

# Control y programación de sistemas automáticos:

## Álgebra de Boole



Se denomina así en honor a **George Boole**, matemático inglés 1815 - 1864, que fue el primero en definirla como parte de un sistema **lógico**, a mediados del siglo XIX. El álgebra de Boole fue un intento de utilizar las técnicas algebraicas para tratar expresiones de la lógica proposicional.

El álgebra opera con **variables booleanas**, que son aquellas que sólo pueden tomar dos valores (0 y 1), estos valores no representan números si no estados. Ejemplo: pueden simbolizar si un interruptor está abierto (0), o cerrado (1), si conduce o no conduce, si hay tensión o no.

En la actualidad, esta herramienta se aplica de forma generalizada en el ámbito del diseño de control electrónico de procesos industriales. Estos métodos de control se basan en el uso de sistemas denominados puertas lógicas.

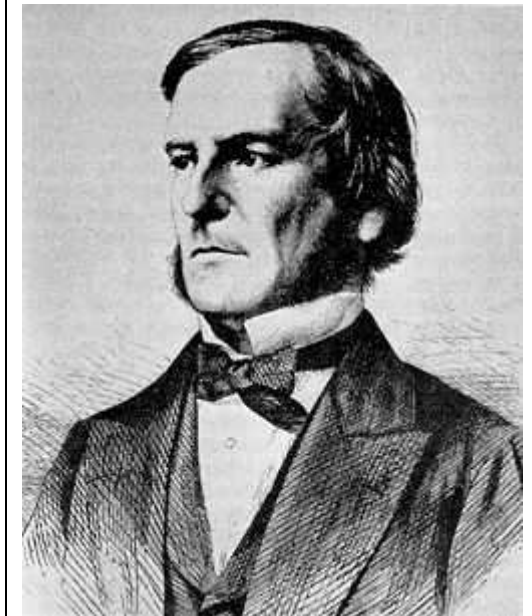


imagen 01. [Wikipedia](#). Creative Commons

Claude Shannon fue el primero en aplicarla en el diseño de circuitos de conmutación eléctrica **biestables**, en 1948.

# 1. Operaciones básicas en el álgebra de Boole



En primer lugar definiremos unos conceptos que necesitamos utilizar:



## Importante

### **Función lógica:**

Toda aquella variable binaria cuyo valor depende de una **expresión algebraica** constituida por otras variables que se encuentran relacionadas entre si por determinadas operaciones.

Por ejemplo sea la expresión:

$$S = A \cdot B + C$$

La interpretación de esta función será que la salida S, tomará el valor "uno" cuando lo hagan las variables A y B o la variable C.



## Importante

### **Función canónica:**

Expresión lógica en la que todos sus términos contienen todas las variables de entrada, bien afirmadas o bien negadas.

Por ejemplo 1 :

La siguiente función lógica **S**, está compuesta por tres términos, y cada término tiene en su composición las tres variables del sistema (A, B y C) . Esta función S, está expresada en **forma canónica**.

$$S = A \cdot B \cdot C + A \cdot \bar{B} \cdot C + A \cdot B \cdot \bar{C}$$

Por ejemplo 2 :

La siguiente función lógica **S**, está compuesta por tres términos, pero no todos estos términos tienen en su composición las tres variables del sistema (A, B y C) . Esta función **S**, está expresada en forma **No canónica**.

$$S = A \cdot B \cdot C + A \cdot \bar{B} + B \cdot \bar{C}$$



## Importante

### Tabla de verdad:

Tabla formada por **n** columnas de entradas o variables (**A, B, C...**) y otra más de salida (**f**), y por  $2^n$  filas, de modo que cada una de las filas representa cada una de las combinaciones posibles que pueden tener las variables de entrada, y el valor que toma la variable de salida para cada combinación de las variables de entrada.

Los posibles valores que pueden tomar las distintas variables representan estados diferentes de un dispositivo, y están representados por los valores lógicos "**cero**" (apagado, abierto, desconectado, ausencia, nivel bajo,...) o "**uno**" (encendido, cerrado, conectado, presencia, nivel alto,...).

Cuando en una tabla de verdad una variable está representada por el valor "x" quiere decir que su valor no está definido o que es indiferente el valor que tome para la respuesta de la función lógica.

En la figura adjunta se representa la tabla de verdad correspondiente a la función lógica:

$$S = A \cdot \bar{B} + \bar{A} \cdot B$$

ENTRADAS		SALIDA
A	B	f
0	0	0
0	1	1
1	0	1
1	1	0

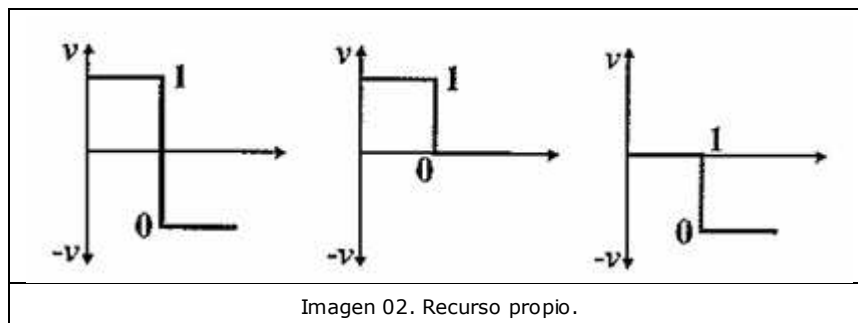


## Importante

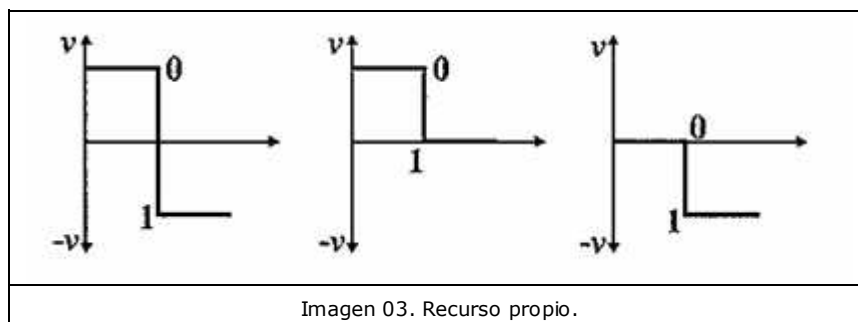
### Lógica positiva y lógica negativa:

Las variables lógicas pueden tomar exclusivamente los valores 0 y 1. Estos valores se pueden conseguir eléctricamente por dos procedimientos diferentes, según como se asignen los niveles de tensión se habla de lógica positiva o negativa.

Si se asigna al 1 lógico el valor más positivo de tensión y el 0 lógico al valor de menor tensión, estaremos ante **lógica positiva**.



Si se asigna al 1 lógico el valor más negativo de tensión, estaremos ante **lógica negativa**.



## 1.1. Operaciones básicas



En el álgebra de Boole se definen tres operaciones básicas que son:

- Producto lógico, o intersección.
- Suma lógica o unión.
- Negación, complementación o inversión.

Las operaciones básicas del álgebra de Boole se suelen implementar por medio de circuitos electrónicos, a estos componentes se les llama **puertas lógicas**.

Veamos ahora cada una de ellas por separado:



### Importante

#### **Producto lógico. Puerta AND. Y. &**

Función representada por la expresión:

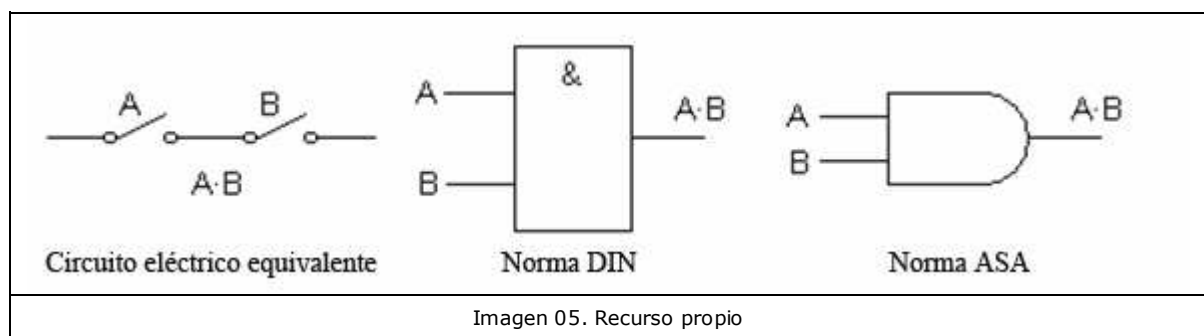
$$S = A \cdot B$$

Esta función responde a la tabla de verdad que se acompaña, debe entenderse como: La salida correspondiente a la función lógica producto toma valor 1 solamente cuando también son 1 los valores de las entradas A y B.

dec	A	B	$S = A \cdot B$
0	0	0	0
1	0	1	0
2	1	0	0
3	1	1	1

Imagen 04. Recurso propio

Esta operación puede representarse a través de un circuito eléctrico equivalente que se muestra en la siguiente tabla. En ella también se muestran los símbolos de este tipo de puertas según sean normas DIN o ASA (ANSI):



## Importante

### Suma lógica. Puerta OR. O.

Función representada por la expresión:

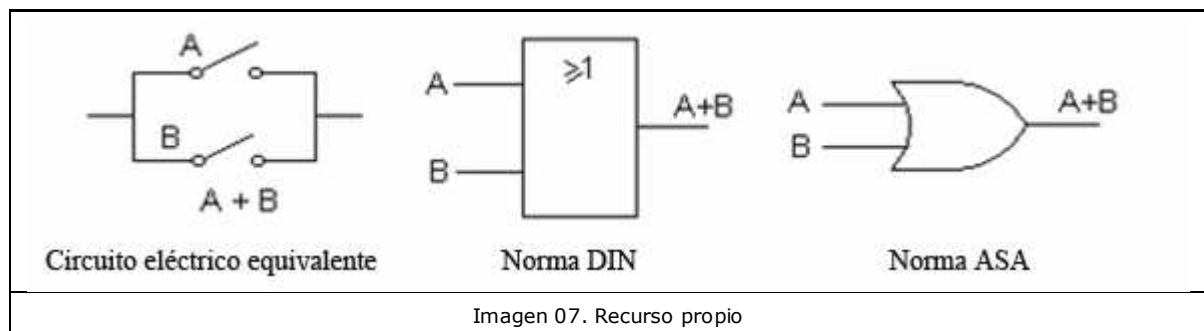
$$S = A + B$$

Esta función responde a la tabla de verdad que se acompaña, debe ser leída como: La salida correspondiente a la función lógica suma toma valor 1 cuando también son 1 cualquiera de las entradas A y B, o las dos simultáneamente.

dec	A	B	$S = A + B$
0	0	0	0
1	0	1	1
2	1	0	1
3	1	1	1

Imagen 06. Recurso propio

Esta operación puede representarse a través de un circuito eléctrico equivalente que se muestra en la siguiente tabla. En ella también se muestran los símbolos de este tipo de puertas según sean normas DIN o ASA:



## Importante

### Negación, complementación o inversión. NOT.

Función representada por la expresión:

