EN1014 Electronic Engineering

Design of Combinational Logic Circuits

Chamira U. S. Edussooriya
B.Sc.Eng. (Moratuwa), M.A.Sc. (UVic), Ph.D. (UVic)

Department of Electronic and Telecommunication Engineering
University of Moratuwa

Some of the tables and figures included in this presentation have been extracted from
Digital Design: With an Introduction to the Verilog HDL (M. M. Mano and M. D. Ciletti, Prentice Hall, 2012)

Introduction

- The central task in designing a combinational logic circuit is finding an optimal gate-level implementation of the Boolean functions that describe a digital circuit.

Introduction

- The central task in designing a combinational logic circuit is finding an optimal gate-level implementation of the Boolean functions that describe a digital circuit.
- Such a design is difficult to carry out by manual methods when the circuit has more than a few inputs.
- Computer-based logic synthesis tools can minimize large sets of Boolean equations efficiently and quickly.
- However, it is important that a designer understands the underlying mathematical description and the solution of the problem.

The Karnaugh Maps (K-Maps)

- Simplification of a Boolean expression algebraically is not straightforward since it lacks specific rules to predict each succeeding step in the manipulation process.

The Karnaugh Maps (K-Maps)

- Simplification of a Boolean expression algebraically is not straightforward since it lacks specific rules to predict each succeeding step in the manipulation process.
- The graphical method, known as the Karnaugh maps (or K-maps) provides a simple and straightforward procedure for minimizing Boolean functions.
- A K-map is a diagram made up of squares, with each square representing one minterm or maxterm of the function that is to be minimized.

The Karnaugh Maps (K-Maps)

- Simplification of a Boolean expression algebraically is not straightforward since it lacks specific rules to predict each succeeding step in the manipulation process.
- The graphical method, known as the Karnaugh maps (or K-maps) provides a simple and straightforward procedure for minimizing Boolean functions.
- A K-map is a diagram made up of squares, with each square representing one minterm or maxterm of the function that is to be minimized.
- The simplified expressions produced by the K-maps are always in one of the two standard forms: sum of products or product of sums.

The Karnaugh Maps (K-Maps) *cont'd*

- In order to identify main properties of the K-maps, let us consider a three-variable K-map.

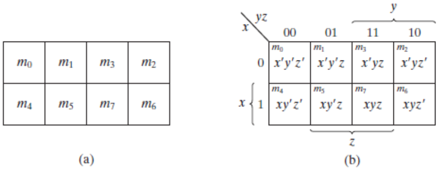

FIGURE 3.3
Three-variable K-map

- In order to identify main properties of the K-maps, let us consider a three-variable K-map.

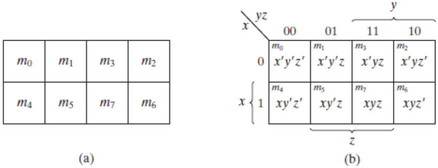

FIGURE 3.3
Three-variable K-map

- Note that
  - the minterms are arranged in a sequence similar to the Gray code (only one bit changes in value from one adjacent column to the next).

The Karnaugh Maps (K-Maps) *cont'd*

- In order to identify main properties of the K-maps, let us consider a three-variable K-map.

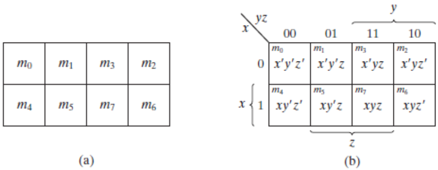

FIGURE 3.3
Three-variable K-map

- Note that
  - the minterms are arranged in a sequence similar to the Gray code (only one bit changes in value from one adjacent column to the next).
  - Any two adjacent squares in the map differ by only one variable, which is complemented in one square and uncomplemented in the other.

The Karnaugh Maps (K-Maps) *cont'd*

- In order to identify main properties of the K-maps, let us consider a three-variable K-map.

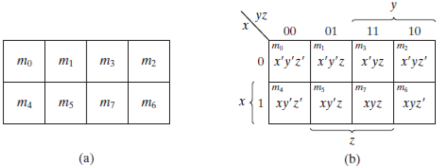

FIGURE 3.3
Three-variable K-map

- Note that
  - the minterms are arranged in a sequence similar to the Gray code (only one bit changes in value from one adjacent column to the next).
  - Any two adjacent squares in the map differ by only one variable, which is complemented in one square and uncomplemented in the other.
  - By using the postulates of Boolean algebra, it can be shown that the sum of two minterms in adjacent squares can be simplified to a single product term consisting of only two literals.

- Examples:
  - Simplify the following Boolean functions:
    (a) $F(x, y, z) = \Sigma(2, 3, 4, 5)$
    (b) $F(x, y, z) = \Sigma(3, 4, 6, 7)$
    (c) $F(x, y, z) = \Sigma(0, 2, 4, 5, 6)$
    (d) $F(a, b, c, d) = \Sigma(0, 1, 2, 6, 8, 9, 10)$.

The Karnaugh Maps (K-Maps) *cont'd*

- Note that in choosing adjacent squares in a map, we must ensure that
  - all the minterms of the function are covered
  - the number of terms in the expression is minimized
  - there are no redundant terms.

The Karnaugh Maps (K-Maps) *cont'd*

- Note that in choosing adjacent squares in a map, we must ensure that
  - all the minterms of the function are covered
  - the number of terms in the expression is minimized
  - there are no redundant terms.
- A simplified expression for a given Boolean function, which satisfies the above three conditions, may obtained from the logical sum of
  - all the essential prime implicants
  - other prime implicants that may be needed to cover any remaining minterms.

