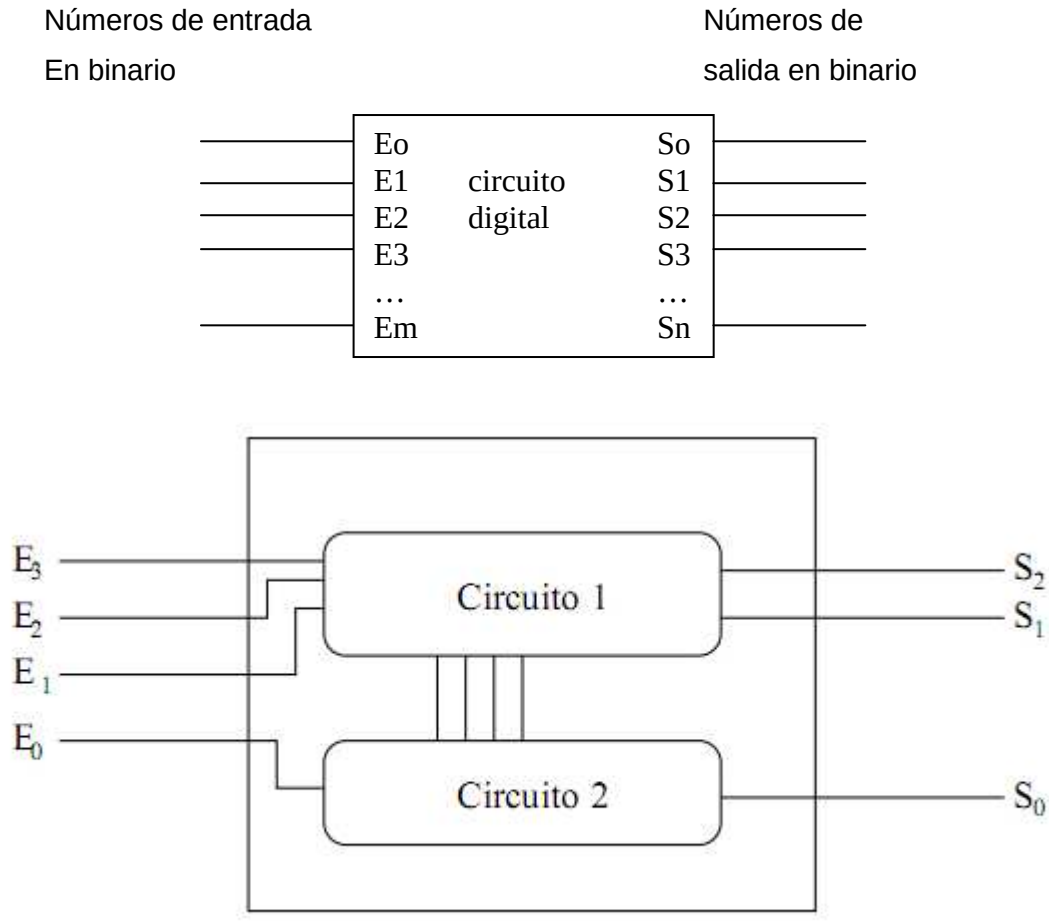
CIRCUITOS COMBINACIONALES Y SECUENCIALES

Los circuitos digitales son circuitos electrónicos que trabajan con números y con la tecnología con la que está realizados, estos números están representados en binario.
Si tomamos un circuito genérico y miramos en su interior, podemos ver que está constituido por otros circuitos más simples, interconectados entre sí.

Estos circuitos se pueden clasificar en dos tipos:

1. Circuitos combinacionales
2. Circuitos secuenciales



Todo circuito digital genérico tendrá una parte combinacional y otra parte secuencial. Primero vamos a ver circuitos combinacionales para después pasar a los secuenciales.

Los **Circuitos Combinacionales** se caracterizan porque no almacenan información. Las salidas están relacionadas con las entradas a través de una función booleana. En cambio algunos circuitos digitales poseen una memoria que les permite recordar su historia anterior, los cambios y evoluciones que hayan experimentado previamente. Son los llamados **Circuitos secuenciales,** cuyo comportamiento es diferente de los Combinacionales.

Las puertas lógicas son los elementos que usamos para construir estos circuitos, y como las funciones booleanas las podemos realizar mediante puertas lógicas, esto se denomina Implementación de funciones booleanas.

CIRCUITOS INTEGRADOS

Las puertas lógicas se encuentran encapsuladas dentro de circuitos integrados o también conocidos como chips. Coloquialmente hablando “cucarachas”, porque son negros con patas.

Hay una familia de circuitos integrados, 74XX, estandarizada. Así pueden existir multitud de fabricantes, pero todos respetando el mismo estándar.

Por las patas denominadas VCC y GND se introduce la alimentación del chip, que normalmente son 5v. Por el resto de patas entra o sale información binaria codificada según la tecnología empleada. Por ejemplo se puede asociar 5v al dígito 1 y 0v al digito 0.

A la hora de fabricar un diseño, estos chips se insertan en una placa y se interconectan las patas con el resto de chips o partes de nuestro circuito. La interconexión se realiza por medio de cables. Cuando se realiza una placa profesional, las interconexiones entre los chips son pistas de cobre en la superficie de la placa. Estas placas reciben el nombre de placas de circuito impreso, o por sus siglas en inglés PCB (printed circuit Board).

Una placa en su parte inferior:

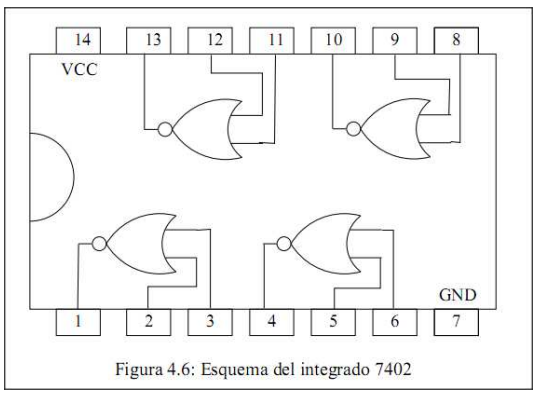
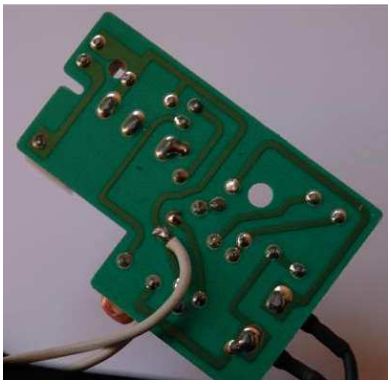


Figura 4.6: Esquema del integrado 7402



Por los agujeros están interconectados por pistas de cobre.
Además existe una capa de un barniz verde para que las pistas no estén al aire y se puedan producir cortocircuitos.

La electrónica ha avanzado muchísimo y en los chips en los que antes sólo se podían integrar unas pocas puertas lógicas, ahora se pueden integrar muchísimas más. Los chips tradicionalmente se han clasificado según el número de puertas que pueden integrar. Así tenemos la siguiente clasificación de chips:

- SSI (Small Scale Integration). Chips con menos de 12 puertas
- MSI (Medium Scale Integration). Entre 12 y 100 puertas.
- LSI (Large Scale Integration). Entre 100 y 10.000 puertas.
- VLSI (Very Large Scale Integration). Más de 10.000 puertas

Los VLSI se corresponden con los microprocesadores y los microcontroladores. Muchos diseños que antes se realizaban solo con electrónica digital, ahora es más sencillo y barato hacerlos con un microprocesador o microcontrolador y programarlos. Es decir, hacer software en vez de hardware.
Existen otras maneras de implementar circuitos digitales sin utilizar los chips tradicionales, es decir, sin tener que recurrir a los chips de la familia 74XX. Esta nueva forma de diseñar se denomina lógica programable.
Existen unos circuitos integrados genéricos (PALs,GALs, CPLDs,FPGAS), que contienen en su interior muchas puertas lógicas y otros componentes. El diseñador especifica los circuitos digitales que quiere diseñar utilizando un lenguaje de descripción hardware. Una herramienta software, conocida como sintetizador, convierte esta descripción en un formato que indica cómo se deben interconectar los diferentes elementos de este chip genérico. El chip “se configura” según se indica en el fichero sintetizado, de manera que ¡¡¡¡nuestra descripción del hardware se ha convertido en un circuito que hace lo que hemos indicado!!!! ¡¡¡Con esta técnica se pueden diseñar desde circuitos simples hasta microprocesadores!!!.

1. CIRCUITOS COMBINACIONALES

En un circuito combinacional el estado lógico de sus salidas, en cada instante depende únicamente del estado de sus entradas. Por consiguiente, en este tipo de circuitos no es necesario tener en cuenta la noción de tiempo. Son funciones lógicas, representables en una tabla de verdad y simplificables mediante la lógica booleana, o por métodos como el de Karnaugh.

En estos sistemas no es posible almacenar el estado de las entradas en un instante y utilizarlo para tomar decisiones posteriormente.

Las aplicaciones de los circuitos combinacionales son de dos tipos:

1. Realización de funciones lógicas, por ejemplo en sistemas de control, donde se procesan entradas y con ello se dan salidas a relés, válvulas …

2. Realización de sistemas en los que, mediante ciertos códigos, se procesan datos representativos de magnitudes numéricas, los cuales se transforman y se someten a operaciones lógicas o aritméticas.

APLICACIONES DE LOS CIRCUITOS COMBINACIONALES. CIRCUITOS DISPONIBLES COMERCIALMENTE.

En el caso de funciones sencillas resulta apropiado realizar los circuitos mediante puertas lógicas. Pero si se trata de funciones más complejas, es más eficaz emplear la gran variedad de circuitos integrados existentes, en combinación con las puertas.

Se intenta sustituir las puertas lógicas por bloques más complejos. El criterio de minimización pretende, de esta manera, conseguir el menor número posible de circuitos integrados.

Los más importantes son:

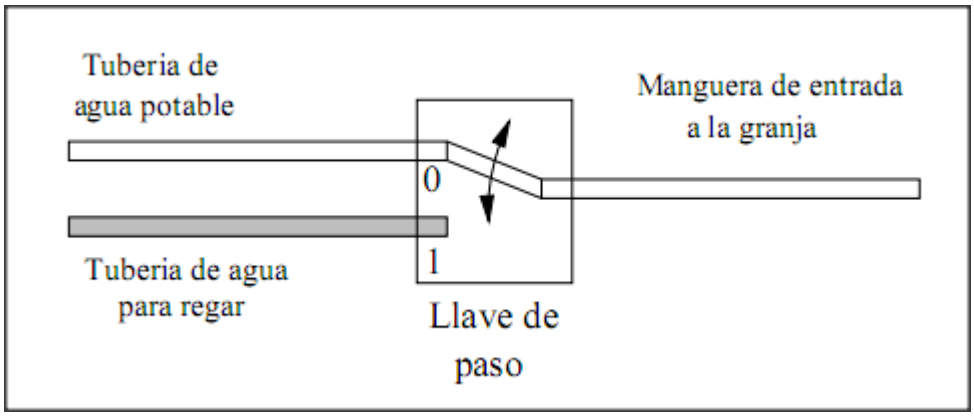
Comparadores
Sumador total y semisumador
Codificadores
Decodificadores
Multiplexores
Demultiplexores

1.1 CIRCUITOS MSI: MULTIPLEXORES Y DEMULTIPLEXORES

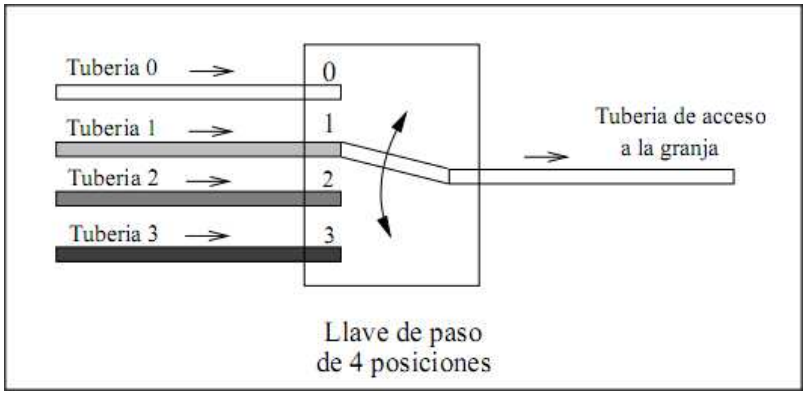
Los circuitos MSI son aquellos que están constituidos por un número de puertas lógicas comprendidos entre 12 y 100.

Un multiplexor es un circuito combinacional al que entran varios canales de datos, y solo uno de ellos, el que hallamos seleccionado, es el que aparece por la salida. Es un circuito que nos permite seleccionar que datos pasan a través de dicho componente.

Vamos a ver un ejemplo NO electrónico, para entender el funcionamiento de un multiplexor. Imaginemos que hay dos tuberías (canales de datos) por el que circulan distintos fluidos (datos). Una transporta agua para regar y la otra agua potable. Estas tuberías llegan a una granja, en la cual hay una única manguera por la que va a salir el agua (bien potable o bien para regar), según lo que seleccione el granjero posicionando la llave de paso en una u otra posición. En la figura se muestra un esquema. Las posiciones son la 0 para el agua potable y 1 para el agua de regar.

