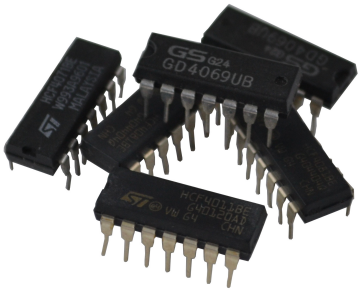
electrónica digital

Rocío Leira Rodríguez
Jorge Gómez Suárez

Definición

La **electrónica digital** se encarga de sistemas electrónicos en los cuales la información está codificada en dos únicos estados. A estos estados se les puede llamar "verdadero" o "falso", HIGH (H) o LOW(L) o más comúnmente 1 y 0. Realmente lo que estamos haciendo es transformar las señales del circuito (sonido, luz, temperatura,…) en números, por lo que una vez conseguido, es muy sencillo compararlas, operar, contar, almacenar, tratar, etc las señales, bien usando el sistema binario, el código BCD (Binary Codex Decimal), el hexadecimal o el ASCII, dependiendo de la función digital a realizar.

Como vemos en el esquema, la electrónica digital parte de una señal analógica que convierte en digital (código binario) para tratarla y finalmente la vuelve en convertir en analógica.



Los circuitos digitales son circuitos encapsulados en circuitos integrados

Los **sistemas digitales** pueden realizar diferentes funciones: aplicar lógica, operar, memorizar, comparar, contar… y son clasificados en:

- **Sistemas digitales combinacionales:** Un circuito combinacional es aquel que está formado por funciones lógicas elementales en los que el valor de la salida o salidas dependen exclusivamente de las entradas. Son circuitos sin memoria, la salida solo depende de lo que ocurre en ese momento, no de lo que pasó con anterioridad. Dentro de los circuitos combinacionales tenemos:

    - **puertas lógicas:** realizan las operaciones lógicas si-no-y-o
    - **codificadores/decodificadores:** modifican las señales binarias de manera que de una señal de 4 bits pasamos a tener una de 9 bits (codificador) o al revés (decodificador).
    - **multiplexor/demultiplexor:** son un tipo especial de codificadores y decodificadores.
    - **comparador:** son capaces de comparar dos señales digitales del mismo número de bits indicando si son iguales o cual es mayor.

- **Sistemas digitales secuenciales:** el valor de la salida no solo depende del estado de las entradas, sino también de los valores y estados almacenados anteriormente. Son circuitos con **memoria**. Ejemplos de este tipo de circuitos son:

    - **biestables:** flip-flop en inglés, circuito utilizado como memoria en electrónica digital.
    - **contadores:** construido a partir de biestable y puertas lógicas capaz de almacenar y contar los impulsos.

- **Sistemas digitales microprogramables:** Un sistema microprogramable es un sistema electrónico digital capaz de interpretar y ejecutar secuencialmente las órdenes contenidas en un programa y a una velocidad muy elevada. Diferenciamos el **hardware** o parte física y el **software**, conjunto de instrucciones. Un ejemplo de este tipo de circuito es la placa Arduino. Este tipo de circuitos está compuesto por puertas lógicas, comparadores, contadores, conversores analógico- digital, memorias, codificadores, etc, y simplemente se activan unos u otros en función del programa introducido. Además llevan un software especial, denominado **firmware**, que traduce el programa a código máquina (binario).

Circuitos digitales

El factor más importante a la hora de trabajar con circuitos digitales es saber cuántos bits vamos a utilizar. Bien sea porque trabajamos directamente con una señal digital, por ejemplo cuando pulsamos la tecla de una calculadora y ese pulso se traduce en un código binario, o bien cuando digitalizamos una señal analógica, por ejemplo al convertir lectura de un sensor de temperatura o de luz a código binario, lo más importante es determinar cuántos bits necesito para que el sistema funcione correctamente.

En el caso de una calculadora, ¿cuántas teclas de números tiene? Pues tenemos diez dígitos, del 0 al 9. ¿Cuántos bits necesito para representarlos?

Si uso dos bits tengo 4 posibles combinaciones:

Está claro que son insuficientes para representar los diez dígitos que necesito, tengo que usar más bits.

bit 1	bit 2	número decimal
0	0	0
0	1	1
1	0	2
1	1	3

$2^n=2^2=4$

Tres bits tampoco me llegarían, ya que obtengo ocho combinaciones posibles, por lo que necesito usar cuatro bits, aunque solo usaría diez de las dieciséis combinaciones posibles.

bit 1	bit 2	bit 3	bit 4	número decimal
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9

$2^n=2^2=4$ ✗

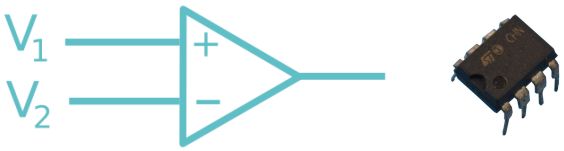
$2^n=2^3=8$ ✗

$2^n=2^4=16$ ✓

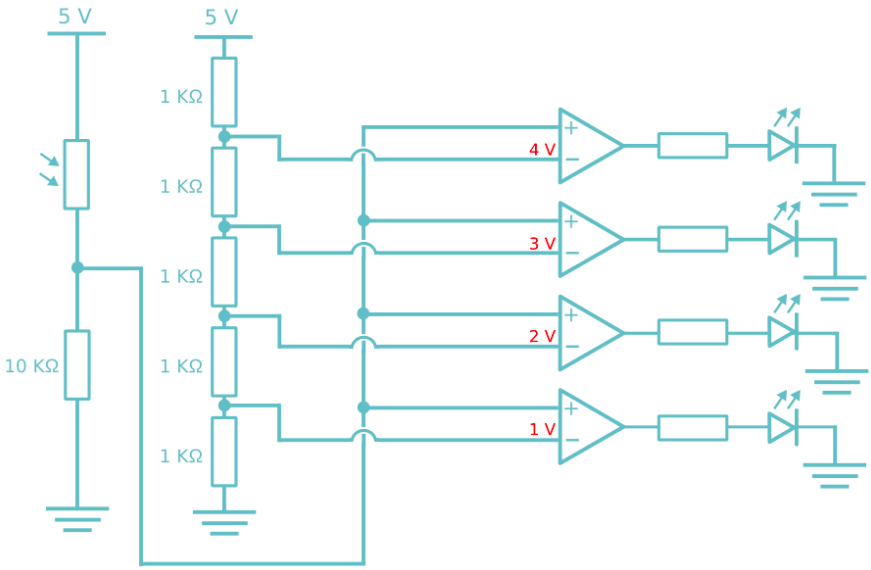
Si convierto una señal analógica en digital, en función del número de bits que utilice tendré mejor o peor calidad en la digitalización. Por ejemplo, al hacer una foto con una cámara digital podemos guardar el color de cada pixel de la foto con cuatro bits, ocho, dieciséis, … y en función de esto podremos almacenar 16 colores diferentes, 256 o incluso 16 millones. En contrapartida cuantos más bits utilice mayor complejidad tendrá el circuito para guardarlos, operar con ellos, etc.

