# Lecture 5: Logic Simplication & Karnaugh Map

Professor Peter Cheung
Department of EEE, Imperial College London

(Floyd 4.5-4.11)
(Tocci 4.1-4.5)





## Points Addressed in this Lecture

- Standard form of Boolean Expressions
  - Sum-of-Products (SOP), Product-of-Sums (POS)
  - Canonical form
- Boolean simplification with Boolean Algebra
- Boolean simplification using Karnaugh Maps
- "Don't cares"





## Forms of Boolean Expressions

- Sum-of-products form (SOP)
  - first the product (AND) terms are formed then these are summed (OR)
  - eg: ABC + DEF + GHI
- Product-of-sum form (POS)
  - first the sum (OR) terms are formed then the products are taken (AND)
  - eg: (A+B+C) (D+E+F) (G+H+I)
- It is possible to convert between these two forms using Boolean algebra (DeMorgan's)





## Canonical Form

- Canonical form is not efficient but sometimes useful in analysis and design
- In an expression in canonical form, every variable appears in every term

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

- note that the dot (meaning AND) is often omitted

– An SOP expression can be forced into canonical form by ANDing the incomplete terms with terms of the form $(X + \overline{X})$ where X is the name of the missing variable

– eg: $f(A,B,C) = AB + BC$

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

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

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

– The product term in a canonical SOP expression is called a '**minterm**'





### A Notation using Canonical Form

- Previous example:
- Construct the truth table for this function
  - use a 0 when the variable is complemented, 1 otherwise

$$f(A,B,C) = ABC + AB\overline{C} + \overline{A}BC$$

| Row Number | A | B | C | f |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 |
| 2 | 0 | 1 | 0 | 0 |
| 3 | 0 | 1 | 1 | 1 |
| 4 | 1 | 0 | 0 | 0 |
| 5 | 1 | 0 | 1 | 0 |
| 6 | 1 | 1 | 0 | 1 |
| 7 | 1 | 1 | 1 | 1 |

  - *f* can be written as the sum of row numbers having TRUE minterms

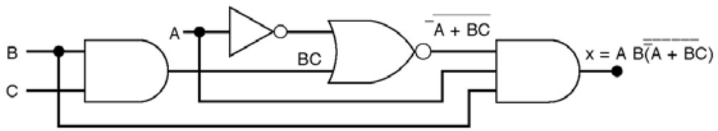
$$f = \sum(3,6,7)$$



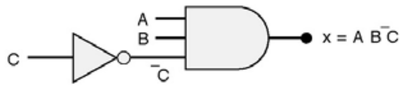


## Simplifying Logic Circuits

- First obtain one expression for the circuit, then try to simplify.
- Example:




(a)




(b)

- Two methods for simplifying
  - Algebraic method (use Boolean algebra theorems)
  - Karnaugh mapping method (systematic, step-by-step approach)





### Method 1: Minimization by Boolean Algebra

- Make use of relationships and theorems to simplify Boolean Expressions
  - perform algebraic manipulation resulting in a complexity reduction
  - this method relies on your algebraic skill
  - 3 things to try

- a) Grouping
  - Given

$$A + AB + BC$$

  - write it as

$$A(1+B) + BC$$

  - then apply

$$1 + B = 1$$

  - Minimized form

$$A + BC$$





- b) Multiplication by redundant variables
  - Multiplying by terms of the form $A + \overline{A}$ does not alter the logic
  - Such multiplications by a variable missing from a term may enable minimization
  - eg:

$$\begin{aligned} AB + A\overline{C} + BC &= AB(C+\overline{C}) + A\overline{C} + BC \\ &= ABC + AB\overline{C} + A\overline{C} + BC \\ &= BC(1+A) + A\overline{C}(1+B) \\ &= BC + A\overline{C} \end{aligned}$$





- c) Application of DeMorgan's Theorem
  - Expressions containing several inversions stacked one upon the other may often by simplified by applying DeMorgan's Theorem.
  - DeMorgan's Theorem "unwraps" the multiple inversion
  - eg:

$$\begin{aligned} \overline{\overline{A\overline{B}C} + \overline{ACD} + B\overline{C}} &= \overline{(\overline{A}+B+\overline{C}) + (\overline{A}+\overline{C}+\overline{D}) + B\overline{C}} \\ &= \overline{(\overline{A}+B+\overline{C}+\overline{D}) + B\overline{C}} \\ &= \overline{(\overline{A}+B+\overline{C}+\overline{D})} \\ &= A\overline{B}CD \end{aligned}$$





## Example of Logic Design

Design a logic circuit having 3 inputs, A, B, C will have its output HIGH only when a majority of the inputs are HIGH.

Step 1 Set up the truth table

Step 2 Write the AND term for each case where the output is a 1.

| A | B | C | x | |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | |
| 0 | 0 | 1 | 0 | |
| 0 | 1 | 0 | 0 | |
| 0 | 1 | 1 | 1 | $\rightarrow \overline{A}BC$ |
| 1 | 0 | 0 | 0 | |
| 1 | 0 | 1 | 1 | $\rightarrow A\overline{B}C$ |
| 1 | 1 | 0 | 1 | $\rightarrow AB\overline{C}$ |
| 1 | 1 | 1 | 1 | $\rightarrow ABC$ |

Step 3 Write the SOP form the output

Step 4 Simplify the output expression

$$x = \overline{A}BC + A\overline{B}C + AB\overline{C} + ABC$$

$$x = \overline{A}BC + ABC + A\overline{B}C + \boxed{ABC} + AB\overline{C} + \boxed{ABC}$$

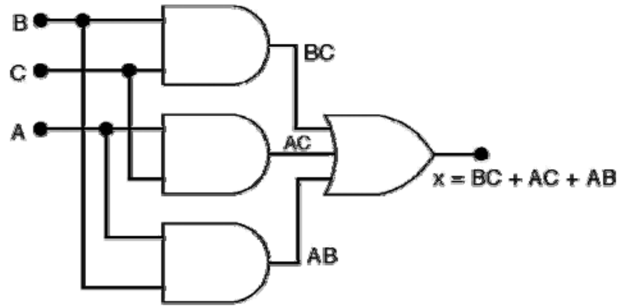
$$= BC(\overline{A} + A) + AC(\overline{B} + B) + AB(\overline{C} + C)$$

$$= BC + AC + AB$$





Step 5 Implement the circuit








## Minimization by Karnaugh Maps

- What is a Karnaugh map?
  - 3 Variable Example:

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

  - A grid of squares
  - Each square represents one minterm
    - eg: top-left represents $\overline{A}.\overline{B}.\overline{C}$ , bottom-right represents $A.B.\overline{C}$
  - The minterms are ordered according to Gray code
    - only one variable changes between adjacent squares
  - Squares on edges are considered adjacent to squares on opposite edges
  - Karnaugh maps become clumsier to use with more than 4 variables





- 4 Variable example

| AB\CD | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | | | | |
| 01 | | ? | ?? | |
| 11 | | | | |
| 10 | | | | |

- The square marked ? represents $\overline{A}.B.\overline{C}.D$
- The square marked ?? represents $\overline{A}.B.C.D$
- Note that they differ in only the C variable.

## Filling out a Karnaugh Map

- Write the Boolean expression in SOP form
- For each product term, write a 1 in all the squares which are included in the term, 0 elsewhere
  - canonical form: one square
  - one term missing: two adjacent squares
  - two terms missing: 4 adjacent squares
- Eg:

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

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





## Minimization Technique

- Minimization is done by spotting patterns of 1's and 0's
- Simple theorems are then used to simplify the Boolean description of the patterns
- Pairs of adjacent 1's
  - remember that adjacent squares differ by only one variable
  - hence the combination of 2 adjacent squares has the form

$$P(A + \overline{A})$$

  - this can be simplified (from before) to just *P*





- Take out previous example (Slide 12)

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

the adjacent squares $ABC$ and $\overline{A}BC$ differ only in *A*

- hence they can be combined into just *BC*
- normally indicated by grouping the adjacent squares to be combined

- Adjacent Pairs
  - The same idea extends to pairs of pairs





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

- "**Cover**" all the 1's with maximum grouping:

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

- The simplified Boolean equation is one that sums all the terms corresponding to each of the group:

$$X = AC + BC + AB$$

# More Examples of grouping

| AB\CD | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 1 | 0 | 0 |
| 01 | 0 | 1 | 0 | 0 |
| 11 | 1 | 1 | 1 | 1 |
| 10 | 0 | 1 | 0 | 0 |

| AB\CD | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 0 | 0 | 0 |
| 01 | 1 | 0 | 0 | 1 |
| 11 | 1 | 0 | 1 | 1 |
| 10 | 0 | 0 | 0 | 0 |

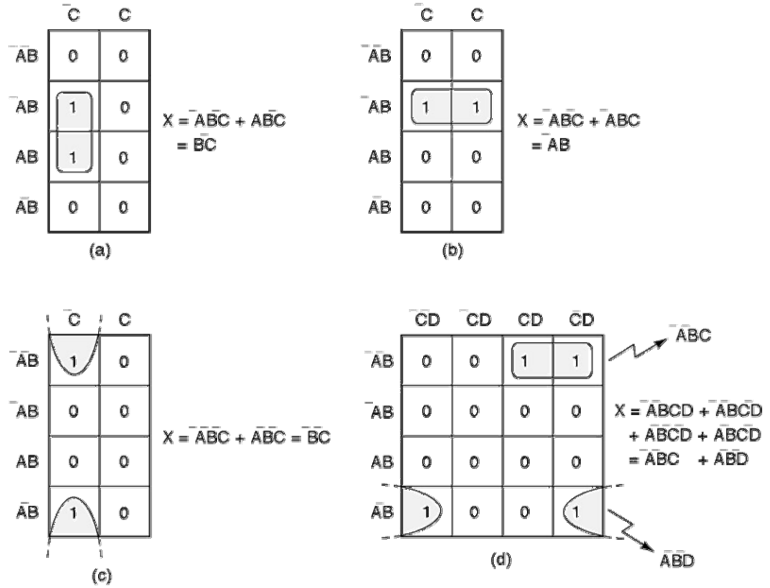
$A\,B + \overline{C}\,D$

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



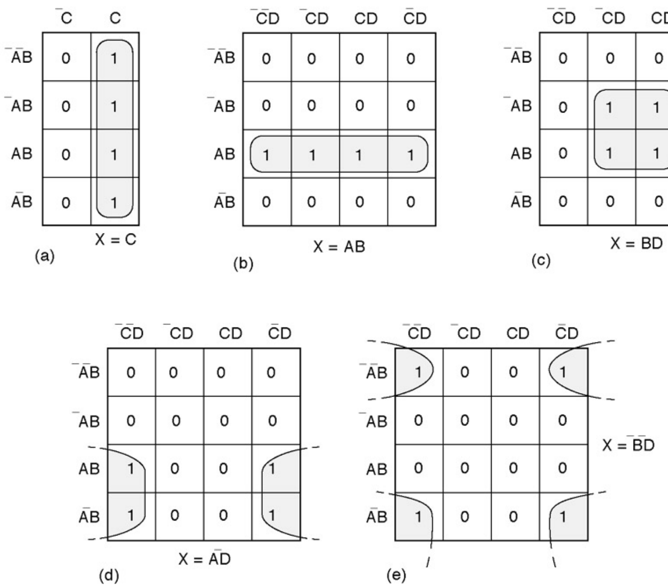


# More examples



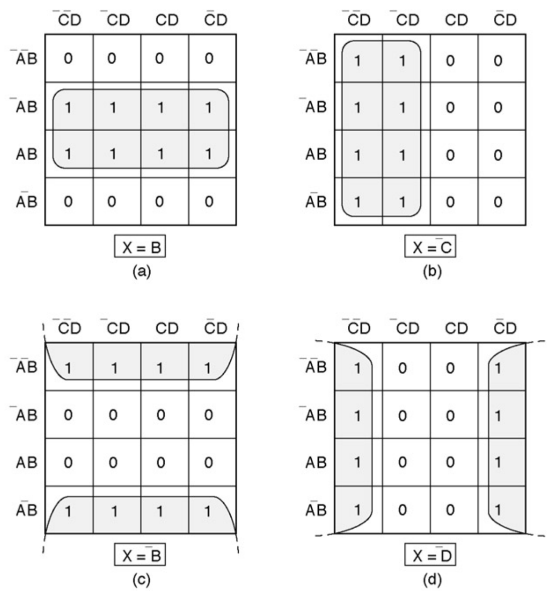








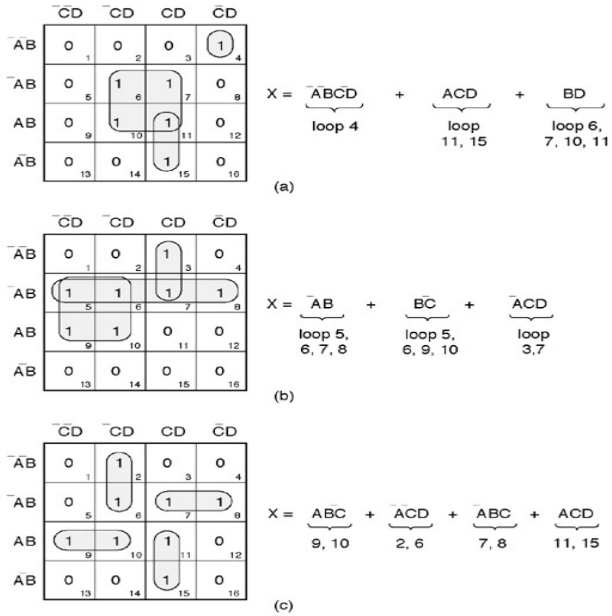
Imperial College London

# Complete Simplification Process

1. Construct the K map and place 1s and 0s in the squares according to the truth table.
2. Group the isolated 1s which are *not* adjacent to any other 1s. (single loops)
3. Group any pair which contains a 1 adjacent to only one other 1. (double loops)
4. Group any octet even if it contains one or more 1s that have already been grouped.
5. Group any quad that contains one or more 1s that have not already been grouped, *making sure to use the minimum number of groups.*
6. Group any pairs necessary to include any 1s that have not yet been grouped, *making sure to use the minimum number of groups.*
7. Form the OR sum of all the terms generated by each group.



Imperial College London






Imperial College London

## Don't Care Conditions

- 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 and X
    - meaning don't care
  - When minimizing an X is like a "joker"
    - X can be 0 or 1 - whatever helps best with the minimization
  - Eg:

| A\BC | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | X |
| 1 | 0 | 0 | 1 | 1 |

  - simplifies to *B* if X is assumed 1

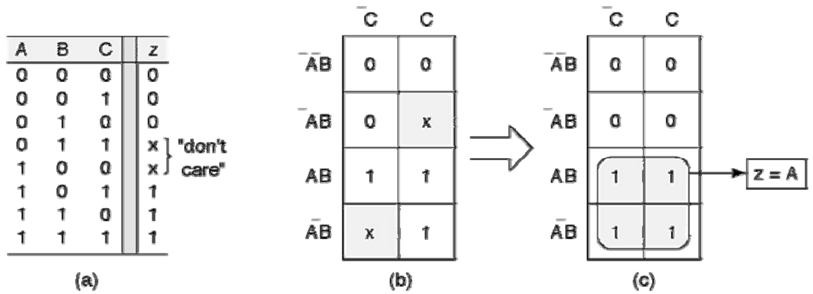
$$X = B$$

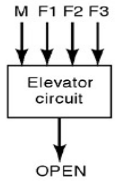


Imperial College London

## 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.






| M | F1 | F2 | F3 | 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 |

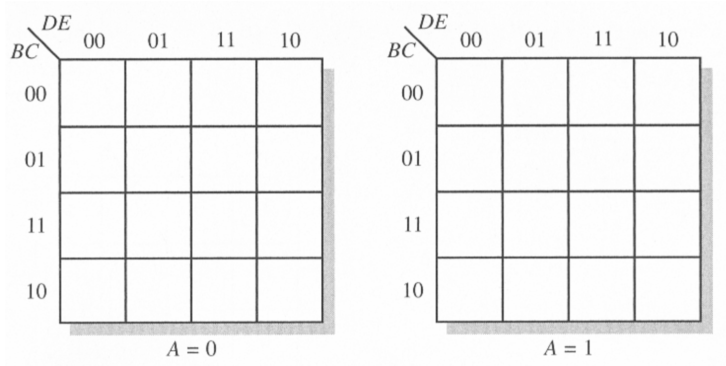
| | $\overline{F2}\,\overline{F3}$ | $\overline{F2}\,F3$ | $F2\,F3$ | $F2\,\overline{F3}$ |
|---|---|---|---|---|
| $\overline{M}\,\overline{F1}$ | 0 | 1 | X | 1 |
| $\overline{M}\,F1$ | 1 | X | X | X |
| $M\,F1$ | 0 | X | X | X |
| $M\,\overline{F1}$ | 0 | 0 | X | 0 |

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





## The Karnaugh Map with 5 variables








## K Map Method Summary

•Compared to the algebraic method, the K-map process is a more orderly process requiring fewer steps and always producing a minimum expression.

•The minimum expression in generally is NOT unique.

•For the circuits with large numbers of inputs (larger than four), other more complex techniques are used.





## Summary

- **SOP and POS –useful forms of Boolean equations**
- **<u>Design of a comb. Logic circuit</u>**

  **(1) construct its truth table, (2) convert it to a SOP, (3) simplify using Boolean algebra or K mapping, (4) implement**
- **<u>K map</u>: a graphical method for representing a circuit's truth table and generating a simplified expression**
- **"Don't cares" entries in K map can take on values of 1 or 0. Therefore can be exploited to help simplification**