



CIRCUITOS LÓGICOS Y ALGEBRA DE BOOLE



CIRCUITOS LÓGICOS

La lógica proposicional y el álgebra de Boole están vinculados y sirven como bases fundamentales para el diseño y análisis de circuitos lógicos en la electrónica digital.

Por un lado, la lógica proposicional se centra en el estudio de las proposiciones y sus interacciones mediante operaciones lógicas como la conjunción, disyunción y negación. Por otro, el álgebra de Boole, desarrollada por George Boole, es una estructura matemática que formaliza estas operaciones en un sistema algebraico.

Esta conexión se torna esencial en los circuitos lógicos, donde las señales pueden considerarse como valores de verdad (verdadero o falso) y se manipulan a través de puertas lógicas como AND, OR y NOT.

Estas puertas realizan operaciones que se asemejan a las operaciones lógicas en la lógica proposicional y pueden modelarse utilizando las reglas del álgebra de Boole.

CIRCUITO EN SERIE

Este circuito tiene la particularidad de que sobre una misma línea se encuentran los interruptores, o sea que:

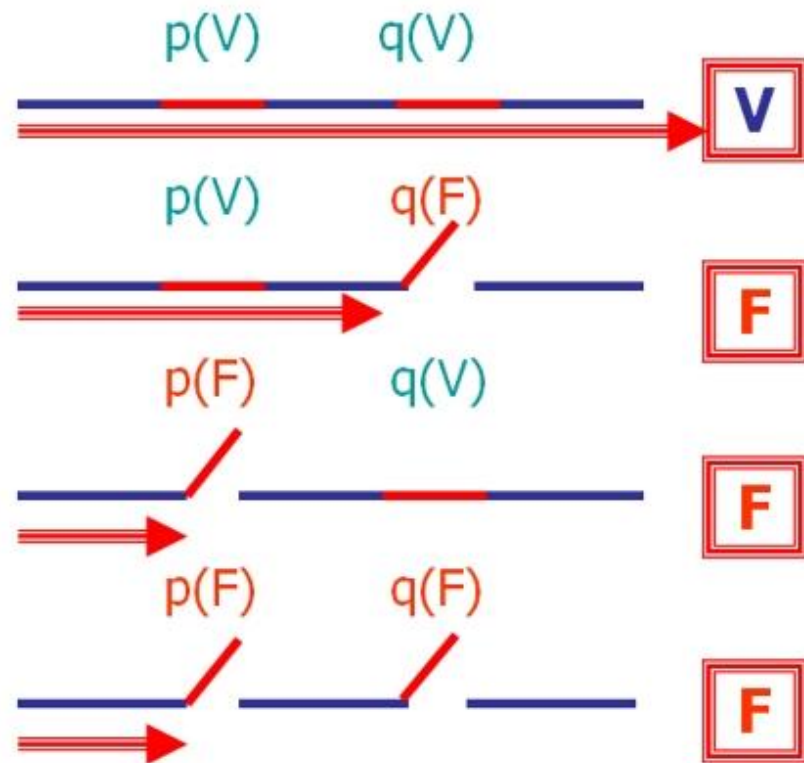


Este circuito se relaciona con la conjunción

p	q	$p \wedge q$
V	V	V
V	F	F
F	V	F
F	F	F

CIRCUITO EN SERIE

p	q	$p \wedge q$
V	V	V
V	F	F
F	V	F
F	F	F





En la primera línea p y q son VERDADEROS, lo que significa los interruptores están cerrados o sea que pasa la corriente por los dos.

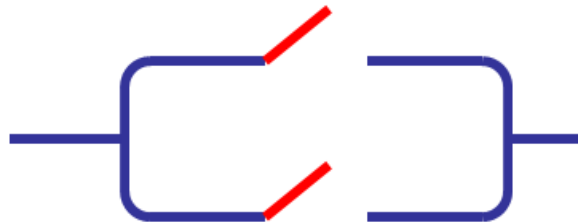
En la segunda línea p es VERDADERO y q FALSO, lo que significa que la corriente pasa hasta antes de q .

En la tercera línea p es FALSO y q VERDADERO, lo que significa que la corriente pasa hasta antes de p .

En la cuarta línea p es FALSO y q FALSO, lo que significa que la corriente pasa hasta antes de p .

CIRCUITO EN PARALELO

Este está formado por dos líneas que tienen el mismo principio y el mismo fin y con un interruptor en cada línea.

