# Puertas lógicas. Técnicas de diseño y simplificación de funciones lógicas.

## Introducción

La electrónica digital está basada en una teoría binaria cuya estructura matemática fue desarrollada por George Boole en el siglo XIX.

En esta estructura se manejan variables y funciones que solo pueden tomar dos valores: verdadero o falso. La asociación de estas funciones, por tanto, sólo tendrá dos posibles estados de salida. Asociando estos valores a los dos estado lógicos de la electrónica digital, 1 (verdadero) y 0 (falso), podemos aplicar esta teoría a la asociación de circuitos digitales.

Los sistemas físicos que funcionan según estos principios, reciben el nombre de circuitos lógicos o puertas lógicas.

## Álgebra de Boole

Las operaciones básicas son:

- **Suma lógica o unión** → Solamente es cero si todas las variables son cero. S = A + B

**"OR"**

| A | B | S = A+B |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

- **Producto lógico o intersección** → Es uno si todas las variables son uno. S = AB

**"AND"**

| A | B | S = AB |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

- **Complementación o negación lógica** → Hay dos estados posibles (0 y 1), y cada uno es el complemento de otro, por tanto, complementar una variable equivale a cambiarla de estado.

| **A** | **S = $\overline{A}$** |
|---|---|
| 0 | 1 |
| 1 | 0 |

"$\overline{A}$"

Como en toda estructura algebraica, existe una prioridad en las operaciones. La complementación es la operación prioritaria sobre las demás, el producto es el siguiente en grado de prioridad y, por último, la suma.

En un sistema de dos estados, al estado complementario se le denomina estado negado o variable negada.

Prioridad en las operaciones. ⟶ Complementación - producto - la suma.

## Propiedades del Álgebra de Boole

En general se puede afirmar que cumplen las propiedades de cualquier álgebra:

### Propiedad interna:

El resultado de una operación entre dos variables booleanas es otra variable booleana.

### Propiedad de idempotencia:

$A*A = A$

$A+A = A$

### Asociativa:

$(A+B)+C = A+(B+C)$
$(AB)C = A(BC)$

Conmutativa:

A+B = B+A
AB = BA

Distributiva I (respecto al producto):

A(B+C) = AB+AC

Distributiva II (respecto a la suma):

A+(BC) = (A+B)(A+C)

Elemento neutro:

A+0 = A
A*1 = A

Elemento inverso u opuesto:

A+A’ = 1

A*A’ =0

Ley de involución:

A’’ = A

Ley de absorción:

A+A*B = A

A*(A+B) = a

Teoremas de Morgan:

$(\overline{A+B}) = \overline{A}\ \overline{B}$

$\overline{AB} = \overline{A} + \overline{B}$

## Funciones lógicas

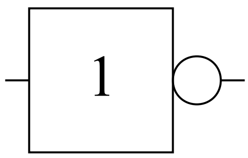
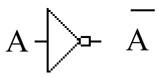
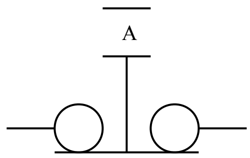
Una función lógica es una agrupación de variables relacionadas entre sí mediante las operaciones básicas (suma, producto y complementación).

Cualquiera de estas funciones queda perfectamente reflejada por su tabla de verdad. Esta no es más que una tabla en la que se dan las salidas que tiene la función para "todos" los valores de entrada posibles de las variables que componen esta función.

De todas las funciones posibles las más simples son las llamadas puertas lógicas.

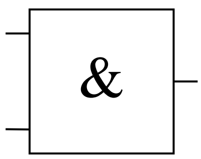
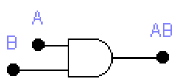
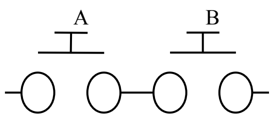
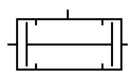
## Puertas lógicas

### Puerta NO, NOT o negadora

| **Representaciones** | | |
|---|---|---|
| DIN | ASA | Eléctrica |
|   |   |   |

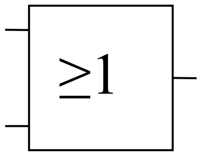
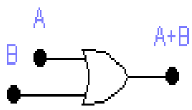
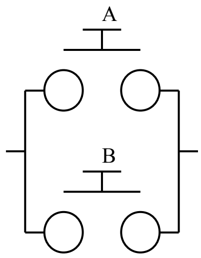
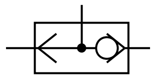
| $A$ | $\overline{A}$ |
|---|---|
| 0 | 1 |
| 1 | 0 |

### Puerta Y, AND o multiplicadora

| **Representaciones** | | | |
|---|---|---|---|
| DIN | ASA | Eléctrica | Neumo-hidráulica |
|   |   |   |  |

| A | B | AB |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

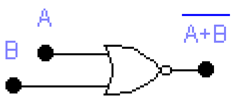
## Puerta O, OR o sumadora

| **Representaciones** | | | |
|---|---|---|---|
| DIN | ASA | Eléctrica | Neumo-hidráulica |
|  |  |  |  |

| A | B | A+B |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

## Puerta O Negada o NOR

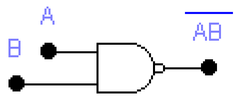
La menciono porque tiene una representación especial, pero no sería más que una OR seguida de un inversor.




Como se aprecia se le pone un circulito a la salida. Igual se haría con la representación DIN

## Puerta Y NEGADA o NAND

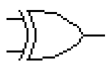
Es un caso análogo al anterior.




## Puerta O EXCLUSIVA o XOR

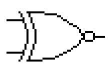
Es uno, cuando una de las dos entradas es uno, y cero, cuando las dos son cero o cuando las dos son uno.

$A \oplus B = \overline{A}\ B + A\ \overline{B}.$



| A | B | $A\overline{B}+\overline{A}B$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

## Puerta O-EXCLUSIVA NEGADA O XNOR



# Representación de funciones

## Mediante su forma algebraica

Esta constituida por las variables que la forman y su relación entre ellas. Por ejemplo: F(A,B,C) = A'B+B'A+A'C'. Existen dos formas particulares de la representación algebraica que son las dos formas canónicas:

- Como suma de productos en cada uno de los cuales están representadas todas las variables de la función. **Minitérminos o minterm**.
- Como productos de sumas en las que están representados todas las variables de la función. M**axitérminos o maxterm.**

## Numéricamente

Es una forma simplificada de representar las expresiones canónicas.

| $A$ | $B$ | $C$ | $F(A,B,C)$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |

Para representar una función canónica en suma de productos utilizaremos el símbolo $\Sigma_n$ (sigma) y en producto de sumas $\Pi_n$ (pi), donde n indicará el número de variables. Así, la representación numérica correspondiente a la tabla de verdad sería:

$$F = \Sigma_3(2, 4, 5, 6) = \Pi_3(0, 1, 3, 7)$$

Matemáticamente se demuestra, que para todo término i de una función, se cumple la siguiente ecuación:

$$F = [\Sigma_n( i )]' = \Pi_n(2^n\text{-}1\text{-} i )$$

A modo de ejemplo se puede utilizar esta igualdad para obtener el producto de sumas a partir de la suma de productos del ejemplo anterior:

$$F = \Sigma3(2, 4, 5, 6) = [\Sigma_3(2, 4, 5, 6)]'' = [\Sigma_3(0, 1, 3, 7)]' = \Pi_3(0, 4, 6, 7)$$

## Mediante su tabla de verdad

Se trata de dar los valores de salida para todas las posibles entradas mediante una tabla.

| A | B | C | F(A,B,C) |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 |

De esta tabla de verdad se puede extraer su representación algebraica en forma de minterm o maxterm.

## Como suma de productos o minterm

Se trata de coger las salidas que dan uno y considerar los unos como variables sin negar y los ceros como variables negadas. Por ejemplo:

| A | B | C | F(A,B,C) | |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 → | A’B’C’ |
| 0 | 0 | 1 | 0 | |
| 0 | 1 | 0 | 1 → | A’BC’ |
| 0 | 1 | 1 | 1 → | A’BC |
| 1 | 0 | 0 | 1 → | AB’C’ |
| 1 | 0 | 1 | 1 → | AB’C |
| 1 | 1 | 0 | 0 | |
| 1 | 1 | 1 | 0 | |

Así la función en su forma algebraica nos quedaría:

F(A,B,C)=A’B’C’+A’BC’+A’BC+AB’C

## Como productos de sumas o maxterm

Haciéndolo de esta forma debemos coger las salidas nulas y considerar los ceros como variables no negadas y los unos como variables negadas. Así por ejemplo:

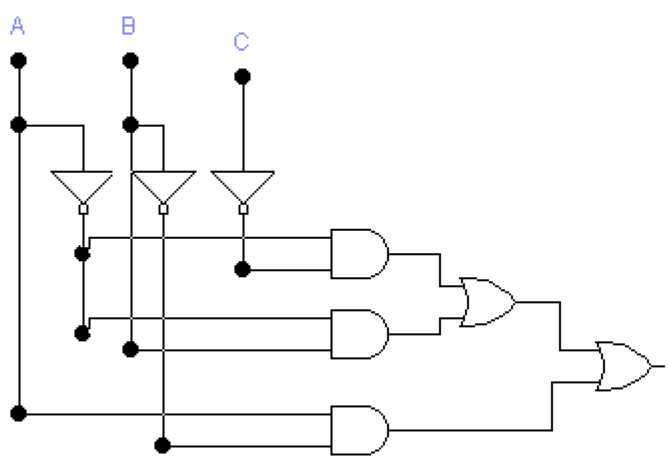
| A | B | C | F(A,B,C) | |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | |
| 0 | 0 | 1 | 0 | A+B+C’ |
| 0 | 1 | 0 | 1 | |
| 0 | 1 | 1 | 1 | |
| 1 | 0 | 0 | 1 | |
| 1 | 0 | 1 | 1 | |
| 1 | 1 | 0 | 0 | A’+B’+C |
| 1 | 1 | 1 | 0 | A’+B’+C’ |

Así nos quedaría la representación algebraica siguiente:

$$F(A,B,C)=(A+B+C')(A'+B'+C)(A'+B'+C')$$

## Representación gráfica

Se trata de implementar la función con la simbología de las puertas lógicas. Así la función anterior nos quedaría:



## Método analítico o algebraico

Este método consiste en la aplicación de las leyes y teoremas del álgebra de Boole para conseguir reducir la función lo máximo posible.

Para paliar la subjetividad de este método se suele emplear una rutina de simplificación basada en las propiedades de los términos adyacentes, es decir, aquellos que sólo difieren en el estado de una de sus variables. Esto nos permite que podamos eliminar ese término al sumar esos dos elementos. Así $ABCD+ABCD'=ABC(D+D')=ABC$

De esta manera podemos ir eliminando términos adyacentes hasta llegar a la expresión simplificada.

## Método de Karnaugh

Se trata de un método gráfico basado en la rutina simplificadora anterior.

Un mapa de Karnaugh es una tabla de verdad dispuesta de manera que la adyacencia quede claramente reflejada de modo que dos casillas contiguas representan miembros adyacentes, es decir, términos que solo se diferencian en el estado de complementación o no de una solo variable.

También se consideran contiguas las que se hallan en casillas que pueden superponerse por alguna línea de las que dividen el mapa de Karnaugh en casillas mediante un solo doblez.

Hagamos por ejemplo la simplificación de la función representada por la tabla de verdad siguiente:

| A | B | C | F(A,B,C) |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 |

El mapa de Karnaugh correspondiente a esta tabla de verdad sería:

| C \ AB | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 |

Los agrupamientos se hacen siguiendo el criterio siguiente:

1. Se toman todos los unos que no se pueden agrupar con ningún otro.
2. Se forman los grupos de dos unos que no pueden formar un grupo de cuatro.
3. Se forman los grupos de cuatro unos que no puedan formar un grupo de ocho
4. Cuando se hayan cubierto todos los unos finaliza el proceso.

Nota: El mismo uno puede formar parte de más de un grupo.

Siguiendo estos pasos obtendremos los grupos marcados en la tabla anterior. Del grupo horizontal se la izquierda tendremos A'C' (esto es porque a A le corresponde cero en las dos casillas y, puesto que la agrupación es de una sola fila, C solo tiene el estado cero, es decir, negado), del vertical de la izquierda saldría, análogamente A'B y del vertical de la derecha AB'. Luego en conjunto tendremos que la ecuación simplificada será: F(A,B,C)=A'C'+A'B+AB'

Esto sería trabajando en minterm.

En maxterm habría que coger los ceros pero se haría análogamente. Interpretando en esta caso que los ceros son las variables no negadas y los unos las variables negadas. Además nos quedaría como productos de sumas.

## Método de tabulación de Quine-McCluskey

Este método es susceptible de tratamiento por ordenador.

Pongamos un ejemplo: Supongamos que queremos simplificar la función: f(A,B,C,D) = A'BC'+A'BD+AB'C'+AB'D+A'BC

Lo primero es poner la función en su forma canónica. Para ello multiplicamos al término que le falte una variable por la suma de ella más su complementaria (esta suma siempre es uno y multiplicar por uno es como no hacer nada). Por ejemplo: si al termino le falta la B se multiplicaría ese término por B+B'.

Otra forma de hacer esto es poner la tabla de verdad y seleccionar los términos que contengan los productos de la función. Nosotros lo haremos de esta última forma para agilizar cálculos.

| A | B | C | D | F |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 (0) |
| 0 | 0 | 0 | 1 | 0 (1) |
| 0 | 0 | 1 | 0 | 0 (2) |
| 0 | 0 | 1 | 1 | 0 (3) |
| 0 | 1 | 0 | 0 | 1 (4) |
| 0 | 1 | 0 | 1 | 1 (5) |
| 0 | 1 | 1 | 0 | 1 (6) |
| 0 | 1 | 1 | 1 | 1 (7) |
| 1 | 0 | 0 | 0 | 1 (8) |
| 1 | 0 | 0 | 1 | 1 (9) |
| 1 | 0 | 1 | 0 | 0 (10) |
| 1 | 0 | 1 | 1 | 1 (11) |
| 1 | 1 | 0 | 0 | 0 (12) |
| 1 | 1 | 0 | 1 | 0 (13) |
| 1 | 1 | 1 | 0 | 0 (14) |
| 1 | 1 | 1 | 1 | 0 (15) |

$F = \sum_4 m\,(4,5,6,7,8,9,11)$

Ordenamos los minterm poniendo primero los que tienen todos cero, después los que tienen un uno, a continuación los que tienen dos unos, seguidamente los que tienen tres unos y, por último, el que tiene cuatro unos.

| | | |
|---|---|---|
| 0100 (4) | 010_ (4,5) | 01_ _ (4,5,6,7) |
| 1000 (8) | 01_0 (4,6) | 01_ _ (4,6,5,7) |
| 0101 (5) | 100_ (8,9) | |
| 0110 (6) | 01_1 (5,7) | |
| 1001 (9) | 011_ (6,7) | |
| 0111 (7) | 10_1 (9,11) | |
| 1011 (11) | | |

Comparamos los términos que tienen n unos con los que tienen n+1. Los términos que solo discrepan en el estado de una variable se simplifican. Esto se hace análogamente en la segunda columna y se repite hasta que no puedan realizarse más simplificaciones.

Los términos que no hayan sido utilizados más los que ya no admiten más simplificaciones se llaman **primeros implicantes**. Estos términos son suficientes para representar la función pero puede que no todos sean necesarios.

Tabla de primeros implicantes:

| | 4 | 5 | 6 | 7 | 8 | 9 | 11 |
|---|---|---|---|---|---|---|---|
| 100_ (8,9) | | | | | X | X | |
| 10_1 (9,11) | | | | | | X | X |
| 01_ _ (4,5,6,7) | X | X | X | X | | | |

El minterm que esta representado por un solo primer implicante nos dice que ese primer implicante es necesario para representar la función y se llama **primer implicante esencial**.

Si en alguna columna no hubiera marca querría decir que ese minterm no esta incluido en ningún primer implicante esencial y entonces habrá que coger un primer implicante que, sin ser esencial, funcione como tal.

f = AB’C’+AB’D+A’B

Este método se suele usar a partir de 5 variables.

**Términos de indiferencia**. Se llama así a unos términos que pueden aparecer representados de varias maneras (pero que generalmente aparecen como ceros tachados o equis) y que nosotros podemos tomar a la hora de simplificar o bien como ceros o bien como unos según nos convenga.

# Técnicas de diseño

## Homogeneización de circuitos

### Con puertas NAND

El proceso a aplicar es el siguiente:

1. Se pone la función como suma de productos.
2. Se aplican dos inversiones.
3. Se opera una inversión para sustituir las sumas por productos.

Ejemplo:

$$
\begin{aligned}
F(A,B,C,D) &= AB+B'C+(C'A)D \\
&= AB+B'C+C'D+AD \\
&= ((AB+B'C+C'D+AD)')' \\
&= ((AB)'(B'C)'(C'D)'(AD)')'
\end{aligned}
$$

### Con puertas NOR

El proceso es el siguiente:

1. Se pone la función como productos de sumas.
2. Se le aplican dos negaciones.
3. Se opera una negación para quitar los productos.

### Paso de un sistema diseñado con puertas NAND a otro con puertas OR y AND

Para hacer esto se consideran los niveles contados a partir de la salida y se hace lo siguiente:

1. Consideramos como OR todas las NAND que estén en un nivel impar.
2. Consideramos como AND todas las NAND que están en un nivel par
3. Debemos invertir todas las variables que lleguen al circuito en un nivel impar

### Paso de un sistema diseñado con puertas NAND a otro con puertas OR y AND

Se opera de la siguiente manera:

1. Se consideran como AND las NOR de nivel impar
2. Se consideran como OR las NOR de nivel par
3. Complementamos las variables que lleguen al circuito en nivel impar

Nota: El primer nivel es el compuesto por la puerta cuya salida es la función de salida, el segundo será el compuesto por aquellas puertas cuyas salidas están conectadas a una puerta de primer nivel y así sucesivamente.

## Codificadores

Un codificador es un circuito combinacional con $2^n$ entradas y n salidas, cuya función es presentar a la salida el código binario correspondiente a la entrada activa.

Existen dos tipos fundamentales de codificadores:

- Los que solo admiten una entrada activa, codificando en la salida el valor binario de la misma y cero cuando no existe ninguna activa.
- En los que puede haber más de una entrada activada, existiendo prioridad en aquella de valor decimal más alto. Estos se denominan codificadores con prioridad.

## Decodificadores

Realizan la función inversa a los codificadores. Presentan una salida activa dependiendo del valor binario de sus entradas.

## Multiplexores

Son circuitos con $2^n$ entradas de información, n entradas de selección y una salida de información, siendo su objetivo colocar en la salida el dato presente en la entrada seleccionada por las entradas de control.

El circuito actúa como un conmutador múltiple controlado por las entradas de selección.

## Demultiplexor

Realiza la función contraria a los multiplexores.

## Convertidores de código

Son circuitos cuya misión es transformar una información binaria representada en un determinado código y presente en su entrada, en la misma información pero en otro código que generará en su salida.

Un ejemplo típico es un conversor de BCD a 7 segmentos.

Circuitos que realizan operaciones aritméticas

Son circuitos combinacionales que realizan operaciones sencillas de suma y resta con un determinado número de bits en algún método de representación.

## Comparadores

Son circuitos con dos entradas de n bits y tres salidas de las cuales estará activa la correspondiente a la comparación de los dos números de entrada.