Class : 9th

Subject : Computer Science

Chapter : 3 Digital Systems and Logic Design

Book Exercise Complete Solved

Punjab Board

1. Which of the following Boolean expressions represents the OR operation?
✅ **Correct Answer: (b) A + B**

Options:
(a) A·B
✅ (b) A + B
(c) A
(d) A ⊕ B

Explanation:
OR operation is represented by + sign in Boolean algebra → A + B.

---

2. What is the dual of the Boolean expression A·0 = 0?
✅ **Correct Answer: (a) A + 1 = 1**

Options:
✅ (a) A + 1 = 1
(b) A · 1 = A
(c) A + 1 = A
(d) A · 0 = 0

Explanation:
In dual, replace · with +, and `0` with `1`:
A · 0 = 0 → Dual = A + 1 = 1

---

3. Which logic gate outputs true only if both inputs are true?
✅ **Correct Answer: (b) AND gate**

stepacademyofficial.com

CamScanner

Options:
(a) OR gate
✅ (b) AND gate
(c) XOR gate
(d) NOT gate

Explanation:
AND gate outputs 1 only when both inputs are 1.

---

4. In a half-adder circuit, the carry is generated by which operation?
✅ **Correct Answer: (b) AND operation**

Options:
(a) XOR operation
✅ (b) AND operation
(c) OR operation
(d) NOT operation

Explanation:
Carry = $A \cdot B$ → AND operation

---

5. What is the decimal equivalent of the binary number 1101?
✅ **Correct Answer: (c) 13**

Options:
(a) 11
(b) 12
✅ (c) 13
(d) 14

Explanation:
$1101_2 = (1\times8) + (1\times4) + (0\times2) + (1\times1) = 8 + 4 + 0 + 1 = \mathbf{13}$

Short Questions

1. **Define a Boolean function and give an example.**
   A Boolean function uses Boolean algebra and returns either 0 or 1.
   Example: F = A + B
2. **What is the significance of the truth table in digital logic?**
   A truth table lists all possible inputs and their corresponding output, helping to understand and design logical circuits.

stepacademyofficial.com

CS CamScanner

3. **Explain the difference between analog and digital signals.**

- **Analog signal:** Continuous and varies over time (e.g., sound waves).
- **Digital signal:** Discrete values (0 and 1), used in computers and electronics.

4. **Describe the function of a NOT gate with its truth table.**
   The NOT gate inverts the input.

Input (A)	Output (¬A)
0	1
1	0

5. **What is the purpose of a Karnaugh map in simplifying Boolean expressions?**
   Karnaugh maps (K-maps) visually group terms in a Boolean expression to eliminate redundancy and simplify logic circuits.

Long Question

1. Explain the usage of Boolean functions in computers.

Answer:
Boolean functions play a fundamental role in computers. They are used in:

- **Logic gates:** Every operation in a computer (AND, OR, NOT, etc.) is defined by Boolean functions.
- **Decision making:** Computers use Boolean expressions to make decisions (e.g., if-else).
- **Digital circuits:** Boolean functions are the building blocks of circuits like adders, multiplexers, and memory.
- **Arithmetic operations:** Adders and ALUs (Arithmetic Logic Units) use Boolean logic to perform calculations.

Example:
A + B = Sum using OR gate
A · B = Carry using AND gate

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

Answer:

A **truth table** is a table that shows **all possible input combinations** of a Boolean expression and the **corresponding output** for each combination. It helps in analyzing and designing logic circuits.

stepacademyofficial.com

CamScanner

Steps to Construct a Truth Table

Step 1: Identify the number of variables

- Count how many input variables are in the Boolean expression.
- **Example:** If the expression is `F = A · B`, there are **2 variables** (A and B).

Step 2: Calculate total combinations

- Use the formula: 2^n, where `n` is the number of variables.
- For 2 variables: $2^2 = 4$ combinations.

Step 3: List all input combinations

- Write down all the binary combinations of inputs from `00` to `11`.

Step 4: Solve the Boolean expression for each row

- Apply the Boolean logic (AND, OR, NOT, etc.) to each combination.

Example: Boolean Expression → F = A + B

This expression uses the **OR operation** (+).

