Álgebra de Boole y Puertas Lógicas

Electrónica Digital — 2º GIERM (Mecatrónica) Apuntes propios · Tema 2: Teoría de la conmutación

> Notebook autocontenido. Solo requiere `numpy`, `matplotlib` y `schemdraw`.
> Filosofía *worked-example*: primero el resultado/ejemplo resuelto, luego la teoría que lo justifica.

Índice

Configuración del entorno

Importamos lo necesario. `schemdraw.logic` contiene las puertas (`And`, `Or`, `Not`, `Nand`, `Nor`, `Xor`, `Xnor`, `Buf`). Definimos también una función auxiliar para imprimir **tablas de verdad** con `numpy`.

```
Entorno listo. schemdraw 0.22
```

1. Idea de partida: del razonamiento al circuito

Queremos circuitos que **repliquen razonamientos lógicos**:

- hacer algo si se cumple *una* condición,
- si se cumplen *dos a la vez*,
- si se cumple *una u otra*,
- si se cumple *una pero no otra*, etc.

Ejemplo motivador (del temario): una puerta debe abrirse *siempre* pulsando un botón interior, y *también* si se pulsa el botón exterior *siempre que no sea de noche*.

Señales (todas digitales, valen 0 ó 1):

- `a` = botón interior
- `c` = botón exterior
- `b` = es de noche
- `f` = orden de abrir (1 = abrir)

En lenguaje lógico: *abrir si (interior)* ***O*** *(exterior* ***Y NO*** *noche)*:

$$f = a + c \cdot \overline{b}$$

Lo resolveremos al final como ejemplo integrador. Primero necesitamos el formalismo.

2. Álgebra de Boole

Formalización matemática del razonamiento lógico (G. Boole, *An Investigation of the Laws of Thought*, 1854). Cada proposición es **V o F**, que escribimos **1 ó 0**.

Un **álgebra de Boole** es un conjunto B con dos operaciones, suma lógica `+` y producto lógico `·`, que cumple los **axiomas**. La suma se lee *OR*, el producto *AND* y la complementación $\overline{x}$ es *NOT*.

Axiomas (definición del álgebra)

#	Axioma	Operación `+`	Operación `·`
1	Cierre	$a + b \in B$	$a \cdot b \in B$
2	Elemento identidad	$a + 0 = a$	$a \cdot 1 = a$
3	Conmutatividad	$a + b = b + a$	$a \cdot b = b \cdot a$
4	Distributividad	$a + (b \cdot c) = (a + b) \cdot (a + c)$	$a \cdot (b + c)$ $= (a \cdot b)$ $+ (a \cdot c)$
5	Elemento complementario	$a + \overline{a} = 1$	$a \cdot \overline{a} = 0$
6	Cardinalidad	hay al menos **dos** elementos distintos en B $(0 \neq 1)$	

> Ojo a la **doble distributividad**: en álgebra de Boole, a diferencia de la aritmética ordinaria, *la suma también distribuye sobre el producto*.

Álgebra de Boole **bivalente**

El caso que usaremos siempre: solo dos valores, $B = \{0, 1\}$. Las operaciones se definen por tabla:

```
OR (suma +)     AND (producto .)    NOT
a b | a+b       a b | a.b           a | ~a
0 0 |  0         0 0 |  0           0 |  1
0 1 |  1         0 1 |  0
1 0 |  1         1 0 |  0           1 |  0
1 1 |  1         1 1 |  1
```

Comprobamos numéricamente algunos axiomas (un axioma es verdadero si se cumple para *todas* las combinaciones de entrada):

```
Axioma 5  a+~a=1 : True
Axioma 5  a.~a=0 : True
Axioma 4  + distribuye sobre . : True
```

3. Teoremas del álgebra de Boole

Se deducen de los axiomas. Todos vienen en **parejas duales** (intercambiar `+` ↔ `·` y `0` ↔ `1` da el otro teorema válido):

#	Teorema	Forma con `+`	Forma con `·`
1	Dualidad	(principio general)	—
2	Elemento dominante	$a + 1 = 1$	$a \cdot 0 = 0$
3	Idempotencia	$a + a = a$	$a \cdot a = a$
4	Involución	$\overline{\overline{a}} = a$	—
5	Absorción	$a + a \cdot b = a$	$a \cdot (a + b) = a$
6	Asociatividad	$a + (b + c) = (a + b) + c$	$a \cdot (b \cdot c) = (a \cdot b) \cdot c$
7	**De Morgan**	$\overline{a + b} = \overline{a} \cdot \overline{b}$	$\overline{a \cdot b} = \overline{a} + \overline{b}$

