Boolean Algebra and Logic Gates

Introduction to Digital Systems

- **Digital World:** Every device you use (computer, phone, calculator) is a digital system.
- **Core Principle:** Digital systems only understand two distinct states: ON or OFF.
- **Binary Code:** These states are represented by the binary digits 1 (ON, True, High Voltage) and 0 (OFF, False, Low Voltage).

The Need for Boolean Algebra

- **Conventional Algebra Limits:** Standard algebra (addition, subtraction) doesn't work for logical statements.
- **Boolean Algebra:** A specialized branch of mathematics developed by George Boole (mid-19th century).
- **Purpose:** Provides a systematic method for analyzing and simplifying logical (binary) statements and circuits.
- **Key Idea:** It allows engineers to design complex circuits using the fewest possible components.

Core Concepts: Binary Variables

- **Boolean Variable:** A symbol (like A, B, X) used to represent a logical quantity.
- **Value Restriction:** A Boolean variable can only have one of two values: 1 (True) or 0 (False).
- **Literals:** A variable in its normal form (A) or its complemented form ($\bar{A}$).

Boolean Operations: The Basics

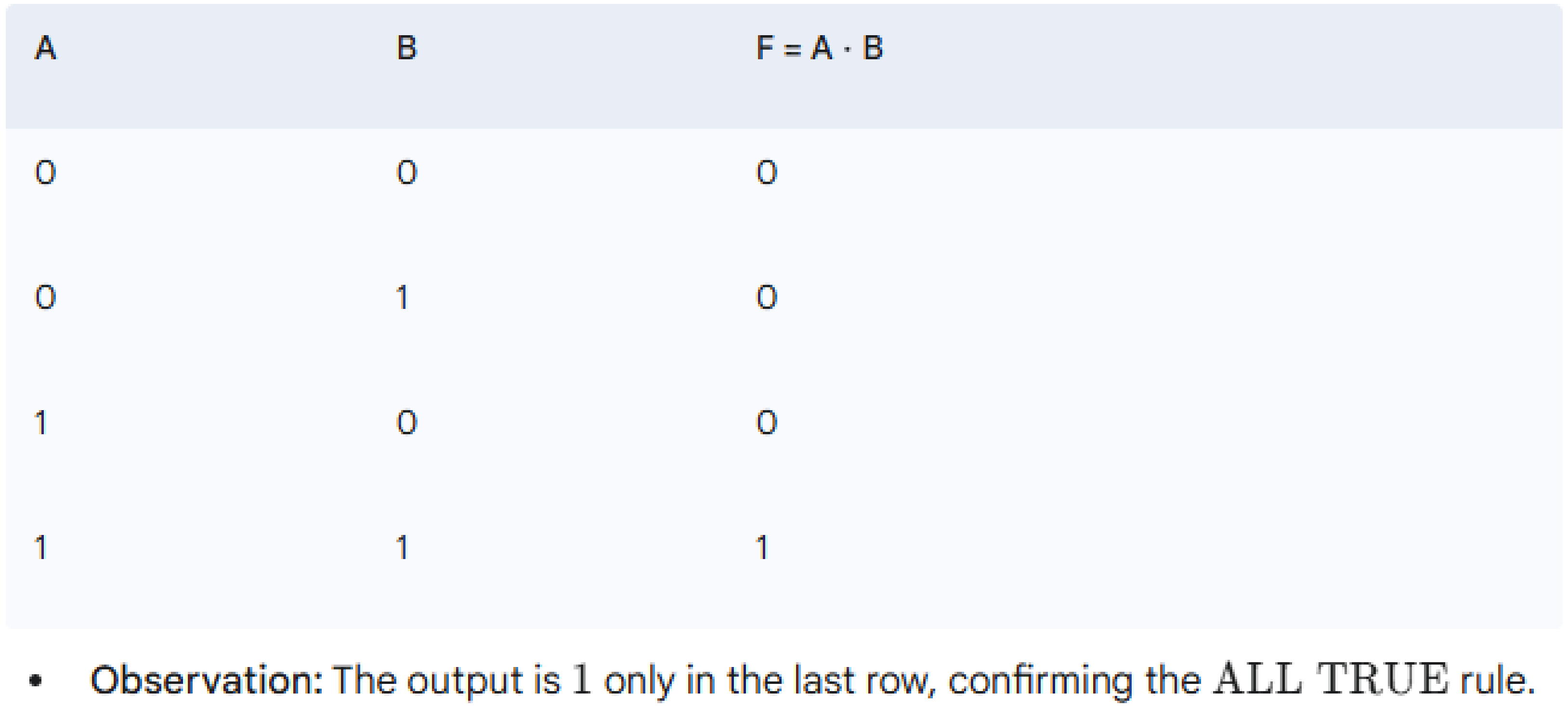
- The Three Fundamental Operations: All digital logic is built upon three basic operations:
- AND (Logical Multiplication)
- OR (Logical Addition)
- NOT (Logical Complement/Inversion)

