

CLASS 9 • COMPUTER SCIENCE

& Entrepreneurship

CHAPTER 3

Digital Systems & Logic Design

Solved Exercise

MCQs • Short Questions • Long Questions

Step-by-step solutions with explanations

STUDY SMARTER

scholareye.xyz — Learn. Practise. Excel.

EXERCISE — Overview

Section	Questions
Multiple-Choice Questions	5
Short Questions	5
Long Questions	9

How to use this PDF: In every MCQ the correct option is **highlighted in green** and marked with a ✓, followed by a short explanation. Short and long questions are solved step by step.

Quick notation: + means OR, · means AND, ⊕ means XOR, and a bar over a letter, like $\bar{A}$, means NOT (complement).

Multiple-Choice Questions (MCQs)

Q1. Which of the following Boolean expressions represents the OR operation?

(a) $A \cdot B$

(b) $A + B$ ✓

(c) $\bar{A}$

(d) $A \oplus B$

✓ **Answer: (b) $A + B$**

Explanation: In Boolean algebra, the plus sign (+) denotes the logical OR operation. Option (a) $A \cdot B$ is the AND operation, (c) $\bar{A}$ is the NOT operation, and (d) $A \oplus B$ is the XOR (Exclusive-OR) operation.

Q2. What is the dual of the Boolean expression $A \cdot 0 = 0$?

(a) $A + 1 = 1$ ✓

(b) $A + 0 = A$

(c) $A \cdot 1 = A$

(d) $A \cdot 0 = 0$

✓ **Answer: (a) $A + 1 = 1$**

Explanation: The duality principle says: swap AND ($\cdot$) with OR ($+$), and swap the identity elements 0 with 1 (and vice versa). Therefore, $A \cdot 0 = 0$ becomes $A + 1 = 1$.

Q3. Which logic gate outputs true only if both inputs are true?

(a) OR gate

(b) **AND gate**



(c) XOR gate

(d) NOT gate

✓ **Answer: (b) AND gate**

Explanation: An AND gate gives a high output (1/True) only when all its inputs are high. If any input is 0, the output is 0. An OR gate outputs true if at least one input is true, and an XOR gate outputs true if the inputs are different.

Q4. In a half-adder circuit, the carry is generated by which operation?

(a) XOR operation

(b) **AND operation** ✓

(c) OR operation

(d) NOT operation

✓ **Answer: (b) AND operation**

Explanation: A half-adder adds two single bits (A and B). The Sum is calculated using XOR ($A \oplus B$), while the Carry is generated using AND ($A \cdot B$). The carry is 1 only when both A and B are 1.

Q5. What is the decimal equivalent of the binary number 1101?

(a) 11

(b) 12

(c) 13



(d) 14

✓ **Answer: (c) 13**

Explanation: Multiply each bit by 2 raised to its position (right to left, starting at 0): $(1 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0) = 8 + 4 + 0 + 1 = 13$.

Short Questions

Q1. Define a Boolean function and give an example.

A Boolean function is a mathematical function that takes one or more binary inputs (0 or 1) and produces a single binary output (0 or 1) based on Boolean logic operations (AND, OR, NOT).

Example:

$$F(A, B) = A \cdot B + \bar{A} \cdot \bar{B}$$

This represents the XNOR function.

Q2. What is the significance of the truth table in digital logic?

A truth table is a tabular representation that lists all possible combinations of input values and the corresponding output values for a logic gate or circuit. Its significance lies in providing a complete, unambiguous functional definition of a logic circuit. It is the primary tool used by engineers to analyze, verify, and design digital systems without needing to physically build them first.

Q3. Explain the difference between analog and digital signals.

- **Analog Signals:** These are continuous signals that can take any value within a given range over time. They vary smoothly (e.g., sound waves, temperature, human voice).
- **Digital Signals:** These are discrete signals that can only take specific, distinct values (usually represented as 0s and 1s). They change in steps rather than continuously (e.g., computer data, digital clocks).

Q4. Describe the function of a NOT gate with its truth table.