Leyes de De Morgan (las más importantes para examen)

$$\boxed{\overline{a + b} = \overline{a} \cdot \overline{b} \qquad \overline{a \cdot b} = \overline{a} + \overline{b}}$$

Regla mnemotécnica: **"se rompe la barra y se cambia el signo"**. Negar una suma = producto de negados; negar un producto = suma de negados. Es la base para convertir cualquier función a solo NAND o solo NOR.

```
De Morgan  ~(a+b) = ~a.~b : True
De Morgan  ~(a.b) = ~a+~b : True
Absorcion  a + a.b = a    : True
Identidad  A + ~A.B = A+B : True
```

4. Variables, funciones y tablas de verdad

Un **circuito digital combinacional** es equivalente a una **función booleana** $F(x_1, \ldots, x_n)$. Es *direccional*: tiene entradas y salidas no intercambiables.

Formas de representar una función

Representación	¿Única?
Expresión algebraica (p.ej. $F = \overline{a}b + ac$)	No
Tabla de verdad	**Sí**
Esquemático con puertas	No
Lenguaje de descripción hardware (VHDL/Verilog)	No

La **tabla de verdad** lista la salida para las 2^n combinaciones de entrada y es la representación canónica única.

Definiciones clave (formas canónicas)

- **Literal**: una variable o su complementaria (a ó $\overline{a}$).
- **Producto canónico**: producto que contiene *todas* las variables (cada una una vez, normal o negada).
- **Mínterm** m_i: producto canónico; vale 1 para **una sola** combinación de entrada.
- **Máxterm** M_i: suma canónica; vale 0 para **una sola** combinación de entrada.
- Relación: $M_i = \overline{m_i}$. Hay el mismo número de mínterms que de máxterms (2^n).

Teorema de la expansión de Shannon

Permite descomponer cualquier función respecto a una variable:

$$F(x, y, z, \ldots) = x \cdot F(1, y, z, \ldots) + \overline{x} \cdot F(0, y, z, \ldots)$$

Aplicándolo recursivamente a todas las variables se llega a la **suma de mínterms** (1ª forma canónica, SOP).

Formas canónicas

$$F = \sum m_i \quad (\text{índices donde } F = 1) \qquad F = \prod M_i \quad (\text{índices donde } F = 0)$$

Ejemplo con tabla de verdad: sea $F = \overline{a}b + ac$. Veamos su tabla y sus mínterms.

```
a | b | c || F
--------------
0 | 0 | 0 ||  0
0 | 0 | 1 ||  0
0 | 1 | 0 ||  1
0 | 1 | 1 ||  1
1 | 0 | 0 ||  0
1 | 0 | 1 ||  1
1 | 1 | 0 ||  0
1 | 1 | 1 ||  1
\nMinterms (F=1):  F = sum m [2, 3, 5, 7]
Maxterms (F=0):  F = prod M [0, 1, 4, 6]
```

5. Puertas lógicas: símbolo, ecuación y tabla

Las **puertas lógicas** son circuitos (a base de transistores) que implementan funciones lógicas. Un **círculo (bolita)** en una entrada o salida indica que la señal está **invertida**.

Dibujamos los símbolos normalizados con `schemdraw.logic`.

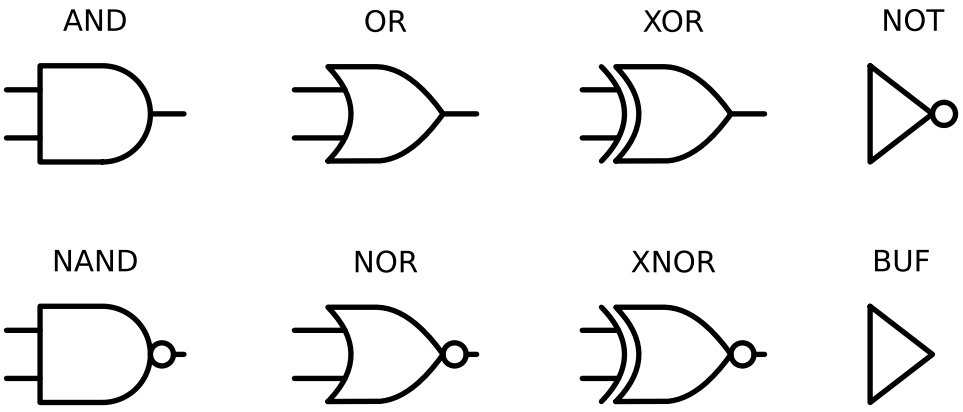


Tabla resumen: ecuación y comportamiento (2 entradas)