Conversión analógica-digital

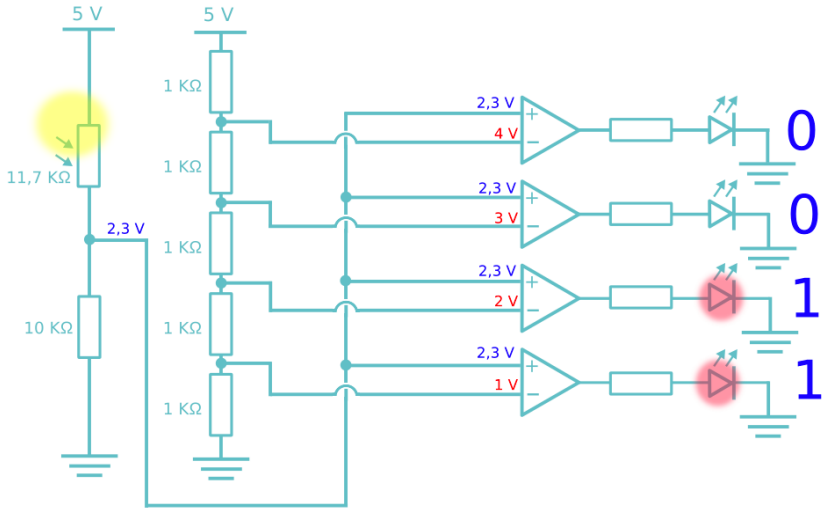
Para entender como se puede realizar la conversión de analógico a digital usaremos como ejemplo el circuito integrado analógico, el **amplificador operacional**, que es un comparador y amplificador de voltajes, aunque en este caso nos interesa su función como comparador. Sabemos que si V_1 es mayor que V_2 el circuito se activa:



Si usamos varios comparadores de este tipo que tengan diferentes voltajes de referencia podremos convertir una señal analógica, por ejemplo la cantidad de luz que incide sobre una LDR, en una señal digital binaria. Ejemplo:



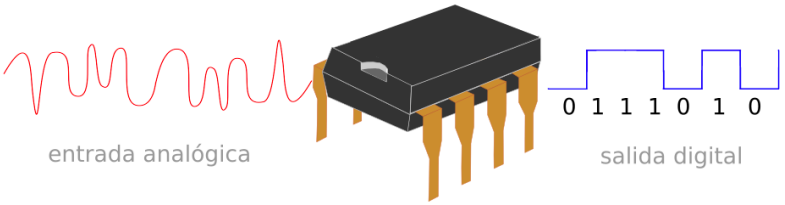
Si la LDR recibe una cantidad de luz tal que ofrece una resistencia de 11,7 KΩ hará que las **entradas no inversoras(+)** de los amplificadores operacionales reciban 2,3 V que compararán con los voltajes de referencia generados por las cincos resistencias en serie de 1 KΩ:



Se encenderán los diodos LED en los que el voltaje de la entrada no inversora(+) sea mayor que la entrada inversora(-).

La luz recibida por la LDR se transforma en el código de cuatro bits: 1100, ya **hemos digitalizado la señal analógica**. En este ejemplo la cantidad de luz podemos codificarla en cinco posibles niveles: 0000 - 1000 - 1100 - 1110 - 1111, por lo que tenemos una digitalización bastante básica, solo podríamos determinar cinco niveles de luz, pero si usáramos más comparadores, tendríamos más bits y por lo tanto mayor precisión.

Hoy en día ya existen unos circuitos integrados específicos para convertir las señales analógicas en digitales y viceversa, por lo que no es necesario montar circuitos tan complejos.



Sistemas de representación

Una vez que tenemos nuestra señal digital con un número de bits determinado, el circuito digital que estamos usando ya podrá realizar operaciones lógicas, matemáticas, almacenarlos, compararlo con otros,… pero para cada una de estas funciones muchas veces necesitamos reducir o aumentar el número de bits utilizados. Para ello en electrónica digital se utilizan diferentes **sistemas de representación**, que facilitan toda esta labor.

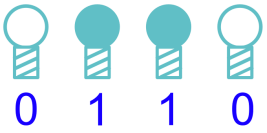
Por ejemplo, cuando trabajamos con números. Todos sabemos lo que es un número, aunque ese número lo podemos representar de varias maneras. Nosotros estamos acostumbrados a representar los números utilizando diez dígitos: 0 - 1 - 2 - 3 - 4 - 5 - 6 - 7 - 8 - 9, por eso nuestro sistema de representación se denomina **sistema decimal** o sistema en base diez. Cualquier número mayor de nueve lo tendremos que representar con la combinación de los diez dígitos anteriores, así el número diez se representa con la combinación del 1 y el 0, es decir, 10.

Pero también existen otros sistemas de representación, que aunque para nosotros no son tan habituales, son muy útiles en los sistemas digitales. Así tenemos el **sistema binario** o de base dos, con solamente dos dígitos: 0 y 1, que se traducen normalmente con el valor a de 5 voltios al dígito 1 y 0 voltios al dígito 0. Cada dígito binario recibe, como ya sabemos, el nombre de **bit** (**bi**nary digi**t**). Los circuitos digitales sólo saben trabajar con números en binario, aunque para nosotros es más cómodo trabajar en decimal. Además se suelen agrupar en grupos de ocho, lo que denominamos **byte**, y sus múltiplos 16, 32, …

Otro sistema muy usado en electrónica digital es el **sistema hexadecimal**, en el que usamos dieciséis dígitos: 0 - 1 - 2 - 3 - 4 - 5 - 6 - 7 - 8 - 9 - A - B - C - D - E - F, en donde las letras representan lo que para nosotros en decimal serían el 10, 11, 12, 13, 14 y 15.

Realmente lo que se hace es aprovechar todas las dieciséis combinaciones al usar cuatro bits.

Si queremos representar con un número el estado en el que están cuatro bombillas, la manera más sencilla sería con el sistema binario, donde 0 sería la bombilla apagada y 1 la bombilla encendida:



En este ejemplo el código 0110 nos serviría perfectamente para saber cuales están encendidas y cuales apagadas. Pero si tenemos veinte bombillas o cien, no está tan clara esa información. En estos casos nos interesa representarlo de una forma más compacta, con menor número de dígitos, por lo que recurriremos a sistemas como el hexadecimal.

bit 1	bit 2	bit 3	bit 4	número hexadecimal
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	A
1	0	1	1	B
1	1	0	0	C
1	1	0	1	D
1	1	1	0	E
1	1	1	1	F

Si tenemos el número binario 0110110000101110 y lo queremos representar en hexadecimal agruparemos los bits en grupos de cuatro y sustituiremos cada grupo por el dígito correspondiente en hexadecimal:

0110 1100 0010 1110

6 C 2 E

Por lo que el equivalente en hexadecimal sería: 6C2E.

