Grade 12 A/L ICT · Lesson 4

4.2 Boolean Algebra and Karnaugh Maps: Laws, SOP, POS and Simplification

Syllabus 4.2. Digital circuits follow **Boolean algebra** – the maths of 0 and 1. In this part you learn the **postulates and laws** (commutative, associative, distributive, identity, redundancy, De Morgan), how to write expressions in **sum-of-products (SOP)** and **product-of-sums (POS)** form, how to **simplify** with theorems, and how a **Karnaugh map (K-map)** groups 1s so you can read a simpler expression by eye.

1. Why simplify Boolean expressions?

A simpler expression needs **fewer gates**, so the circuit is **cheaper, smaller, cooler and faster**. The exam often asks you to simplify with laws or with a K-map.

2. Postulates and laws

Boolean algebra uses only the values **0 and 1**. The basic operations are **AND (·)**, **OR (+)** and **NOT (′)**. Useful laws:



Boolean algebra laws

Law	AND form	OR form
Commutative	A · B = B · A	A + B = B + A
Associative	(A · B) · C = A · (B · C)	(A + B) + C = A + (B + C)
Distributive	A · (B + C) = A·B + A·C	A + (B · C) = (A+B)·(A+C)
Identity	A · 1 = A	A + 0 = A
Null / Annulment	A · 0 = 0	A + 1 = 1
Idempotent	A · A = A	A + A = A
Complement	A · A′ = 0	A + A′ = 1
Redundancy	A + A·B = A	A · (A + B) = A
De Morgan	(A · B)′ = A′ + B′	(A + B)′ = A′ · B′

© 2026 schoolict.net (Ruwan Mangala Suraweera) · All rights reserved

💡 **De Morgan in words:** break the bar and change the sign – an AND bar becomes OR of the complements; an OR bar becomes AND of the complements.

🧱 **Worked example: Check De Morgan for A = 1, B = 0: is (A · B)′ equal to A′ + B′?**
(1 · 0)′ = 0′ = 1. A′ + B′ = 0 + 1 = 1. Equal – De Morgan holds.

3. Standard forms: SOP and POS

Sum of Products (SOP) is an OR of AND terms, for example A·B + A′·C. Each AND term is a **minterm** when it contains every variable (or its complement) once. **Product of Sums (POS)** is an AND of OR terms, for example (A+B)·(A′+C). Each OR term that has every variable is a **maxterm**.

From a truth table: write one minterm for every row where F = 1, then OR them → **SOP**. Write one maxterm for every row where F = 0, then AND them → **POS**. SOP and POS of the same function are equal.

🧱 **Worked example: F(A,B,C) is 1 for minterms 1, 3, 5, 7. Write the SOP.**
m1=A′B′C, m3=A′BC, m5=AB′C, m7=ABC. SOP = A′B′C + A′BC + AB′C + ABC. This simplifies to **C** (every term has C, and the A/B parts cover all).

4. Simplifying with Boolean theorems

🧱 **Worked example: Simplify F = A · B + A · B′.**
Factor A: F = A · (B + B′) = A · 1 = **A**. (Identity and complement laws.)

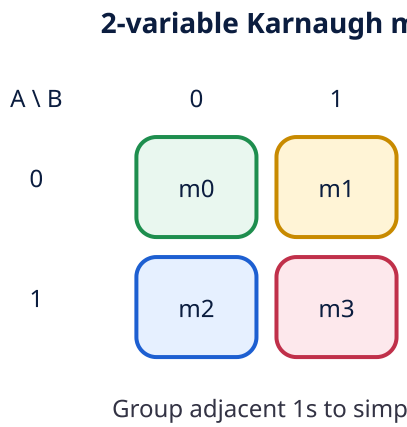
🧱 **Worked example: Simplify F = A + A′ · B.**
Absorption / redundancy: A + A′·B = (A + A′) · (A + B) wait – better: A + A′·B = A·1 + A′·B = A(B+B′) + A′·B … classic result: **A + B**.

🧱 **Worked example: Simplify F = (A + B) · (A + B′).**
Distributive: A + (B · B′) = A + 0 = **A**.

5. Karnaugh maps

A **Karnaugh map** is a truth table redrawn so that **adjacent cells differ by only one bit**. You circle groups of 1, 2, 4 or 8 ones (powers of two). Each group becomes one product term with the variables that stay constant. Larger groups → fewer literals → simpler circuit.