The Karnaugh Maps (K-Maps) *cont'd*

- Note that in choosing adjacent squares in a map, we must ensure that
  - all the minterms of the function are covered
  - the number of terms in the expression is minimized
  - there are no redundant terms.
- A simplified expression for a given Boolean function, which satisfies the above three conditions, may obtained from the logical sum of
  - all the essential prime implicants
  - other prime implicants that may be needed to cover any remaining minterms.
- Note that a prime implicant is a product term obtained by combining the maximum possible number of adjacent squares in the map.
- If a minterm in a square is covered by only one prime implicant, that prime implicant is said to be an essential prime implicant.

- For example, let us consider the simplification of the following Boolean function: $F(A, B, C, D) = \Sigma(0, 2, 3, 5, 7, 8, 9, 10, 11, 13, 15)$.

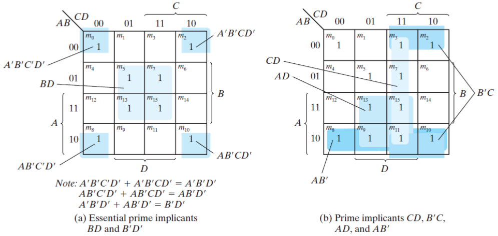

FIGURE 3.11
Simplification using prime implicants

- For example, let us consider the simplification of the following Boolean function: $F(A, B, C, D) = \Sigma(0, 2, 3, 5, 7, 8, 9, 10, 11, 13, 15)$.

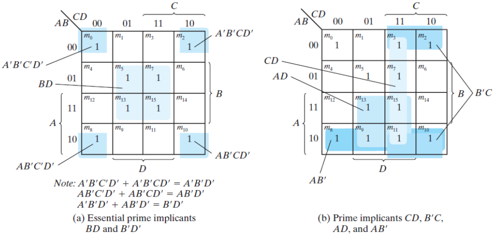

FIGURE 3.11
Simplification using prime implicants

- The four possible simplified expressions:
  - $F = BD + \bar{B}\bar{D} + CD + AD$
  - $F = BD + \bar{B}\bar{D} + CD + A\bar{B}$
  - $F = BD + \bar{B}\bar{D} + \bar{B}C + AD$
  - $F = BD + \bar{B}\bar{D} + \bar{B}C + A\bar{B}$.

- A five-variable K-map may be considered as two four-variable K-maps stacked one on top of the other.

- A five-variable K-map may be considered as two four-variable K-maps stacked one on top of the other.
- Example:
  - Simplify the following Boolean function:
    $F(a, b, c, d, e) = \Sigma(0, 1, 4, 5, 6, 8, 9, 14, 16, 17, 20, 21, 22, 30, 31)$.

The Karnaugh Maps (K-Maps) *cont'd*

- A simplified expression of a Boolean function F can be obtained in the product-of-sums form
  - directly - combining the maxterms of a K-map, and writing the each combination as a sum term. Then, ANDing all the sum terms.
  - indirectly - expressing the complement of the function, i.e. $\bar{F}$, in the sum-of-products form, and taking the complement of $\bar{F}$ using the DeMorgan's theorems.

The Karnaugh Maps (K-Maps) *cont'd*

- A simplified expression of a Boolean function F can be obtained in the product-of-sums form
  - directly - combining the maxterms of a K-map, and writing the each combination as a sum term. Then, ANDing all the sum terms.
  - indirectly - expressing the complement of the function, i.e. $\bar{F}$, in the sum-of-products form, and taking the complement of $\bar{F}$ using the DeMorgan's theorems.
- Example:
  - Express the following Boolean function in the product-of-sums form: $F(x, y, z) = \prod(0, 1, 2, 5, 6)$.

The Karnaugh Maps (K-Maps) *cont'd*

- Don't care conditions appeared in an incompletely specified function can be considered as either 0s or 1s in order to further simplify the function.

The Karnaugh Maps (K-Maps) *cont'd*

- Don't care conditions appeared in an incompletely specified function can be considered as either 0s or 1s in order to further simplify the function.
- Examples:
  - (a) Express the following Boolean function in the sum-of-products form: $f(a, b, c, d, e) = \sum(1, 3, 5, 7, 8, 10, 21, 23, 30, 31) + d(0, 2, 14, 15, 17, 18, 25)$.
  - (b) Express the following Boolean function in the product-of-sums form: $f(a, b, c, d, e) = \prod(1, 3, 5, 7, 10, 17, 19, 26, 30, 31) \cdot D(2, 8, 14, 15, 18, 20, 25)$.

NAND and NOR Implementation

- NAND and NOR gates are easier to fabricate with electronic components and are the basic gates used in all integrated circuit (IC) digital logic families.

NAND and NOR Implementation

- NAND and NOR gates are easier to fabricate with electronic components and are the basic gates used in all integrated circuit (IC) digital logic families.
- Consequently, digital circuits are often implemented with NAND or NOR gates instead of AND and OR gates.

NAND and NOR Implementation

- NAND and NOR gates are easier to fabricate with electronic components and are the basic gates used in all integrated circuit (IC) digital logic families.
- Consequently, digital circuits are often implemented with NAND or NOR gates instead of AND and OR gates.
- Home work:
  - Read Section 3.6 (pp. 90–97) of the text book.