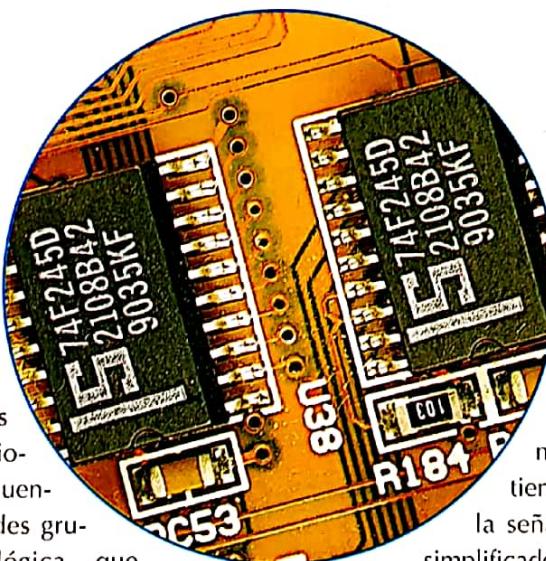


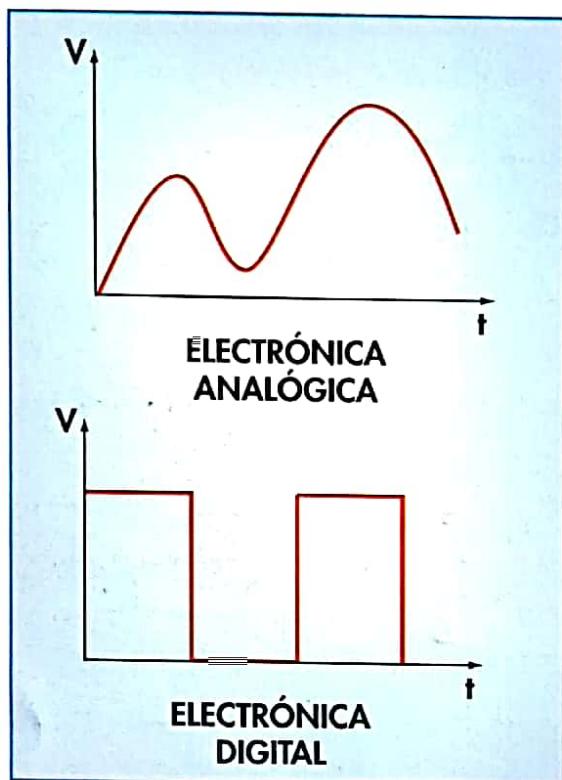
LA ELECTRÓNICA DIGITAL

Como el lector sabe, y como ya hemos comentado en otras ocasiones, la electrónica se encuentra dividida en dos grandes grupos: la electrónica analógica –que hemos estudiado durante todo este tiempo, a través de las resistencias, condensadores, diodos, tiristores, transistores, etc.– caracterizada sobre todo por la variación continua de todos los parámetros, es decir, tensión, corriente, desfases y otros, y la electrónica digital, caracterizada por las variaciones ampliables de sus parámetros.

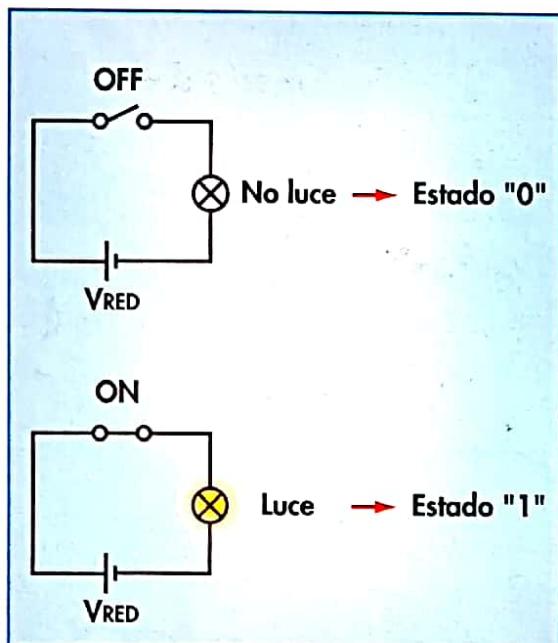
De esta manera, en un circuito analógico la tensión o la corriente pueden adoptar infinitos valores diferentes, mientras que, por el contrario, en un circuito digital sólo es posible encontrar dos valores de tensión o corriente, es decir, dos únicas posibilidades que nos obligan a hablar del carácter binario del mismo.



Cuando estudiábamos la señal en el tiempo de un circuito analógico, tal y como podemos observar en la figura adjunta, teníamos señales continuas en el tiempo, pero, ¿qué ocurre con la señal digital? Sencillo, hemos simplificado los valores y eso nos facilita el tratamiento de la misma para las diversas aplicaciones que necesitemos como, por ejemplo, la captación de la temperatura ambiental a través de un sensor y después la conversión de esa señal analógica a valores digitales, de forma que pueda ser mostrada en un panel, en forma de números. Un ejemplo para entender de forma sencilla la electrónica digital consiste en la simplificación de los valores de la clavija de la luz que tengamos en nuestro salón, es decir, cuando la luz está apagada, el conmutador de encendido/apagado está en OFF, mientras que cuando está encendida tenemos un ON, lo cual nos indica la forma de funcionar de la electrónica digital.



Diferencia entre la electrónica analógica, que es continua en el tiempo, y la electrónica digital, que se caracteriza por los dos niveles distintos de tensión, nivel alto y nivel bajo.



Una forma de simplificar los valores de la tensión de la bombilla es observar cómo se encuentra el interruptor, esto es, si está cerrado tendremos un nivel alto, tenemos tensión, mientras que si está abierto el interruptor tendremos un nivel bajo, ya que a la salida no tenemos tensión.

Aunque todos los componentes estudiados tienen también su aplicación dentro de la electrónica digital, existe uno bastante importante que nos facilita la inmersión dentro del mundo digital, siendo, de hecho, constituyente de una de las células básicas sobre las que se asienta la electrónica digital, ya que todos los circuitos lógicos contienen en mayor o menor medida este componente: hablamos del transistor.

A su vez, la electrónica digital se divide en dos mundos totalmente distintos, es decir, los circuitos secuenciales y los circuitos combinables, que serán estudiados más adelante.

Características del transistor en el mundo digital

Aunque hemos estudiado profundamente las características del transistor en el mundo analógico, existen otras características dentro del mundo digital que tenemos que tener en cuenta y que no eran demasiado importantes en el modo de funcionamiento analógico, tales como la velocidad de conmutación o la rapidez en el paso del estado de corte al estado de saturación, o viceversa.

Una señal digital está caracterizada básicamente porque la información contenida en ella está codificada en impulsos de tensión o de corriente, con lo cual se distinguen dos estados bien definidos (el carácter binario del que hacíamos mención anteriormente), denominados nivel alto –ya que existe la presencia de un determinado valor de tensión o de corriente– y nivel bajo –debido a la ausencia de tensión o de corriente– tal y como se puede ver en la figura adjunta y circunstancias en las que se suele denominar también al estado de nivel alto como "1", y al estado de nivel bajo como "0".

El estado de corte del transistor está provocado por la ausencia de una polarización de la tensión base-emisor, o que no esté polarizada lo suficiente para entrar en conducción, siendo tal situación la que impide el paso de una corriente por los terminales de salida y así se obliga a que la tensión de salida sea la tensión de alimentación; esto es, VCC, es decir, un "1" a la salida ya que tenemos tensión. El caso contrario sucede cuando la tensión de entrada polariza la tensión base-emisor y el transistor se va al estado de saturación, con lo que la tensión de salida tendrá un valor próximo a cero, siendo este valor en el mundo digital un "0".

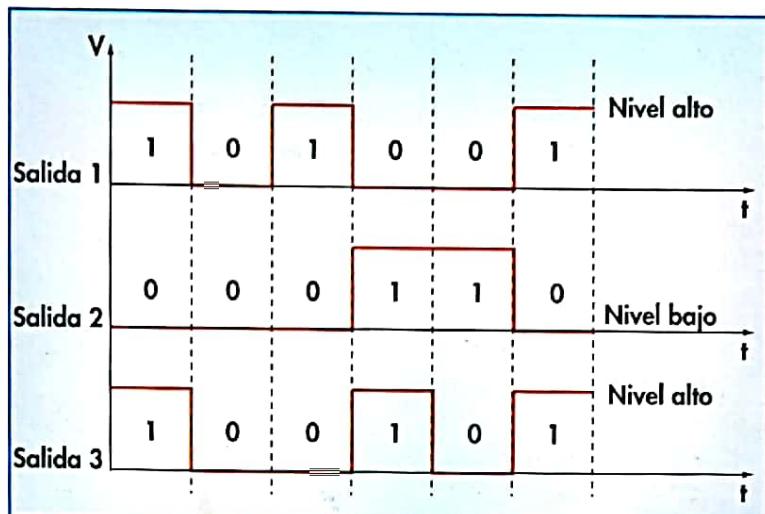
Por lo dicho en el párrafo anterior y tal como se indica en la figura, el transistor trabajando en conmutación nos permite "trabajar" de una nueva forma que simplifica la utilización de herramientas adicionales para diseñar circuitos.

Los valores de tensión

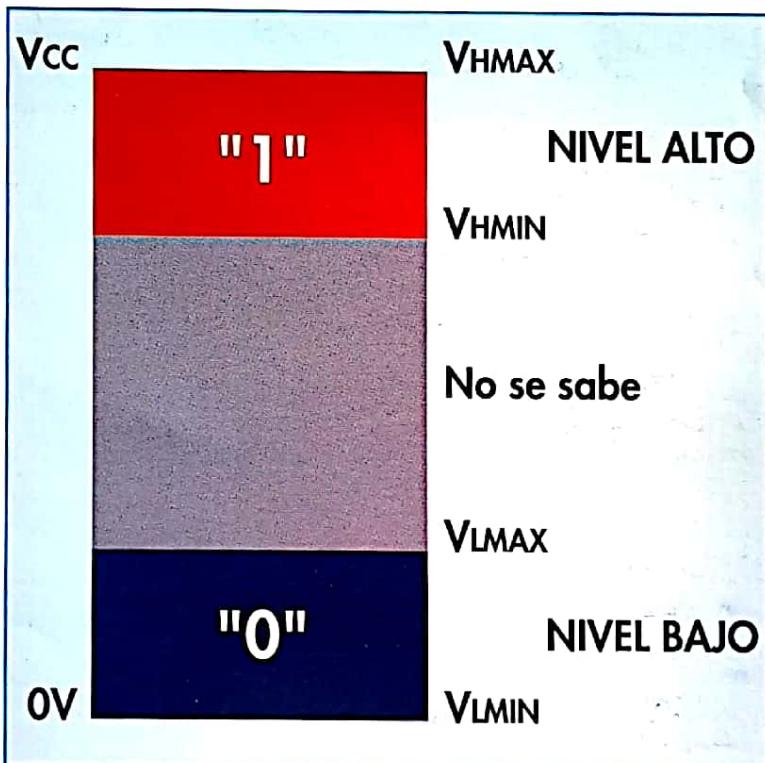
Según el circuito anterior ¿qué alimentación utilizamos? ¿Nos sirve cualquiera? La respuesta es no. En la electrónica digital tenemos los valores de alimentación bastante limitados a diversos valores, que son los que nos ofrecen los dispositivos que se utilizan dentro de esta "nueva electrónica". Independientemente del valor que se utilice, tenemos que tener en cuenta que aunque estemos simplificando los valores de la tensión o de la corriente, en definitiva son valores analógicos con lo que nos aparecerán unos márgenes de tensiones o de corrientes que nos indicarán el estado en el que nos encontramos. Estos valores son:

- V_{HMAX}
- V_{HMIN}
- V_{LMAX}
- V_{LMIN}

Los términos aparecen por los márgenes que nos encontramos dentro de las tensiones digitales. La tensión V_{HMAX} nos indica la tensión a nivel alto máximo, mientras que V_{HMIN} indica la tensión de salida a nivel alto mínimo para que lo entendamos como un "1", al contrario que con V_{LMAX} , que es la tensión máxima a nivel bajo para que tengamos un "0". El último caso es el de la tensión V_{LMIN} , que es el valor de tensión a nivel bajo

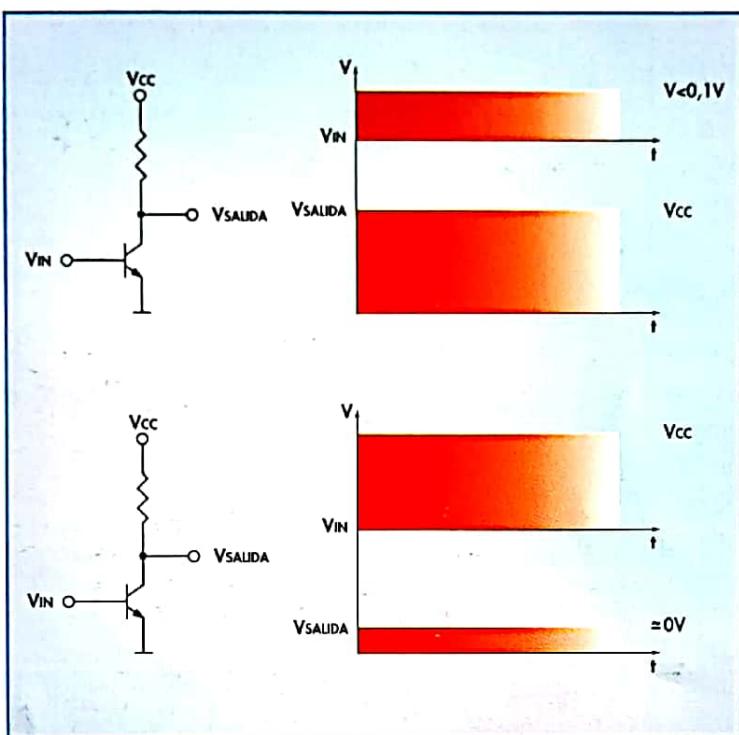


Información de distintas señales codificadas en impulsos de tensión o de corriente, distinguiéndose dos estados bien definidos, es decir, niveles altos y niveles bajos.



Márgenes de tensión que necesitamos para poder decir que tenemos un nivel alto o un nivel bajo, de manera que entre medias existe otra zona de tensiones sobre la que no debemos trabajar, ya que será el propio dispositivo el que aleatoriamente decida la salida.

mínimo. Al lector le sorprenderá el margen de tensión que aparece entre los V_{LMAX} y V_{HMIN} , según se puede observar en la figura adjunta, pero es que, simplemente, no se sabe, esto es, es el propio dispositivo el que marca la tensión de salida aleatoriamente sin que podamos obtener



En la figura de arriba, si no polarizamos la tensión base-emisor (V_{BE}), el transistor se polarizará en estado de corte, de manera que la tensión colector-emisor (V_{CE}) será la tensión de alimentación y por tanto tendremos un "1" a la salida. En cambio, en la figura de abajo la tensión de entrada es la tensión de alimentación y por tanto el transistor se polariza en saturación, con lo que a la salida tendremos un nivel bajo.

LÓGICA POSITIVA

$$V_{SALIDA} = V_{CC} \rightarrow "1"$$

$$V_{SALIDA} = 0V \rightarrow "0"$$

LÓGICA NEGATIVA

$$V_{SALIDA} \approx V_{CC} \rightarrow "0"$$

$$V_{SALIDA} \approx 0V \rightarrow "1"$$

Forma de trabajar en electrónica digital. Debemos elegir la lógica que vamos a utilizar ya que los dispositivos con los que trabajemos deberán ser coherentes con ella.

información previa. Esto nos obliga a trabajar siempre dentro de los valores prefijados para que los dispositivos de las distintas tecnologías que existen dentro de la electrónica no se vuelvan "locos" y nos tiren abajo los diseños realizados.

La lógica positiva y la lógica negativa

Tanto el análisis como la síntesis de circuitos digitales tienen su base y su origen en los estudios realizados en el campo de la lógica por el matemáti-

co inglés George Boole, y publicados en 1937 en su obra "Análisis matemático de la lógica". En sus estudios trata fundamentalmente de conferir una estructura simbólica a los procesos del pensamiento, para así poder manipular dichos símbolos mediante una estructura matemática apropiada.

Sentadas estas bases teóricas de dichas estructuras, podremos proceder a la elaboración práctica de circuitos que emulen el funcionamiento de un estructura de base lógica, con lo que toda la elaboración y diseño de circuitos digitales se reduce, en definitiva, a un sí o no lógicos, de manera que en el mundo de la electrónica se traduzca en los conceptos de ausencia/presencia de tensión o de corriente, en lo que hemos dado en llamar "1" ó "0". Esto nos obliga a realizar una distinción evidente, ya que existen dos formas de referirse a la representación de la ausencia o la presencia de la tensión o de la corriente, denominadas lógica positiva o lógica negativa. La lógica positiva atribuye a la presencia de tensión en la salida como un "1" o nivel alto, y denominar como "0" o nivel bajo a la ausencia de tensión. Sin embargo, la lógica negativa lo realiza el revés, es decir, la presencia de tensión la tomamos

como si fuera un cero, mientras que la ausencia de tensión se toma como un uno. Así, cuando se procede al diseño de un circuito digital, debaremos elegir el principio de la lógica con la que vamos a trabajar, para así poder aplicar los conceptos necesarios que nos permitan determinar los circuitos a emplear.

Parámetros de los niveles

En la presentación de los niveles digitales siempre se obvian las subidas y las bajadas reales, es

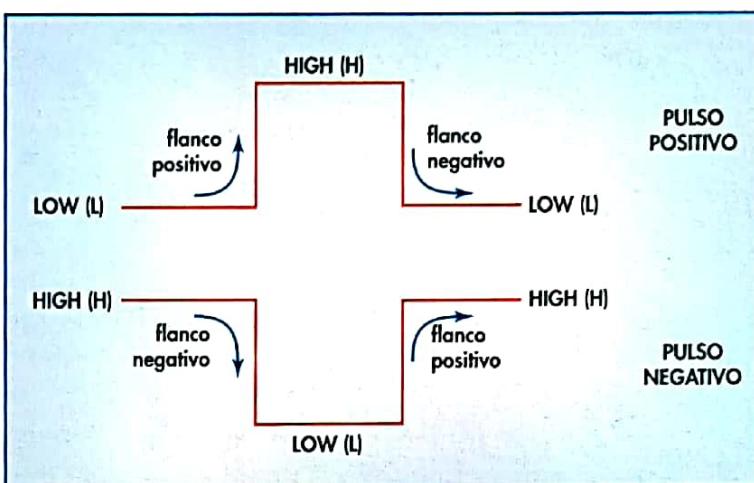
dejar, se dibujan siempre como líneas verticales que unen un estado y otro, pero esto es imposible e inviable en la realidad, ya que está basado en el concepto de que nadie puede estar en el mismo momento en dos sitios distintos. Con esto queremos decir que existen varios parámetros que en las representaciones se obvian pero que están presentes en los componentes, tal y como podemos ver en la figura que corresponde a los cambios de flancos, denominación que se le da a un cambio de nivel alto a nivel bajo y viceversa y, a su vez, teniendo en cuenta la dirección que llevamos, si es de nivel bajo a nivel alto será un flanco positivo y si es a la inversa será negativo.

Según se observa, el parámetro "Rise Time" nos indica el tiempo que tarda la señal en ascender desde un nivel bajo a un nivel alto, entendiéndose como el inicio desde el diez por ciento de la amplitud de la señal hasta el noventa por ciento de la misma, que se considera el final. En cambio, "Fall Time"

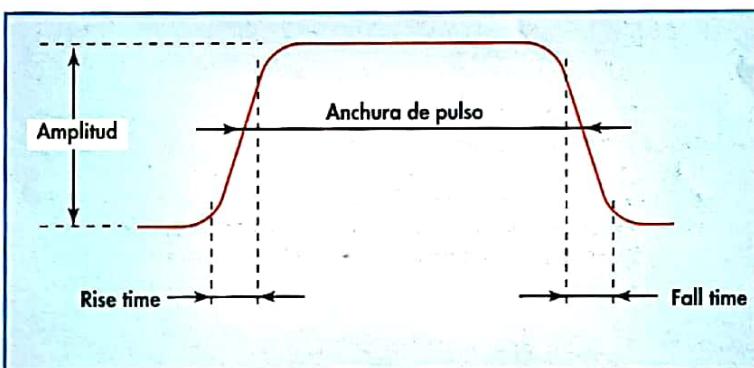
nos presenta el tiempo inverso al anterior, esto es, flanco negativo y con los mismos tiempos que el anterior. En castellano estos parámetros los podemos denominar como tiempo de subida y tiempo de bajada de la señal, además del último dato característico de la misma, la anchura de pulso o "Pulse Width", que se considera desde el 50% de la señal de subida ("Rise Time") hasta el 50% de la señal de bajada ("Fall Time").

Unidades de información

Dentro de la electrónica digital lo que se utiliza son bits, es decir, unos o ceros que nos indican el estado de una señal, siendo éste la unidad mínima de información. Cuando existen muchos bits que caracterizan el estado de una señal, lo que se realiza es una agrupación de los datos, de manera



En la figura de arriba tenemos un flanco positivo, esto es, una transición de nivel bajo a nivel alto, yendo después otra vez a nivel bajo, denominado pues como pulso positivo, mientras que en la figura de abajo tenemos el proceso contrario, un pulso negativo.

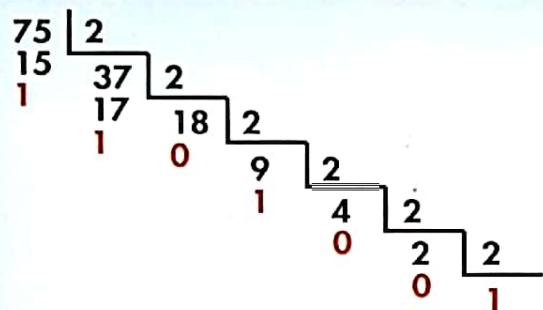


Parámetros que caracterizan a las señales digitales: "Pulse Width" o anchura del pulso, "Rise Time" o tiempo de subida y "Fall Time" o tiempo de bajada.

que dan lugar a otras representaciones que son: el nibble, que corresponde a cuatro bits, esto es, 16 combinaciones posibles que dan lugar a 16 distintas salidas de un circuito. Además, también existen el byte, correspondiente a 8 bits y el word, que son 16 bits.

Los códigos de numeración

Los datos y los números pueden ser representados de distintas maneras, siendo unas más idóneas que otras en las distintas aplicaciones que nos ofrecen. Nosotros estamos acostumbrados a utilizar el sistema decimal (base 10), que nos ofrece un código basado en la utilización de los números naturales (que son: 0, 1, 2, 3, 4, 5, 6, 7, 8 y 9). Sin embargo, existen otras muchas formas de contar que son:



$$75 \text{ DECIMAL} = 1001011 \text{ BINARIO}$$

Proceso de conversión de un número en base decimal a base binaria. Dividimos el número a convertir entre el valor de la base hasta obtener un cociente menor que la base que estamos utilizando, quedándonos entonces con los restos obtenidos para poder formar el número en la base deseada.

SISTEMA DECIMAL	SISTEMA BINARIO	SISTEMA OCTAL	SISTEMA HEXADECIMAL
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Tabla en la que se muestran las equivalencias de valores de la base decimal y la base binaria.