El sistema hexadecimal se usa mucho en los sistemas de almacenamiento digitales, aunque también lo podemos encontrar en el lenguaje HTML para los colores de las páginas web.

Puertas lógicas

Claude Elwood Shannon demostró en 1938 cómo el álgebra de Boole, estructura algebraica que esquematiza las operaciones lógicas (Y, O, NO y SI) se podía utilizar en los circuitos digitales. Las **puertas lógicas** son los circuitos integrados capaces de realizar dichas operaciones lógicas, que combinadas entre sí nos permiten construir circuitos realmente complejos.

Las puertas lógicas son:

SI - NO

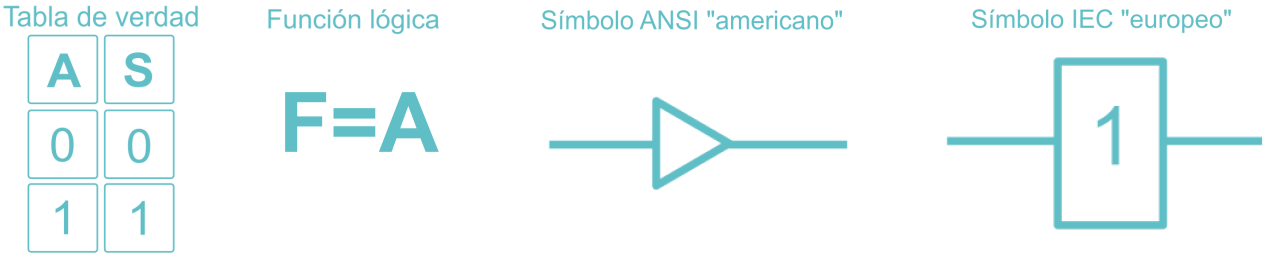
OR - NOR

AND - NAND

XOR - XNOR

Puerta SI o BUFFER

La puerta lógica SI, realiza la función booleana igualdad. Tiene solo una entrada y una salida, siendo ambas siempre iguales. En la práctica se suele utilizar para mantener la tensión sin variaciones.



Puerta NOT o inversora

Realiza la función negación lógica. La función toma valor lógico “1” cuando la entrada a vale “0” y toma el valor “0” cuando la entrada a vale “1”. También se conoce como función inversión. Es la contraria a la SI.

Tabla de verdad

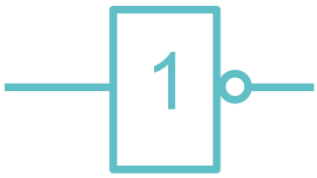
A	S
0	1
1	0

Función lógica

$$F=\overline{A}$$

Símbolo ANSI "americano"

Símbolo IEC "europeo"



Puerta OR

Realiza la función suma lógica o función OR. La función toma valor lógico “1” cuando alguna de las entradas vale “1” y toma el valor “0” cuando las entradas valen “0”.

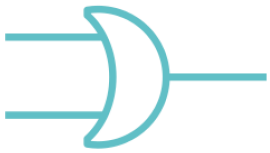
Tabla de verdad

A	B	S
0	0	0
0	1	1
1	0	1
1	1	1

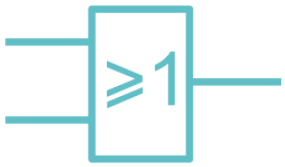
Función lógica

$$F=A+B$$

Símbolo ANSI "americano"



Símbolo IEC "europeo"



Puerta NOR

Realiza la función suma lógica negada o función NOR. Es la función contraria a la OR, por lo tanto la salida siempre es “0” a no ser que las entradas sean todas “0”.

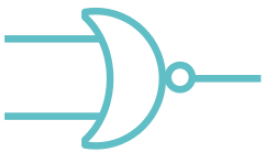
Tabla de verdad

A	B	S
0	0	1
0	1	0
1	0	0
1	1	0

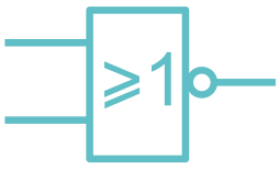
Función lógica

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

Símbolo ANSI "americano"



Símbolo IEC "europeo"



Puerta AND

Realiza la función producto lógico o función AND. La función toma valor lógico "1" cuando todas las entradas valen "1" y toma el valor "0" cuando alguna de las entradas vale "0".

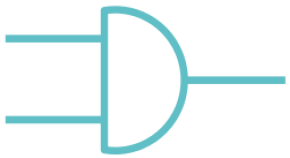
Tabla de verdad

A	B	S
0	0	0
0	1	0
1	0	0
1	1	1

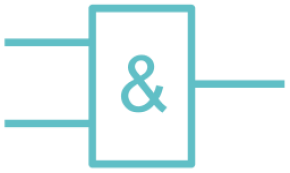
Función lógica

$$F=A.B$$

Símbolo ANSI "americano"



Símbolo IEC "europeo"



Puerta NAND

Realiza la función producto lógico negado o función NAND. Es la función contraria a la AND. La función toma valor lógico "0" cuando todas las entradas valen "1" y toma el valor "1"en el resto de los casos.

Tabla de verdad

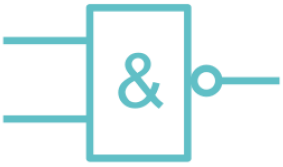
A	B	S
0	0	1
0	1	1
1	0	1
1	1	0

Función lógica

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

Símbolo ANSI "americano"

Símbolo IEC "europeo"



Puerta XOR

Realiza la función OR EXCLUSIVA. La función toma valor lógico "1" cuando las entradas tienen distinto valor y toma el valor "0" cuando las entradas son iguales.

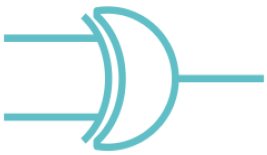
Tabla de verdad

A	B	S
0	0	0
0	1	1
1	0	1
1	1	0

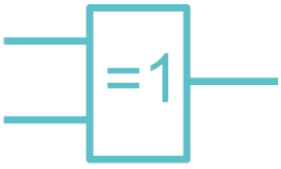
Función lógica

$$F=A\oplus B$$

Símbolo ANSI "americano"



Símbolo IEC "europeo"



Puerta XNOR
Realiza la función NOR EXCLUSIVA. Es la función contraria a la XOR. La función toma valor lógico "1" cuando las entradas tienen el mismo valor y toma el valor "0" cuando las entradas son distintas.