Puerta	Ecuación	Salida = 1 cuando…
AND	$F = a \cdot b$	*todas* las entradas son 1
OR	$F = a + b$	*alguna* entrada es 1
NOT	$F = \overline{a}$	la entrada

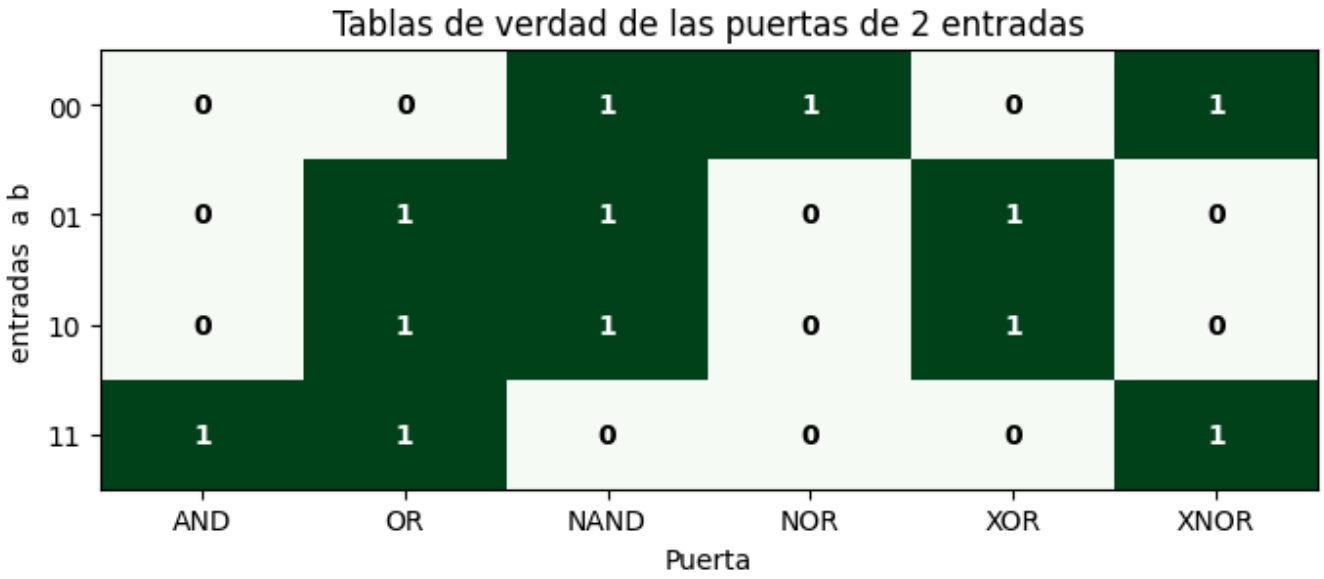
Puerta	Ecuación	Salida = 1 cuando...
		es 0
NAND	$F = \overline{a \cdot b}$	*alguna* entrada es 0 (= NOT de AND)
NOR	$F = \overline{a + b}$	*todas* las entradas son 0 (= NOT de OR)
XOR	$F = a \oplus b = \bar{a}b + a\bar{b}$	nº *impar* de unos (entradas distintas)
XNOR	$F = \overline{a \oplus b}$	nº *par* de unos (entradas iguales)

Generamos **todas las tablas de verdad** programáticamente:

```
=== AND ===
a | b || AND
------------
0 | 0 ||  0
0 | 1 ||  0
1 | 0 ||  0
1 | 1 ||  1

=== OR ===
a | b || OR
-----------
0 | 0 ||  0
0 | 1 ||  1
1 | 0 ||  1
1 | 1 ||  1

=== NAND ===
a | b || NAND
-------------
0 | 0 ||  1
0 | 1 ||  1
1 | 0 ||  1
1 | 1 ||  0

=== NOR ===
a | b || NOR
------------
0 | 0 ||  1
0 | 1 ||  0
1 | 0 ||  0
1 | 1 ||  0

=== XOR ===
a | b || XOR
------------
0 | 0 ||  0
0 | 1 ||  1
1 | 0 ||  1
1 | 1 ||  0

=== XNOR ===
a | b || XNOR
-------------
0 | 0 ||  1
0 | 1 ||  0
1 | 0 ||  0
1 | 1 ||  1

=== NOT (1 entrada) ===
a || NOT
--------
0 ||  1
1 ||  0
```