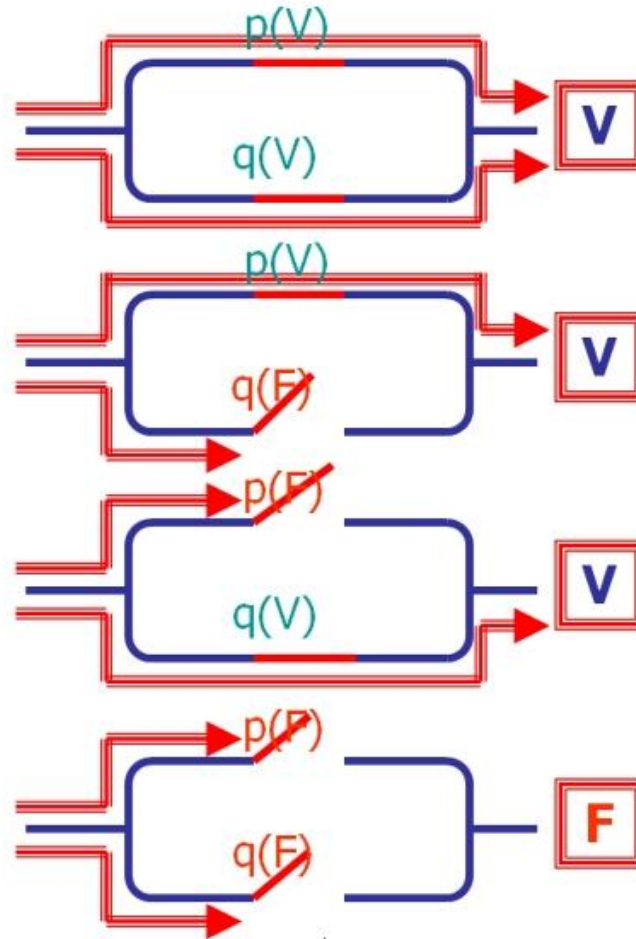


Este circuito se relaciona con la disyunción teniendo en cuenta que la VERDAD de esta proposición está basada en la llegada de la corriente al final del mismo. O sea:

p	q	$p \vee q$
V	V	V
V	F	V
F	V	V
F	F	F

CIRCUITO EN PARALELO

p	q	$p \vee q$
V	V	V
V	F	V
F	V	V
F	F	F






En la primera línea p y q son VERDADEROS, o sea que las llaves están cerradas y pasa la corriente por las dos líneas del circuito.-

En la segunda línea, p es VERDADERO y q FALSO, o sea que solamente la llave p está cerrada, lo que deja pasar la corriente.

En la tercera línea, p es FALSO y q es VERDADERO, lo que significa que la corriente pasa por q, entonces la disyunción es VERDADERA.

En la cuarta línea, p y q son FALSOS, por lo que tanto las llaves p y q están abiertas, entonces la disyunción es FALSA ya que no llega la corriente hasta el final del circuito.



Ahora, el problema se presenta cuando debemos construir un circuito lógico de proposiciones que no son disyunciones ni conjunciones ni negaciones.

En estos casos, se deben aplicar propiedades y leyes lógicas simplificar esas proposiciones compuestas hasta su reducción a conjunciones y disyunciones.-



ALGEBRA DE BOOLE



En la práctica, trabajaremos los circuitos lógicos con la notación de Álgebra de Boole.

- Los valores de verdad se consignarán con 0 y 1

0 \rightarrow Falso

1 \rightarrow Verdadero

- Las proposiciones serán: A, B (letras mayúsculas impresas)
- La negación de la proposición A la indicaremos como A'
- Los conectivos lógicos los representaremos de la siguiente manera:

Conjunción

(conexión en serie)

•

Disyunción

(conexión en paralelo)

+

Así las tablas de verdad quedarán:

Conjunción

(conexión en serie)

A	.	B
1	1	1
1	0	0
0	0	1
0	0	0

Disyunción

(conexión en paralelo)

A	+	B
1	1	1
1	1	0
0	1	1
0	0	0

LEYES EN CIRCUITOS LÓGICOS

Las mismas leyes de la lógica proposicional se verifican para circuitos lógicos.

1. Involución:

$$(A')' = A$$

2. Idempotencia:

$$A + A = A$$

$$A . A = A$$

3. Complemento:

$$A + A' = 1$$

$$A . A' = 0$$

4. Identidad

$$\begin{array}{ll} A + 1 = 1 & A + 0 = A \\ A \cdot 1 = A & A \cdot 0 = 0 \end{array}$$

5. Conmutativa

$$\begin{array}{l} A + B = B + A \\ A \cdot B = B \cdot A \end{array}$$

6. Asociativa

$$\begin{array}{l} A + (B + C) = (A + B) + C \\ (A \cdot B) \cdot C = A \cdot (B \cdot C) \end{array}$$

7. Distributiva

$$\begin{array}{l} A \cdot (B + C) = (A \cdot B) + (A \cdot C) \\ A + (B \cdot C) = (A + B) \cdot (A + C) \end{array}$$

8. Leyes de De Morgan

$$\begin{array}{l} (A \cdot B)' = A' + B' \\ (A + B)' = A' \cdot B' \end{array}$$

FUNCIÓN BOOLEANA:

Una función booleana es una expresión matemática que describe una relación entre variables binarias, es decir, variables que pueden tener uno de dos posibles valores: verdadero (1) o falso (0).