Truth Table for F = A + B

A	**B**	**A + B (F)**
0	0	0
0	1	1
1	0	1
1	1	1

Explanation of Each Row:

- **Row 1:** A = 0, B = 0 → A + B = 0
- **Row 2:** A = 0, B = 1 → A + B = 1
- **Row 3:** A = 1, B = 0 → A + B = 1
- **Row 4:** A = 1, B = 1 → A + B = 1

stepacademyofficial.com

CS CamScanner

(OR gate gives output 1 if **any input is 1**.)

◈ Another Example (With NOT & AND):

Let’s solve: $F = A \cdot B + \neg A$

A	B	¬A	A·B	F = A·B + ¬A
0	0	1	0	1
0	1	1	0	1
1	0	0	0	0
1	1	0	1	1

✓ Conclusion:

Truth tables are **essential tools** in Boolean algebra and computer science.
They help in:

- Verifying Boolean expressions
- Designing and simplifying logic circuits
- Understanding input/output behavior

Agar aap is ka **Urdu tarjuma**, **PDF version**, ya aur koi expression ka truth table chahen tu zaroor batao bhai — mein ready hoon!

You said:

bhai again solve kro aur question ko smjo wo bol kya rha h 3 truth table bna do usy tum kya kar rhy ho

ChatGPT said:

Shukriya bhai! Aap bilkul theek keh rahe ho — question ko properly samajhna zaroori hai.
Question yeh keh raha hai:

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

Aur **aap chahte ho ke 3 different Boolean expressions ke truth tables banaye jayein** — taake concept poora clear ho jaye.

stepacademyofficial.com

CS CamScanner

✅ Detailed Answer: How to Construct a Truth Table (with 3 Examples)

🗒 What is a Truth Table?

A **truth table** shows **all possible combinations** of input variables and the **corresponding output** of a Boolean expression. It is used to check the logic of expressions and circuits.

🔷 Step-by-Step Method:

1. **Identify variables** in the expression (like A, B, C).
2. **Count combinations**: For n variables → 2^n rows in the truth table.
3. **List all binary combinations** of inputs.
4. **Apply the Boolean expression** for each row and write the output.

◆ Example 1: F = A + B

(**OR Operation**)

A	B	A + B (F)
0	0	0
0	1	1
1	0	1
1	1	1

✅ **Explanation:** OR gate gives `1` if **any one input is 1**.

◆ Example 2: F = A · B

(**AND Operation**)

stepacademyofficial.com

CS CamScanner

A	B	A · B (F)
0	0	0
0	1	0
1	0	0
1	1	1

✅ **Explanation:** AND gate gives `1` **only if both inputs are 1**.

◆ Example 3: F = A · B + ¬A

(**AND + OR + NOT Combination**)

A	B	¬A	A · B	F = A·B + ¬A
0	0	1	0	1
0	1	1	0	1
1	0	0	0	0
1	1	0	1	1

✅ **Explanation:**

- Row 1: A=0 → ¬A=1, A·B=0 → F = 0 + 1 = 1
- Row 4: A=1, B=1 → A·B=1, ¬A=0 → F = 1 + 0 = 1

✅ Conclusion:

To construct a truth table:

- Identify variables.
- List all possible binary inputs.
- Apply the Boolean logic.
- Record the result for each case.

stepacademyofficial.com

CS CamScanner

Truth tables help in designing **accurate and error-free logic circuits**.

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

Answer:
The **duality principle** in Boolean algebra states that every Boolean expression remains valid if we:

- Replace AND (·) with OR (+)
- Replace OR (+) with AND (·)
- Replace 0 with 1 and 1 with 0

Example:
Original: $A + 0 = A$
Dual: $A \cdot 1 = A$

Both are valid identities in Boolean algebra.

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

Answer:

Feature	Half-Adder	Full-Adder
Inputs	2 (A, B)	3 (A, B, Carry-in)
Outputs	Sum, Carry	Sum, Carry-out
Carry Handling	Only current bits	Also handles previous carry
Boolean Expr.	$Sum = A \oplus B$	$Sum = A \oplus B \oplus Cin$
	$Carry = A \cdot B$	$Carry = A{\cdot}B + B{\cdot}Cin + A{\cdot}Cin$

Truth Table (Full-Adder):