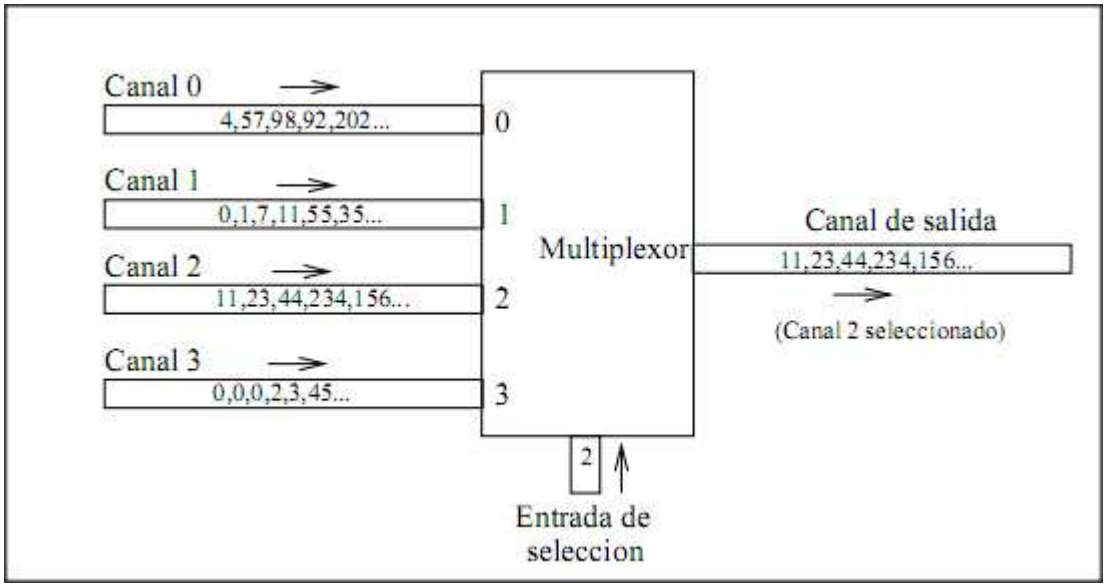


Moviendo la llave de paso, el granjero puede seleccionar si lo que quiere que salga por la manguera es agua potable, para dar de beber al ganado, o agua para regar los cultivos. Según cómo se posicione esta llave de paso, en la posición 0 ó en la 1, seleccionamos una tubería u otra. Pero ¿por qué sólo dos tuberías?. Porque es un ejemplo. A la granja podrían llegar 4 tuberías. En este caso el granjero tendría una llave de paso con 4 posiciones, como se muestra en la siguiente figura.



Esta llave se podría poner en 4 posiciones distintas para dar paso a la tubería 0, 1, 2 ó 3. Obsérvese que sólo pasa una de las tuberías en cada momento, ¡y sólo una!. Hasta que el granjero no vuelva a cambiar la llave de paso no se seleccionará otra tubería.

Por tanto el multiplexor es como una llave de paso que solo conecta uno de los canales de datos de entrada con el canal de datos de salida.
Ahora en vez de tuberías, pensaremos en canales de datos. Fíjate en la figura donde hay 4 canales de datos, y solo uno de ellos es seleccionado por el multiplexor para llegar a la salida.



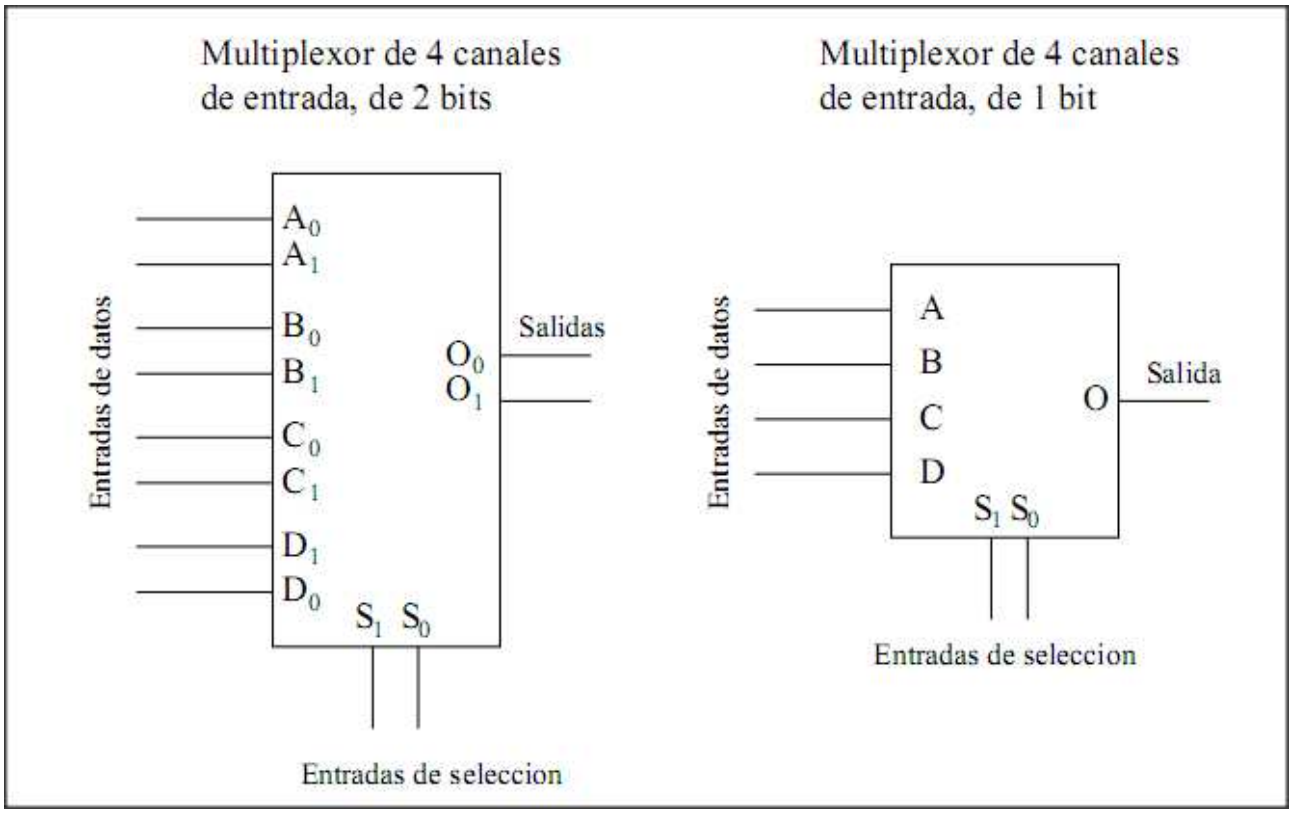
En general, en un multiplexor tenemos dos tipos de entradas:

- Entradas de datos (2^n): por ejemplo las tuberías.
- Entradas de selección (n): indica cual de las entradas se ha seleccionado (posición de la llave de paso).

Al multiplexor le llegan números por las distintas entradas, estos números van siempre expresados en binario y por tanto se podrán expresar mediante bits ¿Cuántos bits? Depende de lo grande que sean los números con los que se quiere trabajar.

En el interior de los microprocesadores es muy normal encontrar multiplexores de 8 bits, que tienen varias entradas de datos de 8 bits. Pero se puede trabajar con multiplexores que tengan 4 bits por cada entrada, o incluso 2, o incluso 1 bit.

En la figura se ven dos multiplexores con 4 entradas de datos y la entrada de selección tiene dos bits (para poder seleccionar entre los 4 canales disponibles). Sin embargo , en uno las entradas de datos son de 2 bits y en el otro de 1 bit.

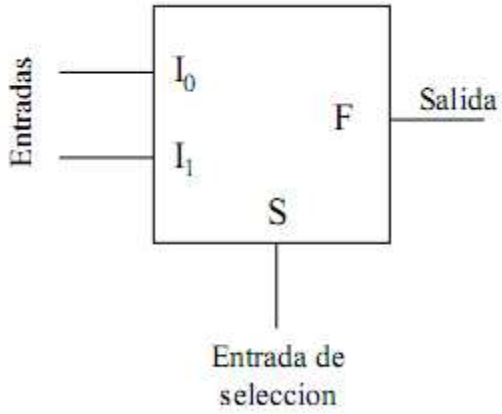


Mirando el número de salidas, podemos conocer el tamaño de los canales de entrada. Así en los dos multiplexores de la figura superior, el de la izquierda tiene 2 bits de salida , por tanto sus canales de entrada son de 2 bits. El de la derecha tiene 1 bit de salida , por tanto los canales de 1 bit.

Nos centraremos en multiplexores con canales de 1 bit.

1.1.1 MULTIPLEXOR CON UNA ENTRADA DE SELECCIÓN

El multiplexor más simple es el que sólo tiene una entrada de selección, S, que permite seleccionar entre dos entradas de datos, según que S=0 ó S=1 . Su aspecto es el siguiente:

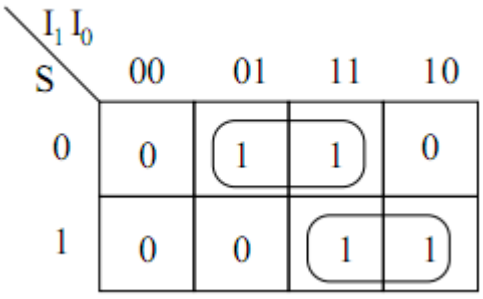


Construyamos la tabla de verdad. Lo primero que nos preguntamos es, ¿Cuántas entradas tenemos? Dos de datos I_0 , I_1 y una es de selección S. Tenemos que rellenar la tabla teniendo en cuenta la definición de multiplexor.

Para S=0 se selecciona la entrada de datos, I_0. Lo que entre por la entrada I_1 será desestimado por el multiplexor. La salida tendrá el mismo valor que la entrada seleccionada I_0. Entonces si I_0=1 entonces F=1.

S	I_1	I_0	F
0	0	**0**	**0**
0	0	**1**	**1**
0	1	**0**	**0**
0	1	**1**	**1**
1	**0**	0	**0**
1	**0**	1	**0**
1	**1**	0	**1**
1	**1**	1	**1**

Se aplica Karnaugh para obtener la expresión más simplificada de F.



$$F = \overline{S} \cdot I_0 + S \cdot I_1$$

La ecuación expresa la definición de multiplexor:

Si S=0 , F=I_0 y si S=1 , F=I_1

El multiplexor se puede expresar de otra forma mediante la siguiente tabla:

S	F
0	I_0
1	I_1

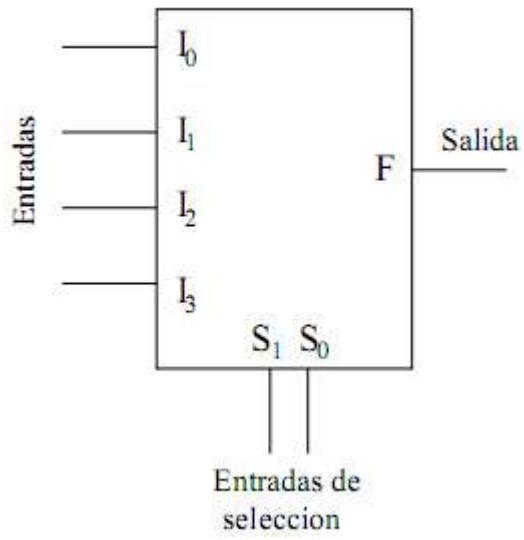
¿Cómo podemos obtener la ecuación del multiplexor a partir de esta tabla?: Aplicando el teorema de expansión,

$$F(S) = S \cdot F(1) + \overline{S} \cdot F(0)$$

F(1) es la salida del multiplexor cuando S=1, es decir, que $F(1)=I_1$ y F(0) es la salida cuando S=0, $F(0)=I_0$. La ecuación del multiplexor es la siguiente:

$$F(S) = S \cdot I_1 + \overline{S} \cdot I_0$$

1.1.2 MULTIPLEXOR DE 4 CANALES o DOS ENTRADAS DE SELECCION



La figura muestra el diagrama de bloques del multiplexor. Las entradas son I_0, I_1, I_2 e I_3 y la selección viene dada por las entradas S_0 y S_1. El valor de la salida *F* depende de los valores lógicos presentes en las entradas de datos y la selección.