- **Alfabeto Octal (base 8), con los números 0, 1, 2, 3, 4, 5, 6 y 7.**
- **Alfabeto Hexadecimal (base 16), que utiliza números y letras debido a la imposibilidad de utilizar números, siendo éstos: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E y F.**
- **Alfabeto Binario (base 2) utilizando los números 0 y 1.**

El caso en el que está basado la electrónica digital es el binario, de manera que tenemos dos posibilidades para un mismo dato, es decir, el cero y el uno.

Todos estos códigos son ponderados, esto es, cada dígito depende de la posición en la que esté para tener un valor u otro más importante, pero para comprenderlo mejor, pongamos un ejemplo:

$$567,32 = 5 \times 10^2 + 6 \times 10^1 + 7 \times 10^0 + \\ + 3 \times 10^{-1} + 2 \times 10^{-2}$$

El cinco está en el tercer lugar empezando desde la coma, de manera que el lugar determina su valor final y así con todos los demás valores. Imaginemos que necesitamos convertir un valor decimal a un binario, cosa bastante probable cuando estamos en la electrónica digital, el proceso consiste en dividir el número decimal entre dos, hasta que obtenemos un cociente de valor menor a la base a la que queramos convertirlo, proceso que se encuentra desarrollado en la imagen adjunta y que se refleja a continuación:

$$75 \text{ DECIMAL} = 1001011 \text{ BINARIO}$$

y cuya comprobación consiste en realizar el proceso inverso, teniendo en cuenta la posición de cada uno de los valores digitales y la base en la que nos encontramos:

$$1 \times 2^6 + 0 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + \\ + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 75$$

De esta forma se pueden realizar distintas equivalencias entre los distintos códigos de numeración que se utilizan, tal y como se puede observar en la tabla adjunta, re-

presentándose los 15 primeros números decimales en los distintos códigos habituales, que son el binario, el octal y el hexadecimal.

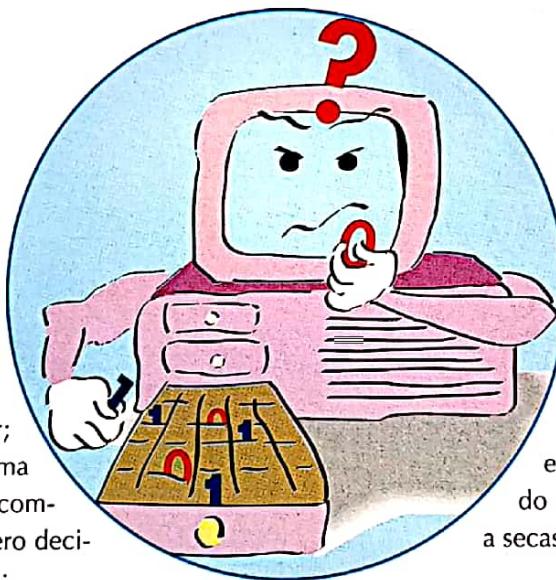
El proceso de conversión de cualquier base a otra distinta consiste en la división del número a convertir entre la base final, tal y como lo hemos desarrollado, hasta obtener un cociente en la división menor a la base por la que se está dividiendo. Mientras que el proceso inverso implica la multiplicación de todos los valores que forman el dígito por la base en la que está y elevado al exponente de la posición que ocupa, debido a la ponderación de los códigos.

LOS NÚMEROS BINARIOS

Los números binarios puros son difíciles de comprender; como ya vimos en la última explicación, es bastante complicado procesar un número decimal en binario y viceversa:

$$75_{\text{DECIMAL}} = 1001011_{\text{BINARIO}}$$

Por ello, dentro de los códigos binarios existen diversos códigos BCD (decimal codificado en binario, en castellano) que realizan la conversión mucho más fácil, como se puede comprobar en la figura adjunta, en la que se muestra el código BCD de cuatro bits para los dígitos decimales del 0 al 9, teniendo en cuenta que es un código ponderado, es decir, el bit más significativo (MSB, en inglés) tiene un peso 8 (2^3), y el menos significativo 1 (2^0). Así, a este código se le denomina BCD 8421, aunque existen otros códigos ponderados



que utilizan otro tipo de pesos, como se puede ver en la siguiente figura. Un ejemplo de cómo sería el número 75_{DECIMAL} en BCD 8421, denominado comúnmente como BCD a secas:

$$75_{\text{DECIMAL}} = 0111\ 0101_{\text{BCD}}$$

También existen otros códigos que no tienen peso, esto es, cada bit no tiene un peso especial y por tanto obedecen a otro tipo de criterios, tales como el exceso a 3 y el código Gray, que se pueden contemplar en la figura.

El código exceso a 3 (XS3) está relacionado con el código BCD 8421 por su naturaleza codificada en binario, de forma que sea el mismo número al que se le suma siempre 3, mientras que el código Gray no tiene nada que ver con el BCD porque cada incremento de un número en decimal corresponde solamente a un cambio de estado en

DECIMAL	BCD 8421
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Tabla de equivalencias entre la base decimal y el código ponderado BCD 8421, en el que se ha de tener en cuenta el peso que existe en la posición del bit, por ejemplo, 8=23, siendo tres la posición desde la derecha del dato.

SISTEMA DECIMAL	BCD 8421		BCD 4221		BCD 5421	
	8421	8421	4221	4221	5421	5421
0	---	0000	---	0000	---	0000
1	---	0001	---	0001	---	0001
2	---	0010	---	0010	---	0010
3	---	0011	---	0011	---	0011
4	---	0100	---	0000	---	0100
5	---	0101	---	0111	---	1000
6	---	0110	---	0100	---	1001
7	---	0111	---	0101	---	1010
8	---	1000	---	0110	---	1011
9	---	1001	---	0111	---	1100
10	0001	0000	0001	0000	0001	0000
11	0001	0001	0001	0001	0001	0001
12	0001	0010	0001	0010	0001	0010
13	0001	0011	0001	0011	0001	0011

Equivalencias entre otros códigos ponderados, como son el BCD 4221 y el BCD 5421, además del ya visto BCD 8421.

un bit, siendo este cambio bastante importante en ciertas aplicaciones de la electrónica digital.

OPERACIONES EN BINARIO

Igual que ocurre cuando operamos con los números decimales, en binario también se pueden realizar operaciones matemáticas, tales como la suma, resta, multiplicación y división, aunque esta última apenas se utiliza.

Para comprender la lógica de las operaciones, vamos a utilizar unos ejemplos para cada una de ellas:

Suma

Debemos tener en cuenta que:

$$0 + 0 = 0$$

$$0 + 1 = 1$$

$$1 + 0 = 1$$

$1 + 1 = 0$ y me llevo 1 (denominado carry)

Tenemos el ejemplo adjunto en la imagen, en la que sumamos cinco más tres.

DECIMAL	EXCESO A 3	CÓDIGO GRAY	DECIMAL	COMPLEMENTO A 1
0	---	0000	-7	1000
1	---	0001	-6	1001
2	---	0011	-5	1010
3	---	0010	-4	1011
4	---	0110	-3	1100
5	---	0100	-2	1101
6	---	0101	-1	1110
7	---	0111	0	1111-0000
8	---	1000	1	0001
9	---	1001	2	0010
10	0100	0011	3	0011
11	0100	0100	4	0100
12	0100	0101	5	0101
13	0100	0110	6	0110
14	0100	0111	7	0111
15	0100	1000		

Se invierten los unos por los ceros y los ceros por los unos

Otros códigos no ponderados son el Exceso a 3 (XS3) y el código Gray, que ofrece distintas utilidades para ciertas aplicaciones de la electrónica digital, como puede ser la transmisión de datos por redes telemáticas.

Para representar números negativos en lógica binaria se utiliza el Complemento a 1 (C1), que tiene la problemática de que posee dos representaciones para el número decimal cero.

Resta

Para esta operación tenemos que:

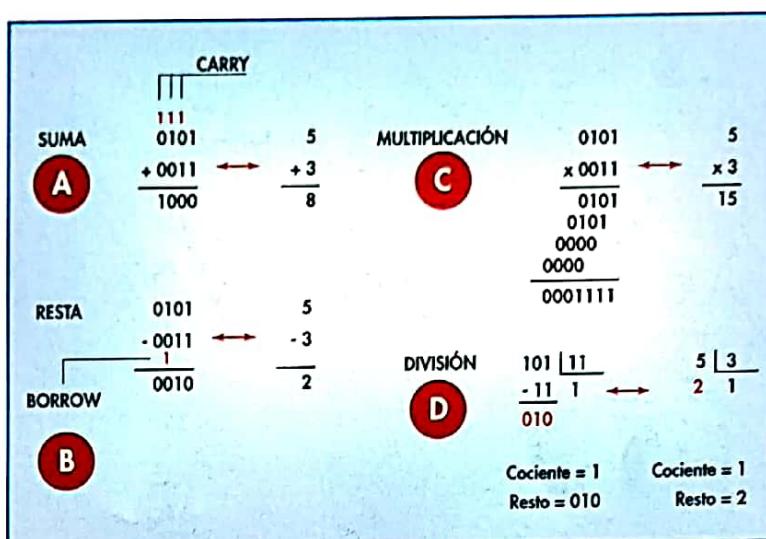
$$0 - 0 = 0$$

$$1 - 0 = 1$$

$0 - 1 = 1$ y de llevada 1
(denominado borrow)

$$1 - 1 = 0$$

El ejemplo lo tenemos en la siguiente figura, en la que se realiza la resta de cinco menos tres.



Multiplicación

Tenemos:

$$0 \times 0 = 0$$

$$0 \times 1 = 0$$

$$1 \times 0 = 0$$

$$1 \times 1 = 1$$

En el ejemplo de esta operación, utilizamos los mismos números anteriores, es decir, multiplicamos cinco por tres.

División

Aunque esta operación no es muy común, la tenemos realizada junto a las otras tres, en la misma figura.

Los números negativos

Para expresar los números negativos en binario, aparecieron lo que se ha dado en llamar el Complemento a 1 (C1) y el Complemento a 2 (C2).

El Complemento a 1 utiliza el bit de mayor peso (MSB) para indicarnos el signo, de manera que perdemos un bit de codificación, es decir, si con 4 bits teníamos 16 combinaciones, ahora en C1, tendremos sólo 8 combinaciones para 4 bits, como se puede ver en la tabla adjunta en la que, además, podremos comprobar la problemática añadida de esta representación: tenemos dos formas de indicar el cero, lo cual

DECIMAL	COMPLEMENTO A 2
-7	1001
-6	1010
-5	1011
-4	1100
-3	1101
-2	1110
-1	1111
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111

Diagram illustrating the conversion of decimal numbers to 4-bit binary representations using the 2's complement method. It shows the addition of 0101 (5) and 1011 (5) to get 1010 (6). The result 1010 is then converted back to decimal as 0101 + 1 = -5.

Para representar números negativos se utiliza de una forma más sencilla el Complemento a 2, que resuelve el problema de la doble representación para el número cero. También se indica el proceso de obtención de un número a Complemento a 2, convirtiendo los unos en ceros y viceversa, además de sumarle 1 al resultado obtenido.

nos impide la realización de aplicaciones de una manera adecuada.