A	B	Cin	Sum	Carry
0	0	0	0	0
0	1	0	1	0
1	0	0	1	0
1	1	0	0	1
0	0	1	1	0
0	1	1	0	1
1	0	1	0	1

stepacademyofficial.com

CS CamScanner

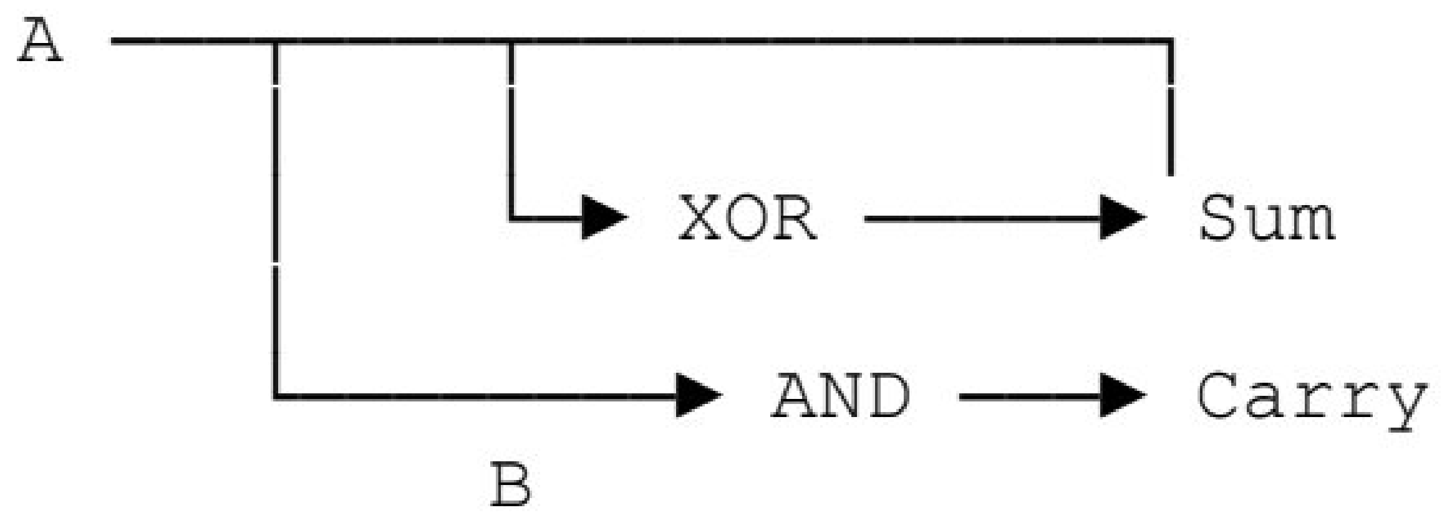
A	B	Cin	Sum	Carry
1	1	1	1	1

Circuit diagrams show Half-Adder using XOR & AND gates; Full-Adder using two Half-Adders and one OR gate.

◆ Circuit Diagrams:

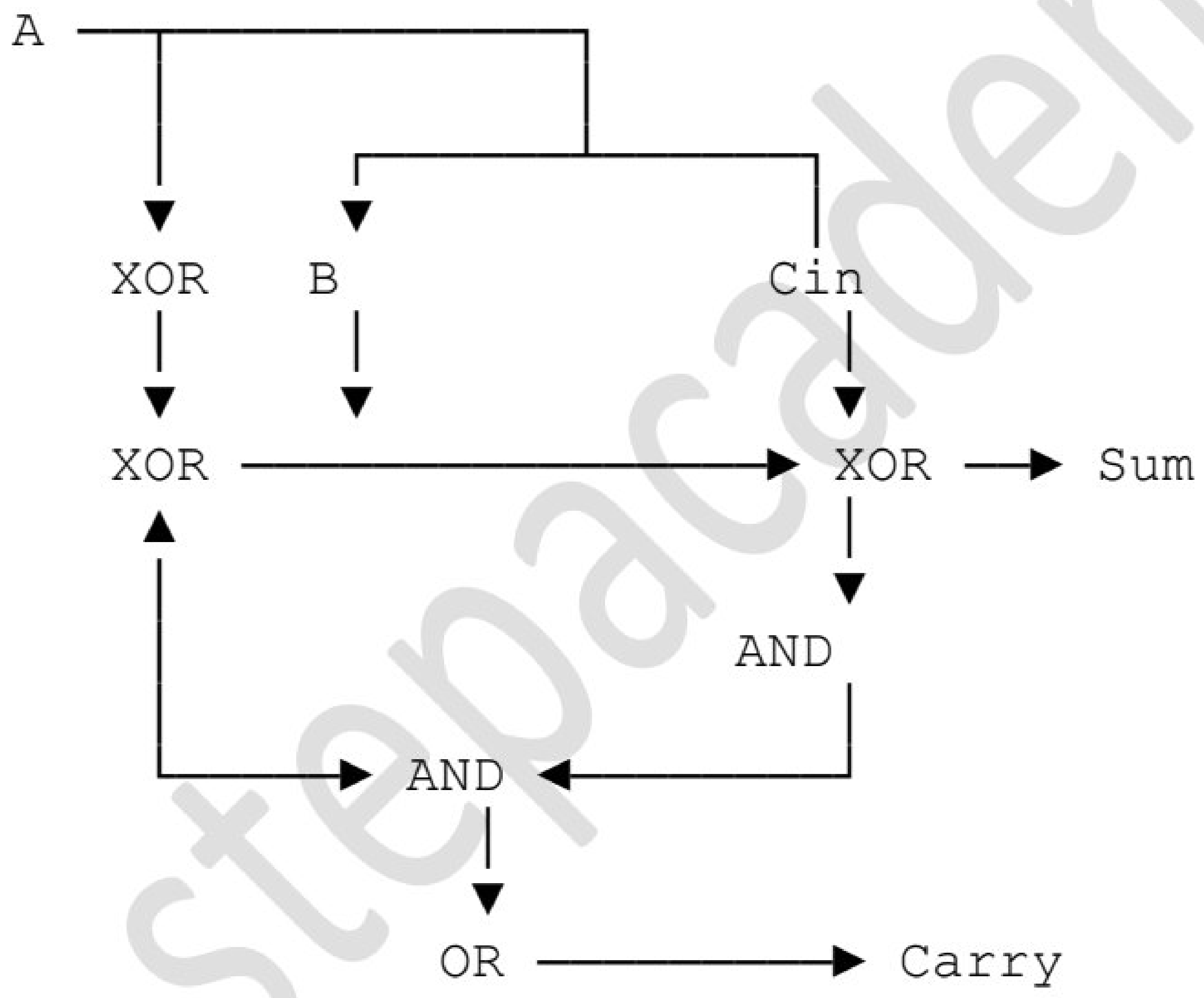
◆ Half-Adder Circuit:

- One **XOR gate** for Sum
- One **AND gate** for Carry



◆ Full-Adder Circuit:

- Two **half-adders**
- One **OR gate** for final Carry



✓ Conclusion:

- A **half-adder** adds only two bits and cannot process carry input.
- A **full-adder** handles carry input and is used in constructing multi-bit adders like **4-bit binary adders**.
- Both are essential components in **digital circuits and arithmetic units**.

stepacademyofficial.com

CamScanner

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

Answer:

Karnaugh Map (K-Map) is a diagram used to simplify Boolean expressions.

Basic Rules:

1. Group only 1s (not 0s)
2. Groups must contain **1, 2, 4, or 8** boxes (powers of 2)
3. Groups should be **as large as possible**
4. Wrap-around grouping is allowed
5. Each 1 must be included at least once

◆ Example: Simplify F(A, B) = Σ(1, 2, 3)

This means the function F is 1 when the minterms (decimal values) **1, 2, and 3** are true.

◆ Step 1: Create the 2-variable K-Map

AB	**00**	**01**	**11**	**10**
F	0	1	1	1

Here:

- Cell 00 → F = 0
- Cell 01 → F = 1 (minterm 1)
- Cell 10 → F = 1 (minterm 2)
- Cell 11 → F = 1 (minterm 3)

◆ Step 2: Draw and Group the 1s

Group the 1s into **largest possible rectangle** of **2 or 4**:

- Group 1: Cell 10 (A=1, B=0) and Cell 11 (A=1, B=1) → Common: A=1
- Group 2: Cell 01 and 11 → Common: B=1

◆ Step 3: Write the Simplified Expression

From groups:

- Group 1 → A

stepacademyofficial.com

CamScanner

- Group 2 → B

✅ **Simplified Expression:** **$F = A + B$**

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

A **4-bit binary adder** is a digital circuit that adds **two 4-bit binary numbers**. For example:

```
A = A3 A2 A1 A0
B = B3 B2 B1 B0
```

It produces a **4-bit sum** (S3 S2 S1 S0) and a **carry-out** from the last bit.

