

ELECTRÓNICA DIGITAL

MAGNITUDES DIGITALES:

Los circuitos electrónicos se pueden dividir en dos amplias categorías: digitales y analógicos. La electrónica digital utiliza magnitudes con valores discretos.

En las aplicaciones electrónicas, las magnitudes digitales tienen ciertas ventajas sobre las analógicas. La principal ventaja es que los datos digitales se pueden procesar y transmitir de forma más eficiente y fiable que los datos analógicos. También poseen otra gran ventaja cuando es necesario proceder a su almacenamiento. Por ejemplo, cuando la música se convierte a formato digital, puede almacenarse de manera más compacta y reproducirse con mayor precisión y claridad de lo que es posible en formato analógico.

El ruido (fluctuaciones de tensión no deseadas) no afecta a los datos digitales tanto como a las señales analógicas.

La electrónica digital utiliza sistemas y circuitos en los que sólo existen dos estados posibles. Estos estados se representan mediante dos niveles de tensión diferentes: **ALTO (HIGH)** y **BAJO (LOW)**. Estos dos estados pueden representarse también mediante niveles de corriente, interruptores abiertos o cerrados, o lámparas encendidas o apagadas. En los sistemas digitales, las combinaciones de estos dos estados se denominan códigos, y se utilizan para representar números, símbolos, caracteres alfabéticos y cualquier otro tipo de información. El sistema de numeración de dos estados se denomina binario y los dos dígitos que emplea son 1 y 0. Un dígito binario se denomina bit.

Dígitos binarios:

Los dos dígitos del sistema **binario**, 1 y 0, se denominan **bits**, que es la contracción de las palabras *binary digit* (dígito binario). En los circuitos digitales, se emplean dos niveles de tensión distintos para representar los dos bits. Un 1 se representa mediante un nivel de tensión más elevado, que se denomina nivel ALTO (HIGH), y un 0 se representa mediante un nivel más bajo de tensión, que se denomina BAJO (LOW). Este convenio recibe el nombre de **lógica positiva**, y es la que se va a emplear a lo largo del libro.

ALTO (HIGH) = 1 y **BAJO (LOW) = 0**

Al sistema, mucho menos común, en el que un 1 se representa por un nivel BAJO y un 0 por un nivel ALTO se lo denomina *lógica negativa*.

Los grupos de bits (combinaciones de 1s y 0s), llamados *códigos*, se emplean para representar números, letras, símbolos, instrucciones y cualquier otra cosa que se requiera en una cierta aplicación.

Niveles Lógicos:

Las tensiones que se utilizan para representar los unos y ceros reciben el nombre de niveles lógicos. Lo ideal sería que un nivel de tensión representara el nivel ALTO (HIGH) y otro nivel de tensión representara el nivel BAJO (LOW). Sin embargo en un circuito digital práctico, un nivel ALTO puede ser cualquier tensión entre un máximo y mínimo especificados. De igual manera, un nivel BAJO puede ser cualquier tensión comprendida entre un máximo y un mínimo especificados. No puede existir solapamiento entre los niveles ALTOS aceptados y los niveles BAJOS aceptados.

La Figura 1 ilustra el rango general de variación para los niveles BAJO y ALTO en un circuito digital. La variable $V_{H(máx)}$ representa el máximo valor para el nivel ALTO y $V_{H(mín)}$ representa el mínimo valor para el nivel ALTO. El valor máximo del nivel BAJO se representa mediante $V_{L(máx)}$ y su valor mínimo $V_{L(mín)}$. Los valores de tensión comprendidos entre $V_{L(máx)}$ y $V_{L(mín)}$ no son aceptables para un funcionamiento correcto. Una tensión dentro de este rango podría interpretarse tanto como nivel ALTO cuanto como nivel BAJO en un determinado circuito. Por tanto, estos valores no aceptables no deben emplearse nunca. Por ejemplo, los valores del nivel ALTO para un cierto tipo de circuito digital denominado TTL