A NOT gate (also called an inverter) is a logic gate with a single input and a single output. Its function is to invert the input signal; if the input is 0, the output is 1, and vice versa.

Truth Table:

Input (A)	Output (Y)
0	1
1	0

Q5. What is the purpose of a Karnaugh map in simplifying Boolean expressions?

A Karnaugh map (K-map) is a graphical tool used to simplify Boolean algebra expressions. Its purpose is to provide a visual method for identifying and eliminating redundant terms in a Boolean expression. By grouping adjacent 1s (for Sum-of-Products) or 0s (for Product-of-Sums) in powers of two, the K-map helps derive the simplest possible logic circuit, reducing the number of gates required.

Long Questions

Q1. Explain the usage of Boolean functions in computers.

Boolean functions form the foundational logic of all computer hardware. At the lowest level, digital circuits are constructed using logic gates (AND, OR, NOT, NAND, NOR, etc.), which are physical implementations of Boolean functions. They are used in:

- **Arithmetic Logic Units (ALUs):** For performing binary addition, subtraction, multiplication, and division.
- **Control Units:** For decoding instructions and making decision-based logic branches (if-else conditions).
- **Memory and State Management:** Flip-flops and latches use Boolean logic to store bits.
- **Data Routing:** Multiplexers and demultiplexers use Boolean functions to route data through a CPU.

Essentially, every operation a computer performs, from booting up to running complex software, is reducible to millions of Boolean logic operations.

Q2. Describe how to construct a truth table for a Boolean expression with an example.

Steps to Construct a Truth Table:

1. Identify the number of input variables (n).
2. Determine the number of rows needed: 2^n . For example, 2 variables = 4 rows, 3 variables = 8 rows.
3. List all possible binary combinations of inputs in ascending order (e.g., 00, 01, 10, 11).
4. Create intermediate columns for complex sub-expressions if needed.
5. Evaluate the final output for each combination.

Example:

$$F = A + \bar{B}$$

A	B	$\bar{B}$	$F = A + \bar{B}$
0	0	1	1
0	1	0	0
1	0	1	1
1	1	0	1

Q3. Describe the concept of duality in Boolean algebra and provide an example to illustrate it.

Concept of Duality: Duality is a fundamental principle in Boolean algebra. It states that if you take a valid Boolean equation and interchange the AND ($\cdot$) operator with the OR ($+$) operator, and interchange the identity elements 0 with 1 (and vice versa), the resulting equation (the dual) is also valid. Variables and their complements remain unchanged.

Example:

$$\text{Original: } A + 0 = A$$

$$\text{Dual: } A \cdot 1 = A$$

Another Example: De Morgan's Law

$$\text{Original: } \overline{A + B} = \overline{A} \cdot \overline{B}$$

$$\text{Dual: } \overline{A \cdot B} = \overline{A} + \overline{B}$$

Q4. Compare and contrast half-adders and full-adders, including their truth tables, Boolean expressions, and circuit diagrams.

Half-Adder: Adds two single binary bits (A, B). It produces a Sum (S) and a Carry (C). It cannot accept a carry input from a previous stage.

Boolean Expressions:

$$S = A \oplus B \quad C = A \cdot B$$

Truth Table:

A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Full-Adder: Adds three single binary bits (A, B, and Carry-In C_{in}). It produces a Sum (S) and a Carry-Out (C_{out}).

Boolean Expressions:

$$S = A \oplus B \oplus C_{in}$$

$$\text{Cout} = A \cdot B + \text{Cin} \cdot (A \oplus B)$$

Truth Table:

A	B	Cin	S	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Circuit Diagrams (block form):



Contrast: A Half-Adder is used only for the least significant bit (LSB) position in multi-bit addition because there is no incoming carry. A Full-Adder is used for all subsequent higher-order bits because it

must account for the carry propagating from the previous stage. A Full-Adder can be constructed using two Half-Adders and an OR gate.

Feature	Half-Adder	Full-Adder
Inputs	2 (A, B)	3 (A, B, Cin)
Outputs	Sum, Carry	Sum, Cout
Carry-in	No	Yes
Used for	LSB only	Higher bits

Q5. How do Karnaugh maps simplify Boolean expressions? Provide a detailed example with steps.