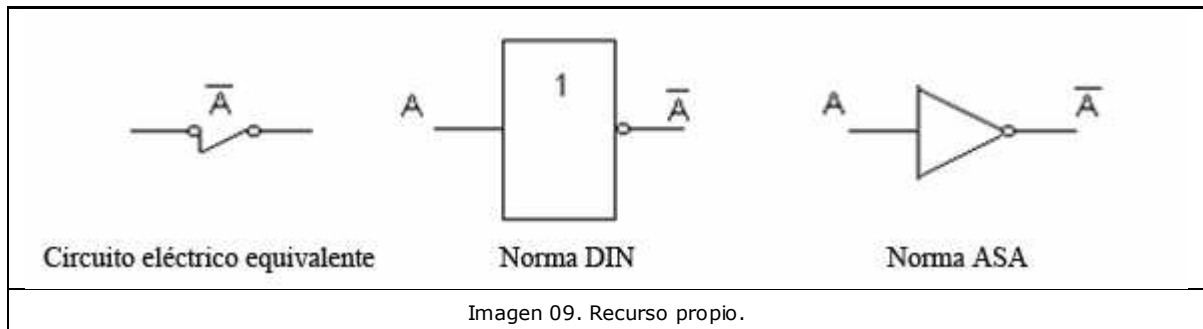
$$S = \bar{A}$$

Esta función responde a la tabla de verdad que se acompaña y debe entenderse como: La salida correspondiente a la función lógica complementación toma valor 1 cuando la entrada es 0 y viceversa.

$A$	$S = \bar{A}$
0	<b>1</b>
1	<b>0</b>

Imagen 08. Recurso propio

Esta operación puede representarse a través de un circuito eléctrico equivalente que se muestra en la siguiente tabla. En ella también se muestran los símbolos de este tipo de puertas según las normas DIN y ASA:





## 2. Otras operaciones lógicas



### Importante

#### Puerta NAND. No Y.

Función representada por la expresión:

$$S = \overline{A \cdot B}$$

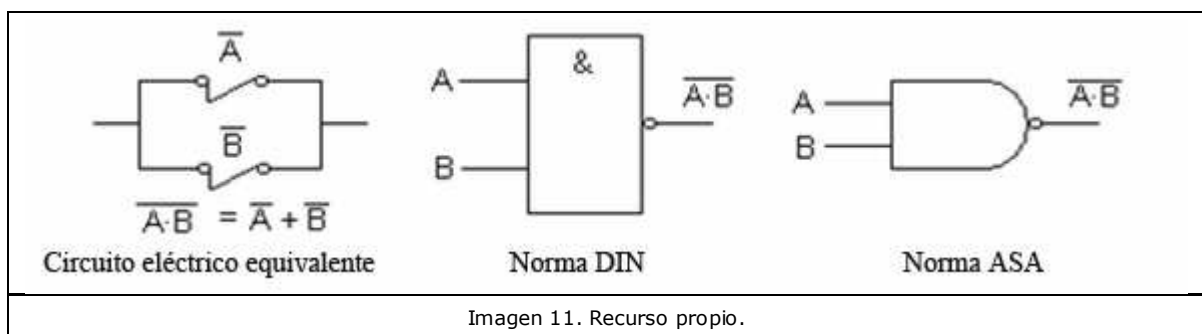
Esta función responde a la tabla de verdad que se acompaña, debe interpretarse como: La salida correspondiente a la función lógica toma siempre valor 1 salvo cuando A y B tienen a la vez valor 1.

La salida es el producto negado de las variables de entrada. Es decir la función NAND es la asociación de una puerta AND y de la función NOT.

dec	A	B	$S = \overline{A \cdot B}$
0	0	0	1
1	0	1	1
2	1	0	1
3	1	1	0

Imagen 10. Recurso propio.

Esta operación puede representarse a través de un circuito eléctrico equivalente que se muestra en la siguiente tabla. En ella también se muestran los símbolos de este tipo de puertas según sean normas DIN o ASA:





## Importante

### Puerta NOR. No O.

Función representada por la expresión:

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

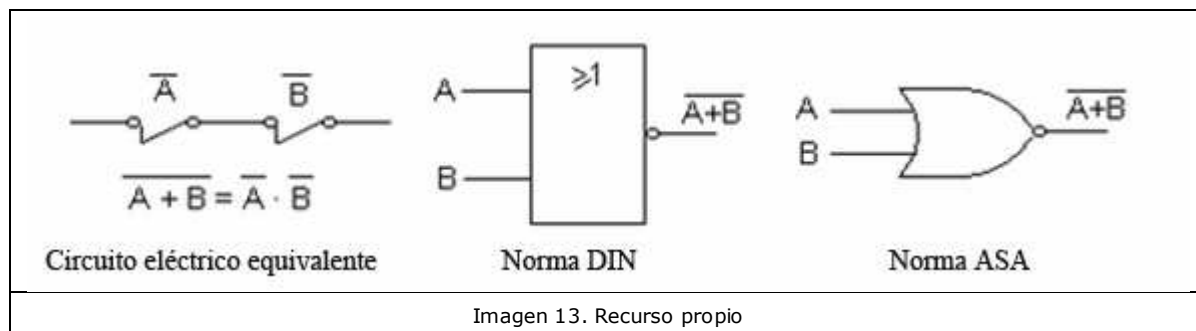
Esta función responde a la tabla de verdad que se acompaña, debe interpretarse como: La salida correspondiente a la función lógica toma valor 1 solamente cuando A y B tienen a la vez valor 0.