Tabla de verdad

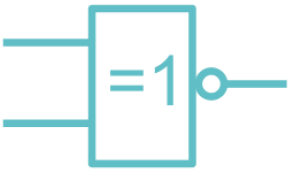
A	B	S
0	0	1
0	1	0
1	0	0
1	1	1

Función lógica

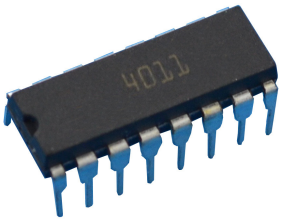
$$F=\overline{A \oplus B}$$

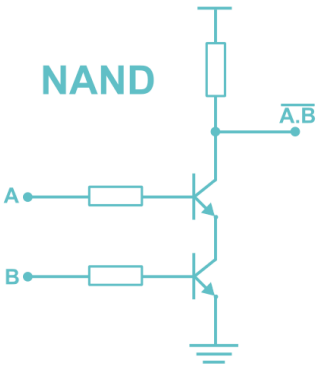
Símbolo ANSI "americano"

Símbolo IEC "europeo"



Las puertas lógicas se usan en forma de circuitos integrados, por ejemplo, un circuito de puertas NAND sería el **4011BE**, que está formado por cuatro puertas NAND de dos entradas. Interiormente este chip contiene resistencias y transistores conectados de tal modo que se cumplen eléctricamente la función lógica correspondiente.

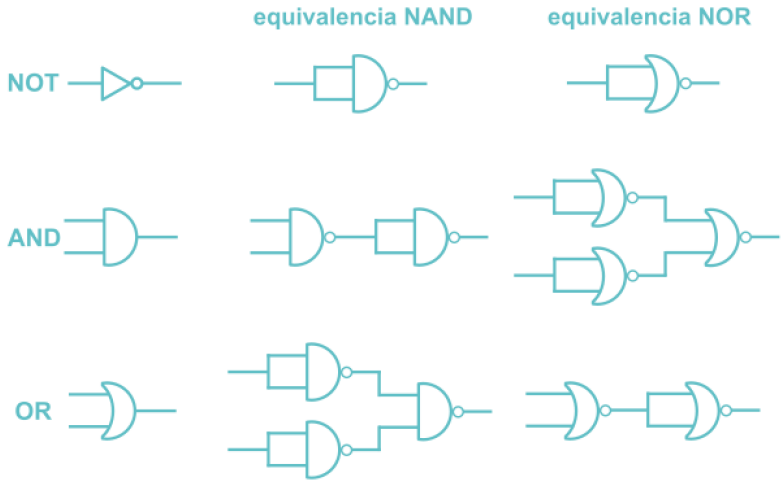




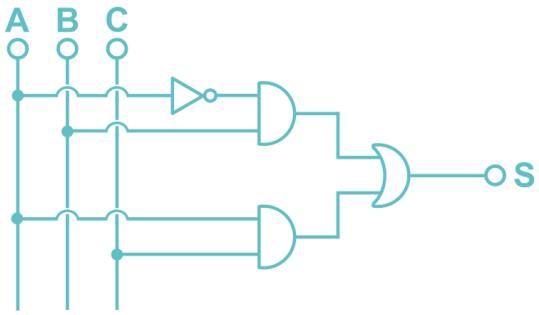
Equivalencia puertas lógicas

Las puertas lógicas NAND y NOR se denominan puertas universales, ya que pueden sustituir al resto de puertas. Esto es algo muy útil a la hora de montar los circuitos, ya que los simplifica mucho, y nos permite reducir el número de circuitos integrados necesarios para el montaje.

Las equivalencias con las puertas NAND y NOR son:

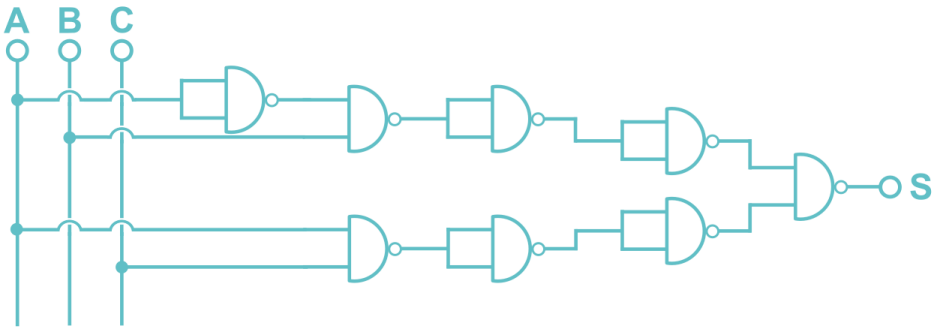


En el siguiente ejemplo vamos a ver como podríamos sustituir las puertas lógicas por las NAND. Partimos del siguiente esquema:

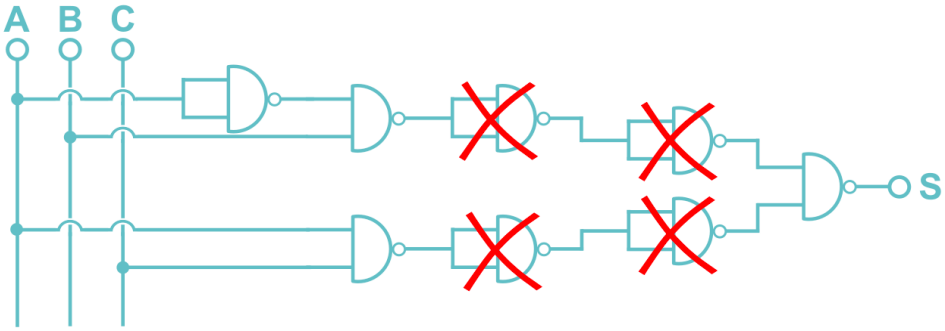


Para montar el circuito necesitaríamos tres circuitos integrados, uno de puertas NOT, otro de AND y otro de OR, aunque solo usaríamos algunas puertas de cada uno:

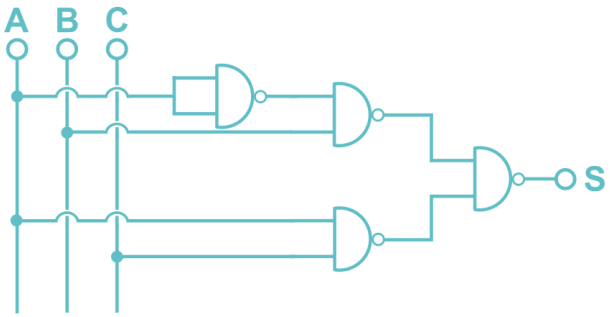
¿Pero qué ocurre si sustituimos las puertas NOT, AND y OR por las NAND?:



En principio parece que estamos complicando el circuito, pero realmente lo estamos simplificando, ya que si negamos dos veces es como si afirmamos:



Al final nos queda un circuito más sencillo, que podríamos montar con solo un circuito integrado, por ejemplo el 4011BE:



Esta es la gran ventaja de usar puertas universales, simplifican el circuito y ahorran costes en el montaje.