The AND Operation (Logical Multiplication)

- **Symbol:** Represented by a dot (·), which is often omitted (e.g., A·B or AB).
- **Rule:** The result is 1 (True) **only if ALL** inputs are 1.
- **Analogy:** Two switches connected in **series**. Both must be ON for the light to turn on.
- **Function:** F=A·B

Truth Table for AND (Two Inputs)

- Truth Table: A list that shows all possible input combinations and the resulting output for a given operation.



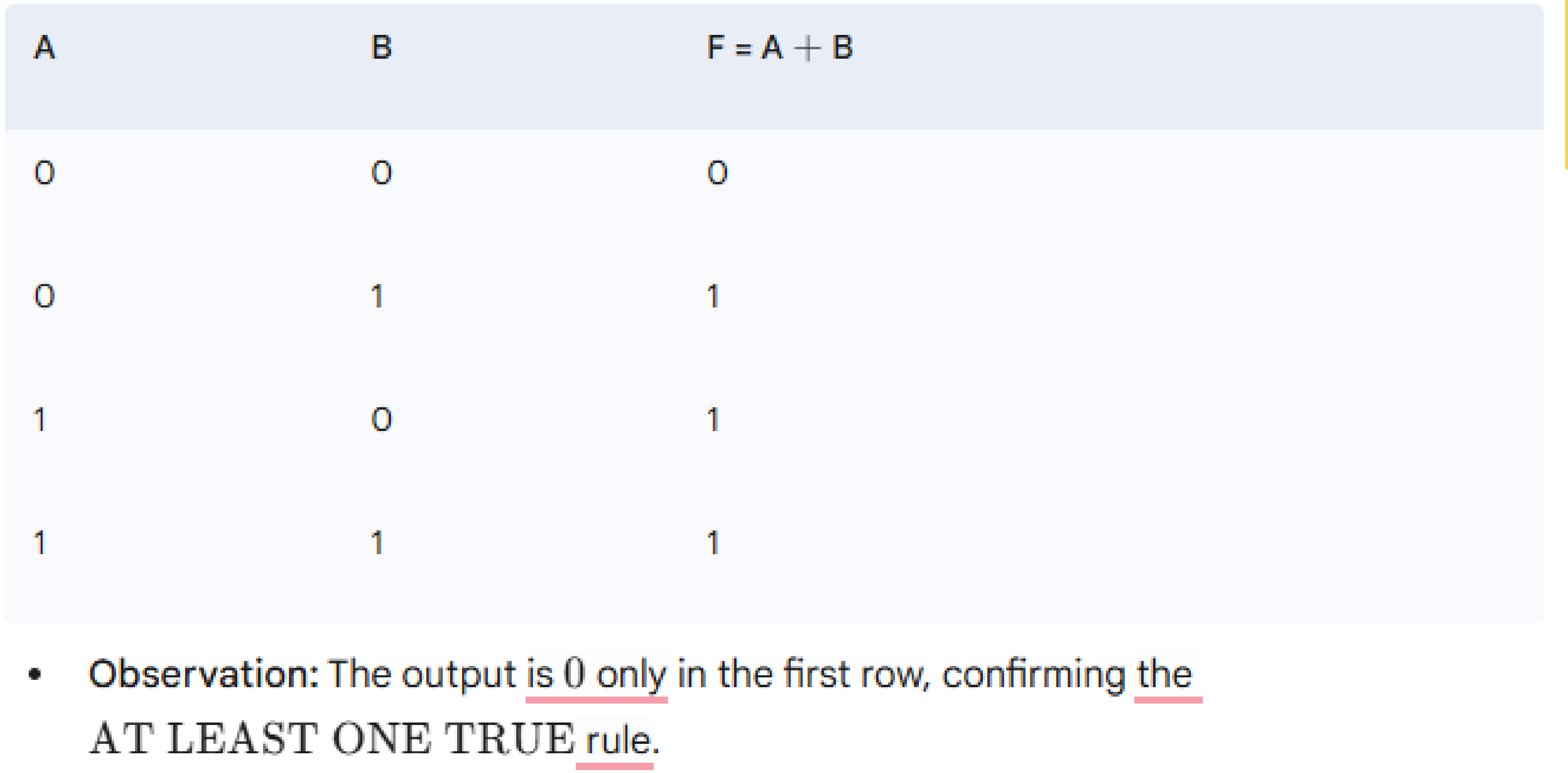
A	B	F = A · B
0	0	0
0	1	0
1	0	0
1	1	1

- **Observation:** The output is 1 only in the last row, confirming the $\mathrm{ALL\ TRUE}$ rule.