La tabla de verdad:

S_1	S_0	F
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

El problema consiste en definir un conjunto de expresiones para construir el circuito lógico. La ecuación en cada fila, se obtiene a partir del dato de entrada y la entrada de selección de datos:

La salida es *F= I_0*, sí *S_1=0* y *S_0=0*. Entonces $F = I_0.\overline{S_1}.\overline{S_0}$

La salida es *F= I_1*, sí *S_1=0* y *S_0=1*. Entonces $F = I_1.\overline{S_1}.S_0$

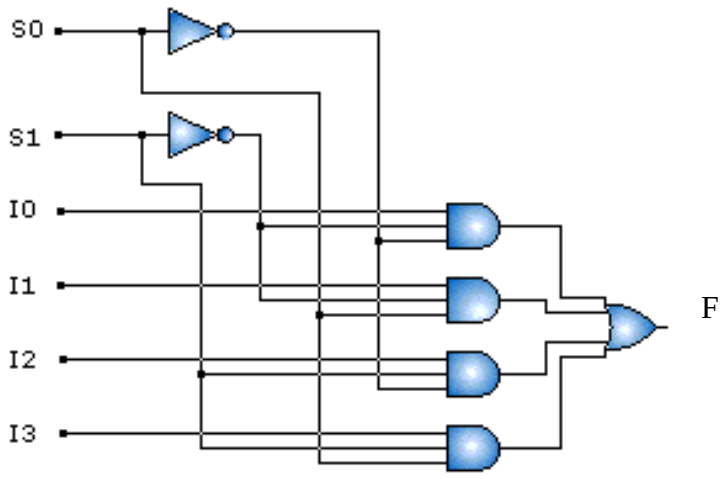
La salida es *F= I_2*, sí *S_1=1 y S_0=0*. Entonces $F = I_2.S_1.\overline{S_0}$

La salida es *F= I_3*, sí *S_1=1* y *S_0=1.* Entonces $F = I_3.S_1.S_0$

Sumando lógicamente las ecuaciones anteriores:

$$F = I_0.\overline{S_1}.\overline{S_0} + I_1.\overline{S_1}.S_0 + I_2.S_1.\overline{S_0} + I_3.S_1.S_0$$

El circuito lógico implementado será:



1.1.3 MULTIPLEXOR 8 CANALES o 3 ENTRADAS DE SELECCION

Multiplexor de 8 entradas de información, 3 entradas de selección y una salida F

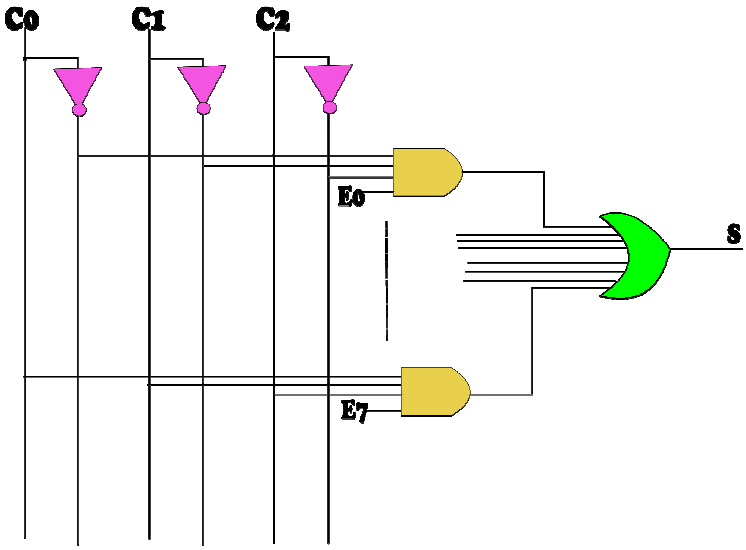
Su tabla de verdad será:

Entrada de selección (control)			**Salida seleccionada**
Co	C1	C2	S
0	0	0	E0
0	0	1	E1
0	1	0	E2
0	1	1	E3
1	0	0	E4
1	0	1	E5
1	1	0	E6
1	1	1	E7

La ecuación de salida será:

$$S = \overline{C0}.\overline{C1}.\overline{C2}.E0 + \overline{C0}.\overline{C1}.C2.E1 + \overline{C0}.C1.\overline{C2}.E2 + \overline{C0}.C1.C2.E3 + C0.\overline{C1}.\overline{C2}.E4 + C0.\overline{C1}.C2.E5 + C0.C1.\overline{C2}.E6 + C0.C1.C2.E7$$

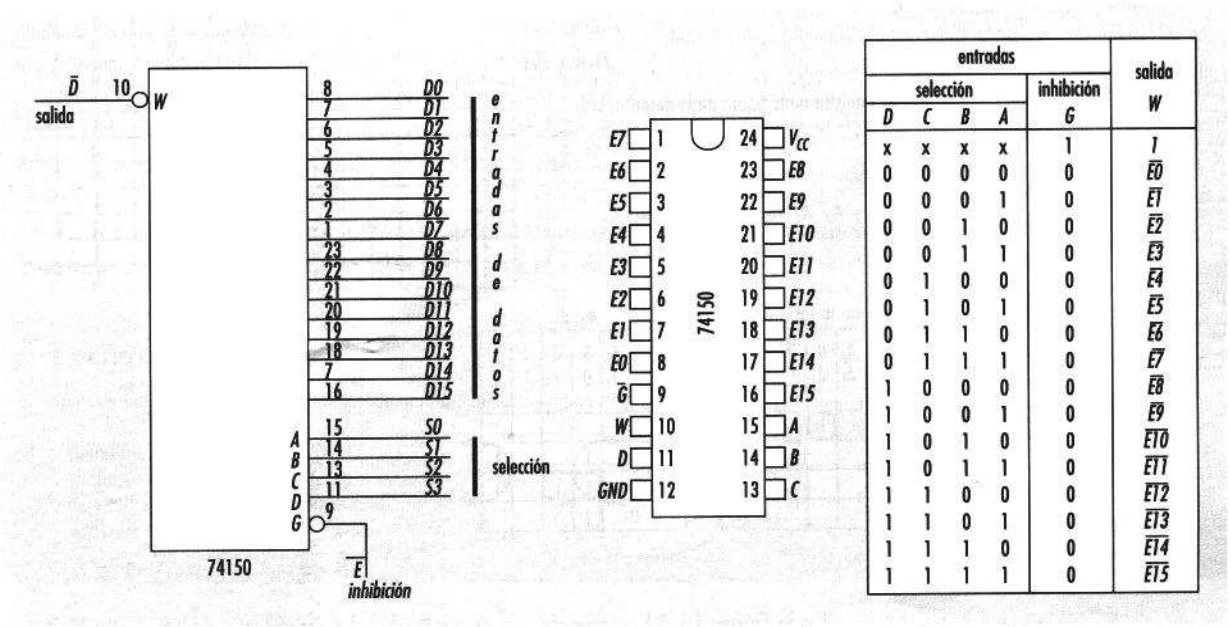
El diagrama lógico será el siguiente:



El circuito consta de ocho puertas AND, conectadas a una puerta OR de ocho entradas, de la cual obtenemos la salida del multiplexador.

Algunos multiplexadores llevan también una **entrada de inhibición**, que hace que la salida sea cero, independientemente de la información que se tenga en las entradas.

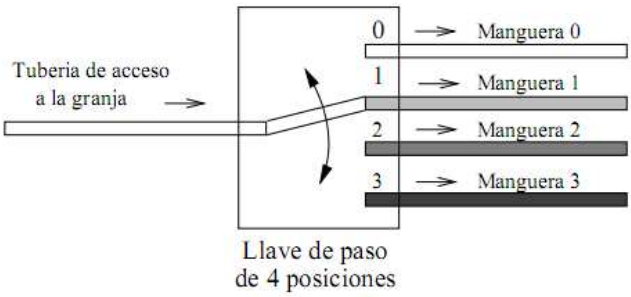
En la siguiente figura se muestra un **multiplexador de 16 entradas** realizado en tecnología TTL del tipo 74150. Incorpora una entrada de inhibición *E* que se activa por nivel bajo y bloquea todas las entradas cuando se encuentra a cero



entradas					salida
selección				inhibición	
D	C	B	A	G	W
x	x	x	x	1	1
0	0	0	0	0	$\overline{E0}$
0	0	0	1	0	$\overline{E1}$
0	0	1	0	0	$\overline{E2}$
0	0	1	1	0	$\overline{E3}$
0	1	0	0	0	$\overline{E4}$
0	1	0	1	0	$\overline{E5}$
0	1	1	0	0	$\overline{E6}$
0	1	1	1	0	$\overline{E7}$
1	0	0	0	0	$\overline{E8}$
1	0	0	1	0	$\overline{E9}$
1	0	1	0	0	$\overline{E10}$
1	0	1	1	0	$\overline{E11}$
1	1	0	0	0	$\overline{E12}$
1	1	0	1	0	$\overline{E13}$
1	1	1	0	0	$\overline{E14}$
1	1	1	1	0	$\overline{E15}$

1.1.4 DEMULTIPLEXORES

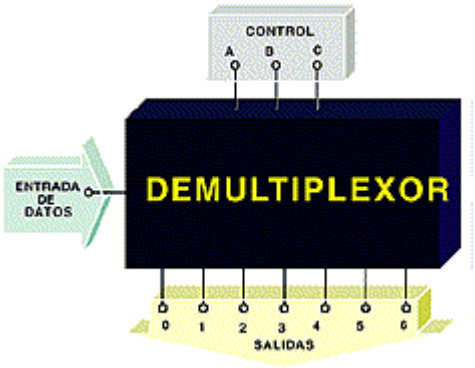
Son circuitos que realizan la función inversa de los anteriores.



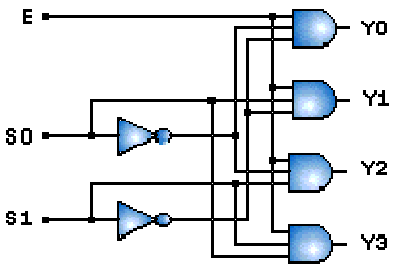
Un demultiplexor es un circuito combinacional que recibe información en una sola línea y la transmite a una de 2^n líneas posibles de salida.

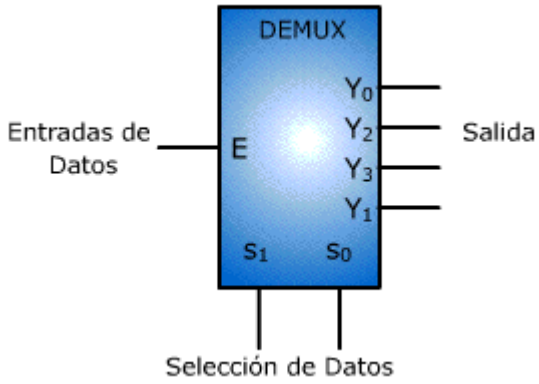
La selección de una línea de salida específica se controla por medio de los valores de los bits de *n* líneas de selección o control. La operación es contraria al multiplexor.

La figura muestra el diagrama de bloques del demultiplexor



Se trata de un **demultiplexor de 1 a 4 líneas**. Las líneas de selección de datos activan una compuerta cada vez y los datos de la entrada pueden pasar por la compuerta hasta la salida de datos determinada. La entrada de datos se encuentra en común a todas las *AND*.





1.2 CODIFICACION, DECODIFICACION Y TRANSCODIFICACION

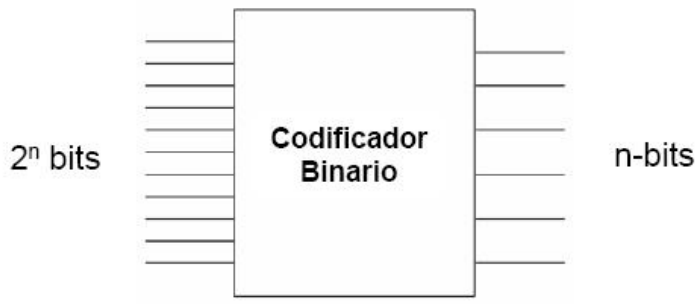
CODIFICAR consiste en establecer una correspondencia entre una información primaria de cualquier tipo, normalmente decimal y una información secundaria siempre en binario. Partimos de una información de cualquier tipo y se obtiene una información binaria. Ejemplo: de decimal a binario o de hexadecimal a binario.

DECODIFICAR: es la operación contraria es decir partiendo de una información binaria obtenemos una información de otro tipo. Ejemplos: de binario a decimal o de binario a hexadecimal.

TRANSCODIFICAR: o convertir el código es partir de una información no binaria a otra información no binaria. Ejemplos: de hexadecimal a decimal o de decimal a hexadecimal.

1.2.1 CODIFICADORES

Un codificador es un circuito integrado combinacional que posee **n salidas** y **2^n entradas**, de forma que al accionarse una de sus entradas, en la salida aparece la combinación binaria correspondiente al número decimal, hexadecimal o binario asignado a dicha entrada. Es decir nos permite compactar la información de entrada.



Los codificadores pueden ser de dos tipos:

a) Sin prioridad: no puede activarse más de una entrada al mismo tiempo, y normalmente no se emplean.

b) Con prioridad: en los que en el caso de producirse una acción simultánea de varias de sus entradas, en la salida se presentará el código de aquella entrada que tenga asignada un mayor peso significativo, normalmente, la de mayor valor decimal.

El valor binario de las salidas puede ser el de cualquiera de los códigos estudiados. En muchas ocasiones si el codificador es de cuatro salidas, el código empleado es el BCD natural o 8421.

