Compuertas Universales (NAND)

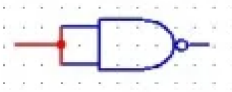
Compuerta NAND = compuerta universal

La compuerta **NAND puede construir cualquier otra compuerta lógica.**

Es suficiente por sí sola para implementar **toda lógica booleana**.

Su función básica es:

NAND(A,B) = ¬(A . B)



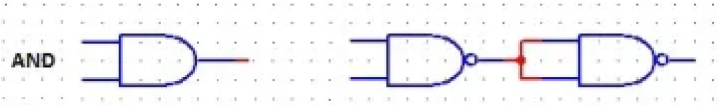
Cómo formar otras compuertas usando solo NAND

NOT con NAND
Usas la misma entrada dos veces.

¬A = NAND(A, A)

AND con NAND
Primero haces NAND, luego lo niegas con un NOT hecho con NAND.

A . B = ¬(NAND(A, B))



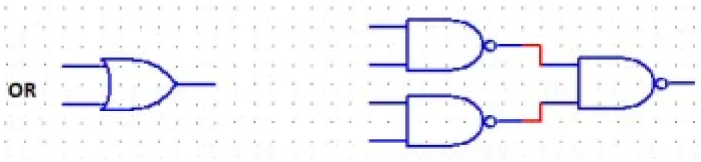
Implementación:

X = NAND(A, B)
AND = NAND(X, X)

OR con NAND

Usa las negaciones de A y B, luego aplica NAND.

A + B = NAND(A, B)



Implementación NAND-solo:

A1 = NAND(A, A)
B1 = NAND(B, B)
OR = NAND(A1, B1)

Una sola compuerta puede implementar **todo**:

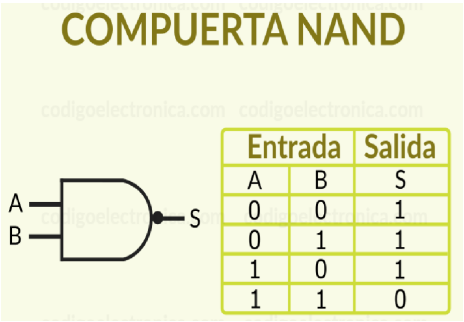
expresiones booleanas,
simplificaciones,
mapas de Karnaugh,
circuitos digitales completos.

Por eso se llaman **universales**.

Funcionamiento básico de la compuerta NAND

La compuerta NAND entrega:

0 solo cuando **ambas entradas son 1**.
En todos los demás casos, entrega **1**.



¿Por qué la NAND es universal?

Porque cumple dos requisitos fundamentales:

1. Puede generar la negación (NOT).

¬A = NAND(A, A)

2. Puede generar las operaciones básicas AND y OR.

Y, una vez que tienes AND, OR y NOT, cualquier circuito digital puede construirse.

Toda la lógica booleana se basa en combinaciones de esas operaciones.

Ventajas de usar puertas NAND

Simplicidad física: en electrónica real, una sola tecnología repetida reduce costos.

Estandarización: muchas familias lógicas (TTL, CMOS) optimizan la velocidad y consumo usando NAND internas.

Implementación de cualquier función: mapas de Karnaugh, expresiones booleanas, sistemas combinacionales y secuenciales.

En la práctica, gran parte de los circuitos integrados están hechos internamente con matrices de puertas NAND.

Relación con mapas de Karnaugh y simplificación booleana

Aunque los mapas muestran expresiones en términos de AND, OR y NOT,
el hardware final casi siempre se implementa en **NAND** porque:

unifica la tecnología,
reduce la cantidad de transistores,
es más rápido,
permite implementar expresiones simplificadas directamente.

Ejemplo:
Si el mapa te da:

F = A . B + ¬C

Puedes transformarlo a NAND de forma inmediata:

NOT C → NAND(C,C)
AND (A,B) → NAND(NAND(A,B), NAND(A,B))
OR → NAND(NOT(AND), NOT(NOT(C)))

Y obtienes un circuito completamente implementado.