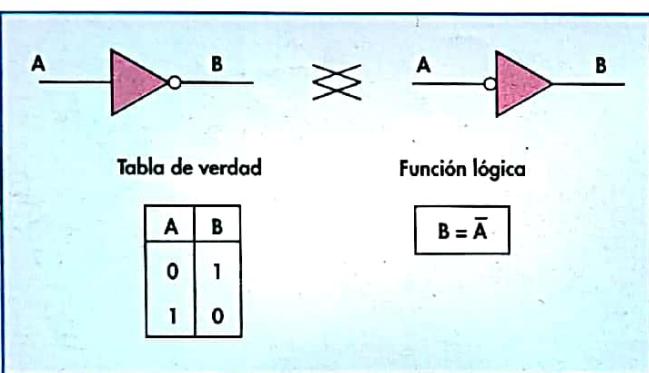
$$\begin{aligned} \text{MSB} = 0 &\Rightarrow \text{número positivo} \\ \text{MSB} = 1 &\Rightarrow \text{número negativo} \end{aligned}$$

Para convertir un número binario a Complemento a 1 (C1) sólo tenemos que observar detenidamente

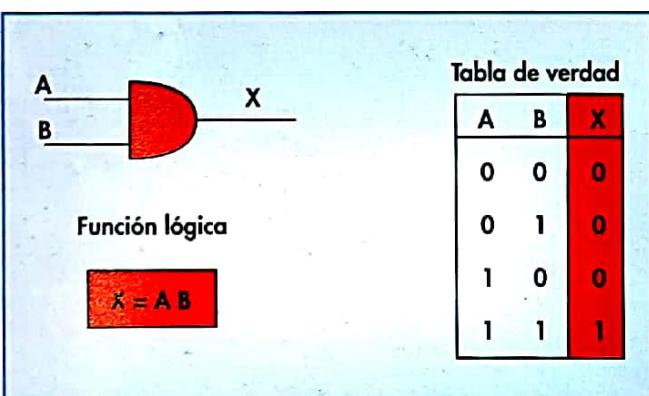
$$\begin{array}{r}
 5 \\
 -3 \\
 \hline
 2
 \end{array}
 \quad
 \begin{array}{r}
 5 \\
 +(-3) \\
 \hline
 2
 \end{array}
 \quad
 \begin{array}{r}
 1\ 1 \\
 0101 \\
 +1101 \\
 \hline
 \cancel{X}0010
 \end{array}$$

¡¡ MÁS SENCILLO !!

El proceso de restar dos números es más sencillo si utilizamos el sustraendo en Complemento a 2, ya que sólo tendremos que realizar una suma.



Puerta lógica NOT junto con su tabla de verdad y su función lógica.



La puerta lógica AND junto con su tabla de verdad y su lógica de funcionamiento.

te la tabla de la que hablamos: los números positivos están en binario natural, mientras que los números negativos sólo tenemos que obtenerlos dando la vuelta a los números positivos, es decir, donde tengamos un cero poner un uno y donde había un uno colocar un cero, y así obtendremos los números negativos.

Sin embargo, el Complemento a 2 (C2), ya no tiene el problema de la doble representación para el cero, lo cual nos ayuda a utilizar los números negativos en operaciones tales como la resta y la suma.

Para obtener el Complemento a 2 de un número sólo tenemos que obtener el complemento a 1 (C1) del número y después sumarle uno, tal y como se puede observar en la tabla. Aunque al lector le pueda resultar extraño la utilización del C2, es de bastante utilidad a la hora de sumar, restar y multiplicar números negativos.

LAS PUERTAS LÓGICAS

Hasta ahora todo lo que hemos visto sólo nos ayuda a comprender la lógica de las operaciones en binario, así como su representación de distintas formas, lo cual hace que nos estemos preparando el camino para lo que viene ahora.

En electrónica digital, tanto las entradas como las salidas de los circuitos siempre están representadas a través de letras o letras con subíndices para indicar si se trata de líneas sueltas (esto es, no tienen nada que ver entre ellas pero son entradas comunes a un circuito) o para indicar que es un bus de datos (es decir, en el supuesto que tengamos bytes o word). Vamos a verlo de una forma más clara mediante el estudio de lo que se han llamado las puertas lógicas y sus posibles estados

La puerta NOT

La representación esquemática de esta puerta se encuentra en la figura adjunta, en la que se puede observar que tiene dos formas distintas de indicarla, ya sea con el "circulito" a la salida o el triángulo a la entrada. En matemática discreta una puerta NOT nos indica la negación del

estado actual que tengamos a la entrada, dicho de otra forma, le damos la vuelta al estado y ponemos el contrario. En castellano esta puerta se denomina más fácilmente como "negador" o como "inversor".

Supongamos que a la entrada del negador, tal y como está en la figura de la que estamos hablan-

do, tenemos un estado A a la entrada del inversor, de manera que desconozcamos qué nivel lógico es, ya sea un nivel alto o un nivel bajo. La función de esta "puerta lógica", es la de darnos a su salida el nivel contrario, esto es, el complemento o la negación de A, expresado en lenguaje lógico como \bar{A} . Por tanto, si observamos la "tabla de verdad" (tabla en la que se indican todos los estados posibles del circuito, tanto sus entradas como sus salidas), obtendremos la tabla que tenemos a su izquierda, en la que a la salida siempre tenemos el contrario de lo que tengamos a la entrada.

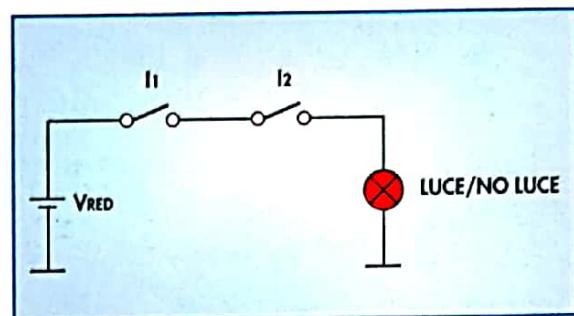
Si colocásemos en serie dos puertas negadoras, obtendríamos el mismo valor de la señal que teníamos ya que, al pasar por la primera puerta, obtendríamos el valor complementario, esto es, dicho en lenguaje digital, obtendríamos "A negado", mientras que después de la segunda puerta, volveríamos a tener A.

La puerta AND

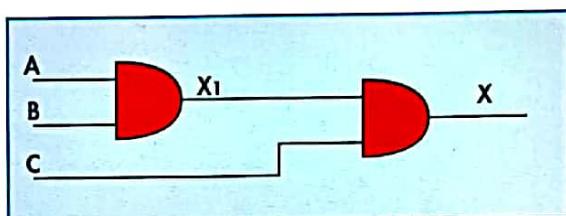
La traducción literal al castellano es "y", lo cual puede darnos una idea de la función que realiza esta puerta lógica, cuya representación se encuentra en la figura adjunta. Realiza el producto lógico de las señales de entrada, denominadas en el gráfico como A y B, con lo que obtendremos la tabla de verdad de una manera sencilla, igual que sucede con los números decimales, siempre que tengamos un cero a la entrada tendremos un cero a la salida, con lo que será necesario tener unos a la entrada para obtener un cero a la salida, como se puede comprobar.

Un símil de funcionamiento lo podemos encontrar en dos interruptores que estén en serie y que iluminen una bombilla, de forma que para que luzca la misma tendrán que estar los dos interruptores cerrados (tener dos unos), ya que en caso contrario la bombilla no lucirá (en el momento en que una de las dos entradas sea cero, o las dos).

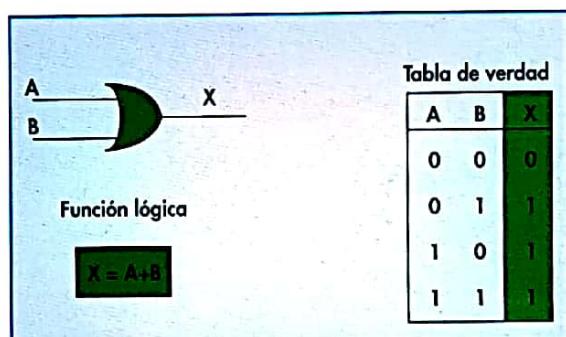
Si necesitásemos hacer una multiplicación de tres variables, supongamos A, B y C, podríamos realizar perfectamente el producto lógico entre dos de ellas, por ejemplo, A y B, mientras que su salida se conectaría directamente a otra puerta AND en donde estuviera conectada en la otra entrada la señal C, y así sucesivamente todas las veces que



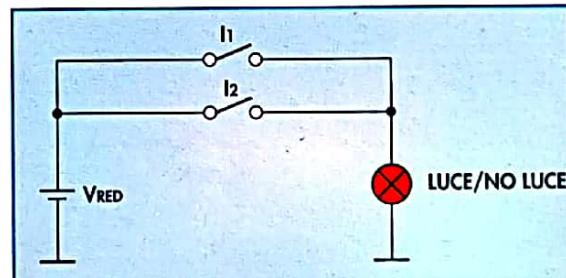
El símil de una puerta AND se trata de dos interruptores en serie, ya que para que luzca la bombilla es necesario cerrar los interruptores. En la puerta AND, para tener un uno a la salida, es necesario que las entradas tengan todos un uno.



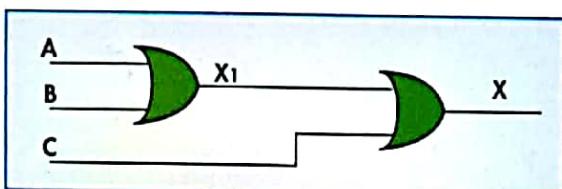
Si tenemos que realizar una triple multiplicación $X=ABC$, podemos realizar con puertas AND de dos entradas, de manera que tengamos que utilizar dos puertas lógicas para realizar la operación.



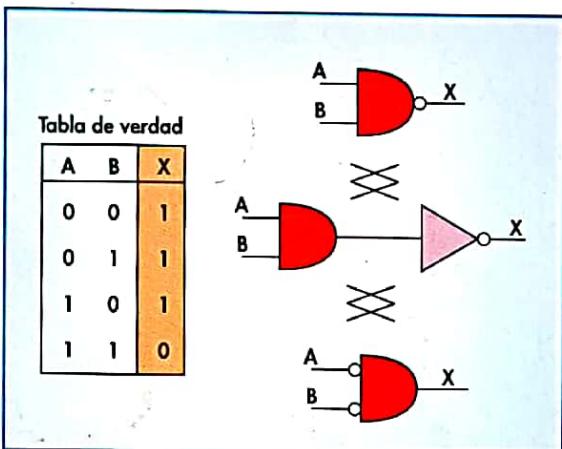
Puerta lógica OR, que realiza una suma lógica. También se indica su tabla de verdad y su función lógica.



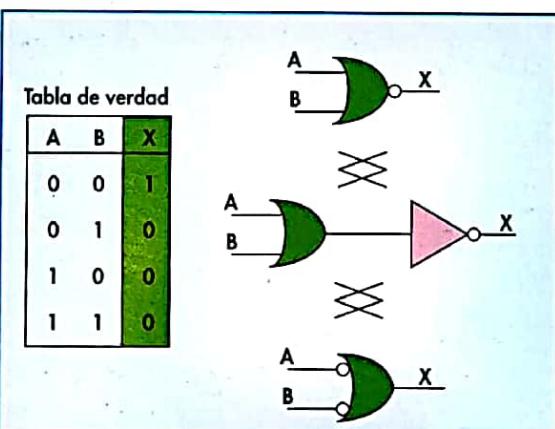
En este caso el símil de la puerta OR son dos interruptores en paralelo, ya que cualquiera de los dos enciende la bombilla. Utilizando una puerta OR, podremos obtener un uno a la salida si cualquiera de las dos entradas, o las dos, tienen un uno.



Si la operación a realizar es $X = A + B + C$, lo podemos realizar utilizando dos puertas OR conectándolas como aparece.



La puerta NAND consiste en una puerta AND y en serie una puerta NOT que niegue el resultado de la primera puerta. También podemos encontrar otras formas de representarla. Se ofrece su tabla de verdad.



En este caso, la puerta NOR, es una puerta OR y en serie otra NOT para negar el resultado, pudiéndose encontrar gráficamente de las formas que aparece.

Se ofrece también su tabla de verdad.

sean necesarias, como podemos ver en la siguiente figura adjunta.

La puerta OR

Si la puerta anterior realizaba el producto lógico, esta otra, al contrario, realiza la suma lógica, teniendo en cuenta su tabla de verdad que se

puede comprobar en la figura y en la que, además, se puede ver su esquema gráfico.