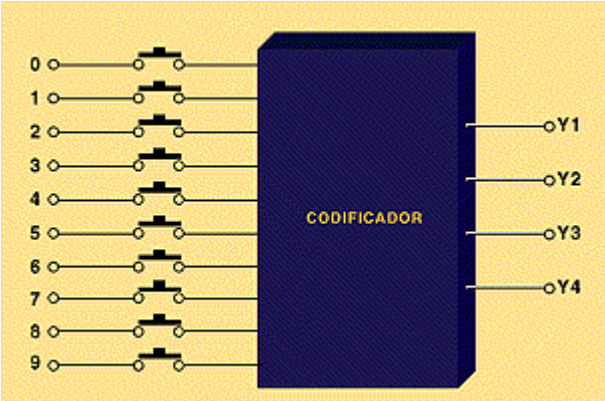
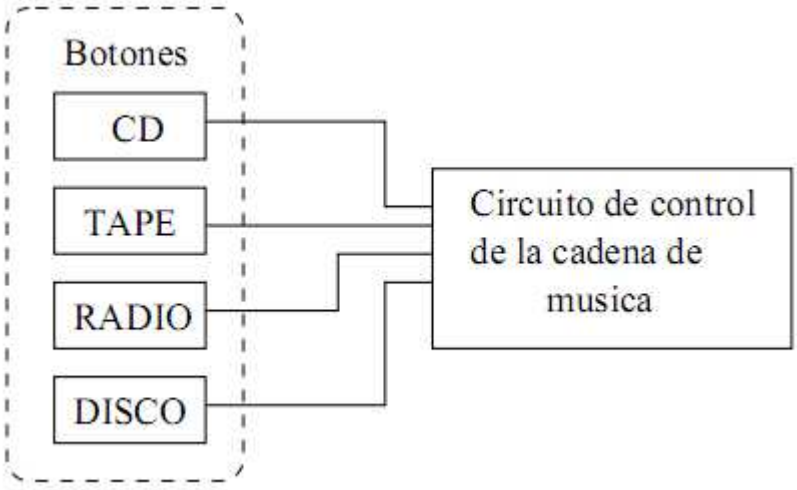


Diagrama de bloques de un codificador de 10 entradas y 4 salidas

EJEMPLOS

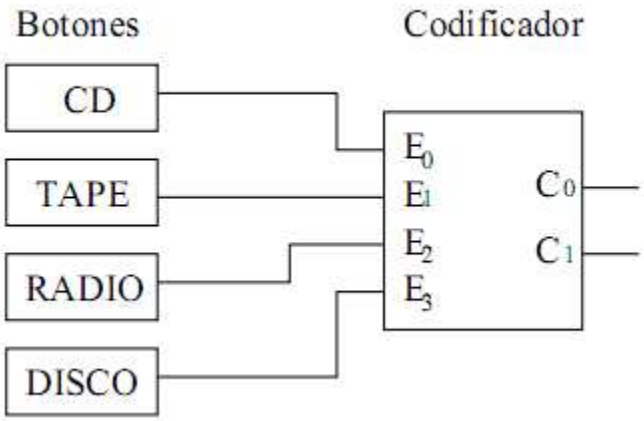
1) Imaginemos que estamos diseñando un circuito digital que se encuentra en el interior de una cadena de música. Este circuito controlará la cadena, haciendo que funcione correctamente.

Una de las cosas que hará este circuito de control será activar la radio, el CD, la cinta o el Disco según el botón que haya pulsado el usuario. Imaginemos que tenemos 4 botones en la cadena, de manera que cuando no están pulsados, generan un '0' y cuando se pulsan un '1' (Botones digitales). Los podríamos conectar directamente a nuestro circuito de control la cadena de música, como se muestra en la figura.



Sin embargo, a la hora de diseñar el circuito de control, nos resultaría más sencillo que cada botón tuviese asociado un número. Como en total hay 4 botones, necesitaríamos 2 bits para identificarlos.

Para conseguir esta asociación utilizamos un codificador, que a partir del botón que se haya pulsado nos devolverá su número asociado:



Fijémonos en las entradas del codificador, que están conectadas a los botones. En cada momento, sólo habrá un botón apretado, puesto que sólo podemos escuchar una de las cuatro cosas. Bien estaremos escuchando el CD, bien la cinta, bien la radio o bien un disco, pero no puede haber más de un botón pulsado. Tal y como hemos hecho las conexiones al codificador, el CD tiene asociado el número 0, la cinta el 1, la radio el 2 y el disco el 3 (Este número depende de la entrada del codificador a la que lo hayamos conectado). A la salida del codificador obtendremos el número del botón apretado. La tabla de verdad será así:

E_3	E_2	E_1	E_0	C_1	C_0	**Botón**
0	0	0	1	0	0	**CD**
0	0	1	0	0	1	**TAPE**
0	1	0	0	1	0	**RADIO**
1	0	0	0	1	1	**DISCO**

El circuito de control de la cadena ahora sólo tendrá 2 bits de entrada para determinar el botón que se ha pulsado. Antes necesitábamos 4 entradas. El codificador que hemos usado tiene 4 entradas y 2 salidas, por lo que se llama codificador de 4 a 2.

Existen codificadores de mayor número de entradas.

2) Imaginemos que ahora queremos hacer un circuito para monitorizar la situación de un tren en una vía. En una zona determinada, la vía está dividida en 8 tramos. En cada uno de ellos existe un sensor que indica si el tren se encuentra en ese tramo (el sensor devuelve 1) o fuera de él (valor 0). Se ve claramente que cuando uno de los sensores esté activado, porque que el tren se encuentre en ese tramo, el resto de sensores devolverán un '0' (No detectan al tren).

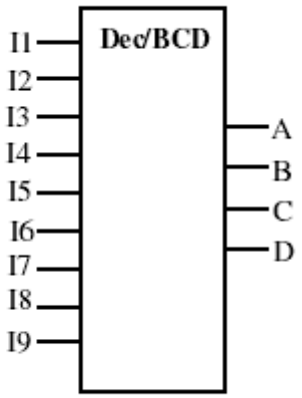
Si conectamos todas las entradas de los sensores a un **codificador de 8 a 3**, lo que tendremos es que a la salida del codificador saldrá un número que indica el tramo en el que se encuentra el tren. El circuito de control que conectemos a las salidas de este codificador sólo necesita 3 bits de entrada para conocer el tramo en el que está el tren, y no es necesario 8 bits. ¡¡Su diseño será más simple!!. La tabla de verdad es:

E_7	E_6	E_5	E_4	E_3	E_2	E_1	E_0	C_2	C_1	C_0	**Tramo**
0	0	0	0	0	0	0	1	0	0	0	**0**
0	0	0	0	0	0	1	0	0	0	1	**1**
0	0	0	0	0	1	0	0	0	1	0	**2**
0	0	0	0	1	0	0	0	0	1	1	**3**
0	0	0	1	0	0	0	0	1	0	0	**4**
0	0	1	0	0	0	0	0	1	0	1	**5**
0	1	0	0	0	0	0	0	1	1	0	**6**
1	0	0	0	0	0	0	0	1	1	1	**7**

En este caso se trata de un codificador completo de **8** bits, o también llamado **codificador de 8 a 3 líneas**. Cuando solo una de las entradas está activa para cada combinación de salida, se le denomina **codificador completo**.

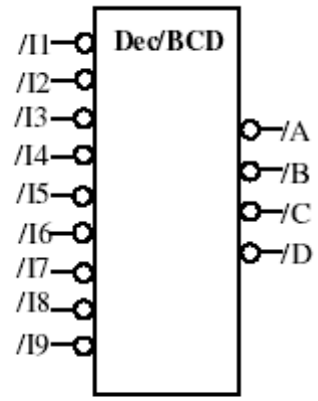
3) En la siguiente figura se representa el diagrama lógico de un **codificador completo de Decimal a BCD Natural,** junto a su tabla de funcionamiento.

I1	I2	I3	I4	I5	I6	I7	I8	I9	A	B	C	D
0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	1
0	1	0	0	0	0	0	0	0	0	0	1	0
0	0	1	0	0	0	0	0	0	0	0	1	1
0	0	0	1	0	0	0	0	0	0	1	0	0
0	0	0	0	1	0	0	0	0	0	1	0	1
0	0	0	0	0	1	0	0	0	0	1	1	0
0	0	0	0	0	0	1	0	0	0	1	1	1
0	0	0	0	0	0	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0	0	1	0	0	1



Por otro lado la figura siguiente representa el diagrama lógico del circuito 74147, que es un **codificador de prioridad de Decimal a BCD Natural**; en la tabla de funcionamiento adjunta se puede notar la diferencia con el anterior.

/I1	/I2	/I3	/I4	/I5	/I6	/I7	/I8	/I9	/A	/B	/C	/D
X	X	X	X	X	X	X	X	0	0	1	1	0
X	X	X	X	X	X	X	0	1	0	1	1	1
X	X	X	X	X	X	0	1	1	1	0	0	0
X	X	X	X	X	0	1	1	1	1	0	0	1
X	X	X	X	0	1	1	1	1	1	0	1	0
X	X	X	0	1	1	1	1	1	1	0	1	1
X	X	0	1	1	1	1	1	1	1	1	0	0
X	0	1	1	1	1	1	1	1	1	1	0	1
0	1	1	1	1	1	1	1	1	1	1	1	0
1	1	1	1	1	1	1	1	1	1	1	1	1



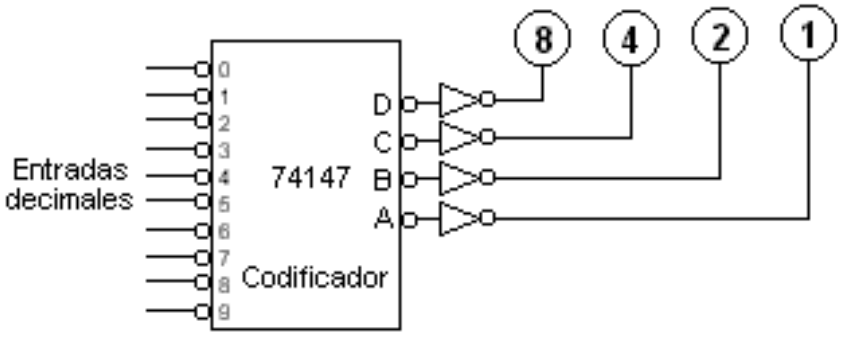


Figura 2: Símbolo lógico del codificador 74147

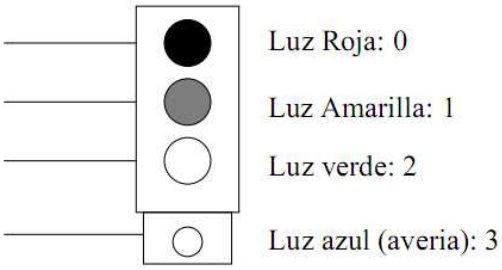
1.2.2. DECODIFICADORES

Un decodificador es un circuito integrado por el que se introduce un número y se activa una y sólo una de las salidas, permaneciendo el resto desactivadas.
Este circuito realiza la operación inversa a la de un codificador de datos y es análoga a la de un demultiplexor, pero sin entrada de información.

Poseen **n entradas** y un **número de salidas menor o igual a 2^n**, y básicamente convierten información codificada en cualquier tipo de código binario en información en otro código que puede ser decimal, hexadecimal (16 salidas) o de 7 segmentos.

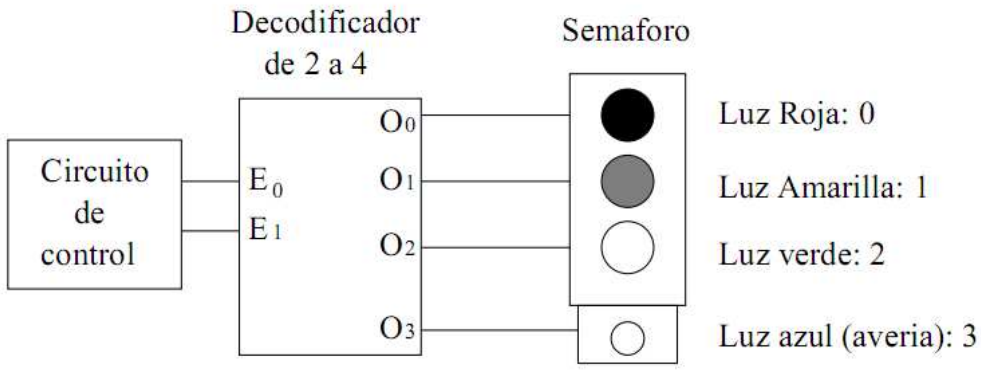
EJEMPLO:

Imaginemos que queremos realizar un circuito de control para un semáforo. El semáforo puede estar verde, amarillo, rojo o averiado. En el caso de estar averiado, se activará una luz interna "azul", para que el técnico sepa que lo tiene que reparar. A cada una de estas luces les vamos a asociar un número. Así el rojo será el 0, el amarillo el 1, el verde el 2 y el azul (averiado) el 3.



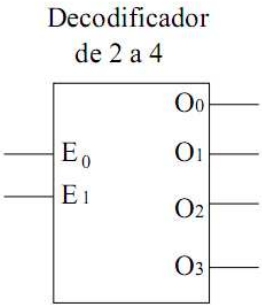
Para controlar este semáforo podemos hacer un circuito que tenga 4 salidas, una para una de las luces. Cuando una de estas salidas esté a '1', la luz correspondiente estará encendida. Sin embargo, ocurre que NO PUEDE HABER DOS O MAS LUCES ENCENDIDAS A LA VEZ.
Por ejemplo, no puede estar la luz roja y la verde encendidas a la vez!!!!.

Si utilizamos un decodificador de 2 a 4, conseguiremos controlar el semáforo asegurándonos que sólo estará activa una luz en cada momento. Además, el circuito de control que diseñemos sólo tiene que tener 2 salidas. El nuevo esquema será:



El funcionamiento es muy sencillo. Si el circuito de control envía el número 2 (E_1=1, E_0=0), se encenderá la luz verde (que tiene asociado el número 2) y sólo la luz verde!!!.
Un decodificador activa sólo una de las salidas, la salida que tiene un número igual al que se ha introducido por la entrada. En el ejemplo del semáforo, si el circuito de control envía el número 3, se activa la salida O_3 y se encenderá la luz azul (y sólo esa!!).
A la hora de diseñar el circuito de control, sólo hay que tener en cuenta que cada luz del semáforo está conectada a una salida del decodificador y que por tanto tiene asociado un número diferente.