La salida es la suma negada de las variables de entrada. Es decir la función NOR es la asociación de una puerta OR y de la función NOT.

dec	A	B	$S = \overline{A+B}$
0	0	0	1
1	0	1	0
2	1	0	0
3	1	1	0

Imagen 12. Recurso propio

Esta operación puede representarse a través de un circuito eléctrico equivalente que se muestra en la siguiente tabla. En ella también se muestran los símbolos de este tipo de puertas según sean normas DIN o ASA:





## Importante

### Puerta OR EXCLUSIVA. XOR.

Función representada por la expresión:

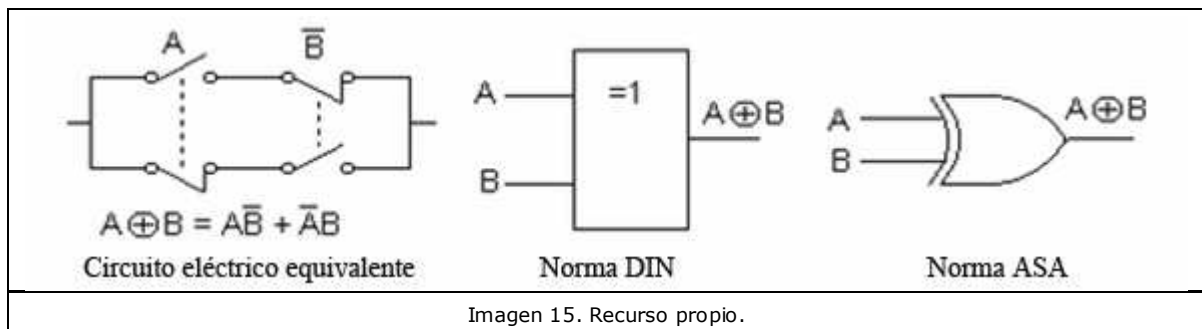
$$S = \bar{A} \cdot B + A \cdot \bar{B}$$

Esta función responde a la tabla de verdad que se acompaña, que debemos leer como: La salida correspondiente a la función lógica toma valor 1 cuando A ó B pero no las dos, toman el valor 1, es decir cuando una variable está afirmada y la otra está negada, o viceversa.

dec	A	B	$S = \bar{A} \cdot B + A \cdot \bar{B}$
0	0	0	0
1	0	1	1
2	1	0	1
3	1	1	0

Imagen 14. Recurso propio.

Esta operación puede representarse a través de un circuito eléctrico equivalente que se muestra en la siguiente tabla. En ella también se muestran los símbolos de este tipo de puertas según sean normas DIN o ASA:





## Importante

### Puerta NOR EXCLUSIVA. XNOR.

Función representada por la expresión:

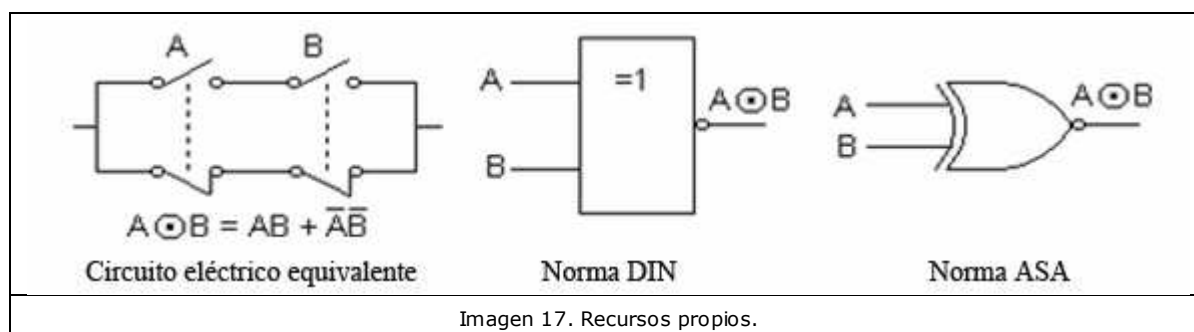
$$S = A \cdot B + \bar{A} \cdot \bar{B}$$

Esta función responde a la tabla de verdad que se acompaña, que debemos leer como: La salida toma el valor 1 cuando las dos entradas son 1 o las dos entradas son 0.

dec	A	B	$S = A \cdot B + \bar{A} \cdot \bar{B}$
0	0	0	1
1	0	1	0
2	1	0	0
3	1	1	1

Imagen 16. Recurso propio

Esta operación puede representarse a través de un circuito eléctrico equivalente que se muestra en la siguiente tabla. En ella también se muestran los símbolos de este tipo de puertas según sean normas DIN o ASA:





## Para saber más

- ▶ Comercialmente las puertas lógicas se distribuyen en circuitos integrados (CI), que disponen de catorce patillas o pines.
- ▶ Todos ellos presentan una muesca que se debe ver a la izquierda para empezar a contar las patillas desde abajo de la muesca y en sentido contrario a las agujas de un reloj.
- ▶ La patilla número 7 se conecta a masa y la número 14 a tensión de alimentación +Vcc, por lo que quedan 12 patillas para utilizar distintas entradas y salidas.
- ▶ Cada circuito integrado se identifica mediante un código, que indica el número y el tipo de puertas de que consta.

En la fotografía se muestra un circuito integrado que contiene cuatro puertas NAND de dos entradas, con las conexiones como se puede observar en el esquema adjunto.