Visualización conjunta de las tablas con matplotlib

Una "matriz de salidas": filas = combinación de entradas `ab`, columnas = puerta. Verde = 1, claro = 0.



6. Generalización a *n* entradas

Puerta	Salida 0 si...	Salida 1 si...
AND	alguna entrada es 0	todas las entradas son 1
OR	todas las entradas son 0	alguna entrada es 1
NAND	todas las entradas son 1	alguna entrada es 0
NOR	alguna entrada es 1	todas las entradas son 0
XOR	nº *par* de entradas a 1	nº *impar* de entradas a 1
XNOR	nº *impar* de entradas a 1	nº *par* de entradas a 1

Es decir, XOR de *n* entradas = **paridad impar**. Lo comprobamos para 4 entradas:

```
XOR de 4 entradas (=1 si nº impar de unos):
0 0 0 0 -> XOR = 0   (nº de unos = 0 -> par )
0 0 0 1 -> XOR = 1   (nº de unos = 1 -> impar )
0 0 1 0 -> XOR = 1   (nº de unos = 1 -> impar )
0 0 1 1 -> XOR = 0   (nº de unos = 2 -> par )
0 1 0 0 -> XOR = 1   (nº de unos = 1 -> impar )
0 1 0 1 -> XOR = 0   (nº de unos = 2 -> par )
0 1 1 0 -> XOR = 0   (nº de unos = 2 -> par )
0 1 1 1 -> XOR = 1   (nº de unos = 3 -> impar )
1 0 0 0 -> XOR = 1   (nº de unos = 1 -> impar )
1 0 0 1 -> XOR = 0   (nº de unos = 2 -> par )
1 0 1 0 -> XOR = 0   (nº de unos = 2 -> par )
1 0 1 1 -> XOR = 1   (nº de unos = 3 -> impar )
1 1 0 0 -> XOR = 0   (nº de unos = 2 -> par )
1 1 0 1 -> XOR = 1   (nº de unos = 3 -> impar )
1 1 1 0 -> XOR = 1   (nº de unos = 3 -> impar )
1 1 1 1 -> XOR = 0   (nº de unos = 4 -> par )
```

7. Universalidad de NAND y NOR (conjuntos completos)

> **Todas** las funciones lógicas se pueden construir usando **solo NAND** o **solo NOR**. Por eso NAND y NOR se llaman **conjuntos completos** (o puertas universales). También son completos `{AND, NOT}` y `{OR, NOT}`, **pero no** solo AND ni solo OR.

La idea: con NAND (o NOR) se pueden fabricar NOT, AND y OR, y con esos tres se construye cualquier función desde su forma canónica SOP.

NOT, AND y OR usando solo NAND

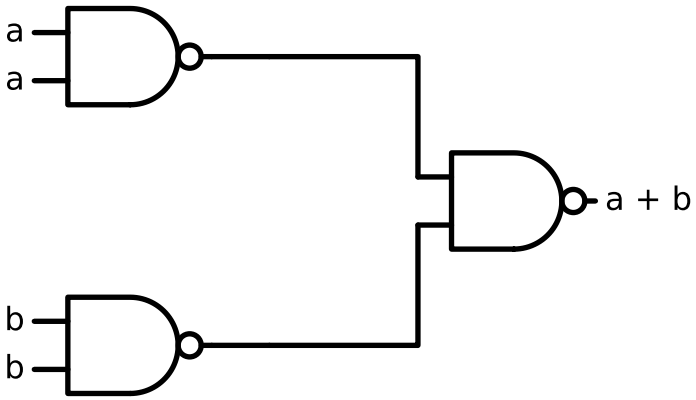
Quiero	Con NAND
NOT a	$\overline{a} = \overline{a \cdot a} = \text{NAND}(a, a)$
AND a,b	$a \cdot b = \overline{\overline{a \cdot b}} = \text{NAND}(\text{NAND}(a, b), \text{NAND}(a, b))$
OR a,b	$a + b = \overline{\overline{a} \cdot \overline{b}} = \text{NAND}(\overline{a}, \overline{b})$ (De Morgan)

Lo verificamos en código:

```
NOT solo con NAND : True
AND solo con NAND : True
OR  solo con NAND : True
=> NAND es un conjunto completo: True
```

Esquemático: OR construida solo con NAND (tres NAND).