The OR Operation (Logical Addition)

- **Symbol:** Represented by a plus sign (+).
- **Rule:** The result is 1 (True) **if AT LEAST ONE** input is 1.
- **Analogy:** Two switches connected in **parallel**. Either one ON is enough to turn the light on.
- **Function:** F=A+B

Truth Table for OR (Two Inputs)



A	B	$F = A + B$
0	0	0
0	1	1
1	0	1
1	1	1

- Observation: The output is 0 only in the first row, confirming the AT LEAST ONE TRUE rule.

The NOT Operation (Complement/Inverter)

- **Symbol:** Represented by a bar over the variable (e.g., $\bar{A}$) or a prime (A').
- **Rule:** The result is the **inverse** (or complement) of the input.
- **Function:** $F=\bar{A}$

Truth Table for NOT (Single Input)

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

- Key Idea: This operation is essential for creating logic circuits that perform comparison or selection.

Basic Boolean Postulates: Idempotent Law

- **Idempotency** means performing an operation multiple times does not change the result.
- **OR Version:**
- $A+A=A$
- **AND Version:**
- $A \cdot A=A$
- **Proof Intuition:** 1+1=1 (in logical terms, not arithmetic).

Properties: Commutative Law

- **Definition:** The order of the variables does not affect the result.
- **OR Version:**
- $A+B=B+A$
- **AND Version:**
- $A \cdot B=B \cdot A$
- **Importance:** Allows flexibility in wiring and component placement in circuits.

Properties: Associative Law for OR

- **Definition:** When performing the same operation multiple times, the grouping of variables does not affect the result.
- **OR Version:**
- $A+(B+C)=(A+B)+C$
- **Benefit:** You can simplify complex OR chains without worrying about the order of operations.

Properties: Associative Law for AND

- **Definition:** The grouping of variables does not affect the result.
- **AND Version:**
- $A \cdot (B \cdot C) = (A \cdot B) \cdot C$
- **Benefit:** Crucial for designing multi-input AND gates.

Properties: Distributive Law

- **Definition:** Allows for the expansion and factoring of Boolean expressions. It links the AND and OR operations.
- **The Key Form:**
- $A \cdot (B+C) = (A \cdot B) + (A \cdot C)$
- **Analogy:** Similar to factoring in regular algebra.

Distributive Law: The Second Form

- **Unusual Form:** Unlike regular algebra, Boolean algebra has a second distributive property:
- $A+(B\cdot C)=(A+B)\cdot(A+C)$
- **Importance:** Used heavily in expressing functions in the Product-of-Sums (POS) form.

Boolean Theorems: Double Negation

- **Theorem:** Complementing a variable twice returns the original variable.
- **Rule:**
- $\bar{A}=A$
- **Analogy:** Two NOT gates in series cancel each other out.

Boolean Theorems: Absorption Law (Form 1)

- **Definition:** A rule that allows for quick simplification by eliminating redundant terms.
- **Rule:**
- $A+A\cdot B=A$
- **Proof Insight:** Since A is a term in both A and A·B, A already guarantees the output is A, making A·B redundant.

Boolean Theorems: Absorption Law (Form 2)

- **The AND Version:**
- $A \cdot (A+B) = A$
- **Proof Insight:** If A is 0, the result is 0. If A is 1, the expression becomes $1 \cdot (1+B) = 1 \cdot 1 = 1$. The result is always equal to A.

De Morgan's Theorem: Introduction

- **Significance:** One of the most important theorems in digital electronics.
- **Function:** It shows how to replace a negated sum or a negated product with a combination of individual complements.
- **Key Concept:** Breaking a long bar (negation over a group) into smaller bars and changing the operator.

De Morgan's Theorem I (The AND Break)

- **Statement:** The complement of a product is equal to the sum of the complements.
- **Rule:**
- $A\cdot B=\bar{A}+\bar{B}$
- **In English:** NOT (A AND B) is the same as (NOT A) OR (NOT B).