◆ Components Used:

- 1 **Half-Adder** for the first (least significant) bit
- 3 **Full-Adders** for the remaining bits

◆ Structure:

Bit Position	Adder Type	Inputs	Outputs
Bit 0 (LSB)	Half-Adder	A0, B0	S0, Carry0
Bit 1	Full-Adder	A1, B1, Carry0	S1, Carry1
Bit 2	Full-Adder	A2, B2, Carry1	S2, Carry2
Bit 3 (MSB)	Full-Adder	A3, B3, Carry2	S3, Carry3 (final carry)

◆ Boolean Expressions:

✅ *Half-Adder (Bit 0):*

- Sum (S0) = $A0 \oplus B0$
- Carry0 = $A0 \cdot B0$

✅ *Full-Adder (Bits 1 to 3):*

For each full-adder:

- Sum (S) = $A \oplus B \oplus Cin$

stepacademyofficial.com

CS CamScanner

- Carry = $(A \cdot B) + (Cin \cdot (A \oplus B))$

◆ Truth Table for Full-Adder (1-bit):

A	**B**	**Cin**	**Sum**	**Carry**
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 Design:

✅ *Half-Adder (Bit 0):*

- XOR gate for Sum $(A0 \oplus B0)$
- AND gate for Carry $(A0 \cdot B0)$

✅ *Full-Adders (Bits 1–3):*

Each full-adder consists of:

- Two XOR gates
- Two AND gates
- One OR gate

Each bit receives:

- Two inputs (A and B)
- One carry from the previous bit

◆ Block Diagram Overview:

```
markdown
CopyEdit
      A3   A2   A1   A0
       |    |    |    |
      FA   FA   FA   HA      ← Adders used
```

stepacademyofficial.com

CamScanner

```
       |   |   |   |
      B3  B2  B1  B0
       |   |   |   |

    Carry3 S3  S2  S1  S0
```

- **HA** = Half-Adder
- **FA** = Full-Adder
- The final output is a **5-bit result**: Carry3 + S3 S2 S1 S0

✓ Conclusion:

A 4-bit binary adder is constructed by:

- Using **1 Half-Adder** for the first bit
- Using **3 Full-Adders** for the next bits
- Connecting the **carry-out of each adder** to the **carry-in of the next**

This circuit is the basic building block of digital devices like **ALUs**, **calculators**, and **processors**.

7. Simplify the following Boolean function using Boolean algebra rules:

$F(A, B) = A \cdot B + A \cdot B$

Answer:

$F = A \cdot B + A \cdot B$

$= A(B + B)$

$= A \cdot 1$

✓ **$F = A$**

8. Use De Morgan's laws to simplify the following function:

$F(A, B, C) = \neg(A + B + AC)$

Answer:

First, simplify inside:
$A + B + AC = (A + B + A \cdot C)$

Apply De Morgan's Law:
$\neg(A + B + AC) = \neg A \cdot \neg B \cdot \neg(A \cdot C)$

stepacademyofficial.com

CS CamScanner

Now apply De Morgan’s to last term:
$\neg(A\cdot C) = \neg A + \neg C$

So,
$F = \neg A \cdot \neg B \cdot (\neg A + \neg C)$

✓ Final simplified expression.

9. Simplify the following expressions:

(a) $A + B \cdot (A + B)$
$= A + A\cdot B + B\cdot B$
$= A + A\cdot B + B$
$= A + B$ (Using Absorption Law)
✓ **A + B**

(b) $(A + B) \cdot (A + B)$
$= A + B$ (Idempotent Law)
✓ **A + B**

(c) $A + A \cdot (B + C)$
$= A$ (Using Absorption Law)
✓ **A**

(d) $A\cdot B + A\cdot B$
$= A$ (Using Rule: $A\cdot B + A\cdot B = A$)
✓ **A**

(e) $(A\cdot B) + (A\cdot B)$
$= A\cdot B$ (Idempotent Law)
✓ **A·B**

stepacademyofficial.com

CS CamScanner

stepacademyofficial.com

CS CamScanner