Diseño de circuitos combinacionales con puertas lógicas

Práctica puertas lógicas

Cuando diseñamos circuitos combinaciones a partir de puertas lógicas partimos de unas condiciones iniciales y obtenemos un esquema que nos indica qué elementos hay que utilizar, así como la interconexión que hay entre ellos. Los pasos que seguiremos para el diseño son los siguientes:

1. **Estudio de las condiciones iniciales y planteamiento del problema.**

Por ejemplo, vamos a diseñar un circuito para que se active el sistema de riego de un invernadero en función de las condiciones de luz, temperatura y humedad de la tierra. Queremos que el sistema active el riego de manera automática cuando se den las siguientes condiciones:

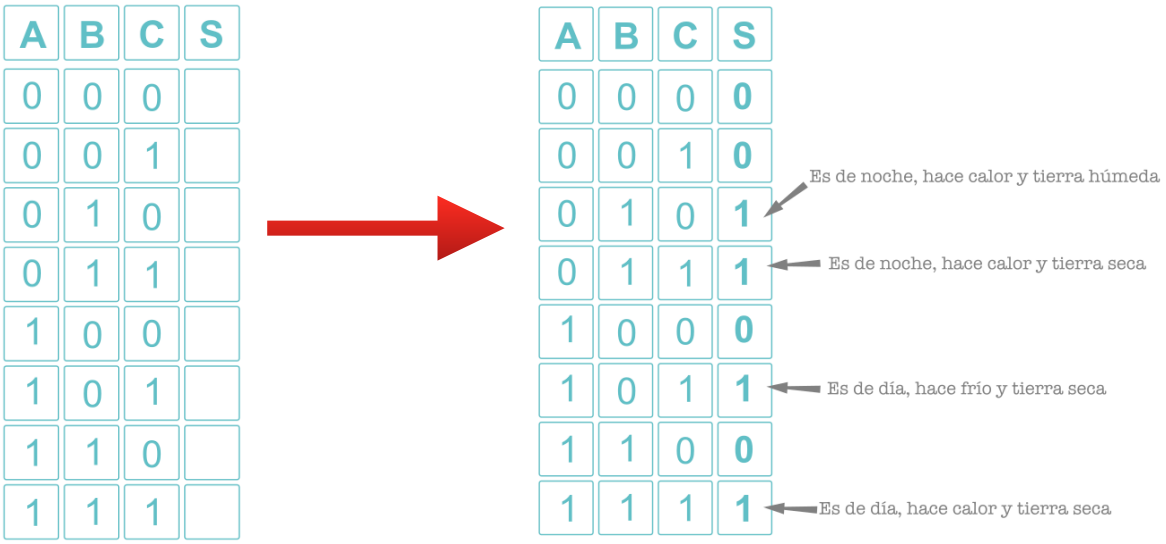
- Siempre que la tierra esté seca, a no ser que sea de noche y haga frío.
- Siempre que haga calor, si es de noche.

Debemos establecer cuántas entradas tenemos y cuántas salidas, nombrarlas y definir cuándo consideramos que son 1 y cuándo 0. En este caso tendremos tres entradas y una salida, y establecemos que:

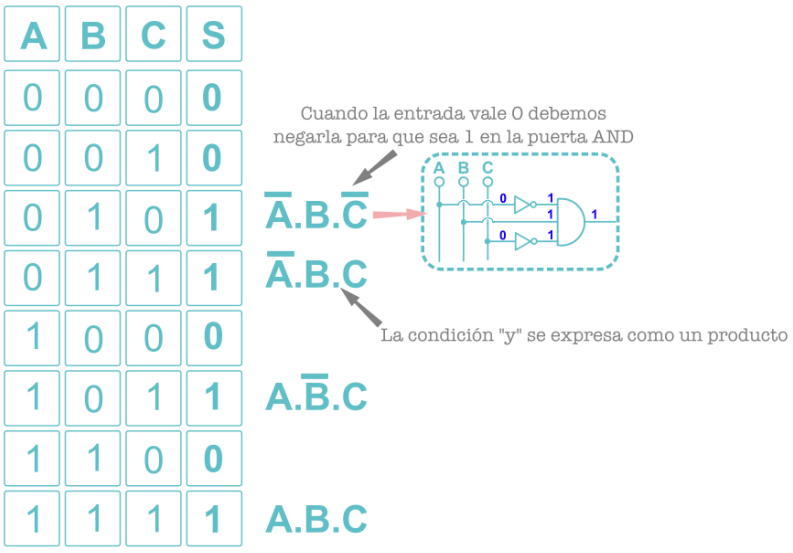
- Entrada A (luz): si hay luz el valor 1 y oscuridad 0
- Entrada B (temperatura): calor es 1 y frío 0
- Entrada C (humedad tierra): tierra seca es 1 y húmeda 0
- Salida (riego): regar es 1 y no regar es 0

2. **Obtención de las tablas de verdad y expresiones booleanas necesarias.**

La tabla de verdad es la representación de todas las posibles combinaciones de las entradas y las posibles salidas.



Vemos que tenemos cuatro combinaciones que activan el riego, y dentro de cada combinación deben darse 3 condiciones simultáneamente. Ahora tenemos que transformar la tabla de verdad en expresiones booleanas que sintetizan lo que queremos expresar:

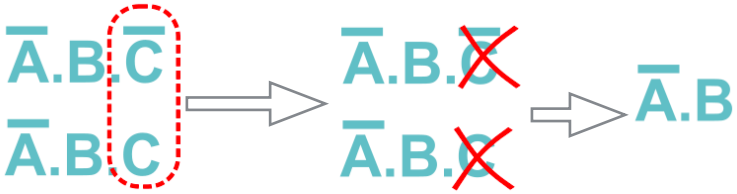


Podemos juntar las cuatro condiciones en una sola expresión:

Qué nos dice esta expresión, que cuando se cumpla la primera condición "o" la segunda "o" la tercera "o" la cuarta el sistema se debe activar. Para que se cumplan cualquiera de las condiciones deben ocurrir simultáneamente tres condiciones, por ejemplo, en el primer caso debe ser de noche "y" hacer calor "y" la tierra está húmeda. Cada "o" es una puerta OR y cada "y" es una puerta AND, para negar las entradas usaremos las puertas NOT.

3. Simplificación de las funciones booleanas.

Antes de montar nuestro circuito tenemos que comprobar si podemos simplificarlo, es decir, eliminar puertas lógicas en su construcción. Si analizamos el ejemplo anterior vemos que el riego se activará siempre sea de noche, hace calor y la tierra está húmeda, pero también cuando es de noche, hace calor y la tierra esta seca. Podemos concluir que para nuestro circuito es indiferente que la tierra este seca o húmeda, siempre que sea de noche y haga calor el sistema regará.

