

**APUNTES DE  
CIRCUITOS  
DIGITALES Y  
ÁLGEBRA BOOLEANA**

**José Juan Urrutia Milán**

# Reseñas

· **Curso:**

<https://www.youtube.com/playlist?list=PLM-p96nOrGcZTxenCCDXiZxPCLQWSFr2Q>

· **Logisim:** Software para simulación de circuitos:

<http://www.cburch.com/logisim/>

# Índice



# Introducción al álgebra booleana

El álgebra booleana nos permite desarrollar componentes digitales.

El álgebra booleana se basa en tres operaciones básicas: AND, OR y NOT.

En álgebra booleana encontramos valores constantes (donde el 0 se puede representar como una toma de tierra) y variables, que usualmente son letras mayúsculas. Las variables sólo pueden almacenar un bit de información.

El álgebra booleana sólo acepta los valores de verdadero (1) y falso (0).

En el álgebra booleana, se hace uso del signo “=” para asignar valores a las variables: **A = 1**.

Además, podemos nombrar a un byte o a un conjunto de bits con una variable, indicando un subíndice a la hora de referirnos a cada cable. Representamos un byte con la letra B:

**B<sub>0</sub> , B<sub>1</sub> , B<sub>2</sub> , B<sub>3</sub> , B<sub>4</sub> , B<sub>5</sub> , B<sub>6</sub> , B<sub>7</sub> .**

# Capítulo I: Álgebra Booleana

## Título I: Tablas de verdad

Cuando trabajamos en álgebra booleana, normalmente haremos uso de funciones a las que le pasamos una serie de variables booleanas.

La forma más fácil de conocer los valores de dicha función es haciendo uso de una tabla de verdad, la cual nos muestra la información de salida a partir de la información de entrada.

En las tablas de verdad necesitamos incluir todos los posibles valores de entrada para así obtener todos los posibles valores de salida.

Los valores de las tablas de verdad suelen estar ordenados.

Estas tablas ordenadas reciben el nombre de tablas canónicas y son las regulares en ingeniería.

El número (x) de filas que debemos agregar para conocer todas las posibles entradas de una función de n variables de entrada es:  $x = 2^n$ . Por tanto, en una función de 3 variables de entrada, su tabla de verdad debe tener 8 ( $2^3$ ) filas.

### Expresar tabla de verdad como vector

Para expresar una tabla de verdad en forma vectorial, simplemente debemos tener la tabla canónica (aquella con los inputs ordenados) y expresar en formato de función los outputs de dicha tabla. Ejemplo con la tabla de verdad de una puerta AND:

A	B	AND
0	0	0
0	1	0
1	0	0
1	1	1

$$\text{AND}(A, B) = (0, 0, 0, 1)$$

De esta forma, si introducimos a la función AND un 2 en binario (10  $\rightarrow$  A=1, B=0), nos va a devolver aquel elemento que se encuentra en la posición 2 (en la tercera, empezamos a contar en 0):

$$\text{AND}(1, 0) = 0$$

Los más común es encontrar tablas de verdad canónicas, pero en este documento haremos uso de la nomenclatura como vector, debido a su tamaño.

## Título II: Operadores lógicos



## XOR

Tiene dos entradas.

Tiene una salida.



Se representa en diagramas con:

En álgebra booleana, se representa con un más dentro de una circunferencia:

$$Z = A \oplus B$$

La salida sólo será verdad si los dos valores de entrada no son iguales:

$$Z(A, B) = (0, 1, 1, 0)$$

## XNOR

En álgebra booleana, se representa con un signo de idéntico:

$$Z = A \equiv B$$

Se trata de una compuerta XOR negada, por lo que su tabla de verdad se invierte.

Se representa con una puerta XOR con una puerta NOT (una circunferencia pequeña) detrás de esta.

La salida sólo será verdad si los dos valores de entrada son iguales:

$$Z(A, B) = (1, 0, 0, 1)$$

## NAND

En álgebra booleana, se representa con una operación AND negada:

$$Z = \overline{A * B}$$

Se trata de una compuerta AND negada, por lo que su tabla de verdad se invierte.

Se representa con una puerta AND con una puerta NOT (una circunferencia pequeña) detrás de esta.

