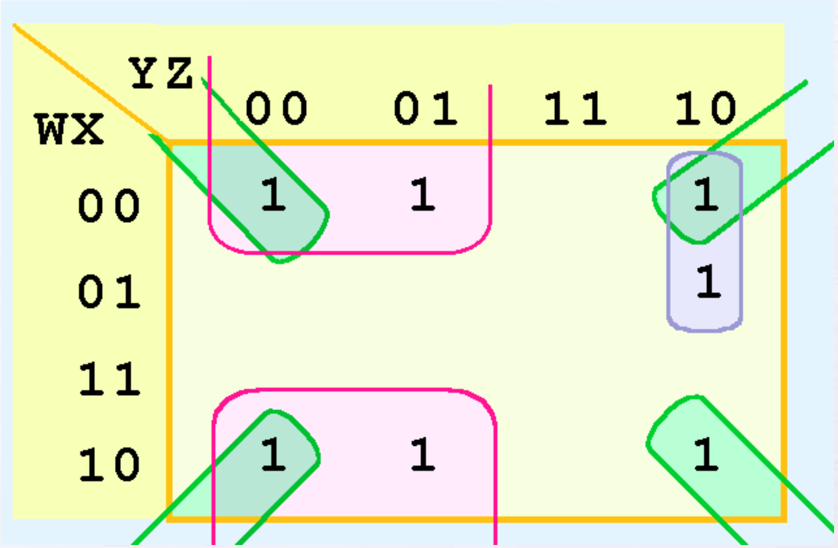
**Recall that groups can overlap and wrap around.**

| WX \ YZ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 1 | 1 | | 1 |
| 01 | | | | 1 |
| 11 | | | | |
| 10 | 1 | 1 | | 1 |

# 3A.3 Kmap Simplification for Four Variables

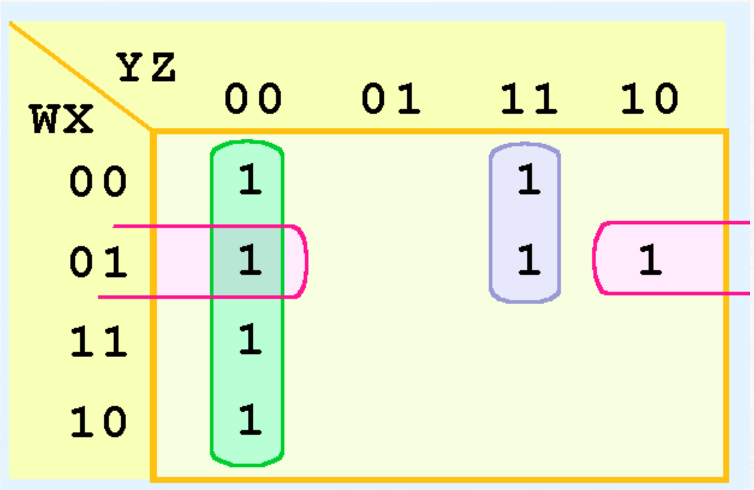
- The three groups consist of:
  - A **purple group** entirely within the Kmap at the right.
  - A **pink group** that wraps the top and bottom.
  - A **green group** that spans the corners.
- We have three terms in our final function:

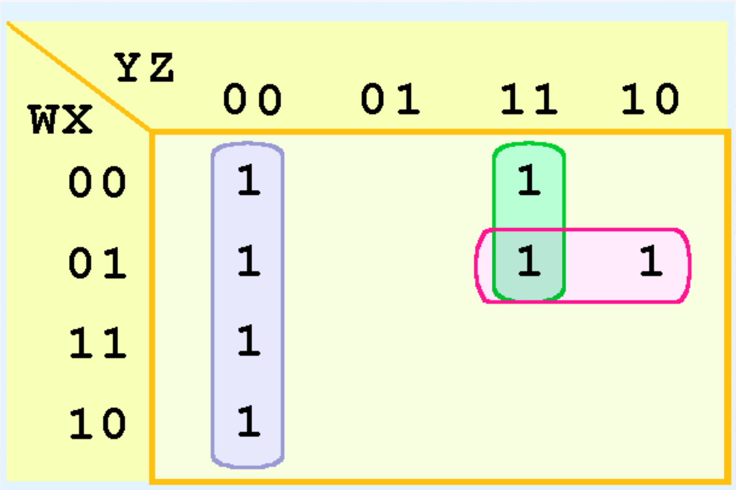
$$F(W,X,Y,Z) = X'Y' + X'Z' + W'YZ'$$



# 3A.3 Kmap Simplification for Four Variables

- It is **possible to have a choice** as to how to pick groups within a Kmap, while keeping the groups as large as possible.
- The **different functions** that result from the groupings below are **logically equivalent**.