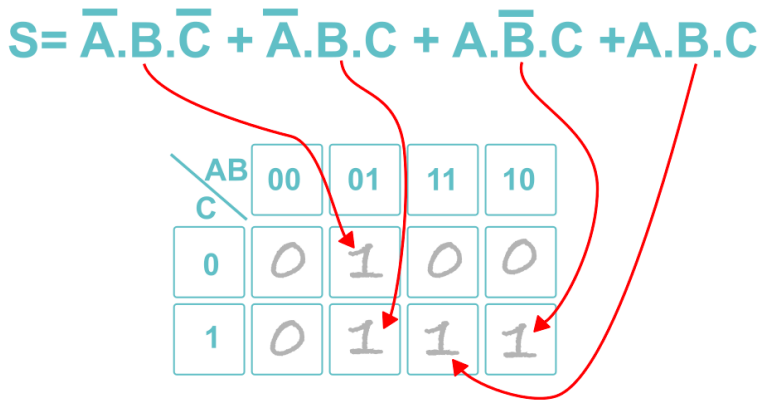


Pero para no cometer errores a la hora de simplificar no debemos hacerlo intuitivamente, sino que debemos usar un método más preciso, es el **método de Karnaugh**, que es un método gráfico en el que utilizaremos unas tablas y siguiendo una serie de reglas, nos permitirá simplificar nuestra función lógica.

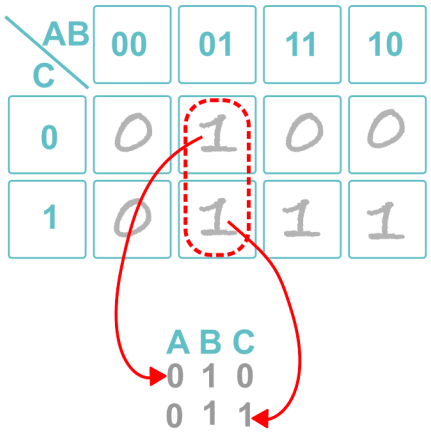
Si nuestro circuito tiene tres entradas la tabla será la siguiente:

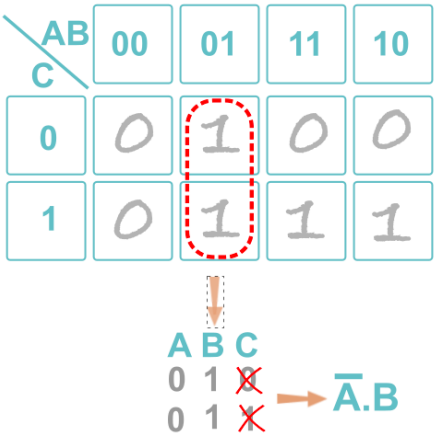
AB / C	00	01	11	10
0				
1				

Colocaremos 1 y 0 en las celdas correspondientes siguiendo lo que nos indica la función lógica que queremos simplificar:

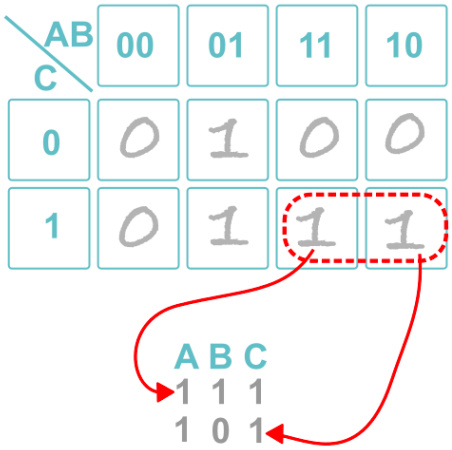


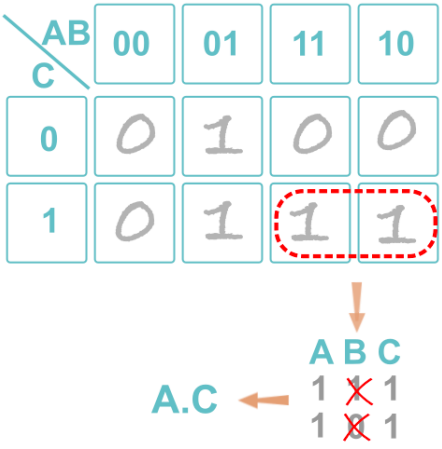
Asociamos los 1 que sean adyacentes en horizontal o vertical, e incluso en esquinas opuestas en la misma fila o columna, siempre en grupos de potencias de 2, es decir, grupos de dos, cuatro u ocho:





También podemos agrupar los otros dos unos que aparecen en horizontal:





Así la función anterior nos quedaría simplificada de la siguiente manera:

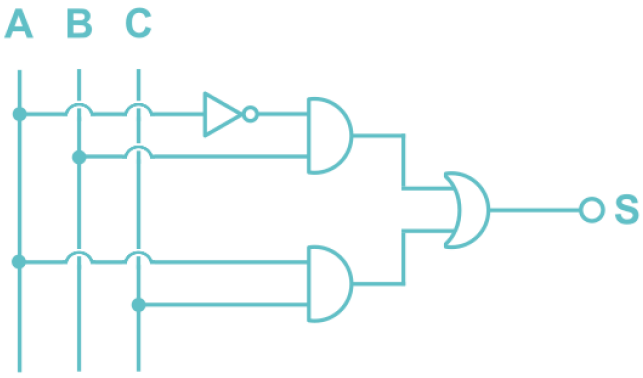
$$S= \overline{A}.B.\overline{C} + \overline{A}.B.C + A.\overline{B}.C + A.B.C$$

$$\downarrow$$

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

4. **Implementación de las funciones booleanas utilizando puertas lógicas.**

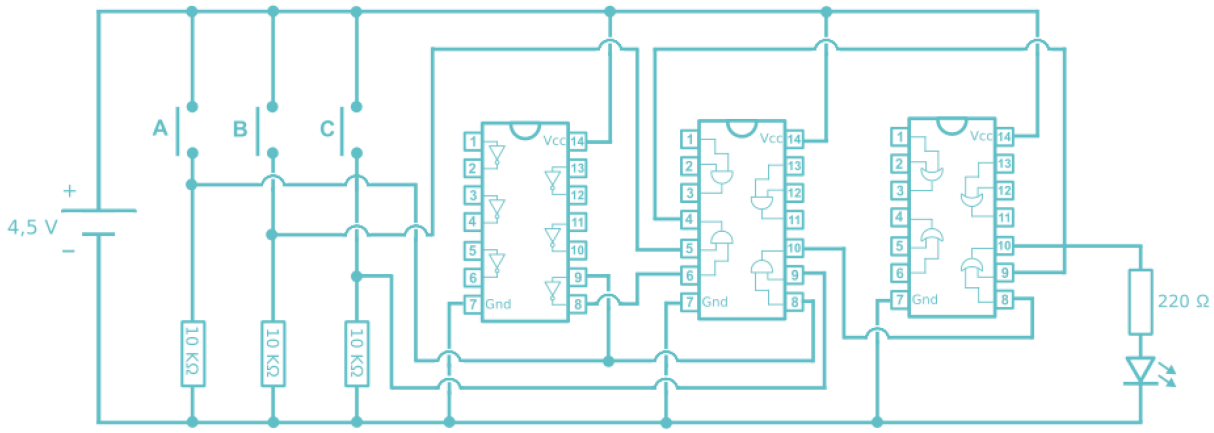
Una vez que tenemos la función lógica simplificada ya podemos implementar el circuito, es decir, representarlo con las puertas lógicas correspondientes. En este caso necesitaremos puertas NOT, AND y OR:



También podríamos sustituir estas puertas lógicas por puertas lógicas universales tipo NOR o NAND como se explicó anteriormente.

5. **Construcción.**

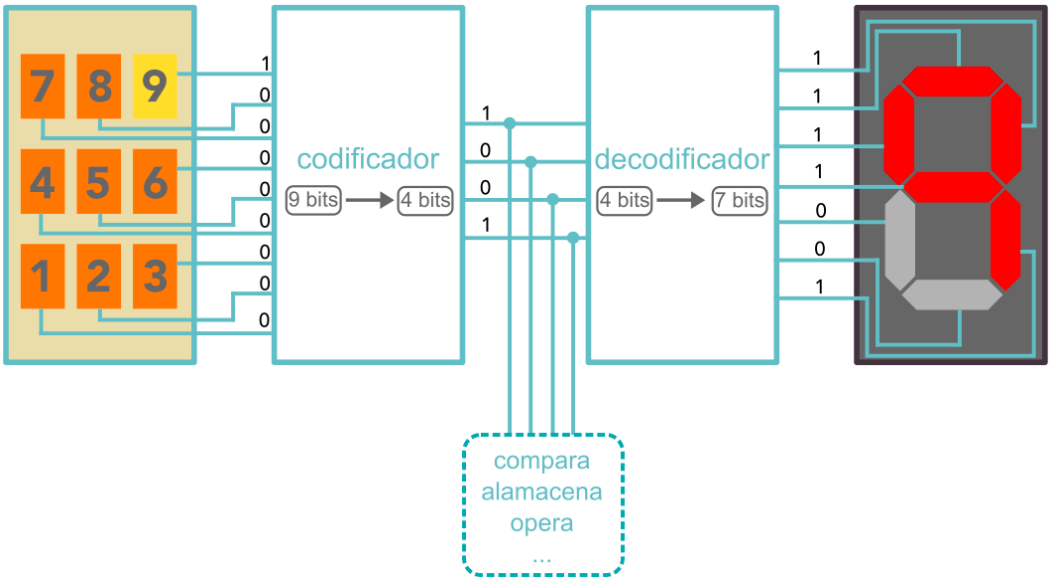
El último paso es llevar este circuito a la realidad, realizando físicamente el montaje. Para ello debemos adquirir los circuitos integrados correspondientes a las puertas lógicas que vamos a usar y conectarlas correctamente:



Otros circuitos digitales: codificadores y decodificadores

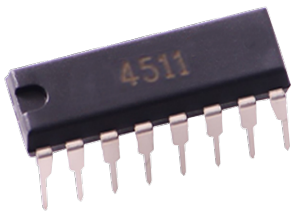
Veamos ahora un ejemplo de uso de un codificador y un decodificador, otro tipo de circuitos combinacionales. La principal característica de este tipo de circuitos es que además de tener múltiples entradas también tienen múltiples salidas, de manera que usamos este tipo de circuitos para transformar una señal digital de 4 bits en otra de 9 bits (codificador) o una de 9 en otra de 7 bits (decodificador).

Por ejemplo, cuando pulsamos un número en una tecla de una calculadora nos aparece en la pantalla dicho número:

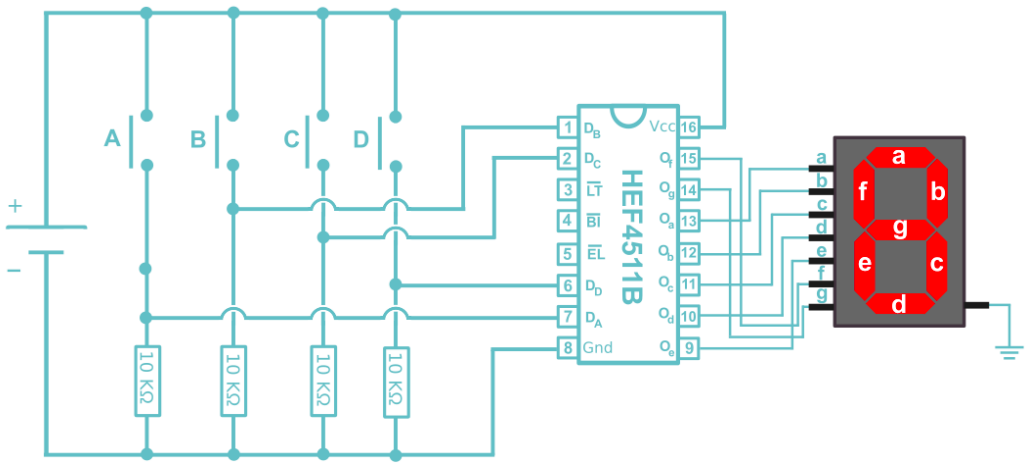


Al pulsar la tecla 9 al sistema llega una señal de 9 bits (100000000) que sería más complicada para realizar operaciones digitales (operar matemáticamente, almacenar en memoria,…) por lo que el **codificador** reduce a 4 bits esa señal (1001), lo que facilita mucho el trabajo al sistema digital. Por último, como queremos que nos aparezca en una pantalla el número pulsado debemos volver a modificar la señal para que tenga 7 bits (1111001). Para pasar de 4 bits a 7 bits usamos el **decodificador**.

Podríamos diseñar nuestro propio codificador o decodificador con puertas lógicas, pero como son de uso muy común ya se comercializan unos circuitos integrados para realizar esa función. Por ejemplo, un decodificador de 4 bits a 7 bits es el circuito integrado HEF4511B, decodificador BCD a display de 7 segmentos.



El código BCD (Decimal Codificado a Binario) es un estándar muy utilizado para representar los dígitos del 0 al 9 en código binario. Para ello necesitamos como mínimo 4 bits. El display de siete segmentos está compuesto por 7 diodos LED nombrados de la "a" a la "g" que según este encendidos o apagados representan los números en sistema decimal.