La salida sólo será falsa si todas las entradas son verdaderas.

$$Z(A, B) = (1, 1, 1, 0)$$

## NOR

En álgebra booleana, se representa con una operación OR negada:

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

Se trata de una compuerta OR negada, por lo que su tabla de verdad se invierte.

Se representa con una puerta OR con una puerta NOT (una circunferencia pequeña) detrás de esta.

La salida sólo será verdadera si todas las entradas son falsas.

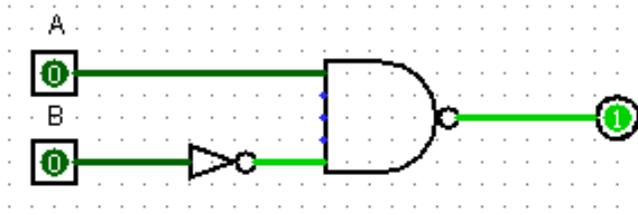
$$Z(A, B) = (1, 0, 0, 0)$$

## Implicación material

Tiene dos entradas.

Tiene una salida.

En diagramas, se trata de un circuito que se construye con una puerta NOT y con una NAND:



En álgebra booleana, representamos la puerta de la siguiente forma:

$$Z = A \rightarrow B$$

Si la primera entrada es verdadera, la salida será el valor de la segunda entrada.

Si la primera entrada es falsa, el valor de la salida puede ser ignorado (se puede poner un 1).

$$Z(A, B) = (-, -, 0, 1) \quad \text{o} \quad Z(A, B) = (1, 1, 0, 1)$$

## Título III: Leyes aritméticas

### Leyes aritméticas simples

#### Ley aditiva

La suma funciona como una puerta lógica OR.

#### Ley multiplicativa

La multiplicación funciona como una puerta lógica AND.

### Jerarquía de operadores

La jerarquía a tomar será la siguiente: NOT > AND > OR, aceptando el uso de los paréntesis.

Esta jerarquía la usaremos para transformar expresiones algebraicas a diagramas ([ver vídeo 12](#)).

Además, la herramienta de Logisim nos permite en: Ventana > Análisis Combinacional, nos permite crear tablas de verdad, diagramas y expresiones algebraicas a partir de cualquiera de estos.

### Identidades algebraicas

(El complemento de una variable es ella misma negada)

El 0 es el elemento neutro de la suma:  $X + 0 = X$

Al sumar 1 a una variable, el resultado será 1:  $X + 1 = 1$

Al sumar una variable a sí misma, el resultado será ella misma:  $X + X = X$

Al sumar una variable a su complemento, el resultado será 1:  $X + /X = 1$

El 1 es el elemento neutro de la multiplicación:  $X \cdot 1 = X$

Al multiplicar 0 a una variable, el resultado será 0:  $X \cdot 0 = 0$

Al multiplicar una variable por sí misma, el resultado será ella misma:  $X \cdot X = X$

Al multiplicar una variable por su complemento, el resultado será 0:  $X \cdot /X = 0$

## Propiedades algebraicas

$$X + Y = Y + X$$

$$X \cdot Y = Y \cdot X$$

$$X + Y + Z = X + (Y + Z)$$

$$X \cdot Y \cdot Z = X \cdot (Y \cdot Z)$$

$$X \cdot Y + X \cdot Z = X \cdot (Y + Z)$$

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

$$//A = A$$

$$A + (A \cdot B) = A$$

$$A \cdot (A + B) = A$$

$$A + (/A \cdot B) = A + B$$

$$A \cdot (/A + B) = A \cdot B$$

$$A \cdot B + /A \cdot B = B$$

## Título IV: Transformar ecuaciones a tablas de verdad

([ver vídeo 14](#))

### Suma de productos

Si queremos pasar una ecuación en modo de suma de productos a tablas de verdad, primero hemos de identificar el producto canónico (aquel donde se multiplican todas las variables que aparecen en la ecuación, sin importar si se encuentran negadas o no).

De esta forma, iremos rellenando la tabla, viendo cuando se cumple que los productos canónicos son verdaderos (ya que con que un producto canónico sea verdadero, la ecuación será verdadera, al tratarse de sumas de productos (puertas OR)).

Los términos canónicos producirán una fila en la tabla de verdad.

Posteriormente, vemos término por término cuando es que este hace que la ecuación sea verdadera y proseguimos rellenando la tabla de verdad.

Los términos no canónicos pueden producir más de dos filas en la tabla de verdad.