Una expresión booleana es una forma de representar operaciones y relaciones lógicas utilizando variables y operadores, una función booleana es un mapeo que asigna entradas booleanas a salidas booleanas específicas.

Las expresiones booleanas a menudo se utilizan para describir funciones booleanas, pero una función booleana es más que simplemente una expresión, ya que describe cómo se comporta una operación lógica en términos de sus entradas y salidas.

VARIABLE:

Cada elemento de la expresión con distinto nombre.

$A \cdot B' + A' C + A (D+E) \rightarrow 5 \text{ variables}$

LITERAL:

cada aparición de una variable o de su complemento.

$A \cdot B' + A' C + A (D+E) \rightarrow 7 \text{ literales}$

Cuando se trabaja con expresiones booleanas puede interesar que éstas estén expresadas como:

- a. Sumas de Productos o Minitérminos o Forma Normal Disyuntiva (FND)
- b. Productos de Sumas o Maxitérminos o Forma Normal Conjuntiva (FNC)

SUMA DE PRODUCTOS (FND)

Minitérmino: es un producto booleano en el que cada variable aparece una vez

Ej.: $A.B.C'$ y $A'.B.C$

La suma de productos consiste en dos o más minitérminos que se suman mediante la suma booleana

PRODUCTOS DE SUMAS (FNC)

Maxitérmino: es una suma booleana en la que cada variable aparece una vez

Ej.: $A+B+C'$ y $A'+B+C$

El producto de sumas consiste en dos o más maxitérminos que se multiplican mediante la multiplicación booleana

CONVERSIÓN DE TABLAS DE VERDAD A FND

Una función booleana puede representarse algebraicamente a través de la tabla de verdad formando la suma de todos los minitérminos que producen un 1 en la función.

X	Y	Z	F
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

$$F = \overline{X}\overline{Y}\overline{Z} + \overline{X}Y\overline{Z} + X\overline{Y}Z + XYZ$$

Suma expandida de productos

CONVERSIÓN DE TABLAS DE VERDAD A FNC

Una función booleana también puede representarse algebraicamente a través de la tabla de verdad formando el producto de todos los maxitérminos que producen un 0 en la función.

X	Y	Z	F
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

$$\bar{F}(X,Y,Z) = \bar{X}\bar{Y}Z + \bar{X}YZ + X\bar{Y}\bar{Z} + XY\bar{Z}$$

CÓMO CONSTRUIR FND A PARTIR DE EXPRESIONES LÓGICAS

$$f = (xy' + yz)' + x'$$

no tiene la forma normal disyuntiva, pero puede ser convertida a esta forma mediante el siguiente proceso :

- Aplicar las leyes de D'Morgan para eliminar los complementos de los paréntesis,
- Aplicar la ley distributiva del producto sobre la suma para reducir la función a un polinomio;
- si en algún término falta una variable, este se multiplica por la suma de la variable ausente más su complemento, sin afectar el resultado (puesto que se está multiplicando por 1). En este ejemplo

$$\begin{aligned} f &= (xy' + yz)' + x' = (xy')' (yz)' + x' = (x' + y)(y' + z') + x' = x'y' + x'z' + yz' + x' \\ &= x'y'(z + z') + x'(y + y')z' + (x + x')yz' + x'(y + y')(z + z') \\ &= x'y'z + x'y'z' + x'yz' + xyz' + x'yz \end{aligned}$$

CÓMO CONSTRUIR FNC A PARTIR DE EXPRESIONES LÓGICAS

La función: $f = (xy' + yz)' + x'$ no tiene la forma normal conjuntiva.