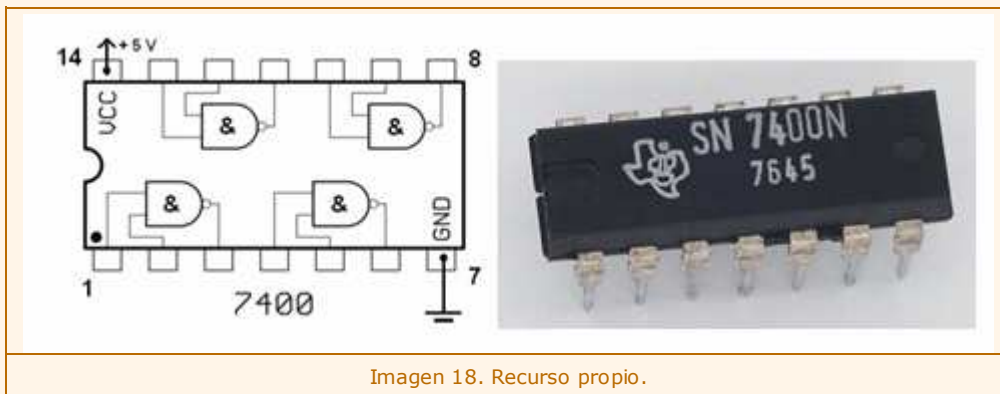


Imagen 18. Recurso propio.

### 3. Obtención de la función lógica a partir de la tabla de verdad



Una vez que hemos obtenido la tabla de verdad de un sistema o problema, existen dos métodos distintos para obtener la función lógica que la representa. En ambos métodos obtenemos una forma canónica. Como recordarás una forma canónica es la expresión de la función lógica sin ninguna simplificación, es decir, en cada término estarán presentes todas las variables de entrada, negadas o sin negar.

Los dos métodos son:



#### Importante

##### **Productos lógicos "Minterms"** (implementación por unos)

El proceso consiste en tomar todas las combinaciones de las variables de entrada que provocan que la función lógica presente un 1 en la salida.

Cada una de estas combinaciones de las variables de entrada será un sumando, constituido por el producto de todas las variables de entrada, en el que las variables estarán negadas cuando tomen el valor 0 y estarán afirmadas cuando tomen el valor 1.

La expresión de la función canónica será la suma de todos los productos equivalentes a las combinaciones de las variables de entrada que hacen que la salida sea un 1.

##### **Sumas lógicas "Maxterms"** (implementación por ceros)

El proceso consiste en tomar todas las combinaciones de las variables de entrada que provocan que la función lógica presente un 0 en la salida.

Cada una de estas combinaciones de las variables de entrada será un producto, constituido por la suma de todas las variables de entrada, en el que las variables estarán negadas cuando tomen el valor 1 y estarán afirmadas cuando tomen el valor 0. La expresión de la función canónica será el producto de todas las sumas equivalentes a las combinaciones de las variables de entrada que hacen que la salida sea un 0.

Cualquiera de los dos métodos es igual de eficaz y rápido, aunque suele tenerse tendencia a resolver los problemas utilizando el método de minterms, es decir suma de productos, o el método de los unos, aunque realmente deberíamos escoger un método u otro según el que produjese funciones canónicas más cortas.

Parece complicado, verás que es mucho más sencillo de entender si se hace un ejemplo.



## Ejercicio resuelto

Obtén la función lógica que se corresponde con la siguiente tabla de verdad:

A	B	C	F	
0	0	0	1	*
0	0	1	1	*
0	1	0	1	*
0	1	1	0	
1	0	0	0	
1	0	1	1	*
1	1	0	0	
1	1	1	1	*

Imagen 19. Recurso propio.



## Ejercicio resuelto

Obten la función canónica en el siguiente ejercicio:

[Descarga pdf](#)

## 4. Cronogramas



En ocasiones en vez de emplearse la tabla de verdad para indicar las combinaciones de las variables de entrada que provocan un determinado valor de la salida, se emplea un método gráfico en el que con un eje de tiempo coincidente para todas las variables de una función lógica, se representa por niveles altos y bajos (1 y 0) que van tomando las variables.

Vamos a verlo con un ejemplo.

Para transformar el siguiente cronograma de una determinada función lógica, en el que E1, E2 y E3 corresponde a las entradas y S es la salida, en primer lugar vamos a obtener su tabla de verdad, y a partir de esta información conseguimos la función canónica.



### Ejercicio resuelto

En el siguiente cronograma de una función lógica, E1, E2 y E3 corresponden con las entradas y S la salida. Obtén su función canónica.

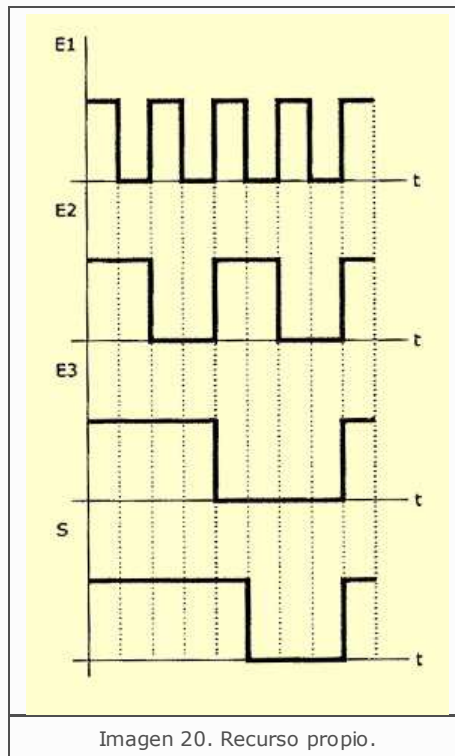


Imagen 20. Recurso propio.





## Ejercicio resuelto

1. A partir del cronograma adjunto, escribe su tabla de verdad e implementa el circuito lógico resultante con cualquier tipo de puertas.

[Descargar pdf](#)

2. La salida Y de un circuito lógico depende de dos entradas A Y B y su función puede describirse mediante el cronograma de la figura adjunta.

[Descargar pdf](#)

## 5. Álgebra de Boole



Vamos a enumerar las distintas operaciones, postulados, leyes y teoremas, para familiarizarnos con el manejo del álgebra de Boole.

### Postulados:

Respecto a la suma	Respecto al producto
$A + 0 = A$	$A \cdot 0 = 0$
$A + 1 = 1$	$A \cdot 1 = A$
$A + A = A$	$A \cdot A = A$
$A + \bar{A} = 1$	$A \cdot \bar{A} = 0$
$\bar{\bar{A}} = A$	

Imagen 22. Recurso propio

### Leyes y teoremas:

Leyes/Teoremas	Respecto a la suma	Respecto al producto
Commutativa.	$A + B = B + A$	$A \cdot B = B \cdot A$
Asociativa	$A + (B + C) = (A + B) + C$	$A \cdot (B \cdot C) = (A \cdot B) \cdot C$
Distributiva	$A \cdot (B + C) = A \cdot B + A \cdot C$	$A + (B \cdot C) = (A + B) \cdot (A + C)$
Cancelación	$(A \cdot B) + A = A$	$(A + B) \cdot A = A$
De Morgan	$\overline{A + B} = \bar{A} \cdot \bar{B}$	$\overline{A \cdot B} = \bar{A} + \bar{B}$

Imagen 23. Recurso propio.



### Autoevaluación

Comprueba que se cumplen las leyes de DeMorgan, mediante el método de las tablas de verdad, e implementa con los distintos tipos de puertas lógicas los resultados que has obtenido.

## 6. Implementación de circuitos con puertas NAND y NOR



Para la implementación de circuitos lógicos se pueden utilizar cualquier tipo de puertas. Sin embargo la tendencia más común es implementar un circuito empleando solamente un tipo de puertas. De este modo se abaratan costes.

Este método de implementación solo se puede realizar con puertas NAND o NOR, ya que solo estas dos puertas lógicas son **universales**, es decir se puede realizar cualquier circuito lógico y sustituir cualquier puerta empleando únicamente este tipo de puertas, para ello debemos seguir un cierto protocolo aprovechando que una doble negación es igual a una afirmación ( $A = \overline{\overline{A}}$ ).

Siempre podemos negar una expresión lógica dos veces, tantas veces como necesitemos y ésta quedará inmutable, si luego aplicamos el teorema de DeMorgan a una de las dos negaciones anteriores, conseguimos que un producto negado se convierta en una suma de variables negadas, o bien que una suma negada se convierta en un producto de variables negadas.

Ejemplo de ejecución de cualquier puerta empleando solamente puertas NOR.



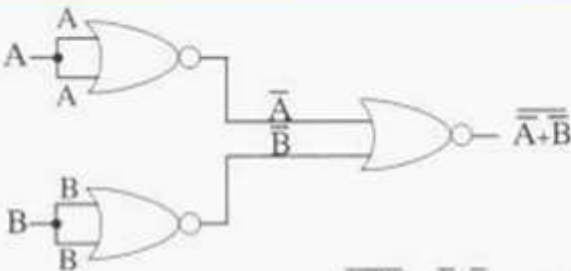
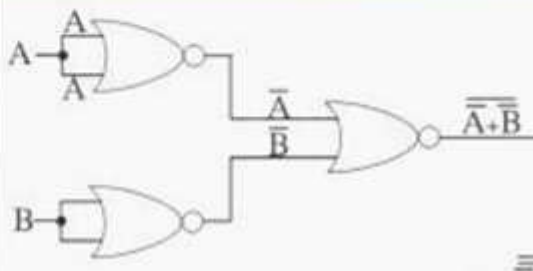
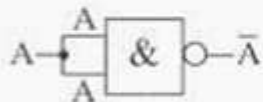
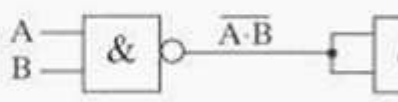
puerta NOT	puerta OR
 <p>Nota: <math>\overline{A+A} = \overline{A}</math></p>	
puerta AND	puerta NAND
 <p>(i) <math>\overline{\overline{A+B}} = \overline{\overline{A} \cdot \overline{B}} = A \cdot B</math></p>	 <p>(i) <math>\overline{\overline{A+B}} = \overline{\overline{A} \cdot \overline{B}} = A \cdot B</math></p>

Imagen 24.Recurso propio.

De igual forma podemos utilizar puertas lógicas NAND:

puerta NOT	puerta AND
 <p>Nota: <math>\overline{A \cdot A} = \overline{A}</math></p>	
puerta OR	puerta NOR



## Ejercicio resuelto

La expresión adjunta corresponde a una determinada función lógica:

$$F = A \cdot \bar{B} + B \cdot C + \bar{A} \cdot C$$

Se desea expresar esta función lógica en forma de productos negados (puertas NAND) o en forma de sumas negadas (Puertas NOR)



## Ejercicio resuelto

Implementar la función lógica que se adjunta empleando únicamente puertas NAND.  
Repetir el ejercicio empleando únicamente puertas NOR.

$$F = A(C + B \cdot D)$$



## Ejercicio resuelto

Resuelve el siguiente problema implementando puertas NOR

[Descarga pdf](#)

## 7. Simplificación de funciones lógicas



Una vez obtenida la función canónica de una expresión lógica, se debe buscar una expresión simplificada de ésta, con el menor número de términos. Con ello se consigue minimizar el número de errores posibles y abarata su implementación.

Existen dos métodos para conseguir simplificar funciones lógicas.



### Importante

#### **Simplificación por el método algebraico:**

Consiste en utilizar todos los postulados, leyes y teoremas enunciados anteriormente. Con este método se consiguen simplificaciones óptimas, pero hay que tener cierta práctica, no obstante cuando las funciones tienen una expresión grande el procedimiento algebraico puede provocar el que cometamos errores por ser complejo y pesado de utilizar, sobre todo si quien lo realiza no es un experto.

#### **Simplificación por el método de Karnaugh:**

Más rápido y eficaz, sobre todo ante problemas complejos. El método se basa en la construcción de unas tablas con la característica de que entre cada celda y su contigua o adyacente solamente cambia el valor de una variable de entrada.

En este punto vamos a mostrar el funcionamiento del método de Karnaugh. A continuación se adjuntan los mapas de Karnaugh para tres y cuatro variables, indicando en el interior de cada celda a que combinación de variables de entrada corresponde:

$\begin{smallmatrix} AB \\ C \end{smallmatrix}$	00	01	11	10
0	$ABC$	$ABC$	$ABC$	$ABC$
1	$ABC$	$ABC$	$ABC$	$ABC$

$\begin{smallmatrix} AB \\ CD \end{smallmatrix}$	00	01	11	10
00	$AB\bar{C}\bar{D}$	$AB\bar{C}D$	$AB\bar{C}\bar{D}$	$AB\bar{C}\bar{D}$
01	$AB\bar{C}\bar{D}$	$AB\bar{C}D$	$AB\bar{C}\bar{D}$	$AB\bar{C}\bar{D}$
11	$AB\bar{C}\bar{D}$	$AB\bar{C}D$	$AB\bar{C}\bar{D}$	$AB\bar{C}\bar{D}$
10	$AB\bar{C}\bar{D}$	$AB\bar{C}D$	$AB\bar{C}\bar{D}$	$AB\bar{C}\bar{D}$

Imagen 28. Recurso propio.

Se colocará un 1 en las celdas correspondientes a las combinaciones de las variables de entrada que hacen que la salida sea un 1.

Hay que tener en cuenta que las tablas son contiguas por los flancos extremos, es decir en una tabla de cuatro variables de entrada, las celdas de la columna de la derecha son contiguas a las de la columna de la izquierda, e igualmente ocurre con las casillas de la fila superior y las de la fila inferior.

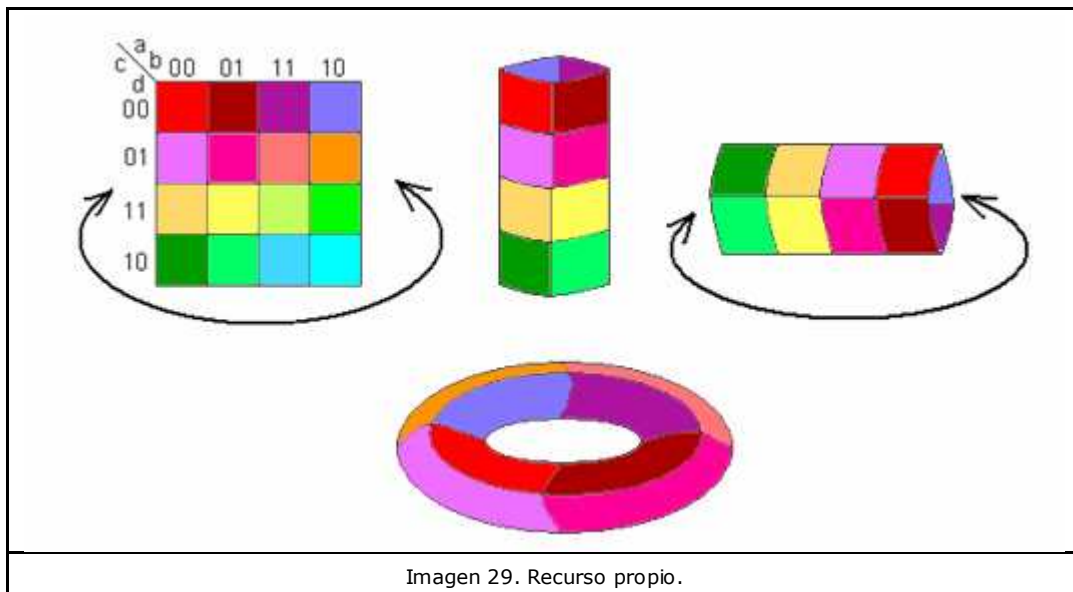


Imagen 29. Recurso propio.

Es decir se puede considerar que se doblan formando un cilindro, como se observa en la figura anterior, y se vuelve a doblar de nuevo formando un [toroide](#).

Las celdas que ocupan los extremos también son adyacentes, por lo anteriormente descrito.

El método se basa en hacer agrupamientos con el mayor número de celdas posible, siempre que sean potencias de dos (16, 8, 4, 2), tratando que todas las celdas que contengan 1 pertenezcan a agrupaciones de celdas.

Se debe tratar de que no haya celdas comunes a varias agrupaciones, siempre que sea posible. Pero no está prohibido que haya celdas que pertenezcan a más de una agrupación.

Las agrupaciones serán horizontales y verticales; las diagonales no están permitidas.

Aunque si están permitidos las verticales y horizontales que lleguen al final de la fila o la columna, y vuelvan a enlazarse otra vez al inicio, o viceversa.

Se debe procurar que haya el menor número de agrupaciones con el mayor número de unos posible.

La función simplificada tendrá tantos términos como agrupaciones o bolsas tenga el mapa de Karnaugh.

Cada término se obtiene eliminando la o las variables que cambien de estado dentro de la misma bolsa.

- Una bolsa de  $2=2^1$  celdas provoca una simplificación de una variable.
- Una bolsa de  $4=2^2$  celdas provoca una simplificación de dos variables.
- Una bolsa de  $8=2^3$  celdas provoca una simplificación de tres variables.
- Una bolsa de  $16=2^4$  celdas provoca una simplificación de cuatro variables.

Si hubiese una celda aislada que no perteneciese a ninguna agrupación, este término quedaría sin simplificar.

De nuevo vuelve a parecer algo complicado. Veamos algunos ejemplos y lo comprenderás mejor:

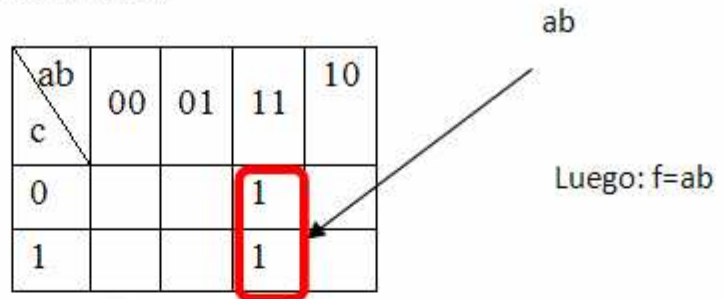


## 7.1. Ejemplos simplificación por el método de Karnaugh

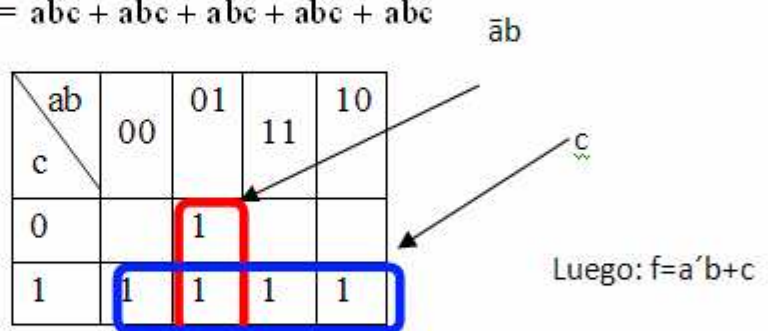


A continuación se muestran cinco ejemplos correspondientes a tablas de Karnaugh de tres y cuatro variables:

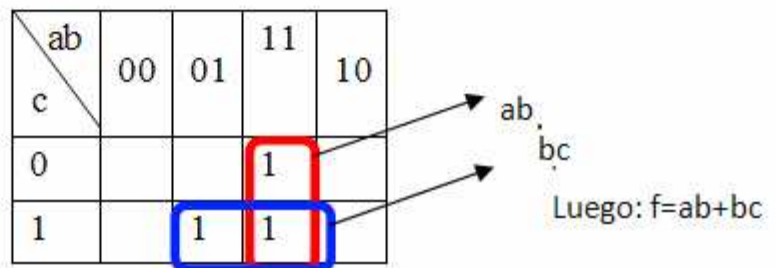
**Ejemplo 1.** Con 3 variables:  $f = ab\bar{c} + abc$



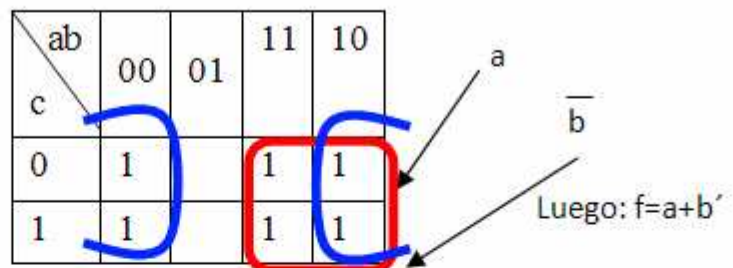
**Ejemplo 2.** Con 3 variables:  $f = \bar{a}\bar{b}\bar{c} + \bar{a}bc + a\bar{b}\bar{c} + abc + a\bar{b}c$



**Ejemplo 3.** Con 3 variables:  $f = \bar{a}bc + abc + ab\bar{c}$



**Ejemplo 4.** Con 3 variables:  $f = \bar{a}\bar{b}\bar{c} + \bar{a}bc + ab\bar{c} + abc + a\bar{b}\bar{c} + a\bar{b}c$



**Ejemplo 5.** Con 4 variables:

$$f = \bar{a}\bar{b}\bar{c}d + \bar{a}bcd + ab\bar{c}d + abcd + \bar{a}\bar{b}cd + \bar{a}bcd + ab\bar{c}d + abcd + \bar{a}\bar{b}cd$$



## Ejercicio resuelto

A partir de la función lógica adjunta:

- ▶ Obtener la máxima simplificación posible.
- ▶ Implementar la función utilizando cualquier tipo de puertas lógicas de dos entradas.
- ▶ Implementar la función empleando únicamente puertas NAND.

$$F = \overline{A}B(\overline{C} + \overline{D}) + BC + \overline{A} \cdot \overline{D}$$



## Ejercicio resuelto

A continuación te planteamos un problema a resolver el método de Karnaugh:

Problema

## 8. Términos indiferentes



En ocasiones, en algunos problemas lógicos, hay combinaciones de las variables de entrada que no se pueden producir, o bien el valor que tome la función para esas combinaciones de las entradas es indiferente para la salida, en ese caso en la tabla de verdad en vez de poner en la salida un 1 o un 0, se pone una x y al simplificar por el método de Karnaugh se considera que el valor es un 1 o un 0 según lo que más favorezca para simplificar. Vamos a verlo con un ejemplo.



### Ejercicio resuelto

Sea un número inferior a diez codificado en binario.

- Obtén la tabla de verdad de la función  $F \leq 5$  (menor o igual de cinco), y escribe su función canónica.
- Simplifica la función obtenida por el método de Karnaugh.