Una vez que analizamos todos los términos, rellenamos la tabla con ceros.

### Ejemplo

$$Y(J, K, L) = J \cdot /K + /J \cdot K \cdot L + J \cdot K \cdot /L + K \cdot L$$

Observamos dos productos canónicos: el segundo y el tercer término.

El segundo término hace verdadera la ecuación en  $Y(0, 1, 1)$ .

El tercer término hace verdadera la ecuación en  $Y(1, 1, 0)$ .

El primer término hace verdadera la ecuación en  $Y(1, 0, x)$ .

El cuarto término hace verdadera la ecuación en  $Y(x, 1, 1)$ .

Una vez conociendo esto, rellenamos la tabla de verdad con 1 en todas estas salidas.

Y finalmente, completamos con ceros aquellas filas restantes:

J	K	L	Y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

### Producto de sumas

En este caso, hemos de hacer algo análogo a lo anterior pero buscando ceros en vez de unos, ya que si una suma nos da cero y esta la multiplicamos por otra, hará cero la ecuación.

#### Ejemplo

$$R(A, B, C) = (A + B + C) \cdot (A + B) \cdot (A + B + C)$$

La primera suma se hace cero en R(1, 0, 0).

La tercera suma se hace cero en R(1, 1, 1).

La segunda suma se hace cero en R(1, 0, x).

Completamos con cero las entradas dichas.

Rellenamos el resto con unos:

A	B	C	R
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

## Título V: Condensar tablas de verdad y obtener expresión algebraica a partir de tabla condensada

Permite reducir la expresión algebraica y reducir los componentes del circuito.

Consiste en observar patrones dentro de las tablas y en ignorar ciertas variables no relevantes en ciertas líneas de la tabla de verdad.

De esta forma, podemos expresar líneas cuya salida sea la misma y en cuya entrada sólo cambia el valor de una variable en líneas que no dependan de dicha variable.

### Ejemplo

$$Y(J, K, L) = J \cdot /K + /J \cdot K \cdot L + J \cdot K \cdot /L + K \cdot L$$

J	K	L	Y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Como observamos, siempre que  $J=1$ , la salida va a ser  $Y=1$ , por lo que podemos sustituir las últimas cuatro líneas por:  $Y(1, x, x) = 1$ .

Además, siempre que  $J=0$  y que  $K=0$ , la salida va a ser  $Y=0$ , por lo que podemos sustituir las dos primeras líneas por:  $Y(0, 0, x)$ .

Sin embargo, en las líneas 3 y 4 no podemos concluir ningún patrón, las conservamos.

De esta forma, nuestra tabla simplificada sería la siguiente:

J	K	L	Y
0	0	x	0
0	1	0	0
0	1	1	1
1	x	x	1

### Obtener expresión algebraica a partir de tabla condensada

Para obtener la expresión algebraica a partir de la tabla condensada, vamos a obtener una suma de productos, donde cada producto es una línea de nuestra tabla de verdad.

De esta forma, por cada fila, incluimos la variable por cada 1, la variable negada por cada 0 y no la incluimos por cada x:

La fila 3 es:  $Y(0, 1, 1) = 1 \rightarrow /J \cdot K \cdot L$

La fila 4 es:  $Y(1, x, x) = 1 \rightarrow J$

Como la salida de las filas 1 y 2 es 0, las ignoramos.

De esta forma, la expresión algebraica sería la siguiente:

$$Y(J, K, L) = J + /J \cdot K \cdot L$$

## Título VI: Teorema de DeMorgan

Nos va a permitir hacer reducciones de circuitos y sustituir puertas OR por AND y viceversa. Así como transformar una suma de productos a un producto de sumas y viceversa.