Decodificador de 2 a 4



La tabla de verdad es la siguiente:

E_1	E_0	O_3	O_2	O_1	O_0
0	0	0	0	0	**1**
0	1	0	0	**1**	0
1	0	0	**1**	0	0
1	1	**1**	0	0	0

Las ecuaciones de las salidas serán:

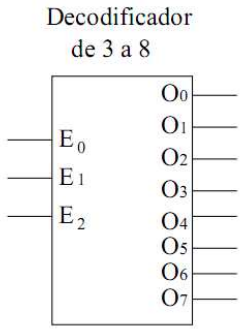
$$O_0 = \overline{E_1} \cdot \overline{E_0}$$

$$O_1 = \overline{E_1} \cdot E_0$$

$$O_2 = E_1 \cdot \overline{E_0}$$

$$O_3 = E_1 \cdot E_0$$

Decodificador de 3 a 8



La tabla de verdad:

E_2	E_1	E_0	Salida Activa
0	0	0	O_0
0	0	1	O_1
0	1	0	O_2
0	1	1	O_3
1	0	0	O_4
1	0	1	O_5
1	1	0	O_6
1	1	1	O_7

Y las ecuaciones son:

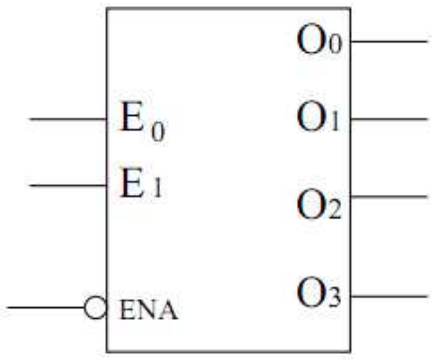
$$O_0 = \overline{E_2} \cdot \overline{E_1} \cdot \overline{E_0}, O_1 = \overline{E_2} \cdot \overline{E_1} \cdot E_0, \ldots, O_7 = E_2 \cdot E_1 \cdot E_0.$$

Decodificador con entrada de validación

Lo mismo que ocurre con los multiplexores y demultiplexores, existe una entrada de validación opcional. Si esta entrada está activada, el decodificador funciona normalmente, pero si está desactivada, sus salidas siempre estarán a ’0’. Existen dos tipos de entrada de validación, las activas a nivel alto y las activas a nivel bajo.

Ejemplo:

Decodificador de 2 a 4 con entrada de validación activa a nivel bajo, por lo el decodificador funcionará siempre que esta entrada esté a ’0’ y todas sus salidas permanecerán desactivadas cuando la entrada de validación esté a ’1’.



Las ecuaciones de este decodificador irán multiplicadas por $\overline{ENA}$, siendo ENA la entrada de validación:

$$O_0 = \overline{E_1} \cdot \overline{E_0} \cdot \overline{ENA}$$

$$O_1 = \overline{E_1} \cdot E_0 \cdot \overline{ENA}$$

$$O_2 = E_1 \cdot \overline{E_0} \cdot \overline{ENA}$$

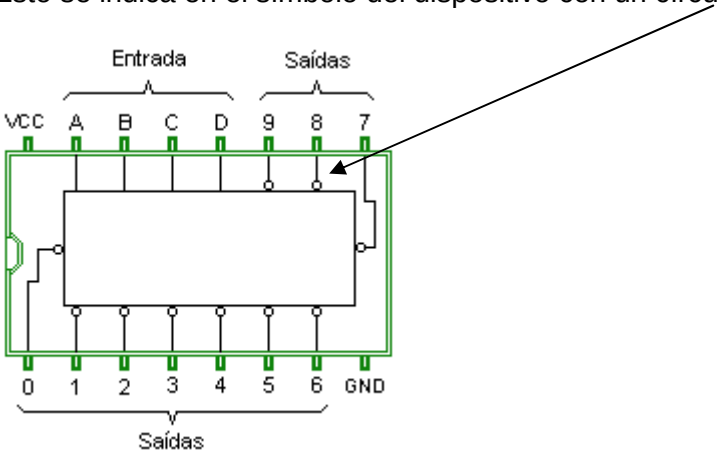
$$O_3 = E_1 \cdot E_0 \cdot \overline{ENA}$$

Cuando por la entrada se introduce un '1' ($ENA = 1$), todas las salidas irán multiplicadas por $\cdot \overline{ENA}$, que vale '0' y todas ellas valdrán '0'. Si se introduce un '1', las ecuaciones serán las de un decodificador de 2 a 4.

Fundamentalmente hay dos tipos de decodificadores:

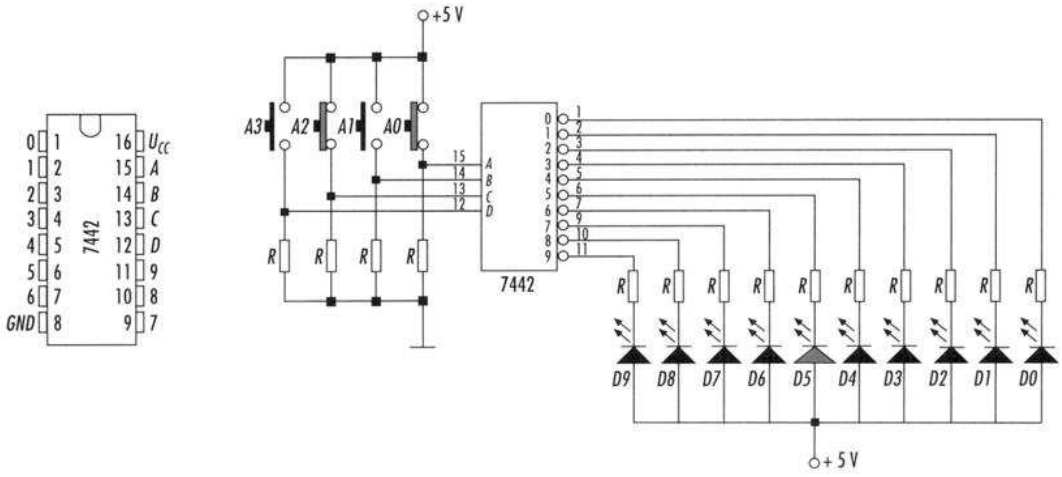
1) Los llamados uno de N en los que para cada combinación de entrada sólo se acciona una de las salidas. Como lo visto anteriormente.

El ejemplo típico es el decodificador BCD-Decimal (de binario a decimal) .Pertenece a la familia TTL. Se denomina 7442. Hay que observar que la salida se activa por nivel bajo (lógica negativa). Esto se indica en el símbolo del dispositivo con un círculo de inversión.



Su tabla de verdad es:

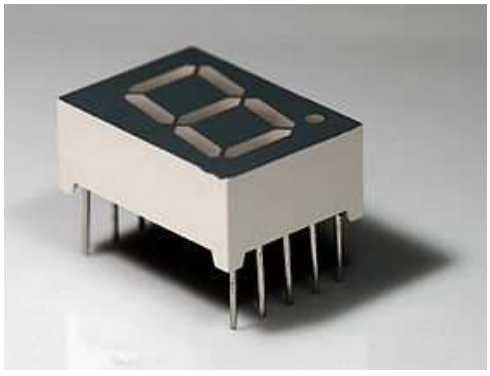
número	entradas				salidas									
	D	C	B	A	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
1	0	0	0	1	1	0	1	1	1	1	1	1	1	1
2	0	0	1	0	1	1	0	1	1	1	1	1	1	1
3	0	0	1	1	1	1	1	0	1	1	1	1	1	1
4	0	1	0	0	1	1	1	1	0	1	1	1	1	1
5	0	1	0	1	1	1	1	1	1	0	1	1	1	1
6	0	1	1	0	1	1	1	1	1	1	0	1	1	1
7	0	1	1	1	1	1	1	1	1	1	1	0	1	1
8	1	0	0	0	1	1	1	1	1	1	1	1	0	1
9	1	0	0	1	1	1	1	1	1	1	1	1	1	0
inválido	1	0	1	0	1	1	1	1	1	1	1	1	1	1
	1	0	1	1	1	1	1	1	1	1	1	1	1	1
	1	1	0	0	1	1	1	1	1	1	1	1	1	1
	1	1	0	1	1	1	1	1	1	1	1	1	1	1
	1	1	1	0	1	1	1	1	1	1	1	1	1	1
	1	1	1	1	1	1	1	1	1	1	1	1	1	1



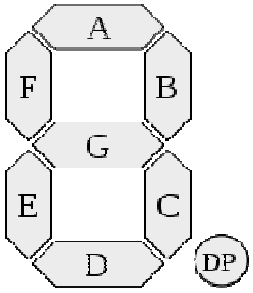
2) Decodificadores en los que se pueden activar varias salidas para una determinada combinación de entrada:

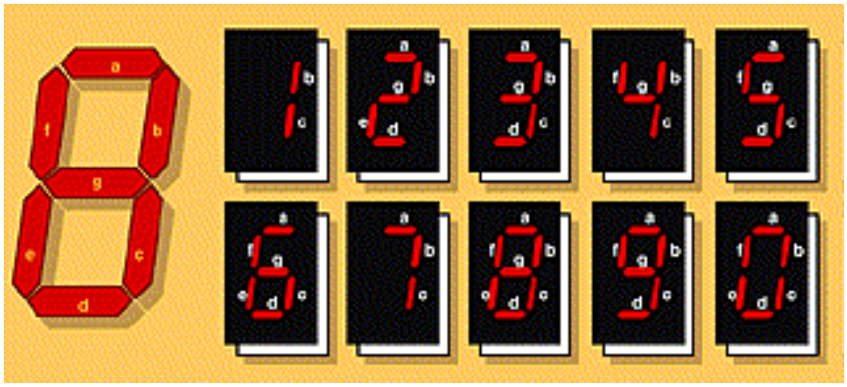
El más representativo BCD-7 segmentos:

Un tipo de decodificador muy empleado. Este circuito decodifica la información de entrada en BCD a un código de siete segmentos adecuado para que se muestre en un visualizador de siete segmentos



El **display de 7 segmentos o visualizador de 7 segmentos** es un componente que se utiliza para la **representación de números** en muchos dispositivos electrónicos debido en gran medida a su simplicidad. Aunque externamente su forma difiere considerablemente de un diodo LED (diodos emisores de luz) típico, internamente están constituidos por una serie de diodos LED con unas determinadas conexiones internas, estratégicamente ubicados de tal forma que forme un número 8.





La representación visual de los diez dígitos decimales se suele realizar a través del denominado código de visualización de siete segmentos

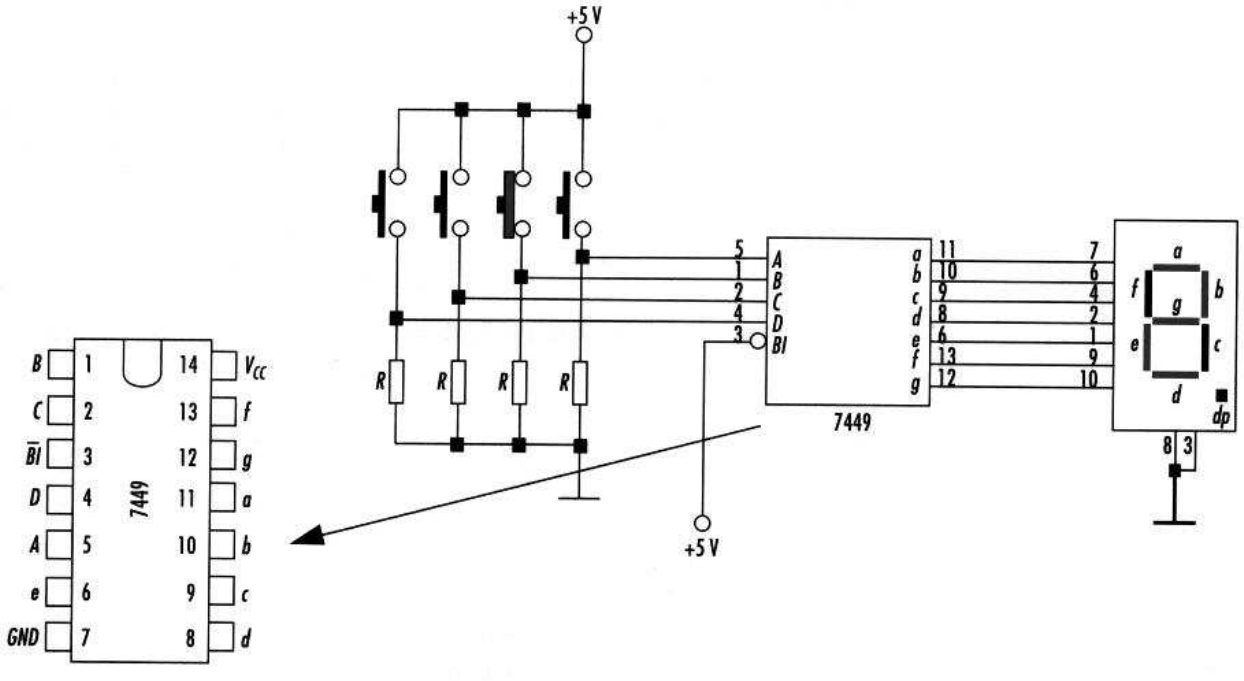
La siguiente figura muestra un decodificador de BCD a código de siete segmentos conectado a un visualizador:

Tanto los segmentos del visualizador como las salidas del decodificador tienen una nomenclatura propia que utiliza las siete primeras letras del alfabeto en minúscula (a, b,c,d,e,f,g). El circuito decodificador es un7449 de la familia TTL. Dispone de una entrada BI activa por 0 para comprobar el estado de los LED. Las salidas son activas por nivel alto .