El símil de funcionamiento de esta puerta consiste en conectar en paralelo dos interruptores, de manera que la luz podrá lucir cuando uno de los dos, o los dos, estén cerrados, esto es, si tenemos un uno en cualquiera de las dos entradas, o incluso en las dos, obtendremos un uno a la salida, como se puede comprobar en el esquema eléctrico del símil explicado.

También podemos realizar la misma conexión anterior en la que hablábamos de tres señales A, B y C, pudiendo conectarlas de la misma forma para realizar la suma lógica de las tres variables de estado y así todas las veces que sean necesarias.

La puerta NAND

Si tenemos en cuenta el funcionamiento de la puerta AND y de una puerta NOT, entenderemos el funcionamiento de una NAND, ya que consiste en la conexión en serie de una puerta AND y a continuación una puerta NOT, con lo que obtendremos, a su vez, una tabla de verdad de la puerta lógica totalmente al contrario que la primera, esto es, si antes teníamos que tener dos unos a la entrada de la puerta AND para que ésta a su salida nos diera un uno, ahora, es necesario tener al menos un cero a la entrada para obtener un uno a la salida de la misma. El esquema de conexión de esta puerta lo tenemos en la figura adjunta, en la que se observa, además, el esquema gráfico de la misma y el "circulito" que caracteriza la negación de las salidas.

La puerta NOR

Si la puerta NAND consistía en la conexión en serie primero de una puerta AND y después una puerta NOT, en este caso la conexión consiste en colocar primero una puerta OR y después la misma puerta NOT, con lo que obtendremos el funcionamiento contrario a la puerta OR. Cuando el resultado de la suma lógica nos dé un "1", el negador nos dará un "0", de manera que la única forma de obtener a la salida un nivel alto es la de tener a la entrada dos niveles bajos de tensión, esto es, dos ceros.

En la figura adjunta se encuentra su tabla de verdad, así como su correspondiente diagrama electrónico.

La puerta XOR

Una vez vistas todas las puertas lógicas básicas que podemos encontrar en el mercado de la electrónica digital, falta por ver dos combinaciones lógicas bastante interesantes y que están denominadas como XOR y XNOR. La primera de ellas, la puerta XOR, también denominada puerta OR-EXCLUSIVA, de la que se puede ver su símbolo eléctrico en la imagen adjunta, nos permite conocer cuándo tenemos un uno en cualquiera de las dos entradas posibles, en su configuración más básica. La tabla de verdad de la puerta lógica nos lo indica, pero también podemos comprobarlo a través de su función:

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

Siendo otra de sus representaciones:

$$F = A \oplus B$$

Suponiendo que tenemos un nivel alto en las dos entradas, es decir, un "1" en A y en B, la salida será:

$$F = 1 \bar{1} + \bar{1} 1$$

$$F = 0 + 0$$

$$F = 0$$

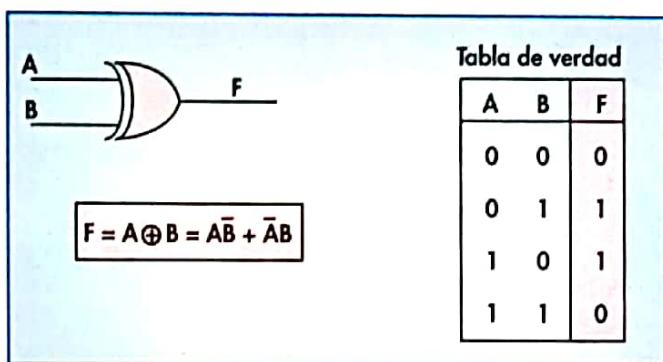
Si las dos entradas tuvieran también un nivel lógico bajo a su entrada, también nos sucedería lo mismo, mientras que si las dos tuvieran un estado contradictorio, esto es, una un nivel alto y la otra un nivel bajo, independientemente de la entrada, nos ocurriría:

$$F = 1 \bar{0} + \bar{1} 0$$

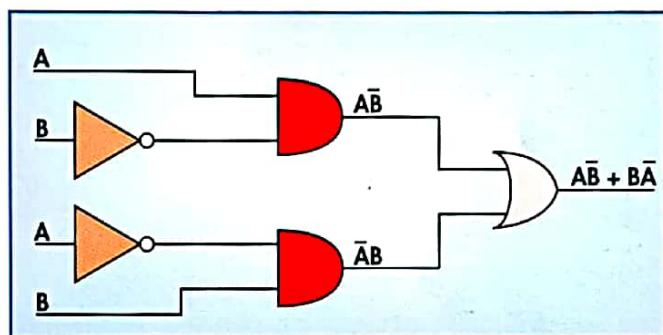
$$F = 1 + 0$$

$$F = 1$$

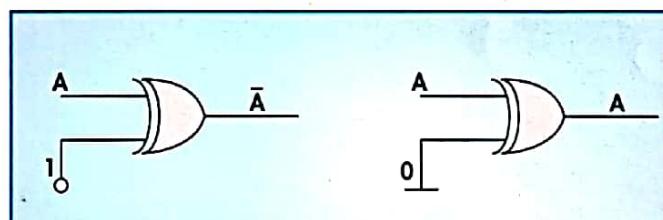
Lo que nos permite conocer cuándo dos entradas tienen un nivel lógico distinto y utilizarlo en determinadas aplicaciones, que serán comentadas más adelante cuando se avance más en el temario.



Símbolo de la puerta X-OR, también conocida como OR-EXCLUSIVA, teniendo como función la vista en su tabla de verdad, indicándonos con un uno a la salida cuando los valores de las entradas son contrapuestos, y con un cero a la salida cuando las dos entradas tienen el mismo valor.



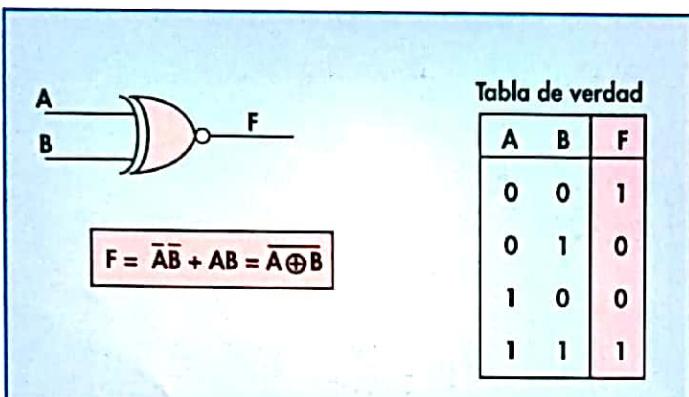
Forma de construir una puerta OR-EXCLUSIVA a base de puertas lógicas básicas, dos negadoras, dos AND's y dos OR's.



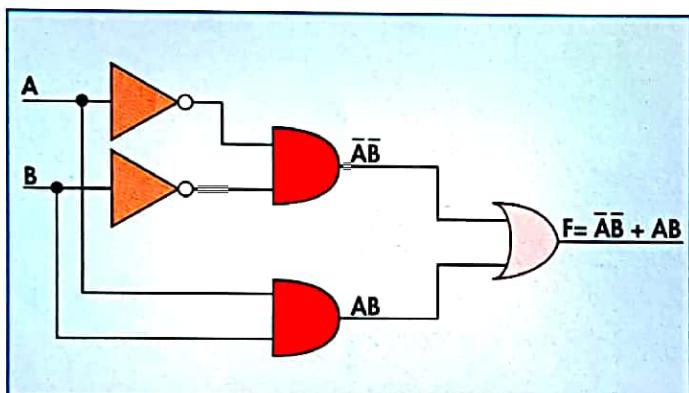
Las puertas X-OR también pueden realizar negaciones con que lleven uno de los dos pines de entrada a un uno lógico, mientras que si lo llevamos a cero, obtendremos a la salida el mismo valor de la entrada.

Esta puerta también puede ser creada como combinación de otras, tal y como se puede comprobar viendo su función, siendo por tanto formado por dos negadoras (NOT), dos multiplicadoras (AND) y una sumadora (OR), como se puede ver en la figura adjunta.

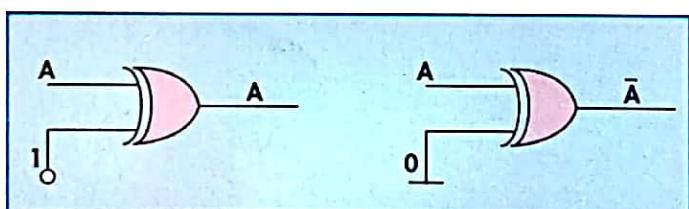
A su vez, la puerta XOR también nos permite otra aplicación bastante interesante y que vamos a desarrollar a continuación: si observamos la figura siguiente, podremos comprobar que la



Símbolo de otra puerta no básica, la X-NOR, conocida también como puerta NOR-EXCLUSIVA. Su tabla de verdad, al contrario de lo que sucedía con la puerta X-OR, nos indica con un uno a la salida cuando los dos valores de los pines de entrada son iguales, y con un cero cuando son distintos.



Utilizando puertas básicas también podemos crear una puerta NOR-EXCLUSIVA, pero colocándolas de distinta forma con respecto a la X-OR, cambiando una única puerta NOT de lugar.



La puerta NOR-EXCLUSIVA también nos sirve para realizar negaciones con unir uno de los dos pines de entrada a masa, al contrario de lo que sucedía con la puerta X-OR.

OR-EXCLUSIVA nos sirve también como negadora, siempre y cuando utilicemos uno de los dos pines a un nivel lógico 1:

$$\begin{aligned} F &= A \bar{1} + \bar{A} 1 \\ F &= 0 + \bar{A} \\ F &= \bar{A} \end{aligned}$$

Mientras que si ese mismo pin lo colocamos a masa el resultado será el contrario:

$$\begin{aligned} F &= A \bar{0} + \bar{A} 0 \\ F &= A + 0 \\ F &= A \end{aligned}$$

La puerta XNOR

En este caso, la puerta NOR-EXCLUSIVA realiza una función parecida a la anterior, es decir, nos permite conocer cuándo dos pines tienen el mismo nivel lógico, ya sea un nivel alto o un nivel bajo. La puerta XNOR nos dará a la salida un uno cuando tenga el mismo nivel en las dos entradas, mientras que tendremos un cero cuando tenga un estado distinto, siendo el caso contrario al que nos proporcionaba la puerta XOR. La función lógica que realiza esta puerta es:

$$F = \bar{A} \bar{B} + AB$$

Otra de sus representaciones es la siguiente:

$$F = \overline{A \oplus B}$$

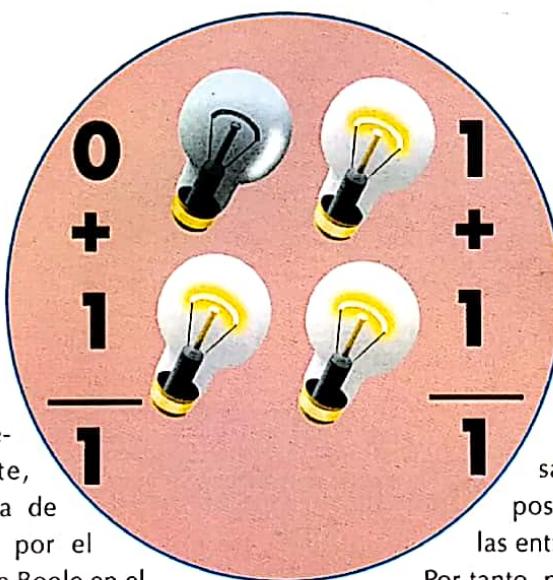
Igual que con la puerta XOR, también podemos realizar la XNOR a través de puertas lógicas más básicas, como son dos negadoras (NOT), dos multiplicadoras (AND) y una sumadora (OR), pero en este caso sólo tendríamos que conectarlas de una forma distinta a la anterior, para así formar la combinación necesaria.