**El inverso de la suma es igual al producto de los complementos de los términos:**

$$\overline{(x + y)} = \overline{x} \cdot \overline{y}$$

**El inverso del producto es igual a la suma de los complementos de los términos:**

$$\overline{(x \cdot y)} = \overline{x} + \overline{y}$$

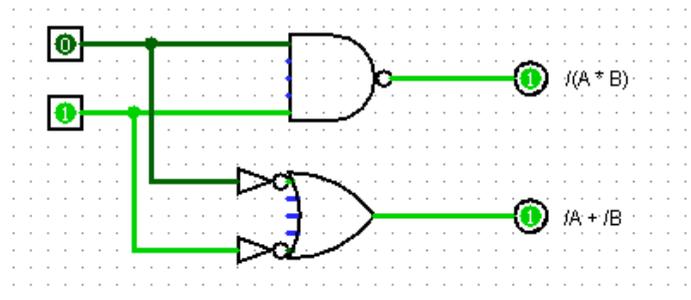
### Empujar burbujas

Teorema de DeMorgan visto de forma gráfica:

Podemos “empujar la burbuja” de una puerta NAND (la puerta lógica NOT que se encuentra después de la AND) a la entrada de la puerta AND, negando todas sus entradas.

Para que esto sea posible, debemos sustituir la puerta AND por una OR (y viceversa).

De esta forma, una puerta lógica NAND es equivalente a una puerta lógica OR con todas sus entradas negadas:



Además, una puerta NOR es equivalente a una puerta AND con todas sus entradas negadas.

### Aplicado a AND o OR

Podemos usar la doble negación para aplicar el teorema de DeMorgan a puertas AND o OR, ya que una puerta AND cuya salida está doblemente negada es equivalente a una puerta AND.

Una vez tenemos estas dos negaciones, empujamos una de las burbujas al inicio.

De esta forma, una puerta lógica AND es equivalente a una NOR con todas sus entradas negadas y una puerta lógica OR es equivalente a una NAND con todas sus entradas negadas.

### Aplicado a expresiones algebraicas

Para ello, haremos uso de la doble negación y de las propiedades vistas al inicio del título:

$$R = x \cdot y + A \cdot B + J \cdot K$$

$$R = \overline{\overline{(x \cdot y + A \cdot B + J \cdot K)}}$$

$$R = \overline{(\overline{(x + \overline{y})} \cdot \overline{(A + \overline{B})} \cdot \overline{(J + \overline{K})})}$$

## Título VII: Formalización de funciones

Nos permite realizar reducciones en las funciones y expresiones algebraicas.

### Minterms

Nombre que recibe aquel término canónico que pertenece a una suma de productos, el cual se puede simplificar con un símbolo del orden:  $m_i$  (donde  $i$  es un número natural).

De esta forma, podemos hacer uso de la siguiente tabla:

X	Y	Z	Expresión	Índice Minterm	Símbolo
0	0	0	$\bar{X}\bar{Y}\bar{Z}$	0	$m_0$
0	0	1	$\bar{X}\bar{Y}Z$	1	$m_1$
0	1	0	$\bar{X}Y\bar{Z}$	2	$m_2$
0	1	1	$\bar{X}YZ$	3	$m_3$
1	0	0	$X\bar{Y}\bar{Z}$	4	$m_4$
1	0	1	$X\bar{Y}Z$	5	$m_5$
1	1	0	$XY\bar{Z}$	6	$m_6$
1	1	1	$XYZ$	7	$m_7$

Para reducir la longitud de nuestras expresiones.

De esta forma, la ecuación:  $Y(X, Y, Z) = XYZ + \bar{X}Y\bar{Z} + XY\bar{Z}$

Se puede expresar como:  $Y(X, Y, Z) = m_7 + m_2 + m_6$

O como:  $Y(X, Y, Z) = \Sigma m(7, 2, 6)$

Además, esta notación la podemos emplear para condensar tablas.

### Maxterms

Nombre que recibe aquel término canónico que pertenece a un producto de sumas, el cual se puede simplificar con un símbolo del orden:  $M_i$  (donde  $i$  es un número natural).

De esta forma, la ecuación:  $Y(X, Y, Z) = (X + Y + Z) \cdot (\bar{X} + Y + \bar{Z})$

Se puede expresar como:  $Y(X, Y, Z) = M_0 \cdot M_5$

O como:  $Y(X, Y, Z) = \Pi M(0, 5)$

Si nos damos cuenta, no es necesario hacer uso de la tabla; podemos convertir las expresiones a binario (donde la negación es un 0 y la no negación un 1), y dicho número en decimal será el subíndice del maxterm (o del minterm). Los maxterms se hacen al contrario que los minterms, las negadas son unos y los no negados ceros.:  $X + Y + Z \rightarrow 1 1 1 \rightarrow 0 0 0 \rightarrow M_0$ .