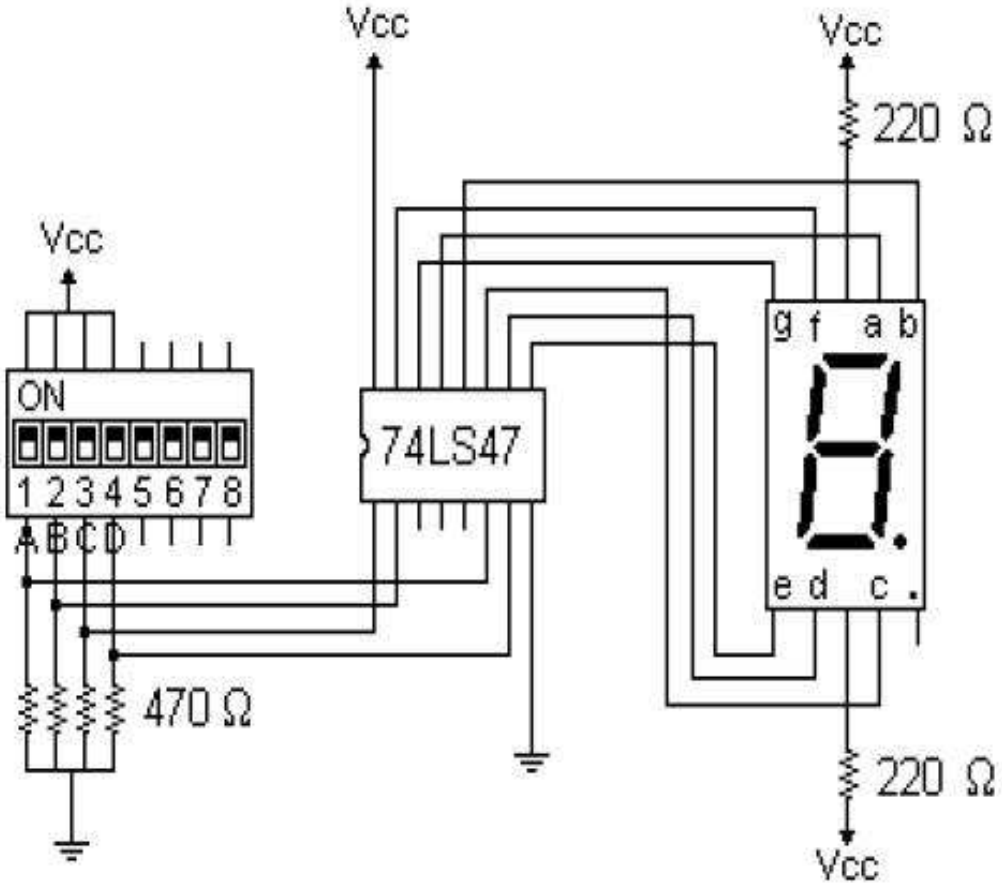
Su tabla de verdad:

número decimal	entradas					salidas						
	D	C	B	A	$\overline{BI}$	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	1	0
1	0	0	0	1	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	1	0	0	1
4	0	1	0	0	1	0	1	1	0	0	1	1
5	0	1	0	1	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	0	1	1	1	1	1
7	0	1	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	0	1	1
10	1	0	1	0	1	0	0	0	1	1	0	1
11	1	0	1	1	1	0	0	1	1	0	0	1
12	1	1	0	0	1	0	1	0	0	0	1	1
13	1	1	0	1	1	1	0	0	1	0	1	1
14	1	1	1	0	1	0	0	0	1	1	1	1
15	1	1	1	1	1	0	0	0	0	0	0	0
	x	x	x	x	0	0	0	0	0	0	0	0

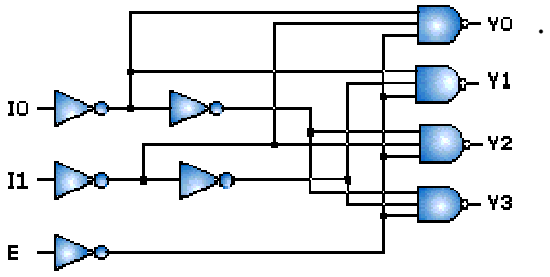
▲ y ◄ Figura 12.28. Descodificador de BCD a siete segmentos (7449).



Otro decodificador posible es:



El decodificador de la figura inferior funciona como un demultiplexor si la línea *E* se toma como línea de entrada de datos y las líneas I_0 e I_1 como líneas de selección. Observe que la variable de entrada E tiene un camino a todas las salidas, pero la información de entrada se dirige solamente a una de las líneas de salida de acuerdo al valor binario de las dos líneas de selección I_0 e I_1.
Por ejemplo si la selección de las líneas $I_0I_1 = 10$ la salida Y_2 tendrá el mismo valor que la entrada *E*, mientras que las otras salidas se mantienen en nivel bajo



E	I_0	I_1	Y_0	Y_1	Y_2	Y_3
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

En consecuencia, como las operaciones decodificador y demultiplexor se obtienen del mismo circuito, un decodificador con una entrada de activación se denomina *decodificador/demultiplexor;* siendo la entrada de activación la que hace al circuito un demultiplexor.

1.3. COMPARADORES

Un comparador digital es un circuito lógico combinacional que es capaz de detectar las relaciones mayor (>), igual (=) y menor (<) entre dos configuraciones binarias.

En esencia, una comparación digital presenta:

- Dos grupos de n líneas de entrada (A y B). Cada grupo de líneas canaliza hacia la entrada del comparador una palabra binaria de n bits.

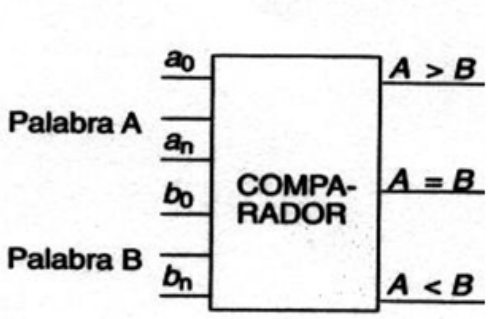
- Tres líneas de salida. Al comparar las dos palabras binarias introducidas en el comparador, el sistema combinacional responderá activando una de las tres salidas siguientes:

A>B : Cuando la palabra binaria A sea de magnitud superior a la de B.

A=B Cuando la palabra binaria A sea igual a la B

A<B Cuando la palabra binaria A sea de menor magnitud a la B

Un comparador digital genérico podría ser:



Realicemos el diseño de un comparador de dos palabras de un bit cada una. En principio realizamos la tabla de verdad correspondiente:

Salida M . Cuando A>B

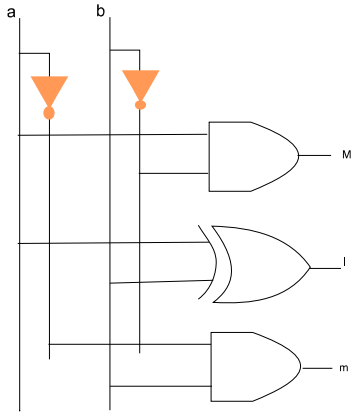
Salida I. Cuando A=B

Salida m. Si A<B

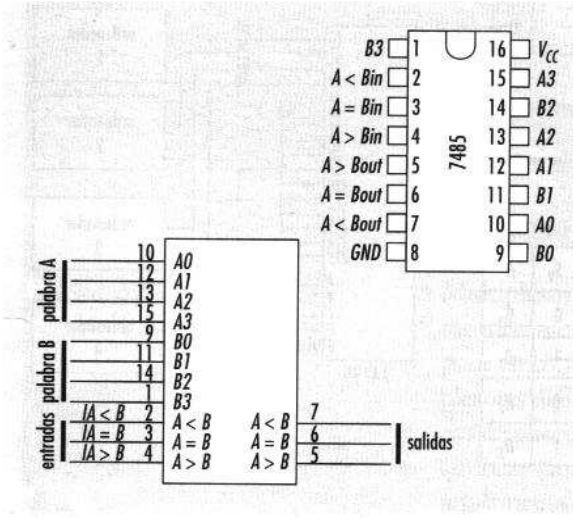
A	B	M	I	M
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

Las ecuaciones de M, I m serán:

$$M = a.\overline{b}$$
$$I = \overline{a}.\overline{b} + a.b = a \otimes b$$
$$m = \overline{a}.b$$

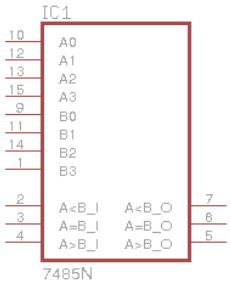


Comparador del tipo 7485 de la familia TTL que permite comparar dos palabras de cuatro bits (8 entradas de datos) y que dispone de 3 entradas auxiliares que permiten conectar los comparadores en cascada para trabajar con palabras de más de 4 bits.

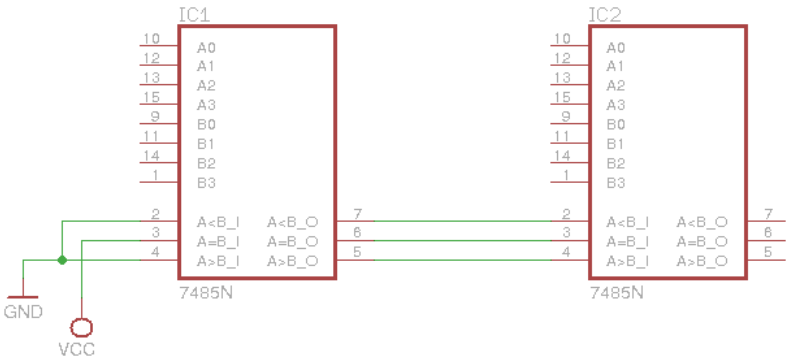


comparadores entradas				cascada entradas			salidas		
A3, B3	A2, B2	A1, B1	A0, B0	A > B	A < B	A = B	A > B	A < B	A = B
A3 > B3	x	x	x	x	x	x	1	0	0
A3 < B3	x	x	x	x	x	x	0	1	0
A3 = B3	A2 > B2	x	x	x	x	x	1	0	0
A3 = B3	A2 < B2	x	x	x	x	x	0	1	0
A3 = B2	A2 = B2	A1 > B1	x	x	x	x	1	0	0
A3 = B3	A2 = B2	A1 < B1	x	x	x	x	0	1	0
A2 = B3	A2 = B2	A1 = B1	A0 > B0	x	x	x	1	0	0
A3 = B3	A2 = B2	A1 = B1	A0 < B0	x	x	x	0	1	0
A3 = B3	A2 = B2	A1 = B1	A0 = B0	1	0	0	1	0	0
A3 = B3	A2 = B2	A1 = B1	A0 = B0	0	1	0	0	1	0
A3 = B3	A2 = B2	A1 = B1	A0 - B0	x	x	1	0	0	1
A3 = B3	A2 = B2	A1 = B1	A0 = B0	1	1	0	0	0	0
A3 = B3	A2 = B2	A1 = B1	A0 = B0	0	0	0	1	1	0

x representa cualquier estado



Comparador de 4 bits



1.4. SUMADOR TOTAL Y SEMISUMADOR

SEMISUMADOR

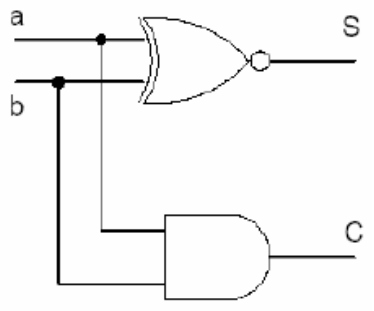
La suma aritmética de dos bits resulta muy sencilla porque estos solo pueden tomar el valor cero y uno. La tabla de la suma en el sistema de base dos es:

Sumandos		Suma binaria	Acarreo
a	b	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

La suma binaria toma el valor uno cuando uno solo de los sumandos tiene dicho valor. Cuando ambos sumandos tienen el valor uno, la suma es cero y se produce un acarreo. De dicha tabla se deducen las expresiones algebraicas de S y C:

$$S = a\bar{b} + \bar{a}b = a \oplus b$$
$$C = ab$$

En las ecuaciones algebraicas de S y C observamos que la suma binaria S es equivalente a la función O-exclusiva y el acarreo C al producto lógico. Al circuito lógico que realiza ambas funciones se le denomina semisumador porque solo permite la suma de dos bits.



SUMADOR TOTAL

Es un sistema combinacional que nos permite sumar dos bits que forman parte de un número binario, para ello es necesario sumar a ambos el acarreo procedente de la suma de los bits de peso inmediato inferior. Su tabla de verdad se muestra a continuación, donde:

C_n representa el acarreo procedente de la etapa anterior.
C_{n+1} el acarreo generado mediante la suma de los bits a , b y el acarreo C_n .