How K-maps Simplify: K-maps simplify expressions by visually grouping adjacent cells containing 1s (for SOP) or 0s (for POS) into rectangles of size 1, 2, 4, 8, etc. (powers of 2). Groups must be as large as possible and can wrap around edges. By grouping, variables that change state within the group are eliminated, leaving only the variables that remain constant.

Detailed Example: Simplify

$$F(A, B, C) = \Sigma(1, 3, 5, 7)$$

1. **Draw the K-map:** Create a 2×4 grid for 3 variables. Label rows for A (0, 1) and columns for BC (00, 01, 11, 10).
2. **Plot the 1s:** Place a 1 in the cells corresponding to minterms 1 (001), 3 (011), 5 (101), and 7 (111).
3. **Group the 1s:** Group the four 1s in the columns where $C = 1$ (the 01 and 11 columns). This forms a group of 4.
4. **Extract the simplified term:** In this group, variable A changes from 0 to 1, and variable B changes from 0 to 1. Only variable C remains constant as 1.
5. **Result:** $F = C$

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

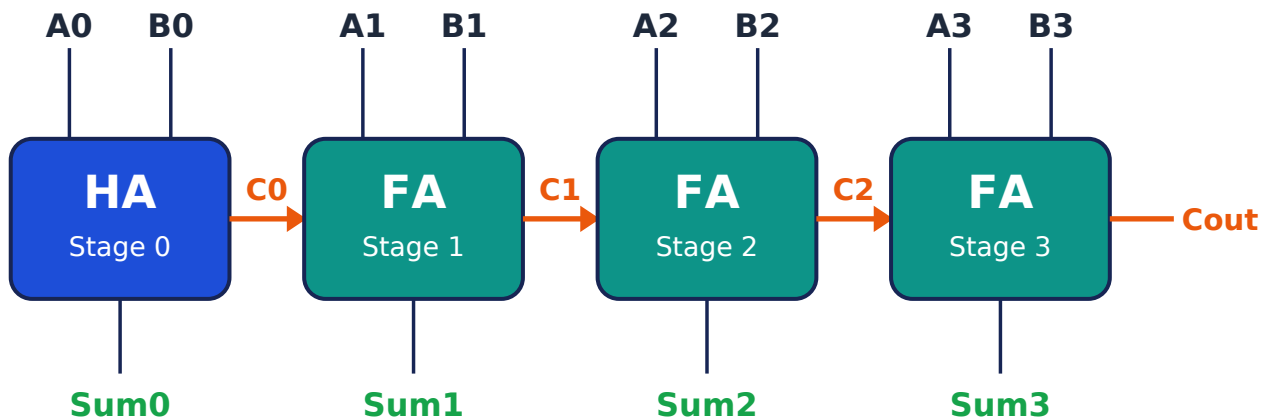
Green group of four 1s $\rightarrow F = C$

$$F = C$$

Q6. Design a 4-bit binary adder using both half-adders and full-adders. Explain each step with truth tables, Boolean expressions, and circuit diagrams.

Design Steps:

1. **Architecture:** A 4-bit adder requires 4 stages to add $A[3:0]$ and $B[3:0]$ along with an initial Carry-In (C_{in}), producing $Sum[3:0]$ and a final Carry-Out (C_{out}).
2. **Stage 0 (LSB):** Use a **Half-Adder** (or a Full-Adder with $C_{in} = 0$) to add A_0 and B_0 . This produces Sum_0 and $Carry_0$.
3. **Stages 1, 2, and 3:** Use **Full-Adders** because they need to accept the carry from the previous stage.
 - **Stage 1:** Adds A_1 , B_1 , and $Carry_0$. Produces Sum_1 and $Carry_1$.
 - **Stage 2:** Adds A_2 , B_2 , and $Carry_1$. Produces Sum_2 and $Carry_2$.
 - **Stage 3:** Adds A_3 , B_3 , and $Carry_2$. Produces Sum_3 and $Carry_3$ (Final C_{out}).
4. **Circuit Diagram:** The circuit is drawn by cascading the blocks: $HA \rightarrow FA \rightarrow FA \rightarrow FA$. The Carry-Out of each stage connects directly to the Carry-In of the next.