También la puerta XNOR puede funcionar como negadora, tal y como se puede ver en la figura adjunta, teniendo un comportamiento contrario al de

la puerta XOR, de manera que cuando conectamos uno de los dos pines a un nivel lógico bajo obtendremos a la salida el valor complementario (comúnmente denominado negado) de la otra entrada, mientras que si el mismo pin lo tenemos a un nivel alto, a la salida siempre tendremos el mismo valor de la entrada.

EL ÁLGEBRA DE BOOLE

Como ya comentamos anteriormente, el álgebra de Boole fue desarrollado por el matemático inglés George Boole en el año 1937 dentro de su obra Análisis matemático de la lógica, basado en la naturaleza binaria de los elementos, teniendo en cuenta que pueden adoptar el valor cero ("0"), o el valor uno ("1"), pudiéndose además relacionar por tres operaciones básicas que ya hemos desarrollado, es decir, la suma lógica (la puerta OR), el producto lógico (la puerta AND) y la negación o el complemento (la puerta NOT). La relación de las variables lógicas con las operaciones vistas ya del álgebra de Boole recibe el nombre de función lógica o también función Booleana, utilizándose en electrónica digi-



tal, como ya hemos visto, para representar el estado de las salidas o salida dependiendo de los posibles estados que tengan las entradas o entrada.

Por tanto, para representar numéricamente una función que desarrolla un circuito, o que lo desarrollará, se emplean las tablas de verdad, concepto visto para el desarrollo y explicación de las puertas lógicas, que ahora vamos a ampliar. La tabla de verdad nos indica el valor lógico que adoptará una salida para cada posible combinación de las variables de entrada, con lo que las tablas de verdad constarán de tantas columnas como variables de entrada y salida tenga la función, y tantas filas como posibles combinaciones puedan formarse con las variables de entrada, resultado bastante fácil de obtener teniendo en cuenta que si tenemos

Postulados del Álgebra de Boole

$$A+1=1$$

$$A+0=A$$

$$\bar{\bar{A}}=A$$

$$A \cdot 1 = A$$

$$A \cdot 0 = 0$$

$$A \cdot \bar{A} = 0$$

$$A \cdot A = A$$

$$A + \bar{A} = 1$$

$$A + A = A$$

Propiedades del Álgebra de Boole

Comutativa

$$A+B=B+A$$

$$A \cdot B = B \cdot A$$

Asociativa

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

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

Distributiva

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

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

Teoremas del Álgebra de Boole

Ley de absorción

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

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

Ley de Morgan

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

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

Resumen de todas las propiedades, postulados y teoremas incluidos en el Álgebra de Boole y que nos sirve para simplificar al máximo las funciones a realizar en los distintos diseños electrónicos.

N variables de entrada, tendremos 2^N posibles salidas, repartiéndose dentro de la tabla de la siguiente forma: las entradas aparecerán en la parte izquierda de la misma, mientras que las salidas estarán a la derecha, como podemos

comprobar en la tabla adjunta, mostrándonos también la relación de nombres que se les dan a las entradas y a las salidas.

Los postulados del álgebra de Boole son:

- La suma de una variable A más 1 es siempre igual a 1:

FUNCIÓN:

$$A + 1 = 1$$

DEMOSTRACIÓN:

$$0 + 1 = 1$$

$$1 + 1 = 1$$

- La suma de una variable A más cero es igual a la propia variable:

FUNCIÓN:

$$A + 0 = A$$

DEMOSTRACIÓN:

$$0 + 0 = 0$$

$$1 + 0 = 1$$

- El producto de una variable A por 1 es igual al valor de la variable:

FUNCIÓN:

$$A \cdot 1 = A$$

DEMOSTRACIÓN:

$$0 \cdot 1 = 0$$

$$1 \cdot 1 = 1$$

- El producto de una variable A por cero es siempre cero:

FUNCIÓN:

$$A \cdot 0 = 0$$

DEMOSTRACIÓN:

$$0 \cdot 0 = 0$$

$$1 \cdot 0 = 0$$

- La suma de una variable A con ella misma es la propia variable:

FUNCIÓN:

$$A + A = A$$

DEMOSTRACIÓN:

$$0 + 0 = 0$$

$$1 + 1 = 1$$

- El producto de una variable A por sí misma es dicha variable:

FUNCIÓN:

$$A \times A = A$$

DEMOSTRACIÓN:

$$0 \times 0 = 0$$

$$1 \times 1 = 1$$

- La suma de una variable A con su complemento es siempre 1:

FUNCIÓN:

$$A + \bar{A} = 1$$

DEMOSTRACIÓN:

$$0 + \bar{0} = 0 + 1 = 1$$

$$1 + \bar{1} = 1 + 0 = 1$$

- El producto de una variable A con su complemento es siempre igual a 0:

FUNCIÓN:

$$A \times \bar{A} = 0$$

DEMOSTRACIÓN:

$$0 \times \bar{0} = 0 \times 1 = 0$$

$$1 \times \bar{1} = 1 \times 0 = 0$$

- Una variable A negada dos veces es siempre igual a A:

FUNCIÓN:

$$\bar{\bar{A}} = A$$

DEMOSTRACIÓN:

$$\bar{\bar{0}} = \bar{1} = 0$$

$$\bar{\bar{1}} = \bar{0} = 1$$

A continuación plantearemos las propiedades que adquieren los elementos del álgebra de Boole:

- Propiedad comutativa de una suma lógica:

FUNCIÓN:

$$A + B = B + A$$

DEMOSTRACIÓN:

$$0 + 1 = 1 + 0$$

- Propiedad comutativa del producto lógico:

FUNCIÓN:

$$A \times B = B \times A$$

DEMOSTRACIÓN:

$$0 \times 1 = 1 \times 0$$

- Propiedad asociativa de la suma lógica:

AB	S	ABC	S	ABCD	S
00	0	000	0	0000	0
01	1	001	1	0001	1
10	0	010	0	0010	0
11	1	011	1	0011	1
		100	0	0100	0
		101	1	0101	1
		110	0	0110	0
		111	1	0111	1
				1000	0
				1001	1
				1010	0
				1011	1
				1100	0
				1101	1
				1110	0
				1111	1

Diferentes tipos de tablas de verdad de distintos circuitos, en los que se puede observar que las posibles salidas están directamente relacionadas con el número de entradas, esto es, $2N$ salidas con N entradas.

FUNCIÓN:

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

DEMOSTRACIÓN:

$$1 + 0 + 1 = 1 + (0 + 1)$$

- Propiedad asociativa del producto lógico:

FUNCIÓN:

$$A \times B \times C = A \times (B \times C)$$

DEMOSTRACIÓN:

$$0 \times 1 \times 0 = 0 \times (1 \times 0)$$

- Propiedad distributiva de la suma lógica:

FUNCIÓN:

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

DEMOSTRACIÓN:

$$1 \times (0 + 1) = (1 \times 0) + (1 \times 1)$$

- Propiedad distributiva del producto lógico:

FUNCIÓN:

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

DEMOSTRACIÓN:

$$1 + (0 \times 1) = (1 + 0) \times (1 + 1)$$

Una vez finalizadas las propiedades, pasamos a explicar los teoremas de dicho álgebra, siendo en concreto dos, la Ley de Absorción y las Leyes de Morgan:

- Ley de la Absorción de la suma lógica:

FUNCIÓN:

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

DEMOSTRACIÓN:

$$A + (A \times B) = A \times (1 + B) = A \times 1 = A$$

- Ley de la Absorción del producto lógico:

FUNCIÓN:

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

DEMOSTRACIÓN:

$$\begin{aligned} A \times (A + B) &= (A \times A) + (A \times B) = \\ &= A + (A \times B) \end{aligned}$$

Si aplicamos a este resultado la Ley de la Absorción de la suma lógica, obtenemos lo mismo que antes:

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

- Ley de Morgan para la suma lógica:

FUNCIÓN:

$$\overline{A + B} = \bar{A} \times \bar{B}$$

DEMOSTRACIÓN:

$$\overline{0 + 1} = \bar{1} \times \bar{0}$$

- Ley de Morgan para el producto lógico:

FUNCIÓN:

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

DEMOSTRACIÓN:

$$\overline{0 1} = \bar{1} + \bar{0}$$



Existen multitud de circuitos integrados con diferentes lógicas que nos permiten realizar todo tipo de funciones y que serán vistos más adelante dentro de las características de las familias digitales de circuitos integrados.

ABCDE	S	ABCDE	S
00000	0	10000	0
00001	1	10001	1
00010	0	10010	0
00011	1	10011	1
00100	0	10100	0
00101	1	10101	1
00110	0	10110	0
00111	1	10111	1
01000	0	11000	0
01001	1	11001	1
01010	0	11010	0
01011	1	11011	1
01100	0	11100	0
01101	1	11101	1
01110	0	11110	0
01111	1	11111	1

5 entradas
2⁵ posibles salidas

DIFÍCIL SIMPLIFICACIÓN

Cuantas más entradas tenga el circuito más difícil será realizar la simplificación, por lo que existen métodos alternativos para la simplificación al máximo de tablas de verdad, de manera que sea un proceso más sencillo.

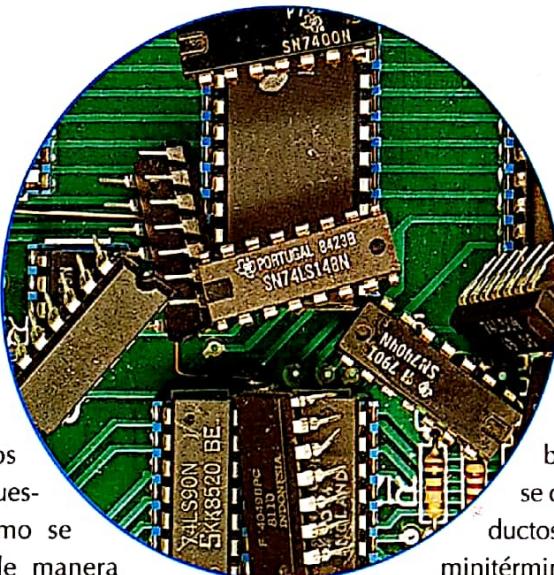
Una vez explicado todo el contenido de las Leyes de Boole, lo único que nos queda es explicar el por qué y el para qué de las mismas, esto es, los postulados, junto con las propiedades y los teoremas vistos, nos sirven para minimizar el boceto de los circuitos digitales que tengamos que diseñar, de manera que podamos simplificar al máximo las funciones a obtener, con lo que nos aseguramos reducir al máximo el número de componentes necesarios, es decir, puertas lógicas tales como las NOT, OR, AND, NOR, NAND, XOR y XNOR.

El proceso de obtención de una función simplificada implica primero la realización de la tabla de verdad, de manera que podamos obtener una función no simplificada pero que sí nos dé idea del posible resultado del circuito y, a partir de aquí, aplicar todas las leyes de Boole posibles para simplificar el circuito al máximo, de manera que tengamos que utilizar pocos circuitos (puertas lógicas) en la realización del mismo.

LAS TABLAS DE VERDAD

Una vez que en los capítulos anteriores se han visto las tablas de verdad, nos damos cuenta de lo que debemos desarrollar, nos creamos nuestra tabla de verdad, como se explicó anteriormente, de manera que obtengamos el primer paso de creación de la misma. Un ejemplo de tabla de verdad lo encontramos en la figura adjunta, en la que se puede observar que tenemos cuatro entradas denominadas A, B, C y D y una salida, denominada como F. Si un circuito digital tuviera más de una salida, lo único que tendríamos que hacer sería colocar a la derecha de la salida F otra columna que nos indicara la otra salida, que podría ser denominada como F₂, siendo la salida F renombrada como F₁, según puede verse en la figura siguiente.