C_n	a	b	S	C_{n+1}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

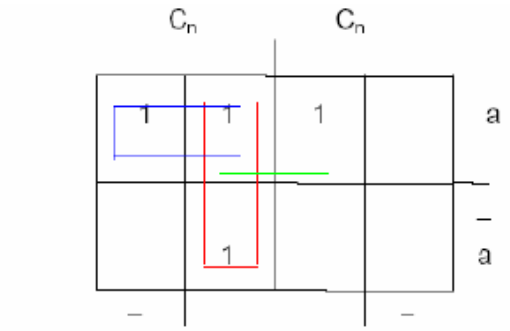
De la tabla se deducen las expresiones de sumas de C_{n+1} y S.

$$S = \overline{C_n}\,\overline{a}\,b + \overline{C_n}\,a\,\overline{b} + C_n\,\overline{a}\,\overline{b} + C_n\,a\,b = C_n\,(\overline{a}\,\overline{b} + a\,b) + \overline{C_n}\,(\overline{a}\,b + a\,\overline{b})$$

$$C_{n+1} = \overline{C_n}\,a\,b + C_n\,\overline{a}\,b + C_n\,a\,\overline{b} + C_n\,a\,b =$$

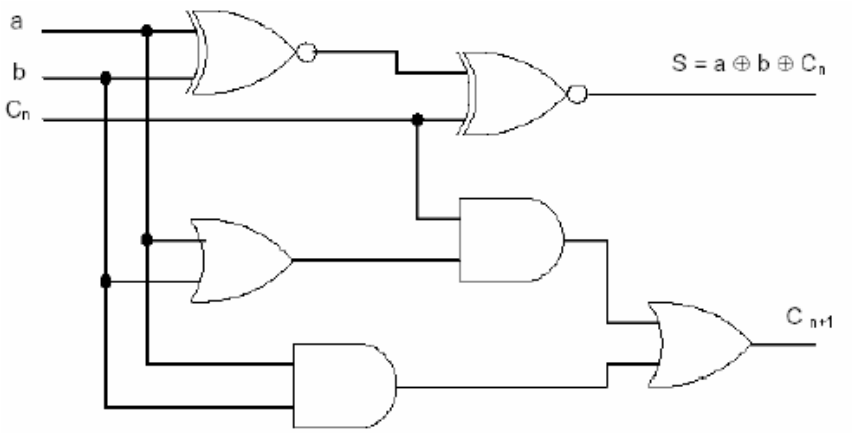
La expresión de S es equivalente a la función o-exclusiva de tres variables.

$$S = a \oplus b \oplus C_n$$

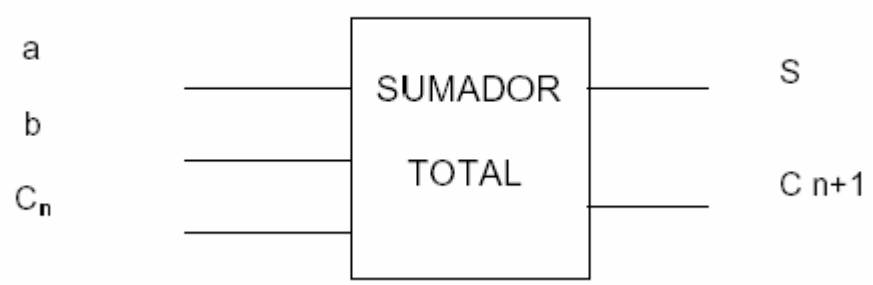


$$C_{n+1} = C_n\,a + C_n\,b + a\,b = C_n\,(a + b) + a\,b$$

La expresión de C_n se puede simplificar al máximo mediante el método numérico o el mapa de Karnaugh.--- graficar el Mapa de karnaugh correspondiente a la tabla de verdad, y hallar C_{n+1}



El sumador total es un circuito combinacional de aplicación general y por ello ha sido realizado en algunas tecnologías de circuitos integrados formando parte de la escala de integración media. Esto permite su utilización como bloque funcional y simplificar de esta forma al máximo el diseño de los sistemas digitales. En lo sucesivo representaremos al circuito sumador mediante el bloque funcional indicado en la siguiente figura:



2. CIRCUITOS SECUENCIALES

En un circuito secuencial el estado de sus salidas depende del estado de sus entradas, pero también depende del estado interno del circuito y de la secuencia con que se introduzcan sus entradas.

Se dice que tienen memoria. Ejemplos son: los contadores de impulsos, una conexión telefónica, la combinación de apertura de una caja fuerte …

Los circuitos secuenciales más elementales son los biestables, que son circuitos construidos a partir de puertas lógicas, y que son capaces de almacenar información binaria de un bit .

2.1. BIESTABLES (flip-flop)

Un **biestable**, también llamado (***flip-flop*** en inglés), es un dispositivo electrónico capaz de permanecer en un estado determinado o en el contrario durante un tiempo indefinido. Esta característica es ampliamente utilizada en electrónica digital para memorizar información.

El paso de un estado a otro se realiza variando sus entradas. Dependiendo del tipo de dichas entradas los biestables se dividen en:

- **Asíncronos:** sólo tienen entradas de control. El más empleado es el biestable RS.

  Un biestable asíncrono tiene poca utilidad o se utiliza en aplicaciones donde realiza una función individualizada. La mayoría de los biestables comercializados son síncronos o como tales forman un conjunto con una función muy específica, como contadores o registros.

- **Síncronos:** además de las entradas de control posee una entrada de sincronismo o de reloj. Si las entradas de control dependen de la de sincronismo se denominan síncronas y en caso contrario asíncronas. Por lo general, las entradas de control asíncronas prevalecen sobre las síncronas.

La **entrada de sincronismo** puede ser **activada por nivel** (alto o bajo) **o por flanco** (de subida o de bajada).

Un biestable es activado por nivel si sólo es necesario que esté presente un valor característico (nivel lógico) de tensión en su entrada de reloj, para que al presentar un nivel lógico en su entrada de información el biestable se dispare.

Si para disparar el biestable es necesario que estando presente la información la entrada de reloj reciba un flanco ascendente o descendente con el cual se dispara, decimos que el biestable está disparado por flanco y en este caso suele recibir el nombre de biestable Edge-Triggered.

Dentro de los biestables síncronos activados por nivel están los tipos **RS** y **D,** y dentro de los activos por flancos los tipos **JK**, **T** y **D**.

Los **BIESTABLES** nos son necesarios para la síntesis de los **circuitos secuenciales**, que son aquellos cuya salida depende de la entrada actual y de las entradas en momentos anteriores. Los **biestables** serán los encargados de almacenar **(MEMORIA)** el **estado interno** del sistema.

Pero aquí nos aparece un concepto nuevo llamado **estado interno** que para poder entenderlo intuitivamente vamos a poner un ejemplo fuera de la electrónica. Si consideramos el sistema BOLIGRAFO podemos definir:

- *el conjunto de entradas:* PULSAR Y NO PULSAR
- *el conjunto de salidas:* SALE PUNTA, ENTRA PUNTA y NO SE MUEVE PUNTA.
- *el conjunto de **ESTADOS INTERNOS**:* PUNTA DENTRO y PUNTA FUERA.

Como puedo observar los estados internos de un sistema me definen todas las situaciones diferenciadas por las que puede pasar o a las que puede evolucionar mi sistema.

Los **biestables** son circuitos binarios (con dos estados) en los que ambos estados son estables de forma que hace falta una señal externa de excitación para hacerlos cambiar de estado. Esta función de excitación define al tipo de biestable (D,T, RS o JK).

En la **electrónica combinacional** no existía el **tiempo**, sin embargo en la **electrónica secuencial** es esencial, la posición relativa en la que ocurren los sucesos (eventos).

Con la introducción anterior podemos definir formalmente **un biestable** *como un circuito secuencial con dos estados estables, es decir tiene **memoria** y una con una salida que puede permanecer indefinidamente en uno de los dos estados posibles. Al ser secuencial las salidas dependen de las entradas y del estado anterior. Un biestable almacena la información de 1 bit.*

Mediante biestables que son la base de los circuitos secuenciales en combinación con una adecuada lógica combinacional podremos construir: contadores, registros de desplazamiento, temporizadores, memorias y en general cualquier autómata.

2.2.BIESTABLES ASÍNCRONOS RS con puertas NAND y NOR

El estado del circuito biestable será el contenido de la memoria. La memoria se consigue mediante la realimentación, o sea introduciendo la salida otra vez a la entrada. Si Q_t es el estado actual o presente y Q_{t+1} el estado futuro entonces se consigue el estado de memoria:

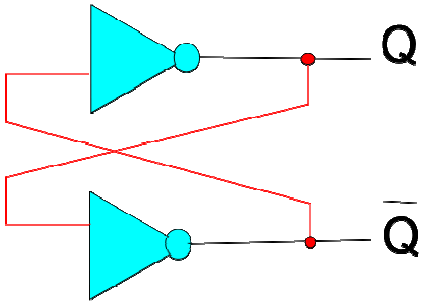


Fig. 1: Configuración básica de estado de memoria

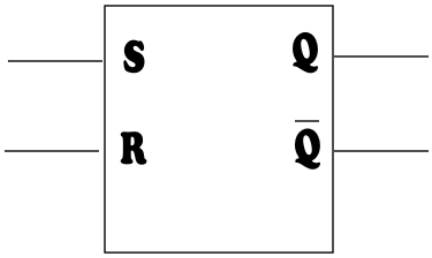
Esta situación de estado de memoria viene dada por la expresión:

$$Q_{t+1} = Q_t$$

$$\overline{Q_{t+1}} = \overline{Q_t}$$

Para poder modificar este estado de memoria debo añadir entradas y así cambiar el estado. Si llamamos a estas entradas **R (reset) y S (set)** obtenemos el **biestable RS**. Los biestables RS se pueden implementar con puertas NOR y NAND.

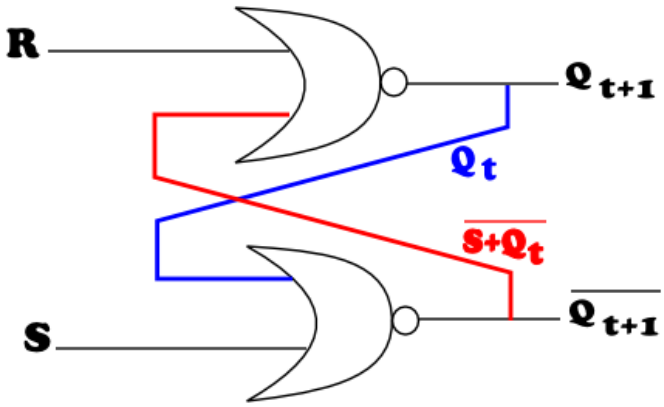
A este tipo de biestables que son activos por nivel se les denomina **LATCH.**



Símbolo biestable RS

CON PUERTAS NOR

Su representación con puertas lógicas es:



La tabla de verdad es la siguiente:

Puesta a 0	Puesta a 1	Estado anterior	Salida	
R	**S**	**Q_t**	**Q_{t+1}**	
0	0	0	0	Memoria
0	0	1	1	
0	1	0	1	Puesta a 1
0	1	1	1	
1	0	0	0	Puesta a 0
1	0	1	0	
1	1	0	X	
1	1	1	x	

La ecuación lógica es la siguiente:

$$Q_{t+1} = \overline{S}.\overline{R}.Q_t + S.\overline{R}.\overline{Q_t} + S.\overline{R}.Q_t$$

Tabla Simplificada:

R	**S**	**Q_{t+1}**
0	0	Q_t
0	1	1
1	0	0
1	1	IND

Simplificamos por Karnaugh:

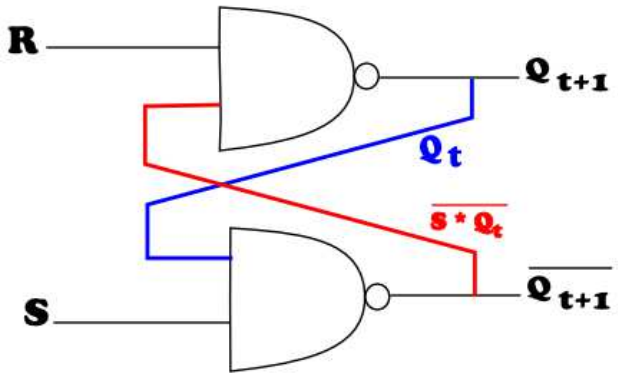
S \ RQ_t	00	01	11	10
0	0	1	0	0
1	1	1	X	x

$Q_{t+1} = S.\overline{R} + \overline{R}.Q_t = \overline{R}(s + Q_t)$ que aplicando la ley de Morgan se convierte en :

$Q_{t+1} = \overline{R + (\overline{S + Q_t})}$ dos puertas Nor de dos entradas.

CON PUERTAS NAND

Su representación con puertas lógicas:



Si analizo la solución del Latch RS con puertas NAND llegaré a la conclusión que se diferencia del anteriormente analizado porque es activo sus entradas a nivel bajo (ceros lógicos).

Tabla de verdad:

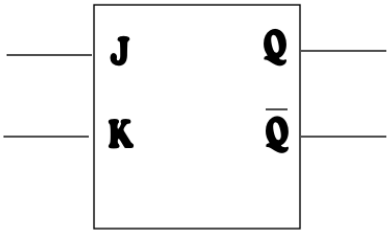
Puesta a 0	Puesta a 1	Estado anterior	Salida
R	**S**	Q_t	Q_{t+1}
0	0	0	X
0	0	1	X
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

2.3.BIESTABLE JK

Consta de dos entradas llamadas J y K y de dos salidas complementadas Q y $\overline{Q}$

El biestable JK se diseña para que la indeterminación que aparece en el RS desaparezca y el valor Q_{t+1} sea el que nosotros queramos.