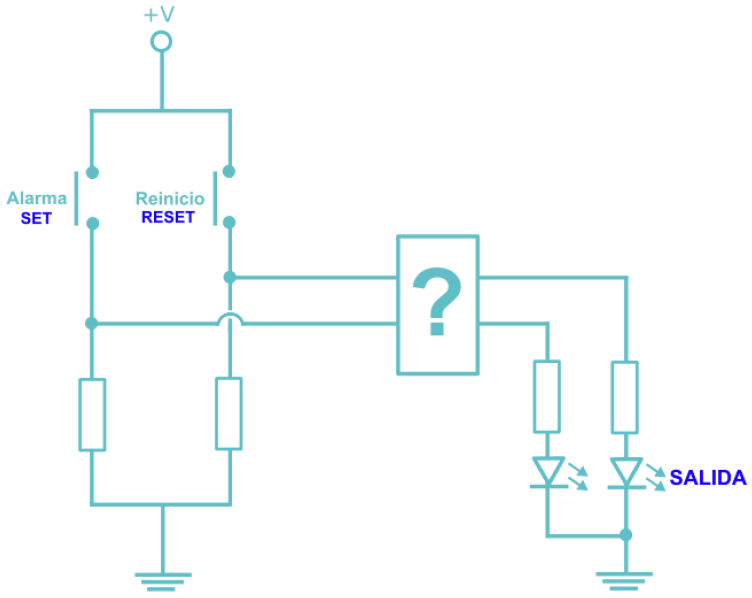


Como decíamos los codificadores y decodificadores tienen múltiples entradas pero también múltiples salidas, que es lo que nos permite en este ejemplo encender o apagar siete diodos LED con un solo chip:

	Código BCD				SALIDA display 7 segmentos							
	D	C	B	A	a	b	c	d	e	f	g	
0	0	0	0	0	1	1	1	1	1	1	0	0
1	0	0	0	1	0	1	1	0	0	0	0	1
2	0	0	1	0	0	0	1	0	0	0	1	2
3	0	0	1	1	1	1	1	1	0	0	1	3
4	0	1	0	0	0	1	1	0	0	1	1	4
5	0	1	0	1	1	0	1	1	0	1	1	5
6	0	1	1	0	1	0	1	1	1	1	1	6
7	0	1	1	1	1	1	1	0	0	0	0	7
8	1	0	0	0	1	1	1	1	1	1	1	8
9	1	0	0	1	1	1	1	1	0	1	1	9

Biestables

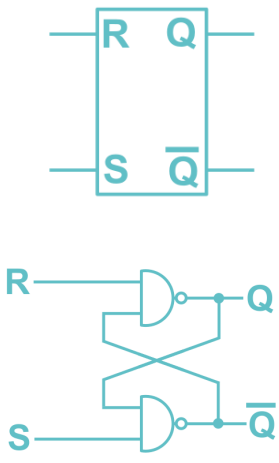
Los biestables son un tipo de circuitos secuenciales, son circuitos con memoria. Imaginemos que queremos diseñar un sistema de alarma en el que cuando se acciona un pulsador se encienda un diodo LED y que permanezca encendido aunque dejemos de accionar ese pulsador. Para esto necesitamos un circuito con memoria, que mantenga el estado de salida pese a que la entrada deje de estar activa, y esto es lo que hacen los **circuitos biestables** o **flip-flop** (en inglés), donde una entrada, denominada SET, activa la salida y se mantiene así hasta que no activemos otra entrada denominada RESET. Además tienen una segunda salida que hace lo opuesto a la primera, aunque no siempre se utiliza.



Existen diferentes circuitos biestables, aquí hablaremos de dos de ellos, la llamada **báscula R-S** y la **báscula J-K**.

Báscula R-S

La báscula R-S recibe este nombre en referencia a las entradas RESET y SET. Tiene el siguiente símbolo, tabla de verdad y circuito equivalente con puertas lógicas NAND:

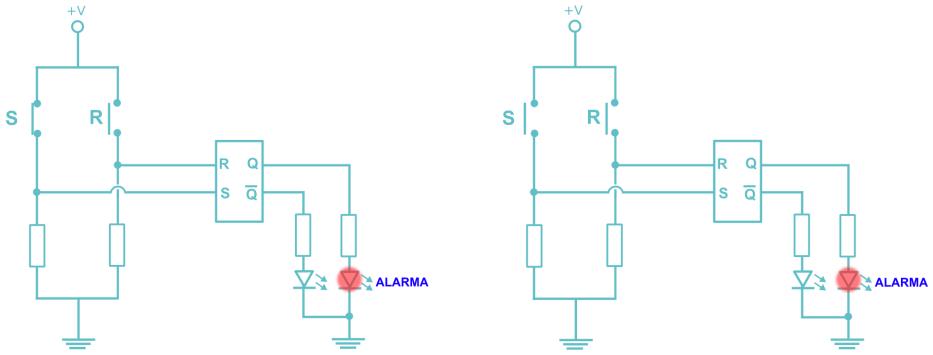


R	S	Q	$\overline{Q}$
0	0	q	N.D.
0	1	1	0
1	0	0	1
1	1	N.D.	q

N.D. = Estado no determinado

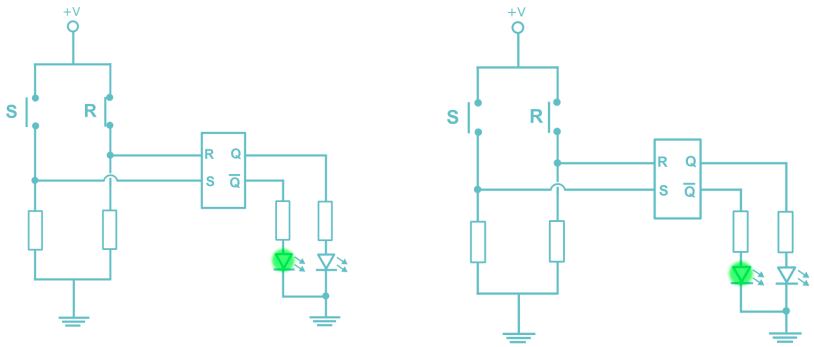
q=Estado anterior

En este circuito al activarse S se activa la salida, por lo que en el ejemplo de la alarma, el diodo LED se encendería:



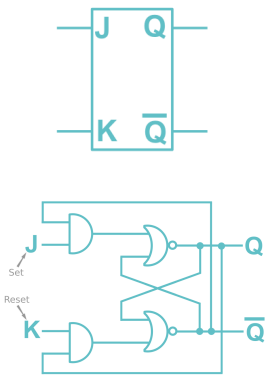
Si deja de estar activa la entrada S, dejando de accionar el pulsador, el diodo LED permanecerá encendido (estamos usando su memoria) mientras no pulsemos RESET y permanecerá en ese estado hasta que se vuelva a pulsar SET:

Este circuito tiene un problema, si pulsamos simultaneamente S y R se produce un estado indeterminado que no se sabe que salida tendría. Para solucionar este problema tenemos la báscula J-K.



Báscula J-K

Se denomina así en honor a **Jack Kilby**, creador del circuito integrado, en la que la entrada J corresponde con el SET y la K con el RESET. El funcionamiento es similar al del circuito anterior salvo que sí tiene una salida para los casos en que pulsamos simultáneamente ambas entradas.



R	S	Q	$\bar{Q}$
0	0	q	$\bar{q}$
0	1	1	0
1	0	0	1
1	1	$\bar{q}$	q