Una vez que nos encontramos con la tabla de verdad, podemos seguir de dos maneras diferentes,



esto es, por MAXITÉRMINOS o por MINITÉRMINOS. De una forma más sencilla podemos decir que los maxitérminos (también llamados maxterminos) se desarrollan en forma de productos de sumas, mientras que los minitérminos (denominados también como mintérminos), son sumas de productos, como se podrá comprobar a continuación. Fijándonos en la última tabla, que tiene las dos salidas F₁ y F₂, podremos comprobar de qué manera deberemos decidir con qué nos interesa desarrollar las funciones a crear, esto es:

- Si existen muchas salidas que contemplan un 1 dependiendo del valor de las entradas, nos interesa realizar la función de salida con respecto a los maxitérminos, ya que nos quedará una función más sencilla.
- Si existen muchas salidas que contemplan lo contrario, esto es, que tienen muchos ceros

como salidas, nos interesa realizar la función de salida a través de los minitérminos.

El lector se preguntará: ¿qué es cada cosa?, es decir ¿qué son y cómo se desarrollan los maxitérminos y los minitérminos? Sencillo. Nosotros tenemos una tabla de verdad con unas entradas y unas salidas, de manera que dependiendo de los valores de la entrada así se activarán o no las salidas. La utilización de minitérminos aparece cuando usamos las salidas a uno lógico ("1"), mientras que cuando se utilizan los maxitérminos cogemos las salidas que se encuentran a cero, sin alterar la función global, es decir, la tabla de verdad define el comportamiento global de una función, ya sea desarrollada a través de las salidas a cero o a través de salidas a uno. Cuando se utiliza una función para obtener unos, implica que para todos los demás valores, la función lógica va a dar un cero, igual que a la inver-

sa, esto es, cuando se utiliza una función para obtener ceros a la salida, para todos los demás valores de la tabla obtendremos unos a la salida. A continuación vamos a desarrollar cada una de las dos posibles combinaciones de la segunda tabla de verdad expuesta anteriormente: Si utilizamos minitérminos para el desarrollo de la función F1, deberemos obtener la combinación de las entradas que producen unos a la salida, esto es:

$$\begin{aligned} F_1 = 1 = & \bar{D}CBA + D\bar{C}BA + DC\bar{B}A + \\ & + DCB\bar{A} + DCBA \end{aligned}$$

O lo que es lo mismo, hemos obtenido sumas de productos, de manera que para simplificarlo deberemos aplicar todo lo estudiado en la anterior explicación, esto es, las propiedades, postulados y teoremas del álgebra de Boole, de manera que:

DCBA	F
0000	0
0001	0
0010	0
0011	0
0100	0
0101	0
0110	0
0111	1
1000	0
1001	0
1010	0
1011	1
1100	0
1101	1
1110	1
1111	1

DCBA	F1	F2
0000	0	1
0001	0	1
0010	0	1
0011	0	1
0100	0	1
0101	0	1
0110	0	0
0111	1	1
1000	0	1
1001	0	1
1010	0	0
1011	1	1
1100	0	1
1101	1	1
1110	1	1
1111	1	1

$F_1 = \bar{D}CBA + D\bar{C}BA + DC\bar{B}A + DCB\bar{A} + DCBA$
 $F_2 = (\bar{D} + C + B + \bar{A})(D + \bar{C} + B + \bar{A})$
F1 = MINITÉRMINOS
F2 = MAXITÉRMINOS

Ejemplo de tabla de verdad en la que se pueden observar cuatro entradas al circuito digital a diseñar con tan sólo una salida que únicamente se pone a uno lógico en cinco de las diecisésis combinaciones posibles de las entradas.

En este caso tenemos el ejemplo de la tabla de verdad anterior pero ampliada, ya que ahora tenemos dos salidas del circuito, en el que se ha desarrollado la función F1 a través de minitérminos y la función F2 a través de maxitérminos.

1) Teniendo en cuenta que $A + \bar{A} = 1$, observamos que:

$$\begin{aligned} F_1 &= \bar{D}CBA + D\bar{C}BA + DC\bar{B}A + DCB(\bar{A} + A) \\ F_1 &= \bar{D}CBA + D\bar{C}BA + DC\bar{B}A + DCB \end{aligned}$$

2) También se observa que tenemos una combinación $\bar{D}C + D\bar{C}$ en los dos primeros términos, es decir, una función realizada a través de una puerta XOR, con lo que:

$$F_1 = BA(\bar{D}C + D\bar{C}) + DC\bar{B}A + DCB$$

3) Podemos sacar factor común a los dos últimos términos DC y obtendremos:

$$F_1 = BA(\bar{D}C + D\bar{C}) + DC(B\bar{A} + B)$$

Siendo ésta una de las posibles funciones a desarrollar, con esto queremos decir que a cada uno se le pueden ocurrir otros pasos para simplificar una misma función y obtener un resultado igual, más sencillo o más complicado, con lo que todo dependerá de la técnica adquirida en la simplificación de las funciones de las tablas de verdad. Para que el lector observe que esto es así, podemos aplicar otras propiedades del álgebra de Boole a la función obtenida en el paso 1, de manera que:

1) Teníamos anteriormente:

$$F_1 = \bar{D}CBA + D\bar{C}BA + DC\bar{B}A + DCB$$

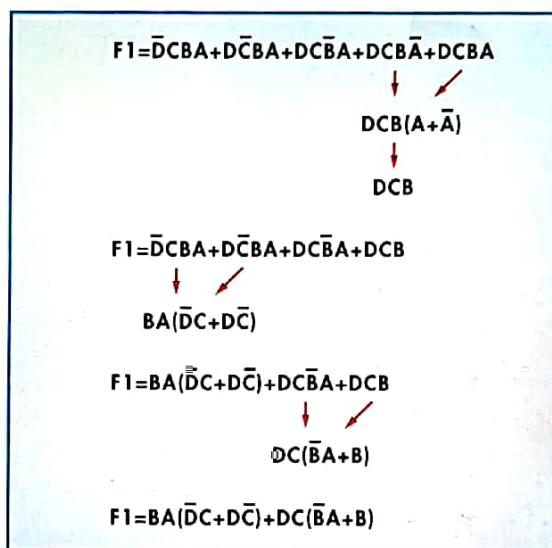
2) Observando entonces que entre el segundo y tercer término tenemos otra función XOR, esto es, $\bar{C}B + C\bar{B}$, con lo que:

$$F_1 = \bar{D}CBA + DA(\bar{C}B + C\bar{B}) + DCB$$

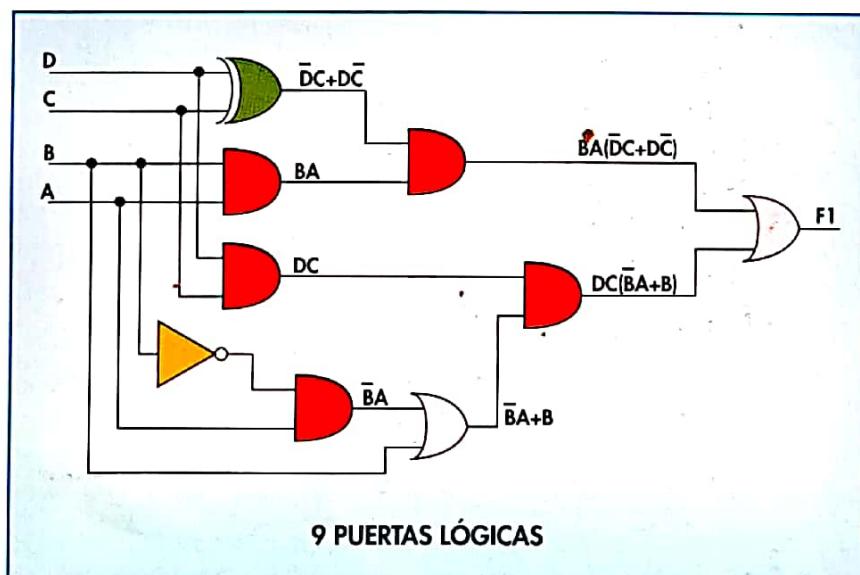
3) Sacando ahora factor común a los términos primero y cuarto tendremos que:

$$F_1 = CB(\bar{D}A + D) + DA(\bar{C}B + C\bar{B})$$

Cada vez que se desarrolle una función, se deberá tener en cuenta la cantidad de puertas lógicas que requieren realizar esta combinación, ya que puede resultar que sea más sencillo con alguna otra simplificación, es decir, que nos ahorremos puertas intentando simplificar la misma función aplicando otras propiedades del álgebra de Boole. En las figuras adjuntas se pueden observar los



Simplificación de la función F_1 obtenida, en la que podemos ver todas las propiedades y leyes aplicadas a la misma para reducirla lo más posible.



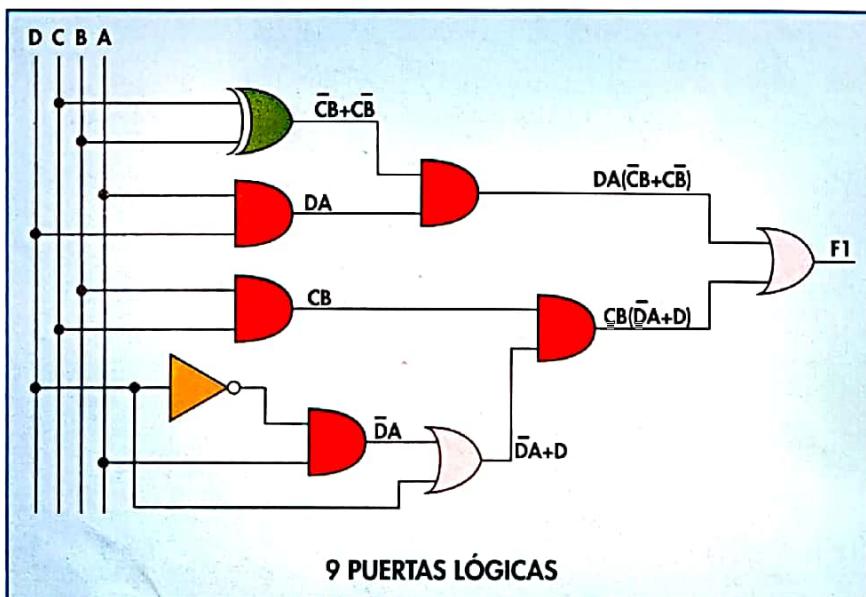
Disposición de las puertas lógicas a emplear en la función lógica anterior una vez simplificada la misma. Se puede observar que existen nueve puertas lógicas, siendo cinco puertas AND's, dos puertas OR's, una puerta NOT y otra puerta XOR.

resultados obtenidos, esto es, el circuito asociado a la función 1 de la tabla de verdad, realizada a través de minitérminos y en la que se puede comprobar que cada simplificación realizada se desarrolla con distintas puertas lógicas y ambas son totalmente válidas.

Con respecto a la segunda función, la denominada F_2 , se va a hacer a través de maxitérminos para que el lector compruebe la diferencia de realización de una a otra, y así pueda aplicar cada uno de los dos métodos en las tablas de verdad que considere necesario. Obtendremos:

$$\begin{aligned} F_1 &= \bar{D}\bar{C}B\bar{A} + \bar{D}\bar{C}B\bar{A} + D\bar{C}\bar{B}\bar{A} + D\bar{C}B \\ &\quad \downarrow \quad \downarrow \\ &= DA(\bar{C}B + C\bar{B}) \\ \\ F_1 &= \bar{D}\bar{C}B\bar{A} + DA(\bar{C}B + C\bar{B}) + D\bar{C}B \\ &\quad \downarrow \quad \downarrow \\ &= CB(D\bar{A} + D) \\ \\ F_1 &= CB(D\bar{A} + D) + DA(\bar{C}B + C\bar{B}) \end{aligned}$$

En este caso tenemos otra posible simplificación de la misma función lógica, en la que se han aplicado otros postulados y leyes no aplicadas anteriormente, obteniendo por tanto un resultado distinto al anterior.