## Título VIII: Mapas de Karnaugh

Son matrices de celdas que nos permiten eliminar términos no relevantes.  
De esta forma, nos permiten condensar funciones algebraicas.

Para ello, partimos de una función algebraica de una suma de productos (si tenemos unos productos de suma, podemos pasarla mediante el Teorema de DeMorgan).

### Distancia de Hamming

Nos sirve para determinar la distancia existente entre dos números binarios, la cantidad de bits diferentes entre dos números binarios:

**A: 10011001**

**B: 10011001**

**C: 11011010**

La distancia de Karnaugh entre A y B es de 0, ya que tienen 0 bits diferentes.

La distancia de Karnaugh entre B y C es de 3, ya que tienen 3 bits diferentes.

De esta forma, podemos ordenar tablas de verdad en vez de en forma canónica, en forma de distancia de Hamming, de forma que cada entrada tenga una distancia de Hamming con el siguiente de 1:

Forma canónica:	00	01	10	11
Forma de dist. de Hamming:	00	01	11	10

### Mapas de Karnaugh de 2 variables

De esta forma, creamos una matriz 2x2 donde las filas tengan el valor de una variable y las columnas la de otra variable, y rellenamos siguiendo la distancia de Hamming.

Una vez hecho esto, observamos que obtenemos todas las posibles combinaciones de ambas variables, cada minterm.

Para hacer el mapa de Karnaugh, colocamos un 1 en todas las celdas donde cada minterm aparece en nuestra expresión, y rellenamos con ceros.

Estas celdas en las que colocamos un 1 reciben el nombre de indicante.

Posteriormente, realizamos el **looping**, es decir, encerramos los indicantes con un círculo. Estos cierres deben ser del orden  $2^n \times 2^m$  (donde n y m son números naturales).

Posteriormente, vamos por cada cierre y comprobamos si las variables se mantienen constantes o varían (Si son constantes en 0, colocamos el complemento de la variable). Si la variable permanece constante, la colocamos. Si varía, no la colocamos. De esta forma, podemos reducir funciones algebraicas de dos variables.

### Ejemplo

$$F(X, Y) = XY + \overline{X}Y$$

Construimos su mapa de Karnaugh:

$x \setminus y$	0	1
0	0	1
1	0	1

$XY$  aparece en  $F(X, Y)$ , por tanto,  $F(1, 1) = 1$ .

$\overline{X}Y$  aparece en  $F(X, Y)$ , por tanto,  $F(0, 1) = 1$ .

Rellenamos el resto con ceros.

Una vez obtenido su mapa, realizamos el looping:

En este caso, sólo podemos rodear los dos unos (resaltados en amarillo). Este rodeo es válido, ya que es del tipo  $1 \times 2$ , que cumple la regla:  $2^n \times 2^m$  ( $2^0 \times 2^1$ ).

En este rodeo, vemos que el valor de  $y$  se mantiene constante (1) mientras que el de  $x$  no. De esta forma, podemos condensar la función inicial, manteniendo  $y$ , ya que es la variable constante:

$$F(X, Y) = Y$$

## Mapas de Karnaugh de 3 variables

En este caso, vamos a utilizar las columnas para dos variables y las filas para una variable. Recordamos que debemos rellenarlas según la distancia de Hamming.

A la hora de trabajar con tres variables, ahora podemos hacer diferentes tipos de loopings. Para obtener una función más eficiente, debemos escoger los loopings lo más grandes posibles.

Las coberturas o loopings pueden cruzarse.

### Ejemplo

$$F(X, Y, Z) = \overline{X}\overline{Y}Z + \overline{X}YZ + \overline{X}YZ + \overline{X}YZ + XYZ + XY/Z$$

Construimos su mapa de Karnaugh:

$x \setminus yz$	00	01	11	10
0	1	1	1	1
1	0	0	1	1

$\overline{X}\overline{Y}Z$  aparece en  $F(X, Y, Z)$ , luego  $F(0, 0, 0) = 1$ .

$\overline{X}YZ$  aparece en  $F(X, Y, Z)$ , luego  $F(0, 0, 1) = 1$ .

$\overline{X}YZ$  aparece en  $F(X, Y, Z)$ , luego  $F(0, 1, 1) = 1$ .

$\overline{X}YZ$  aparece en  $F(X, Y, Z)$ , luego  $F(0, 1, 0) = 1$ .