De Morgan's Theorem II (The OR Break)

- **Statement:** The complement of a sum is equal to the product of the complements.
- **Rule:**
- $A+B=\bar{A}\cdot\bar{B}$
- **In English:** NOT (A OR B) is the same as (NOT A) AND (NOT B).

Using Boolean Algebra for Simplification

- **Goal of Simplification:** To reduce the total number of terms and variables in an expression.
- **Why Simplify?**
- Fewer logic gates are needed.
- This reduces cost, power consumption, and physical size of the circuit.
- Increases circuit speed and reliability.

Simplification Example (Using Distributive & Complement)

- **Expression:** $F = A \cdot B + A \cdot \bar{B}$

- **Step 1 (Distributive):** Factor out A.
- $F = A \cdot (B + \bar{B})$
- **Step 2 (Complement Law):** $B + \bar{B} = 1$.
- $F = A \cdot 1$
- **Step 3 (Identity Law):**.
- **Simplified:** $F = A$

Introduction to Logic Gates

- **What are Gates?** The electronic circuits that physically implement the Boolean algebraic functions (AND, OR, NOT).
- **Construction:** Built using electronic components like transistors, diodes, and resistors.
- **Input/Output:** Gates accept one or more input voltages (0 or 1) and produce a single output voltage (0 or 1).

Classification of Logic Gates

- **Basic Gates:** AND, OR, NOT. (The building blocks)
- **Universal Gates:** NAND, NOR. (Can be used to build any other gate)
- **Special Purpose Gates:** XOR, XNOR. (Used for specific operations like comparison)

The AND Gate (Basic Gate)

- **Function:** Performs the logical multiplication operation.
- **Output:** High (1) only if all inputs are High (1). Otherwise, the output is Low (0).
- **Boolean Expression:** $F=A\cdot B$ (for two inputs)

The OR Gate (Basic Gate)

- **Function:** Performs the logical addition operation.
- **Output:** High (1) if any one or more inputs are High (1).
- **Boolean Expression:** $F=A+B$ (for two inputs)

The NOT Gate (Inverter) (Basic Gate)

- **Function:** Performs the complement or inversion operation.
- **Inputs:** Always has only one input.
- **Output:** The output is the opposite of the input. If input is 1, output is 0, and vice versa.
- **Boolean Expression:** F=A¯

The NAND Gate (Universal Gate)

- **Definition:** A NOT-AND gate. It is the inversion of the AND operation.
- **Output Rule:** The output is 0 only when ALL inputs are 1.
- **Boolean Expression:** $F=A\cdot B$

The NOR Gate (Universal Gate)

- **Definition:** A NOT-OR gate. It is the inversion of the OR operation.
- **Output Rule:** The output is 1 only when ALL inputs are 0.
- **Boolean Expression:** $F=A+B$

The Concept of Universal Gates

- **Significance:** The NAND and NOR gates are called **Universal Gates**.
- **Why?** Because any other logic function or gate (AND, OR, NOT, XOR, XNOR) can be created by using only NAND gates or only NOR gates.
- **Advantage:** Simplifies manufacturing and inventory by allowing circuits to be built using just one type of gate.

The Exclusive-OR (XOR) Gate

- **Function:** Performs a special type of logical comparison.
- **Output Rule:** The output is 1 if the inputs are **DIFFERENT** (01 or 10). The output is 0 if the inputs are the **SAME** (00 or 11).
- **Boolean Expression:** $F=A\oplus B=\bar{A}B+A\bar{B}$

- **Application:** Used primarily in digital adders and error detection circuits (parity checking).

The Exclusive-NOR (XNOR) Gate

- **Definition:** The inverse of the XOR gate (NOT-XOR).
- **Output Rule:** The output is 1 if the inputs are the **SAME** (00 or 11). The output is 0 if the inputs are DIFFERENT.
- **Boolean Expression:** $F=A\oplus B=\bar{A}\bar{B}+AB$

- **Application:** Used as a digital **Equality Detector**.

Summary: From Math to Machine

- **Boolean Algebra:** Provides the mathematical rules for handling binary logic.
- **Postulates & Theorems:** Allow engineers to simplify and optimize complex logical expressions.
- **Logic Gates:** The physical, electronic circuits that execute these algebraic functions.
- **The Engineering Link:** By using simplified Boolean expressions, engineers can minimize the number of gates, leading to smaller, faster, and more efficient processors.

Thank you

Shamna Subaida Khalid
shamnaplpy@gmail.com