Tenemos a su vez las mismas puertas lógicas que en el caso anterior, pero en este caso con una conexión distinta, dependiente de la segunda simplificación realizada.

$$F_2 = 0 = (\bar{D} + C + B + \bar{A})(D + \bar{C} + B + \bar{A})$$

Desarrollando todos los paréntesis conseguiremos una función más factible de simplificar, con lo que tendremos:

$$\begin{aligned} F_2 = 0 &= \bar{D}\bar{D} + \bar{D}\bar{C} + \bar{D}\bar{B} + \bar{D}\bar{A} + CD + C\bar{C} + \\ &+ CB + C\bar{A} + BD + B\bar{C} + BB + B\bar{A} + \bar{A}D + \\ &+ \bar{A}\bar{C} + \bar{A}\bar{B} + \bar{A}\bar{A} \end{aligned}$$

Quitando redundancias tales como BB ó $\bar{D}\bar{D}$, conseguimos:

$$\begin{aligned} F_2 &= \bar{D}\bar{C} + \bar{D}\bar{B} + \bar{D}\bar{A} + CD + CB + C\bar{A} + BD + \\ &+ B\bar{C} + B + B\bar{A} + \bar{A}D + \bar{A}\bar{C} + \bar{A}\bar{B} + \bar{A} \end{aligned}$$

Aplicando la propiedad:

$$\bar{A}B + \bar{A} = \bar{A}(B + 1) = \bar{A}$$

obtenemos que todos los términos que contengan \bar{A} se simplificarán, por lo que conseguiremos que:

$$F_2 = \bar{D}\bar{C} + \bar{D}\bar{B} + CD + CB + BD + B\bar{C} + B + \bar{A}$$

A su vez, encontramos lo mismo con el término $B\bar{C} + B$, con lo que todos los términos que contenga B también se simplificarán, como se ha explicado en el párrafo anterior, con lo que la función será:

$$F_2 = \bar{D}\bar{C} + CD + B + \bar{A}$$

Viendo además que contiene un término $\bar{D}\bar{C} + CD$ que consiste en una puerta XNOR, con lo que el resultado de la función F_2 será:

$$F_2 = (\overline{D \oplus C}) + B + \bar{A}$$

En este caso, y como se puede comprobar en la figura en la que se ha desarrollado con puertas lógicas la función F_2 , hemos llegado al caso más sencillo en el

que ya no se puede seguir simplificando, por lo que no es necesario que probemos a simplificar la función F_2 aplicando otras propiedades del álgebra de Boole.

Además, un detalle a tener en cuenta es que la primera función que se ha obtenido:

$$F_1 = \bar{D}CBA + D\bar{C}BA + DC\bar{B}A + DCB\bar{A} + DCBA$$

y

$$F_2 = (\bar{D} + C + B + \bar{A})(D + \bar{C} + B + \bar{A})$$

son las denominadas funciones canónicas, ya que son la forma más completa en la que se puede obtener la función a desarrollar, mientras que si aplicamos cualquiera de los teoremas del álgebra de Boole, la función se habrá simplificado y por tanto ya no será canónica.

Otro ejemplo que proponemos al lector es el siguiente:

De un bus de datos de 16 líneas denominadas desde A_0 hasta A_{15} , desarrollar un circuito digital con puertas lógicas que dé como resultado un uno a la salida si a la entrada tiene desde la dirección D1C8h hasta la dirección D1CFh, mientras que la salida debe ser un cero si nos encontramos en el resto de direcciones posibles.

Al tener la letra h al final del código nos indica que se trata de un dato hexadecimal, lo que nos proporciona cierta idea del código binario del que se trata con tal sólo convertir el dato a dígitos binarios, de manera que:

$$\begin{aligned} D &= 1101 \\ 1 &= 0001 \\ C &= 1100 \end{aligned}$$

El último dato nos indica el margen de direcciones que tendremos que identificar, esto es:

$$8 = 1000$$

$$F = 1111$$

Por lo tanto el dato D1C8 en hexadecimal será 1101-0001-1100-1000 en binario, y el dato D1CFh será 1101-0001-1100-1111 en binario, con lo que tan sólo tendremos que realizar una pequeña tabla de verdad como la que se indica en la figura adjunta, y en la que se puede observar que todas las X's que aparecen constituyen todos los demás valores que pueden tomar las 16 líneas

$$\begin{aligned} F_2 &= (\bar{D} + C + B + \bar{A})(D + \bar{C} + B + \bar{A}) \\ F_2 &= \bar{D}\bar{D} + \bar{D}\bar{C} + \bar{D}B + \bar{D}\bar{A} + CD + C\bar{C} + CB + C\bar{A} + BD + B\bar{C} + BB + B\bar{A} + \bar{A}D + \bar{A}\bar{C} + \bar{A}B + \bar{A}\bar{A} \\ F_2 &= \bar{D}\bar{C} + \bar{D}B + \bar{D}\bar{A} + CD + CB + C\bar{A} + BD + B\bar{C} + BB + B\bar{A} + \bar{A}D + \bar{A}\bar{C} + \bar{A}B + \bar{A} \end{aligned}$$

$$\text{Ya que: } \bar{D}\bar{D}=0$$

$$C\bar{C}=0$$

$$BB=B$$

$$\bar{A}\bar{A}=\bar{A}$$

$$\text{Además: } \bar{A}B+\bar{A}=\bar{A}(B+1)=\bar{A}$$

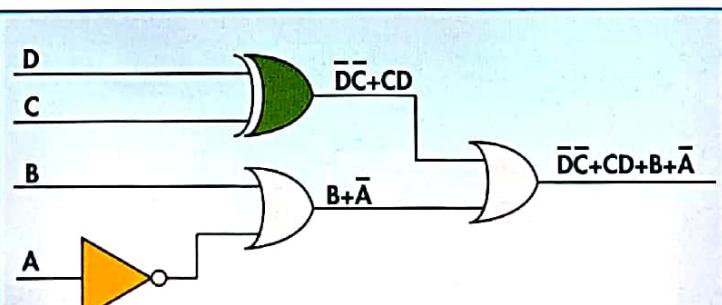
$$F_2 = \bar{D}\bar{C} + \bar{D}B + CD + CB + BD + B\bar{C} + B\bar{A}$$

$$\text{También: } B\bar{C}+B=B(\bar{C}+1)=B$$

$$F_2 = \bar{D}\bar{C} + CD + B + \bar{A}$$

$$F_2 = \overline{D+C+B+\bar{A}}$$

Simplificación realizada a la segunda función de la tabla de verdad inicial, teniendo en cuenta que ahora la función corresponde a máxitérminos y que el proceso será el mismo una vez reducidos los productos de sumas.



Esquema eléctrico de la simplificación anterior, en la que se puede observar la utilización únicamente de cuatro puertas lógicas.

de datos, tanto por encima del número D1CFh como por debajo del número D1C8h, esto es, $2^{16} = 65536$ combinaciones distintas que puede tomar este bus de datos, del que tenemos que identificar ocho valores distintos, de manera que: $65536 - 8 = 65528$ valores distintos que puede tomar el bus de datos que no nos interesan, con lo

que hemos reducido la gigantesca tabla de verdad que obtendríamos a la que aparece en la figura adjunta con tal solo colocar las X.

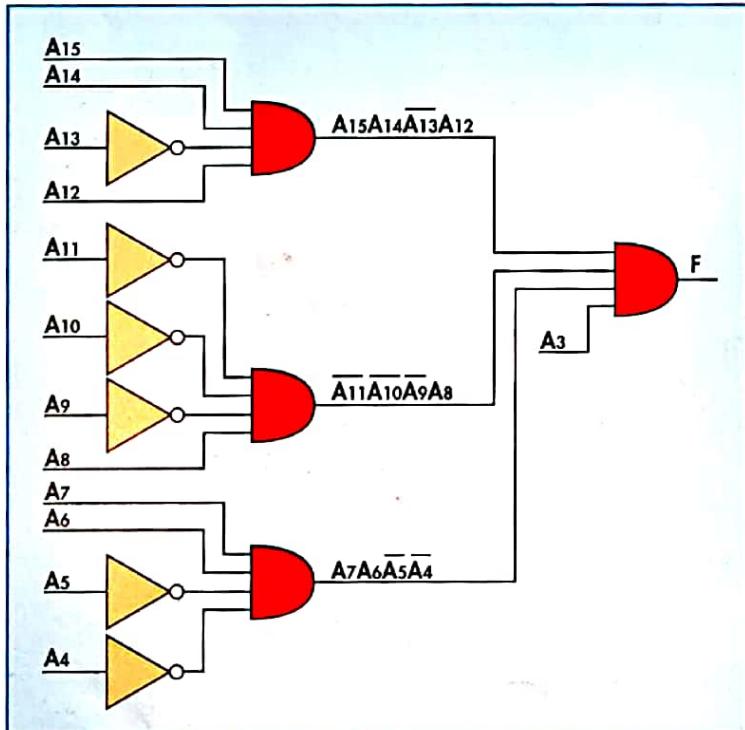
La solución es sencilla si observamos las combinaciones comunes de las direcciones a identificar, de manera que si vemos la tabla de verdad, encontramos que desde la línea A₁₅ hasta la línea A₃ todas son comunes y, tan sólo varían las líneas A₂, A₁ y A₀, por lo que la función a desarrollar está basada en la tabla de verdad simplificada.

La función final que nos dará un uno a la salida cuando a la entrada tengamos una dirección comprendida entre la D1C8h y la D1CFh será:

$$F = A_{15} A_{14} \overline{A}_{13} \overline{A}_{12} \overline{A}_{11} \overline{A}_{10} \overline{A}_9 A_8 \\ A_7 A_6 \overline{A}_5 \overline{A}_4 A_3$$

A ₁₅	A ₁₄	A ₁₃	A ₁₂	A ₁₁	A ₁₀	A ₉	A ₈	A ₇	A ₆	A ₅	A ₄	A ₃	A ₂	A ₁	A ₀	F
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0
1	1	0	1	0	0	0	0	1	1	0	0	0	1	1	1	0
1	1	0	1	0	0	0	1	1	1	0	0	1	0	0	0	1
1	1	0	1	0	0	0	0	1	1	0	0	1	0	0	1	1
1	1	0	1	0	0	0	0	1	1	0	0	1	0	1	0	1
1	1	0	1	0	0	0	0	1	1	0	0	1	0	1	1	1
1	1	0	1	0	0	0	0	1	1	0	0	1	1	0	0	1
1	1	0	1	0	0	0	0	1	1	0	0	1	1	0	1	1
1	1	0	1	0	0	0	0	1	1	0	0	1	1	1	0	1
1	1	0	1	0	0	0	0	1	1	0	0	1	1	1	1	0
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0

Tabla de verdad que nos indica el margen de direcciones que tenemos que identificar a través de la función obtenida. Las X's nos indican todo el resto de direcciones que se encuentran tanto por encima como por debajo, mientras que si la X se encuentra en el valor de una salida, daría igual el estado de la misma, por lo que podría ser tomada como un uno o un cero.



Esquema eléctrico del circuito que identificaría el margen de direcciones entre las D1C8h y la D1CFh, realizado a través de diez puertas lógicas.

Con lo que el circuito final quedaría como se observa en la figura adjunta, en la que se tendrían un total de diez puertas lógicas, entre las que, según se puede observar, tendríamos cuatro puertas AND's de cuatro entradas y seis puertas NOT's que serían las encargadas de negar los datos necesarios para así identificar el margen de direcciones pedido. En el caso de que tuviéramos varios márgenes de direcciones para identificar, el proceso sería prácticamente igual a éste, de manera que la única diferencia sería que tendríamos que realizar tantas sumas finales como márgenes tuviéramos que identificar. Dicho de una forma más sencilla, en el caso en el que hubiera otro margen de direcciones, el proceso de identificación de las líneas sería el mismo y luego tendríamos que sumar ese resultado al obtenido anteriormente, para así completar la tabla de verdad del mismo.