- Dos NAND como inversores: $\overline{a}$ y $\overline{b}$.
- Una NAND final: $\mathrm{NAND}(\overline{a}, \overline{b}) = \overline{\overline{a} \cdot \overline{b}} = a + b$.



8. Ejemplo integrador resuelto (el de la puerta)

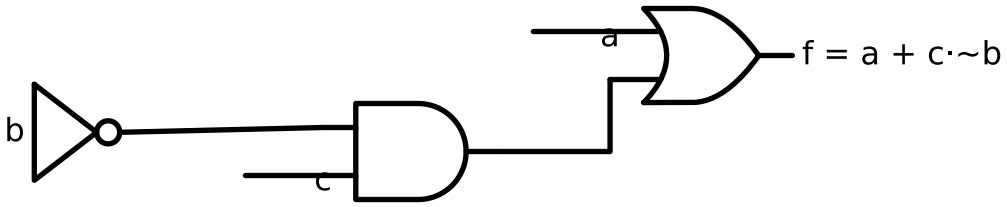
Retomamos el problema inicial: abrir si *(botón interior* `a` *)* **O** *(botón exterior* `c` ***Y NO*** *noche* `b` *)*:

$$f = a + c \cdot \overline{b}$$

Paso 1 — Tabla de verdad.

```
a = boton interior, b = es de noche, c = boton exterior, f = abrir
a | b | c || f
--------------
0 | 0 | 0 ||  0
0 | 0 | 1 ||  1
0 | 1 | 0 ||  0
0 | 1 | 1 ||  0
1 | 0 | 0 ||  1
1 | 0 | 1 ||  1
1 | 1 | 0 ||  1
1 | 1 | 1 ||  1
```

Paso 2 — Esquemático con puertas. Sustituimos cada operación por su puerta: un NOT para $\overline{b}$, un AND para $c \cdot \overline{b}$, y un OR final con a.



Paso 3 — Mínterms (1ª forma canónica, SOP). Los índices donde $f = 1$ (orden de variables `a b c`):

```
f = sum m [1, 4, 5, 6, 7]
  m1: ~a~bc
  m4: a~b~c
  m5: a~bc
  m6: ab~c
  m7: abc
```

9. Chuleta de examen

Axiomas clave

- Doble distributividad (¡también `+` sobre `·` !): $a + bc = (a + b)(a + c)$
- Complemento: $a + \overline{a} = 1,\ \ a \cdot \overline{a} = 0$

Teoremas que más caen

- Dominante: $a + 1 = 1,\ \ a \cdot 0 = 0$
- Idempotencia: $a + a = a,\ \ a \cdot a = a$
- Involución: $\overline{\overline{a}} = a$
- Absorción: $a + ab = a,\ \ a(a + b) = a$
- Identidades: $a + \overline{a}b = a + b,\ \ a(\overline{a} + b) = ab$
- **De Morgan**: $\overline{a + b} = \overline{a}\,\overline{b},\ \ \overline{a \cdot b} = \overline{a} + \overline{b}$ ("rompe la barra, cambia el signo")

Puertas (= cuándo vale 1)

- AND: todas 1 · OR: alguna 1 · NAND: alguna 0 · NOR: todas 0 · XOR: nº impar de 1 · XNOR: nº par de 1

Formas canónicas

- SOP (1ª): $F = \sum m_i$ → mínterms = filas con $F = 1$
- POS (2ª): $F = \prod M_i$ → máxterms = filas con $F = 0$
- $M_i = \overline{m_i}$; hay 2^n de cada

Universalidad

- NAND y NOR son **conjuntos completos** (universales) por sí solos.

- $\{AND, NOT\}$ y $\{OR, NOT\}$ también; **solo AND** o **solo OR** *no*.
- Para pasar a solo-NAND: aplicar De Morgan + doble negación.

Apuntes propios — Electrónica Digital · Álgebra de Boole y puertas lógicas. Generado y verificado ejecutando el notebook.