XYZ aparece en  $F(X, Y, Z)$ , luego  $F(1, 1, 1) = 1$ .

XY/Z aparece en  $F(X, Y, Z)$ , luego  $F(1, 1, 0) = 1$ .

Rellenamos con 0.

Llegados a este punto, podemos hacer looping de diferentes formas.

Para obtener una expresión más reducida, hemos de maximizar la forma de los loopings.

De esta forma, los loopings más grandes que podemos coger son aquellos resaltados en la tabla (uno resaltado en amarillo y otro con el texto en rojo).

Los loopings pueden cruzarse.

Empezamos con el looping resaltado en amarillo, que llamaremos  $F_1$ :

Sólo permanece constante y, luego:  $F_1 = Y$

El looping con el texto en rojo,  $F_2$ :

Sólo permanece constante x, y lo hace en 0, luego:  $F_2 = \neg X$

Una vez analizados todos los loopings, procedemos a construir la función como sumatoria de dichos loopings:

$$F(X, Y) = Y + \neg X$$

Como hemos podido observar, hemos simplificado tanto la función que hasta no nos hacía falta una variable.

## Mapas de Karnaugh de 4 variables

Se realizan de forma análoga a lo anterior, colocamos dos variables en cada fila y dos en cada columna. Rellenamos según la distancia de Hamming.

Hemos de tener en cuenta que a la hora de realizar el looping, los mapas de Karnaugh son periódicos (los de cualquier variable), por lo que podemos “llevar el looping hacia el infinito por arriba” para que salga por abajo (ver ejemplo).

### Ejemplo

$$F(W, X, Y, Z) = \neg W/\neg X/Y/Z + \neg W/X/Y/Z + W/\neg X/Y/Z + W/X/Y/Z + \neg W/X/Y/\neg Z$$

W X \ Y Z	00	01	11	10
00	1			1
01	1			
11				
10	1			1

Realizamos 2 loopings: el resaltado en amarillo y en rojo (los mapas de Karnaugh son infinitos):

$$F_1 = /W/Y/Z$$

$$F_2 = /X/Z$$

$$F(W, X, Y, Z) = /X/Z + /W/Y/Z$$

### Obtener producto de sumas a partir de Mapa de Karnaugh

Una vez obtenido el mapa de Karnaugh de cualquier función (como el del ejemplo anterior), podemos obtener la función reducida en términos de producto de sumas en vez de en suma de productos. Para ello, en vez de fijarnos en los indicantes (celdas con 1) a la hora de hacer el procedimiento de los loopings, nos fijamos en las celdas con 0.

Sólo hay que tener en cuenta que colocaremos los complementos de las variables a la hora de hacer la función.

Según la tabla del ejemplo anterior:

W X \ Y Z	00	01	11	10
00	1	0	0	1
01	1	0	0	0
11	0	0	0	0
10	1	0	0	1

Agrupamos los 0 en tres grupos (amarillo, texto en rojo y subrayado):

Amarillo: Sólo permanece constante Z en uno, pero como es el complemento, incluimos /Z:

$$F_1 = /Z$$

Rojo: Permanece constante X en uno (/X) e Y en uno (/Y):

$$F_2 = /X + /Y$$

Subrayado: Sólo permanecen constantes w y x en 1:

$$F_3 = /W + /X$$

$$F(W, X, Y, Z) = /Z \cdot (/X + /Y) \cdot (/W + /X)$$



# Capítulo II: Binario

## Bit más significativo

El bit más significativo de un número binario es aquel que puede aportar un mayor valor al número. Es decir, el bit que se encuentra más a la izquierda

## Bit menos significativo

El bit menos significativo de un número binario es aquel que menos influye al valor total del número. Es decir, el bit que se encuentra más a la derecha.

## Binario a Decimal

El método más común es usar la siguiente tabla:

$2^9$	$2^8$	$2^7$	$2^6$	$2^5$	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$

## Decimal a Binario

Para ello, usaremos la tabla superior, preguntándonos si el número decimal es mayor o igual a cada posición. Si lo es, colocaremos uno y si no, cero.

# Título I: Aritmética binaria

## Suma

Nota: el más aquí no representa la puerta OR, sino que es una suma en binario.

Al sumar en binario, es común que nos encontremos junto con el resultado otro dígito a modo de **carry**. Este carry (c) es el típico dígito que “nos llevamos” al realizar cualquier operación.

$$0 + 0 = 0 \quad c = 0$$

$$0 + 1 = 1 \quad c = 0$$

$$1 + 0 = 1 \quad c = 0$$

$$1 + 1 = 0 \quad c = 1$$

$$1 + 1 + 1 = 1 \quad c = 1$$

## Resta

(ver segundo complemento).

En la resta, nos volvemos a encontrar una variable de “prestado”, similar al carry en la suma:

$$0 - 0 = 0 \quad p = 0$$

$$0 - 1 = 1 \quad p = 1$$

$$1 - 0 = 1 \quad p = 0$$

$$1 - 1 = 0 \quad p = 0$$

Restamos mediante la siguiente fórmula:  $(M - P) - S$   
Minuendo menos prestado menos sustraendo ([ver video 24](#)).

## Título II: Números negativos

Hay dos técnicas para representar números negativos en binario:

### Primer complemento

Para ello, vamos a observar el bit más significativo:

Si es igual a 1: el número es negativo.

Si es igual a 0: el número es positivo.

Un inconveniente del primer complemento es que nos reduce la cantidad de números que podemos representar a la mitad, lo que también puede provocar problemas de overflow.

Otro inconveniente es que hay dos formas de representar el 0:

00000000 y 10000000

### Segundo complemento

Utilizando el segundo complemento vamos a poder representar todos los números enteros entre -128 y 127 con un byte, evitando los problemas que nos daba el primer complemento.

Para representar un número positivo, lo haremos de la forma tradicional, dejando como 0 el bit más significativo:  $5 \rightarrow 00000101$

Para representar un número negativo, debemos tener su representación en binario.

Luego, realizamos la inversión de todos sus bits.

Finalmente, sumamos un 1:

$5 \rightarrow 00000101$

11111010

11111011  $\rightarrow -5$

El segundo complemento nos permite simplificar la resta de números binarios, ya que en vez de realizarla con la fórmula superior, podemos pasar el sustraendo a segundo complemento y sumar el minuendo más el segundo complemento del sustraendo, obteniendo así el resultado de la resta.

### Segundo complemento a sistema decimal

Para ello, realizamos la inversión del número y le sumamos 1.

Una vez obtenido el número en binario, lo pasamos a decimal, teniendo en cuenta que como su bit más significativo era un 1, es un número negativo:

11111011

00000100

00000101

Lo pasamos a decimal y es un 5

Como su bit más significativo era un 1: 1111011  $\rightarrow$  -5



# Capítulo III: Electrónica

Existen aislantes, que no permiten el paso de corriente, conductores, que sí y los semiconductores.

## Semiconductores

Puede actuar como aislante y como conductor.

El silicio es el semiconductor más usado.

El silicio se dopa con fósforo (tipo N) y boro (tipo P).

## Título I: El diodo

Es un dispositivo semiconductor resultado de unir silicio tipo P con silicio tipo N.

Esto va a hacer que el paso de corriente sólo sea posible en una dirección, funcionando como conductor en esta dirección y como aislante en la contraria.

La terminal (cable) que se encuentra en el lado de tipo P se denomina **ánodo** y la que se encuentra en la parte de silicio tipo N se denomina **cátodo**.

Si conectamos una batería a un diodo, debemos conectar el lado positivo al ánodo y el negativo al cátodo.

Por tanto, el diodo sólo permite el paso de corriente del cátodo al ánodo.



En los diagramas, los diodos se representan de la siguiente forma:

## Título II: El transistor

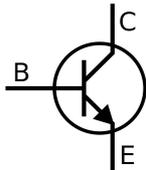
Es un dispositivo conductor resultado de unir tres capas de silicio, del tipo NPN o PNP.

La capa más importante es la base, que es la intermedia y la más delgada. Nos permite controlar la cantidad de corriente que fluye a través del transistor.

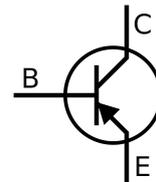
Además, encontramos el colector y el emisor.

Dependiendo del tipo de transistor (NPN o PNP), lo representaremos con un diagrama o con otro:

NPN:



PNP:



El transistor nos sirve a su vez de amplificador: a mayor corriente apliquemos en la base, mayor corriente obtenemos en el colector y emisor.