5. Boolean Expressions:

Stage 0 (Half-Adder):

$$\text{Sum0} = A0 \oplus B0 \quad \text{Carry0} = A0 \cdot B0$$

Stages 1-3 (Full-Adder):

$$S = A \oplus B \oplus \text{Cin}$$

$$\text{Cout} = A \cdot B + \text{Cin} \cdot (A \oplus B)$$

These equations are replicated across stages 1-3, while stage 0 uses the Half-Adder equations. The truth tables of the Half-Adder and Full-Adder are the same as shown in Question 4.

Q7. Simplify the following Boolean function using Boolean algebra rules: $F(A, B) = A \cdot B + A \cdot \overline{B}$

Step 1: Apply the Distributive Law (factoring out A):

$$F = A \cdot (B + \overline{B})$$

Step 2: Apply the Complement Law ($B + \overline{B} = 1$):

$$F = A \cdot 1$$

Step 3: Apply the Identity Law ($A \cdot 1 = A$):

$$F = A$$

Q8. Use De Morgan's laws to simplify the following function: $F(A, B, C) = \overline{A + B} + AC$

Note: Based on the image text, the expression is read as $\overline{A + B} + AC$

Step 1: Identify the part of the expression to apply De Morgan's Law:

$$\overline{A + B}$$

Step 2: Apply De Morgan's Law (the complement of a sum is the product of the complements):

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

Step 3: Substitute back into the original function:

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

This is the simplified Sum-of-Products form. It cannot be simplified further using standard Boolean algebra.

Q9. Simplify the following expressions:**(a)**

$$\overline{A + B} \cdot (A + B)$$

- Let $X = (A + B)$. The expression becomes $\overline{X} \cdot X$.
- According to the Complement Law, a variable ANDed with its complement is always 0.

Answer: 0**(b)**

$$(A + \overline{B}) \cdot (A + B)$$

- Apply the Distributive Law: $A \cdot A + A \cdot B + \overline{B} \cdot A + \overline{B} \cdot B$
- Simplify: $A + AB + A\overline{B} + 0$
- Factor out A: $A(1 + B + \overline{B})$
- Since $1 + \text{anything} = 1$, we get $A \cdot 1$
- Alternatively, using the theorem $(X + Y)(X + Z) = X + YZ$ with $X = A$, $Y = \overline{B}$, $Z = B$: $A + \overline{B} \cdot B = A + 0 = A$.

Answer: A

(c)

$$A + \bar{A} \cdot (\bar{B} + C)$$

- Apply the Distributive Law ($X + YZ = (X + Y)(X + Z)$): $(A + \bar{A}) \cdot (A + \bar{B} + C)$
- Apply the Complement Law ($A + \bar{A} = 1$): $1 \cdot (A + \bar{B} + C)$

Answer: $A + \bar{B} + C$

(d)

$$\bar{A} \cdot B + \bar{A} \cdot \bar{B}$$

- Factor out $\bar{A}$: $\bar{A} \cdot (B + \bar{B})$
- Apply the Complement Law ($B + \bar{B} = 1$): $\bar{A} \cdot 1$

Answer: $\bar{A}$

(e)

$$(A \cdot B) + (\bar{A} \cdot \bar{B})$$

- This expression represents the XNOR (Exclusive-NOR) function.
- It is already in its minimal Sum-of-Products (SOP) form and cannot be simplified further using basic Boolean algebra rules.

Answer: $(A \cdot B) + (\bar{A} \cdot \bar{B})$ (or equivalently, $\overline{A \oplus B}$)



Thank You!

Keep Learning, Keep Growing

Chapter 3 • Digital Systems & Logic Design
Class 9 Computer Science & Entrepreneurship

scholareye.xyz

More solved exercises and study material