© 2026 schoolict.net (Ruwan Mangala Suraweera) · All rights reserved



A 2-variable Karnaugh map

2-variable K-map: rows A=0,1; columns B=0,1. Cells are minterms m0..m3. **3-variable K-map:** usually rows A, columns BC in Gray code order 00, 01, 11, 10 so that neighbours still differ by one bit. Edges wrap around.

🧱 **Worked example: Simplify F(A,B) = Σm(1,2,3) with a K-map.**
Cells m1,m2,m3 are 1. Group m1+m3 (B=1, A varies) → B. Group m2+m3 (A=1, B varies) → A. Result: **A + B**.

🧱 **Worked example: Simplify F(A,B,C) = Σm(0,1,2,3,6) with a K-map.**
The four cells with A=0 (m0–m3) form a group of 4 → **A′**. Cell m6 (ABC=110) groups with m2 (010) → **B · C′**. Result: **A′ + B·C′**.

💡 Never make a group of 3 or 5. Only 1, 2, 4, 8, … Cover every 1 at least once; overlapping is allowed; don’t circle 0s.

🚀 **Beyond the syllabus (2026 update)** *Not required for the exam – read for interest.* Logic synthesis tools in 2026 (Synopsys, Cadence, open-source Yosys) still use Boolean algebra and Espresso-style minimisation behind the scenes, then map the result onto millions of standard-cell gates. Learning K-maps by hand teaches you what those tools are doing.

Key Words

Boolean algebra – Algebra of the values 0 and 1 with AND, OR and NOT.

Commutative law – Order does not matter: A·B = B·A and A+B = B+A.

Associative law – Grouping does not matter for AND or for OR.

Distributive law – A·(B+C) = A·B+A·C and A+(B·C) = (A+B)·(A+C).

De Morgan’s laws – (A·B)′ = A′+B′ and (A+B)′ = A′·B′.

SOP – Sum of Products – an OR of AND terms.

© 2026 schoolict.net (Ruwan Mangala Suraweera) · All rights reserved

POS – Product of Sums – an AND of OR terms.

Minterm – A product term that includes every variable once (true or complemented).

Maxterm – A sum term that includes every variable once.

Karnaugh map – A grid that groups adjacent 1s to simplify a Boolean function.

Literal – A variable or its complement (A or A′).

Redundancy law – A + A·B = A (and the dual A·(A+B) = A).

Sample Q&A

1. State De Morgan's two laws.

> **Answer:** (A · B)′ = A′ + B′ and (A + B)′ = A′ · B′.

2. What is SOP form?

> **Answer:** Sum of Products: the function is written as an OR of AND terms (minterms).

3. Why do we simplify Boolean expressions?

> **Answer:** Fewer gates → smaller, cheaper, cooler and often faster circuits.

4. Simplify A · B + A · B′.

> **Answer:** A · (B + B′) = A · 1 = A.

5. What size groups are allowed on a K-map?

> **Answer:** Powers of two: 1, 2, 4, 8, … (never 3 or 5).

6. How do you get a POS expression from a truth table?

> **Answer:** Write a maxterm for every row where F = 0, then AND those maxterms.