# 3A.6 Don't Care Conditions

- Consider the function F(w,x,y,z) that returns whether the **quotient wx/yz has a remainder**
- Examples:
  - $01_2/10_2$ = 1/2 has a remainder, output: 1
  - $11_2/11_2$ = 3/3 has no remainder, output: 0
  - $10_2/00_2$ = 2/0 is an **invalid input**, output: undefined
- Circuits must be designed such that **invalid inputs** will **never happen**
- In a Kmap, we **don't need to care** whether the output corresponding to an invalid input is 0 or 1

# 3A.6 Don't Care Conditions

- Consider the function F(w,x,y,z) that returns whether the **quotient wx/yz has a remainder**

| WX \ YZ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | X | | | |
| 01 | X | | 1 | 1 |
| 11 | X | | | 1 |
| 10 | X | | 1 | |

- The **cross *X*** indicates **'don't care' cases** (division by 0)
- When building groups, we are **free to include or ignore the *X*'s**

# 3A.6 Don’t Care Conditions

- Consider the function F(w,x,y,z) that returns whether the **quotient wx/yz has a remainder**

| WX \ YZ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | X | | | |
| 01 | X | | 1 | 1 |
| 11 | X | | | 1 |
| 10 | X | | 1 | |

- Simplified function F(w,x,y,z) = xz’ + w’xy + wx’yz
- We included **some** *X*’s in order to get larger groups

# 3A.6 Don't Care Conditions

- Second example:

| WX \ YZ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | X | 1 | 1 | X |
| 01 | | X | 1 | |
| 11 | X | | 1 | |
| 10 | | | 1 | |

# 3A.6 Don’t Care Conditions

- One possible solution:

$$F(W,X,Y,Z) = W'X' + YZ$$

| WX \ YZ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | X | 1 | 1 | X |
| 01 | | X | 1 | |
| 11 | X | | 1 | |
| 10 | | | 1 | |

# 3A.6 Don’t Care Conditions

- Another possible solution:

$$F(W,X,Y,Z) = W'Z + YZ$$

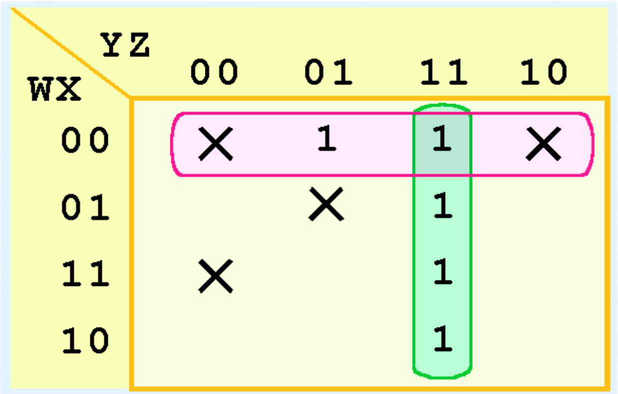
| WX \ YZ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | X | 1 | 1 | X |
| 01 | | X | 1 | |
| 11 | X | | 1 | |
| 10 | | | 1 | |

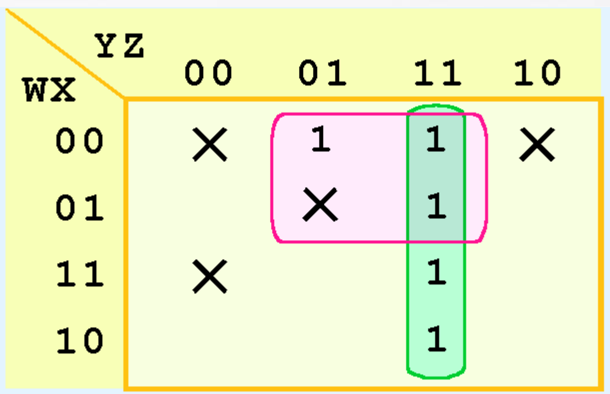
# 3A.6 Don't Care Conditions

- The truth table of: $F(W,X,Y,Z) = W'X' + YZ$

  differs from the truth table of:
  $F(W,X,Y,Z) = W'Z + YZ$

- However, the values for which they differ, are the inputs for which we have **don't care** conditions.



# 3A Summary

- **Kmaps** provide an easy graphical method for **simplifying Boolean functions**.
- A Kmap is a **matrix** consisting **of the outputs** of the minterms of a Boolean function.
- In this section, we have discussed 2- 3- and 4-input Kmaps. This method **can be extended to any number** of inputs through the use of multiple tables.

# 3A Summary

Recapping the **rules of Kmap simplification**:

- Groupings can contain only 1s; no 0s.
- Groups can be formed only at right angles; diagonal groups are not allowed.
- The number of 1s in a group must be a power of 2 – even if it contains a single 1.
- The groups must be made as large as possible.
- Groups can overlap and wrap around the sides of the Kmap.
- Use don't care conditions when you can.

# 3A Exercise

- Build a Kmap corresponding to the following truth table
- Derive the simplified function via building groups

| A | B | C | D | Out |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 |