Logic Gates & Boolean Algebra

The building blocks behind every digital circuit — and how to reason about them on paper

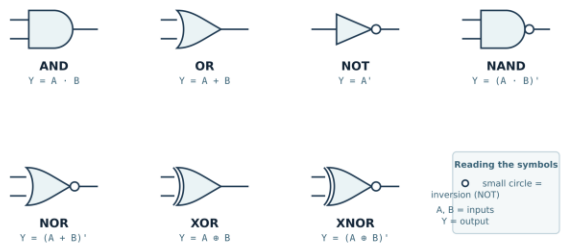
.1. Boolean Algebra & Its Laws

Boolean algebra deals with variables that can only be 1 (true) or 0 (false). Every digital circuit — from a calculator to a laptop's processor — reduces its decisions to AND, OR, and NOT operations on 1s and 0s. The laws below let you simplify any such expression before wiring it into a circuit: fewer terms means fewer gates, which means smaller and cheaper hardware.

Law	AND form	OR form
Identity	`A · 1 = A`	`A + 0 = A`
Null	`A · 0 = 0`	`A + 1 = 1`
Idempotent	`A · A = A`	`A + A = A`
Complement	`A · A′ = 0`	`A + A′ = 1`
Commutative	`A · B = B · A`	`A + B = B + A`
Distributive	`A·(B+C) = A·B+A·C`	`A+(B·C) = (A+B)·(A+C)`
Absorption	`A·(A+B) = A`	`A+(A·B) = A`
De Morgan's	`(A·B)′ = A′ + B′`	`(A+B)′ = A′ · B′`

2. Logic Gates — The Physical Form

Each Boolean operator has a matching gate symbol. Notice the pattern: NAND is AND with a bubble on the output, and NOR is OR with a bubble — that bubble always means "invert this signal."



3. Truth Tables & De Morgan's Theorem

A truth table lists every input combination and the resulting output. For two inputs there are always four rows (2^2):

A	B	AND (A·B)	OR (A+B)	XOR (A⊕B)
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

De Morgan's laws let you push a NOT through a bracket by flipping the operator inside: $(A \cdot B)' = A' + B'$, and $(A+B)' = A' \cdot B'$. In short — break the bracket, flip AND to OR (or OR to AND).

Dr. N.G.Thangamma
Assistant Professor, PG Department of Computer Science
St.Joseph’s College (Arts & Science)