Símbolo:

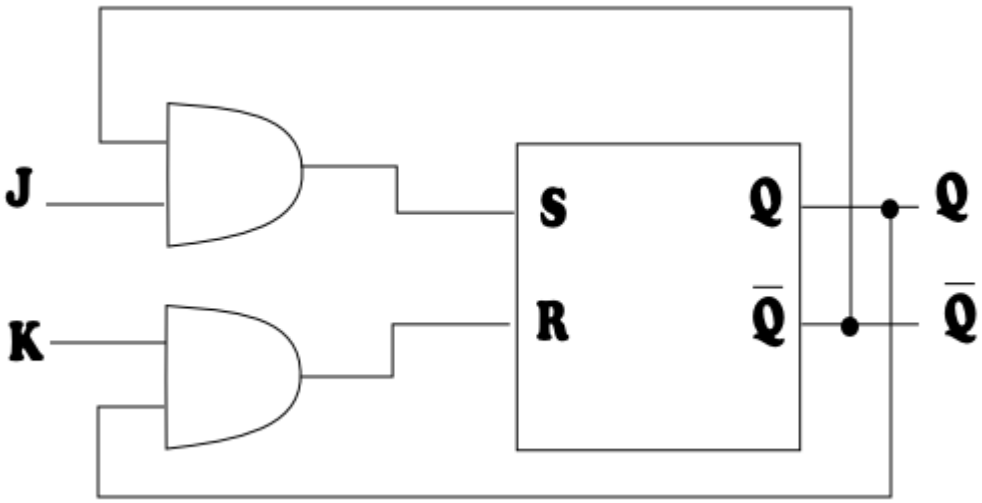


Funciona con los mismos criterios que el biestable RS, pero eliminando la indeterminación en la combinación R=1 S=1.

La tabla de verdad será:

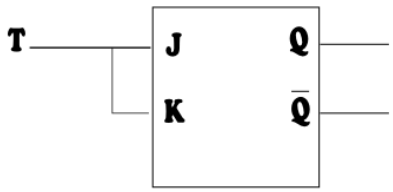
J	K	Q_t	Q_{t+1}	
0	0	0	0	No hay cambios
0	0	1	1	
0	1	0	0	Puesta a cero
0	1	1	0	
1	0	0	1	Puesta a uno
1	0	1	1	
1	1	0	1	Cambia a $\overline{Q_t}$
1	1	1	0	

Su implementación:



2.4. BIESTABLE T

Es un biestable JK que tiene las entradas J y K unidas en una sola T. Cambia de estado cada vez que se active su entrada.



T	Q_t	Q_{t+1}
0	0	0
0	1	1
1	0	1
1	1	0

Su tabla de verdad

Su ecuación lógica: $Q_{t+1} = \overline{Q_t}$

2.5. BIESTABLE D

Es un biestable que tiene una sola entrada denominada D y dos salidas Q y $\overline{Q}$. Se llama D-latch, que significa cerrojo ya que permite pasar a Q_{t+1} lo que hay en D.

Su tabla de verdad:

D	Q_t	Q_{t+1}
0	0	0
0	1	0
1	0	1
1	1	1

2.6. BIESTABLES SÍNCRONOS.

La necesidad de establecer los **instantes de tiempo** en un circuito secuencial basado en biestable nos lleva a la introducción de señales de reloj que nos marcan esos instantes. En cuanto al comportamiento respecto a los instantes de tiempo los circuitos se dividen en:

- **Circuitos asíncronos :** cada variación en las entradas afecta al estado del circuito (es igual a definir un nuevo instante de tiempo)
- **Circuitos síncronos:** Una señal de reloj establece los instantes en los que se modifica el estado del circuito.

Sincronismo por nivel y sincronismo por flanco.

Los circuitos síncronos se dividen a su vez en:

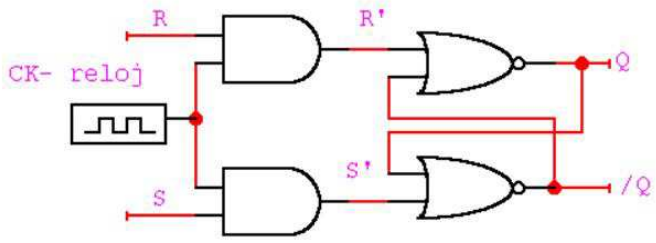
- **Síncronos por nivel:** El instante en el que se modifica el estado del circuito es un semiciclo de reloj.
- **Síncronos por flanco:** El instante en el que se modifica el estado del circuito es un flanco del reloj.

Esto me lleva a la siguiente clasificación de los biestables:

- **Latch:** Se les llama así a los biestables asíncronos o síncrono por nivel. (ver figura 2 el biestable asíncrono RS por nivel).
- **Flip-flop :** Se les llama así a todos los biestables síncronos por flanco.

Biestable RS síncrono por nivel

Se añade una señal de reloj al Latch RS básico (asíncrono) quedando de la siguiente forma (ver figura 3):



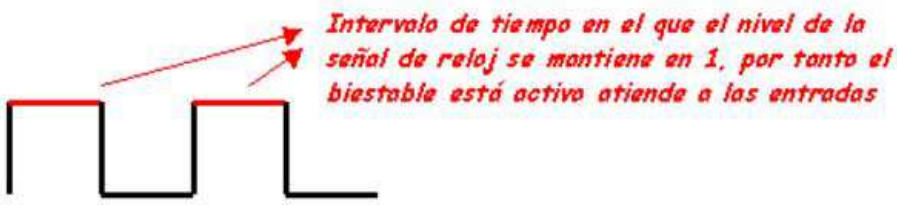


Figura 3. Latch RS síncrono por nivel

Aquí tenemos que:

R'= R.CK

S'= S.CK

Si CK=0 tenemos que R'=S'=0 por lo que nos encontramos es una situación de estado de memoria. Si CK=1 implica que R'=R y S'=S y por tanto el biestable atiende a los valores de entrada y actúa según su tabla de verdad. Todo esto lo resumimos en la siguiente tabla de verdad:

CK	R	S	Q_{n+1}
0	X	X	Q_n
1	0	0	Q_n
1	0	1	1
1	1	0	0
1	1	1	?

Como el tiempo que atiende el biestable a las entradas es todo el semiciclo en alta, si durante ese tiempo se produce un cambio inesperado en las entradas R y S nos puede llevar a una situación errónea. Por tanto para utilizar este tipo de biestables por nivel debo garantizar que las entradas sean estables durante el tiempo que el nivel está en alta.

Una solución a estos problemas es el uso de biestables RS sincronizados por flancos (Flip-flop RS) ya que reduzco el instante de tiempo en el que el biestable atiende las entradas.

Biestables RS síncronos por flancos

En estos biestables se introduce un circuito detector de flancos (ver figura 4):

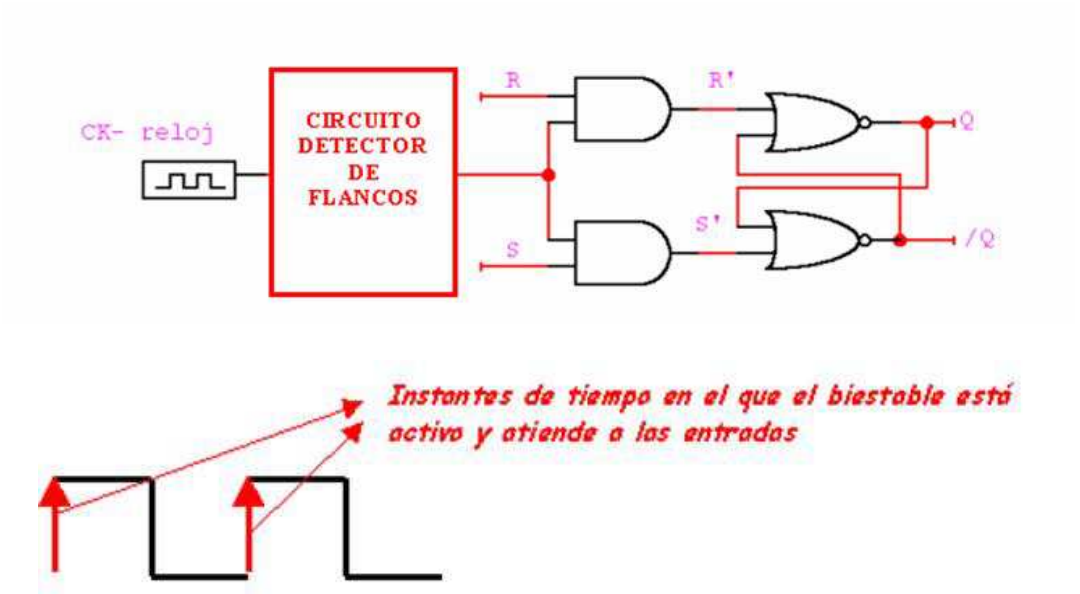
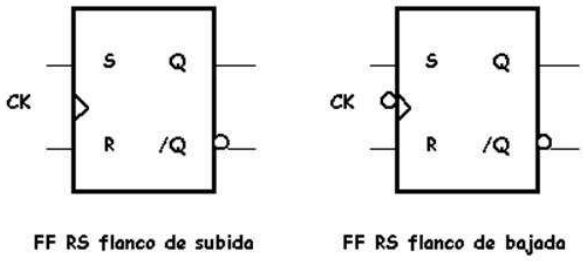


Figura 4. Flip-flop RS

El disparo (activación del FF) se puede dar tanto en el flanco de subida como el de bajada, esta situación viene reflejada en la en la tabla de verdad del FF, como en la siguiente en las que las flechas hacia arriba indican que se utiliza el flanco de subida de la señal de reloj.

CK	R	S	Q_{n+1}
0	X	X	Q_n
↑	0	0	Q_n
↑	0	1	1
↑	1	0	0
↑	1	1	?

De todas formas en la representación del FF RS en los circuitos también podré diferenciarlos



según muestra la siguiente figura:

Figura 5. FF RS por flancos

BIESTABLES JK, T Y D sincronos

Biestable JK

El JK resuelve el caso de indeterminación R=S=1 del RS (la ? de las tablas de verdad) además de ofrecer más posibilidades. Una posible realización del JK sería la siguiente:

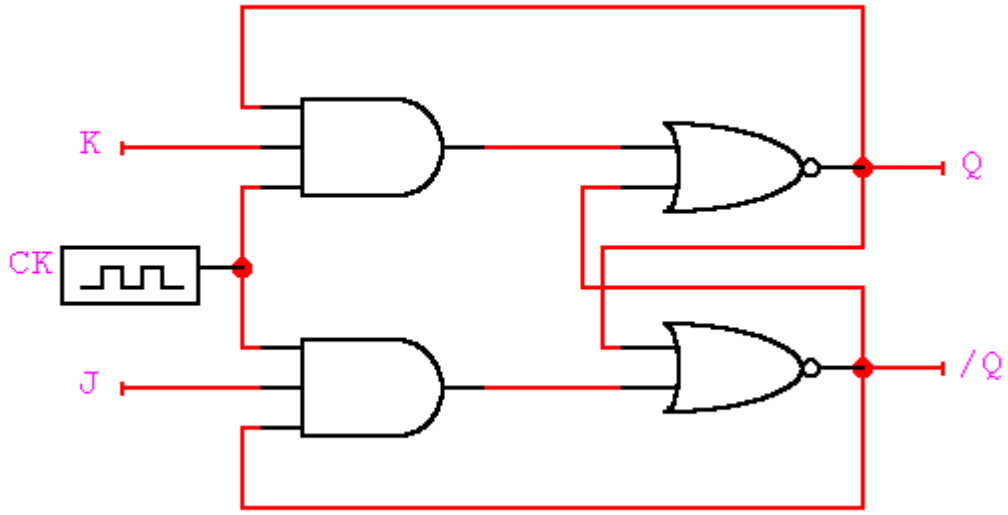


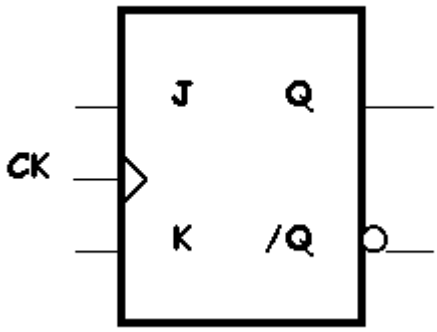
Figura 6. Biestable JK (puede existir versión por flanco o por niveles)

La tabla de verdad o funcionamiento sería la siguiente:

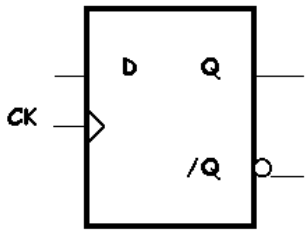
J	K	Qt	Qt+1
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

La ecuación de funcionamiento de la tabla de verdad es:

$$Q_{n+1} = J.\overline{Q_n} + \overline{K}.Q_n$$



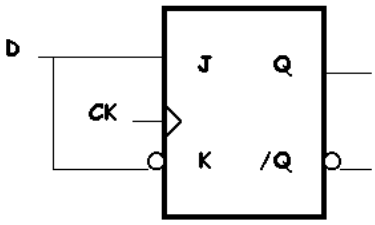
BIESTABLE TIPO D (DELAY = RETARDO)



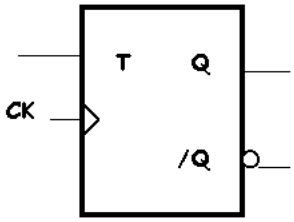
Su tabla de funcionamiento o verdad es la siguiente:

Q_n	D	Q_{n+1}
0	0	0
0	1	1
1	0	0
1	1	1

La ecuación es la siguiente $Q_{n+1} = D$. Puedo obtener un biestable tipo D conectando un JK de la siguiente forma:



BIESTABLE TIPO T (TRIGGER = DISPARO)



La tabla de funcionamiento es la siguiente:

Q_n	T	Q_{n+1}
0	0	0
0	1	1
1	0	1
1	1	0

Siendo su ecuación obtenida de la tabla: $Q_{n+1} = T \oplus Q_n$

También puedo obtener un tipo T a partir de un JK de la siguiente forma: