COM1006 Devices and Networks (Autumn)
COM1090 Computer Architectures

Lecture #5

▶ Simplifying Circuits

Dr Dirk Sudholt
Department of Computer Science
University of Sheffield

d.sudholt@sheffield.ac.uk

http://staffwww.dcs.shef.ac.uk/~dirk/campus_only/com1006/

Based on Section 2.5 in Clements, Principles of Computer Hardware

Aims of this lecture

- To introduce Boolean algebra
- To show how Boolean algebra can be used design and simplify logic circuits
- To introduce Karnaugh maps as a visual aid in simplifying logic circuits
- To show how digital circuits can be implemented in NAND and NOR logic only

Boolean algebra

- George Boole (1815-1864) was an English mathematician and philosopher.
- He developed a logical calculus of truth values, which at the time was a relatively obscure work.
- Subsequently Claude Shannon and others showed how Boolean algebra could be used to implement logic in electrical switches.

Laws of Boolean algebra

- **Commutative law**: AND and OR operators are commutative so the order of the variables in a sum or product group doesn't matter:

  $A+B = B+A$

  $A \cdot B = B \cdot A$

- **Associative law**: AND and OR are associative so the order in which sub-expressions are evaluated doesn't matter:

  $(A \cdot B) \cdot C = A \cdot (B \cdot C)$

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

- **Distributive law**: AND behaves like multiplication and OR behaves like addition. In an expression containing both AND and OR operators, AND takes precedence over OR:

  $A \cdot (B+C) = A \cdot B + A \cdot C$

  $(A \cdot B)+C = (A+C) \cdot (B+C)$

Boolean identities

AND	OR	NOT
$0 \cdot X = 0$	$0 + X = X$	$\overline{\overline{X}} = X$
$1 \cdot X = X$	$1 + X = 1$	
$X \cdot X = X$	$X + X = X$	
$X \cdot \overline{X} = 0$	$X + \overline{X} = 1$	

Combining minterms

- For every Boolean formulas X and Y we have

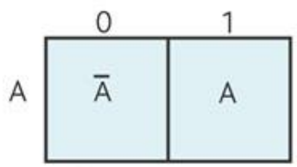
$$
\begin{aligned}
& \mathrm{XY} + \mathrm{X}\overline{\mathrm{Y}} \\
&= \mathrm{X}(\mathrm{Y} + \overline{\mathrm{Y}}) && \text{(distributive law)} \\
&= \mathrm{X} \cdot 1 && (\text{as } \mathrm{Y} + \overline{\mathrm{Y}} = 1) \\
&= \mathrm{X} && (\text{as } \mathrm{X} \cdot 1 = \mathrm{X})
\end{aligned}
$$

- This is a useful tool for simplifying an S-of-P expression.
- Example: $\mathrm{F} = \overline{\mathrm{A}}\mathrm{BC} + \mathrm{A}\overline{\mathrm{B}}\mathrm{C} + \mathrm{AB}\overline{\mathrm{C}} + \mathrm{ABC}$ from the majority circuit.
- Different ways of applying this rule repeatedly can yield different results.

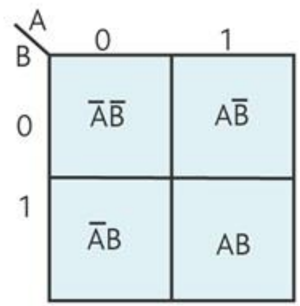
Karnaugh maps

- Systematic and visual method for combining minterms.
  - Invented by Edward Veitch in 1952
  - Refined by Maurice Karnaugh in 1953
- Idea:
  - **map** all minterms on a 2D grid
  - **neighbourhoods** in the grid indicate which terms can be combined.
- Works well for 2, 3 or 4 variables.
- More variables: use algorithm by Quine/McCluskey (out of the scope of this module)

Karnaugh maps

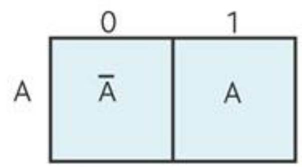


(a) One-variable Karnaugh map.

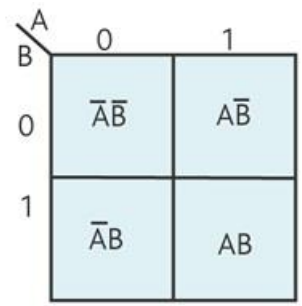


(b) Two-variable Karnaugh map.

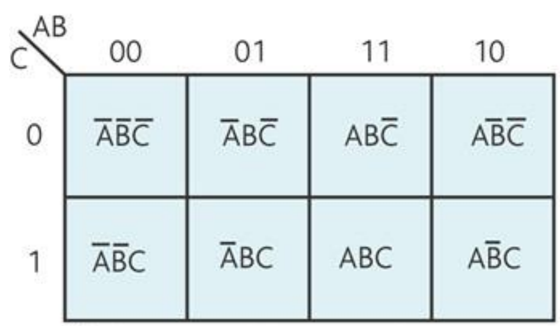
Karnaugh maps



(a) One-variable Karnaugh map.

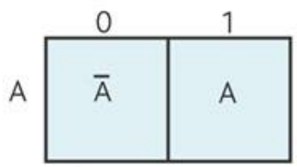


(b) Two-variable Karnaugh map.

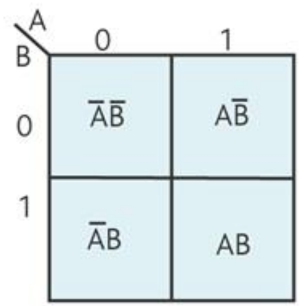


(c) Three-variable Karnaugh map.

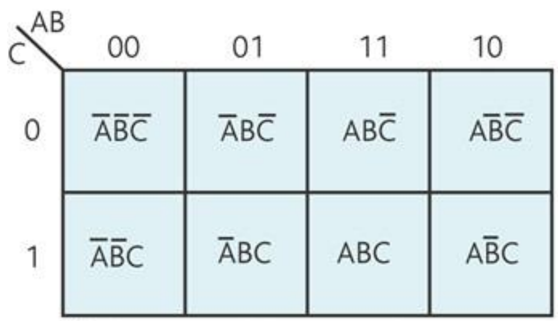
Karnaugh maps



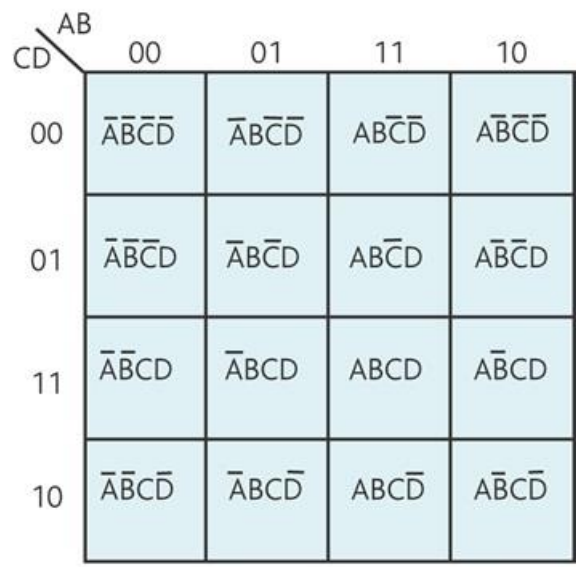
(a) One-variable Karnaugh map.



(b) Two-variable Karnaugh map.



(c) Three-variable Karnaugh map.



(b) Four-variable Karnaugh map.

▶Structure of Karnaugh maps

C \ AB	00	01	11	10
0	$\bar{A}\bar{B}\bar{C}$	$\bar{A}B\bar{C}$	$AB\bar{C}$	$A\bar{B}\bar{C}$
1	$\bar{A}\bar{B}C$	$\bar{A}BC$	ABC	$A\bar{B}C$

CD \ AB	00	01	11	10
00	$\bar{A}\bar{B}\bar{C}\bar{D}$	$\bar{A}B\bar{C}\bar{D}$	$AB\bar{C}\bar{D}$	$A\bar{B}\bar{C}\bar{D}$
01	$\bar{A}\bar{B}\bar{C}D$	$\bar{A}B\bar{C}D$	$AB\bar{C}D$	$A\bar{B}\bar{C}D$
11	$\bar{A}\bar{B}CD$	$\bar{A}BCD$	$ABCD$	$A\bar{B}CD$
10	$\bar{A}\bar{B}C\bar{D}$	$\bar{A}BC\bar{D}$	$ABC\bar{D}$	$A\bar{B}C\bar{D}$

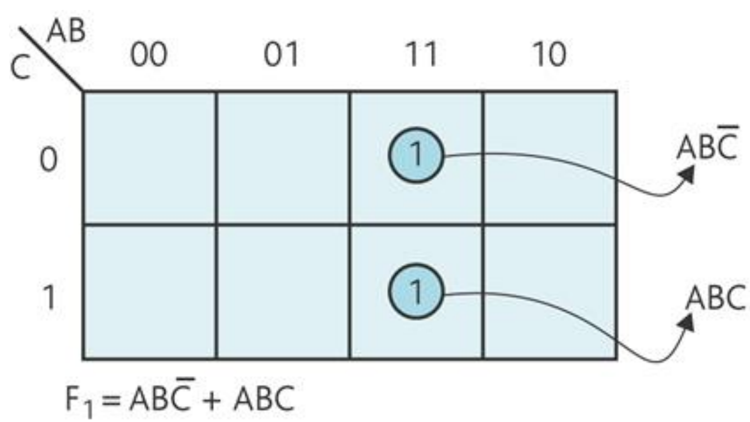
- Subsequent labels for columns/rows differ in exactly 1 bit:

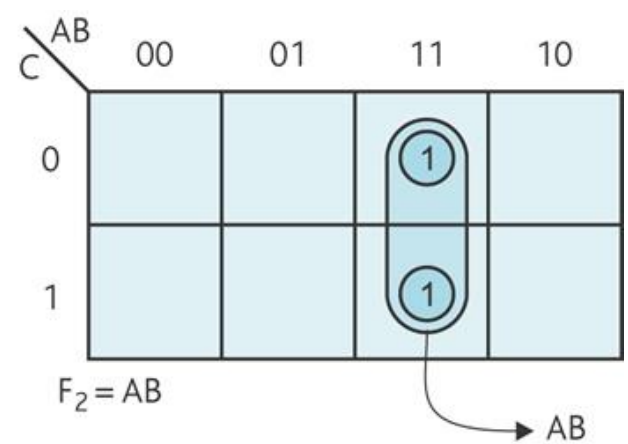
  00 → 01 → 11 → 10 → 00

- As a result, neighbouring cells differ in exactly one bit, i.e. minterms only differ in one negation (e.g. $AB\bar{C}$ vs. ABC).

Karnaugh maps

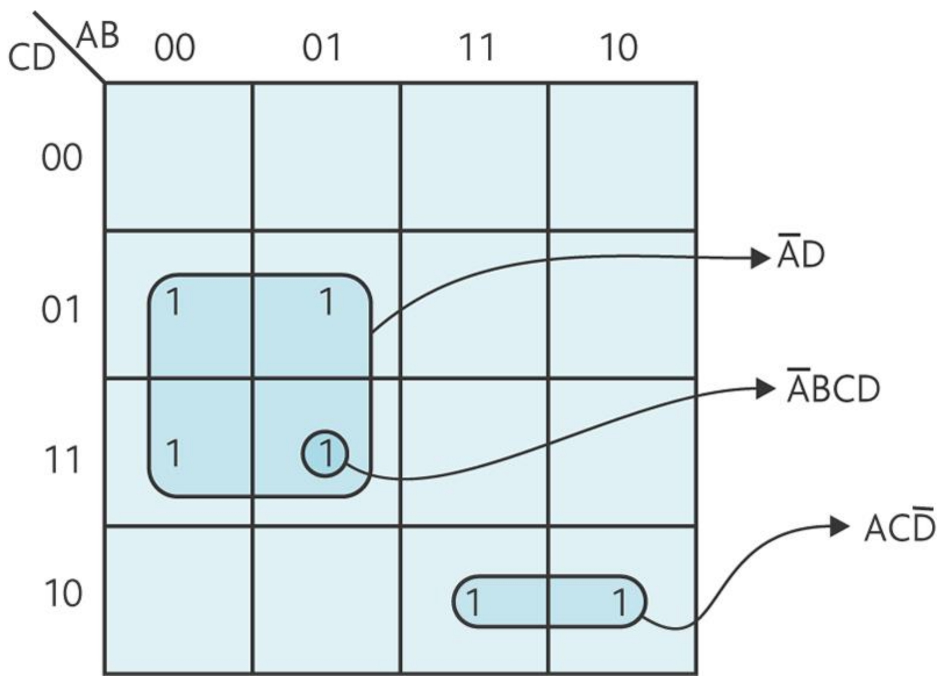
- A Karnaugh map works like a truth table.
- Typically truth values 1 are written down, empty cells are read as 0.
- Neighbouring 1s can be combined to give a simpler formula.





Karnaugh maps: rectangles

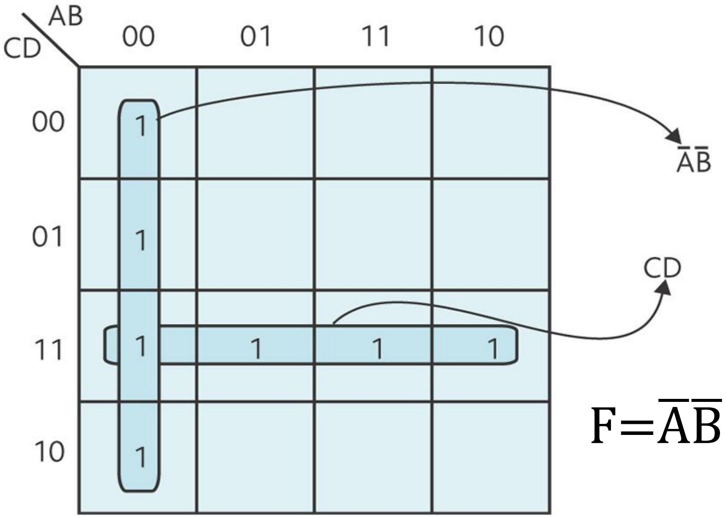
- Also works for larger rectangles if **side lengths** are **1,2, or 4.**
- Larger rectangles have less variables.



How to use Karnaugh maps

1. Enter all 1s according to the function to be simplified.
2. Draw rectangles with side lengths 1, 2, or 4 as to:
   - **cover all 1s and no empty cells (0s)** (to represent the function)
   - using as **few rectangles** as possible (→ few sums)
   - make rectangles as **large** as possible (→ small products)
3. Read off product terms and simplified formula.

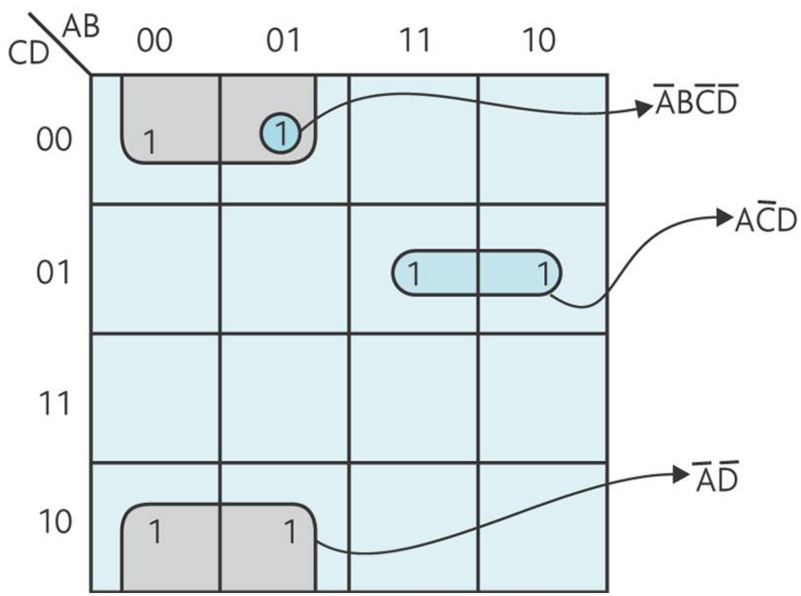
- Recall: side lengths 1, 2, or 4 are OK; 3 isn‘t!
- Rectangles may overlap.

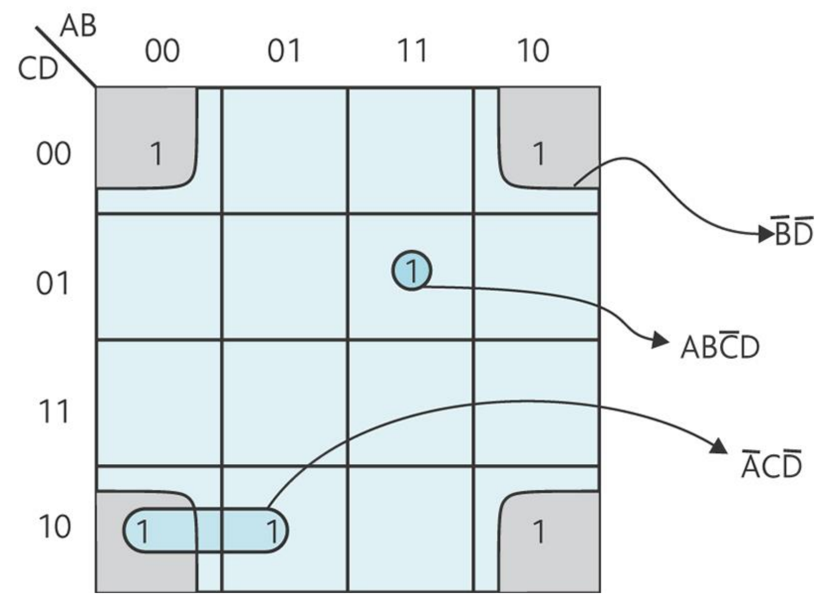


$F=\overline{A}\overline{B}+CD$

Wrapping around

- **Caution:** neighbourhoods “wrap around“ on all sides.





$$F = \overline{A}\,\overline{D} + A\overline{C}D$$

$$F = \overline{B}\,\overline{D} + AB\overline{C}D + \overline{A}C\overline{D}$$

Example: Majority circuit

- Use a Karnaugh map to simplify

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

De Morgan's theorem

- Two further rules of Boolean algebra are collectively known as De Morgan's theorem:

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

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

- In other words, to apply De Morgan's theorem to a Boolean function we change ANDs into ORs, change ORs into ANDs, and variables/literals are complemented.

- Example: Apply De Morgan's theorem to

$$F = \overline{X \cdot Y + X \cdot Z}$$

$$= \overline{X \cdot Y} \cdot \overline{X \cdot Z}$$

$$= (\overline{X} + \overline{Y}) \cdot (\overline{X} + \overline{Z})$$

Implementing in NAND and NOR only

- De Morgan’s theorem is important because it allows:
  - an AND gate to be implemented by an OR gate and an inverter;
  - an OR gate to be implemented by an AND gate and an inverter.
- Hence any device that can be made from a combination of ANDs, ORs and NOTs can be made using a combination of NANDs (or NORs) only.
- This is of practical value because NAND gates operate at higher speed than AND gates, and require fewer components at the chip level.

Example: implementing in NAND/NOR

- Recall the majority device
  $F = AB + BC + AC$

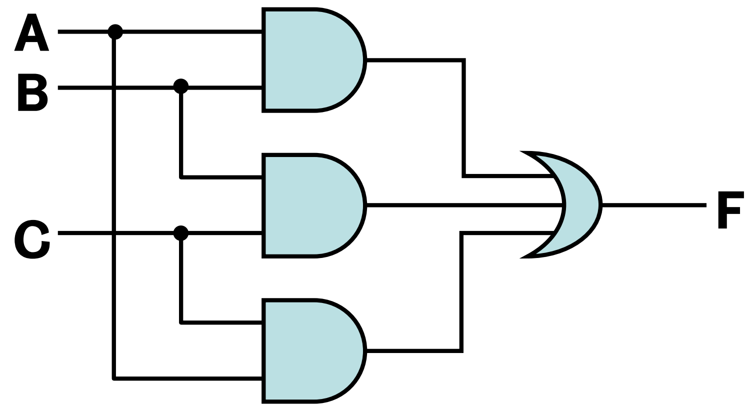
- We note that

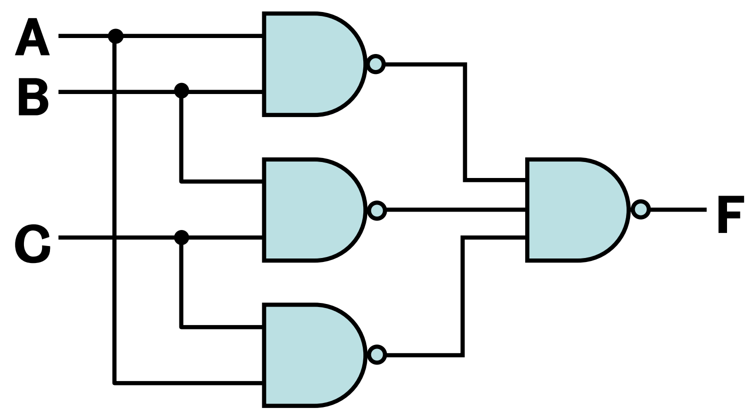
  $$F = \overline{\overline{F}} = \overline{\overline{AB + BC + AC}}$$

- Now apply De Morgan:

  $$F = \overline{\overline{AB} \cdot \overline{BC} \cdot \overline{AC}}$$

Exercise: use a truth table to verify that the two circuits are equivalent





Summary

- Boolean algebra provides a powerful means of describing and simplifying logic devices.
- Karnaugh maps visualise terms that can be grouped, leading to Boolean formulas with less and shorter terms than the canonical S-of-P form.
- De Morgan's theorem is important because it suggests a means by which any digital circuit can be implemented in NAND and NOR logic only.