MCQ

1. De Morgan: (A · B)′ equals

A. A′ + B′
B. A + B
C. A · B
D. A′ · B′
E. (A + B)′

© 2026 schoolict.net (Ruwan Mangala Suraweera) · All rights reserved

2. SOP means

A. Signed Octal Product
B. Sum of Products
C. Product of Sums
D. Sum of Powers
E. Single Output Process

3. POS means

A. Product of Sums
B. Product of Squares
C. Sum of Products
D. Positive Output Signal
E. Parallel Output Sum

4. A · 1 equals

A. 0
B. A · A
C. A
D. 1
E. A′

5. A + A′ equals

A. 0
B. 1
C. A
D. A′
E. A · A′

6. Simplify A · B + A · B′

A. B
B. A · B
C. 0
D. A
E. A + B

7. A K-map group may contain how many 1s?

A. 1, 2, 4 or 8
B. Any odd number
C. Only 4
D. 3 or 5
E. Only 2

© 2026 schoolict.net (Ruwan Mangala Suraweera) · All rights reserved

8. Adjacent K-map cells differ by

A. all bits
B. three bits
C. two bits
D. no bits
E. exactly one bit

9. A + A · B simplifies to

A. A
B. B
C. 1
D. A · B
E. A + B

10. F = Σm(1,2,3) for two variables simplifies to

A. A′
B. A + B
C. B′
D. A · B
E. 1

11. (A + B)′ equals

A. A′
B. A + B
C. A′ + B′
D. A′ · B′
E. A · B

12. A minterm for three variables has how many literals?

A. 1
B. 8
C. 4
D. 2
E. 3

13. Which term means: “Algebra of the values 0 and 1 with AND, OR and NOT”?

A. Minterm
B. Associative law
C. Literal
D. SOP
E. Boolean algebra

© 2026 schoolict.net (Ruwan Mangala Suraweera) · All rights reserved

14. Which term means: “Order does not matter: A·B = B·A and A+B = B+A”?

A. Maxterm
B. Commutative law
C. Redundancy law
D. Distributive law
E. POS

15. Which term means: “Grouping does not matter for AND or for OR”?

A. Boolean algebra
B. De Morgan’s laws
C. Minterm
D. Associative law
E. Karnaugh map

16. Which term means: “A·(B+C) = A·B+A·C and A+(B·C) = (A+B)·(A+C)”?

A. Commutative law
B. Maxterm
C. Distributive law
D. Literal
E. SOP

17. Which term means: “(A·B)′ = A′+B′ and (A+B)′ = A′·B′”?

A. POS
B. Redundancy law
C. Associative law
D. De Morgan’s laws
E. Karnaugh map

18. Which term means: “Sum of Products – an OR of AND terms”?

A. Distributive law
B. Minterm
C. SOP
D. Literal
E. Boolean algebra

19. Which term means: “Product of Sums – an AND of OR terms”?

A. Maxterm
B. Commutative law
C. POS
D. De Morgan’s laws
E. Redundancy law

© 2026 schoolict.net (Ruwan Mangala Suraweera) · All rights reserved

20. Which term means: "A product term that includes every variable once (true or complemented)"?

A. Karnaugh map
B. Boolean algebra
C. Associative law
D. SOP
E. Minterm

Structured Questions

1. State the commutative, associative and distributive laws for AND and OR. *(3 marks)*

> Commutative: A·B=B·A, A+B=B+A (1); Associative: (A·B)·C=A·(B·C), same for + (1); Distributive: A·(B+C)=A·B+A·C and A+(B·C)=(A+B)·(A+C) (1).

2. Explain SOP and POS and how each is obtained from a truth table. *(3 marks)*

> SOP = OR of minterms for F=1 rows (1); POS = AND of maxterms for F=0 rows (1); both represent the same function (1).

3. Simplify F = A + A′ · B using Boolean laws. Show the steps. *(2 marks)*

> A + A′·B = (A + A′)(A + B) is one route, or note redundancy expansion → A + B (2); final answer A + B (1).

4. Using a 2-variable K-map, simplify F = Σm(1,2,3). *(3 marks)*

> Place 1s in m1,m2,m3 (1); group → A and B (1); F = A + B (1).

5. Apply De Morgan to simplify (A · B′)′. *(2 marks)*

> (A · B′)′ = A′ + (B′)′ (1); (B′)′ = B so A′ + B (1).

Summary

- Boolean laws (commutative, associative, distributive, De Morgan, identity, redundancy) let you rewrite expressions.
- SOP is an OR of products; POS is an AND of sums – both come from the truth table.
- Simplifying cuts the number of gates, so circuits are cheaper and faster.
- A K-map groups adjacent 1s in sizes 1, 2, 4, 8 to read a minimal SOP.
- De Morgan: break the bar and change the operator.

MCQ answer key – 4.2

1. **A** – Break the bar, change · to +.

© 2026 schoolict.net (Ruwan Mangala Suraweera) · All rights reserved

2. **B** – OR of AND terms.
3. **A** – AND of OR terms.
4. **C** – Identity law.
5. **B** – Complement law.
6. **D** – A(B+B′)=A.
7. **A** – Powers of two only.
8. **E** – Gray-code adjacency.
9. **A** – Redundancy law.
10. **B** – K-map groups give A + B.
11. **D** – De Morgan for OR.
12. **E** – One literal per variable.
13. **E** – Boolean algebra – Algebra of the values 0 and 1 with AND, OR and NOT.
14. **B** – Commutative law – Order does not matter: A·B = B·A and A+B = B+A.
15. **D** – Associative law – Grouping does not matter for AND or for OR.
16. **C** – Distributive law – A·(B+C) = A·B+A·C and A+(B·C) = (A+B)·(A+C).
17. **D** – De Morgan's laws – (A·B)′ = A′+B′ and (A+B)′ = A′·B′.
18. **C** – SOP – Sum of Products – an OR of AND terms.
19. **C** – POS – Product of Sums – an AND of OR terms.
20. **E** – Minterm – A product term that includes every variable once (true or complemented).

© 2026 schoolict.net (Ruwan Mangala Suraweera). All rights reserved.