Mediante el siguiente proceso una función puede convertirse a la forma normal conjuntiva:

- eliminar los complementos (si los hay) con las leyes de D'Morgan,
- factorizar la función;
- si en algún factor falta una variable, a este se le suma el producto de la variable ausente por su complemento, sin afectar el resultado (puesto que se le está sumando cero). En este ejemplo:

$$\begin{aligned} f &= (xy' + yz)' + x' = (xy')'(yz)' + x' = \\ &= (x' + y)(y' + z') + x' = x' + (x' + y)(y' + z') = \\ &= (x' + x' + y)(x' + y' + z') = (x' + y)(x' + y' + z') \end{aligned}$$


En el primer factor falta z , a este se le suma zz'

$$\begin{aligned} f &= (x' + y + zz')(x' + y' + z') \\ f &= (x' + y + z)(x' + y + z')(x' + y' + z') \end{aligned}$$



COMPUERTAS LÓGICAS




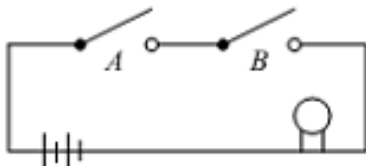

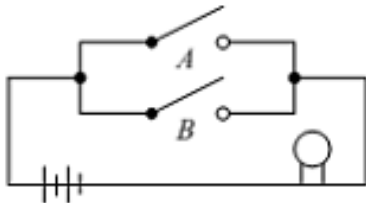
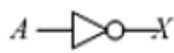
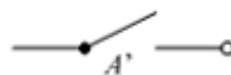

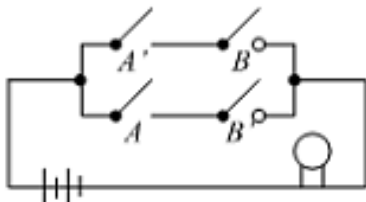



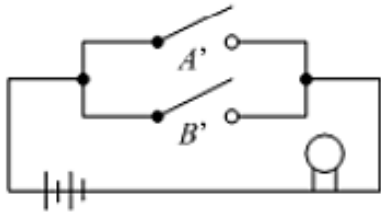

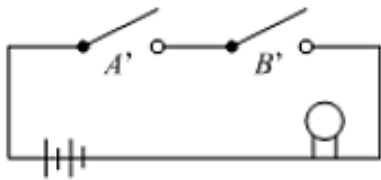
La representación de los circuitos lógicos utilizando interruptores y las puertas lógicas AND, OR y NOT son dos formas diferentes de visualizar y entender el funcionamiento de los mismos principios subyacentes en la electrónica digital.

1. Representación con Interruptores (Modelo Físico): Esta representación se basa en la idea de que los circuitos digitales pueden implementarse utilizando interruptores físicos que pueden estar en dos estados: abierto (representado como 0) o cerrado (representado como 1). Esta representación es útil para comprender la esencia de cómo funcionan los circuitos digitales a nivel físico y cómo las señales eléctricas se manipulan para lograr operaciones lógicas.

2. Puertas Lógicas (Modelo Abstracto): Las puertas lógicas como AND, OR y NOT son abstracciones matemáticas que representan operaciones lógicas fundamentales. Estas puertas se utilizan para describir cómo las señales lógicas se combinan y transforman en circuitos digitales. Esta representación es más abstracta y simplificada en comparación con los interruptores físicos. Permite un análisis y diseño más eficiente de circuitos complejos al proporcionar reglas algebraicas precisas para manipular las señales lógicas.

COMPUERTAS LÓGICAS: representación gráfica de una o más variables de entrada a un operador lógico para obtener una señal de salida

Expresión	Compuerta Lógica	Tabla de Verdad	Circuito de Interruptores															
$X = AB$	 AND	<table><tr><th>A</th><th>B</th><th>X</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	X	0	0	0	0	1	0	1	0	0	1	1	1	
A	B	X																
0	0	0																
0	1	0																
1	0	0																
1	1	1																
$X = A + B$	 OR	<table><tr><th>A</th><th>B</th><th>X</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	X	0	0	0	0	1	1	1	0	1	1	1	1	
A	B	X																
0	0	0																
0	1	1																
1	0	1																
1	1	1																
$X = A'$	 NOT	<table><tr><th>A</th><th>X</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	A	X	0	1	1	0										
A	X																	
0	1																	
1	0																	
$X = A \oplus B$ \Rightarrow $X = A'B + AB'$	 XOR (OR exclusivo)	<table><tr><th>A</th><th>B</th><th>X</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	A	B	X	0	0	0	0	1	1	1	0	1	1	1	0	
A	B	X																
0	0	0																
0	1	1																
1	0	1																
1	1	0																

Expresión	Compuerta Lógica	Tabla de Verdad	Circuito de Interruptores															
$X = (AB)'$	 NAND	<table><tr><th>A</th><th>B</th><th>X</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	A	B	X	0	0	1	0	1	1	1	0	1	1	1	0	
A	B	X																
0	0	1																
0	1	1																
1	0	1																
1	1	0																
$X = (A + B)'$	 NOR	<table><tr><th>A</th><th>B</th><th>X</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	A	B	X	0	0	1	0	1	0	1	0	0	1	1	0	
A	B	X																
0	0	1																
0	1	0																
1	0	0																
1	1	0																