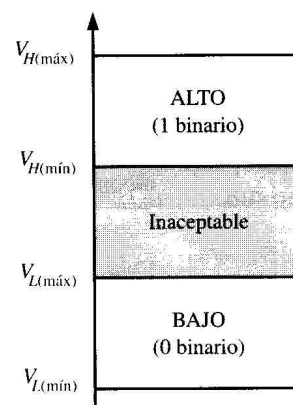


Fig. 1 Rango de niveles lógicos

pueden variar desde 2 V a 5 V mientras que los valores del nivel BAJO lo pueden hacer entre 0 V y 0,8 V. De esta manera, si por ejemplo se aplica una tensión de 3,5 V, el circuito lo interpretará como un nivel ALTO (HIGH) o 1 binario. Si se aplica una tensión de 0,5 V, el circuito lo interpretará como un nivel BAJO (LOW) o 0 binario. Para este tipo de circuito, las tensiones comprendidas entre 0,8 V y 2 V no son aceptables y nunca deben utilizarse.

Formas de onda digitales:

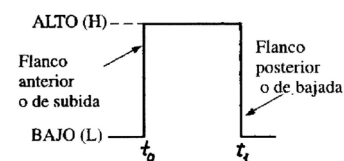
Las señales digitales consisten en niveles de tensión que varían entre los estados o niveles ALTO y BAJO. La Figura 2(a) muestra que un **pulso positivo** se genera cuando la tensión (o la intensidad) pasa de su nivel normalmente BAJO hasta su nivel ALTO, y luego otra vez retorna al nivel BAJO. El pulso negativo de la Figura 2(b) se genera cuando la tensión pasa de su nivel ALTO hasta el nivel BAJO, y vuelve otra vez al nivel ALTO. Una señal digital está compuesta por una serie de pulsos.

Pulsos. Como se indica en la figura 2, un pulso posee dos flancos: un **flanco anterior**, que se produce en el instante t_0 , y un **flanco posterior** que se produce en el instante t_1 . Para un pulso positivo, el flanco anterior es un flanco de subida y el flanco posterior es de bajada. Los pulsos de la figura 2 son ideales, ya que se supone que los flancos de subida y bajada ocurren en un tiempo cero (instantáneamente). En la práctica, estas transiciones no suceden de forma instantánea, aunque para la mayoría de las situaciones podemos asumir que son pulsos ideales.

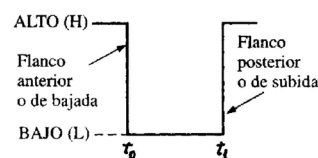
La Figura 3 muestra un pulso real (no ideal). En lo que sigue se han respetado los subíndices de las abreviaturas que se encuentran en los manuales de los componentes digitales con que se trabaja. El tiempo requerido por el pulso para pasar del nivel BAJO al nivel ALTO recibe el nombre de **tiempo de subida** (t_r) (*Rise time*), mientras que el tiempo requerido para la transición del nivel ALTO al nivel BAJO se denomina **tiempo de bajada** (t_f)

(*Fall time*). En la práctica, el tiempo de subida se mide como el tiempo que tarda en pasar del 10% (distancia desde la línea base) al 90% de la **amplitud** del pulso, y el tiempo de bajada se mide como el tiempo que tarda en pasar del 90% al 10% de la amplitud del pulso, como se puede observar en la Figura 3. La razón de que el 10% inferior y el 10% superior no se incluyan dentro de los tiempos de subida y bajada se debe a la no linealidad de la señal en estas áreas. La **anchura del pulso** (t_w) (*Pulse Width*) es una medida de la duración del pulso y, a menudo, se define como el intervalo de tiempo que transcurre entre los puntos en los que la amplitud es del 50% en el flanco de subida y en el de bajada, como se indica en la Figura 3.

En la Figura 4 se señalan las definiciones del tiempo de propagación, en este caso de un inversor. En la parte superior de la figura se muestra el pulso que se aplica a la entrada. En la parte inferior el pulso de salida invertido correspondiente a una puerta inversora, donde se puede observar que los tiempos de elevación y caída (t_{THL} y t_{TLH}) son finitos. Los subíndices tienen el siguiente significado: T transición (*Transition*); L bajo (*low*), H alto (*High*). También transcurre un tiempo entre que se aplica el pulso de entrada y se obtiene el de salida conocido como tiempo de propagación y se especifican el tiempo de propagación de alto a bajo (t_{PHL}) y el tiempo de propagación de bajo a alto (t_{PLH}), donde el nuevo subíndice P corresponde a propagación (*propagation*).



(a) pulso positivo



(b) pulso negativo

Fig.2 Pulsos ideales

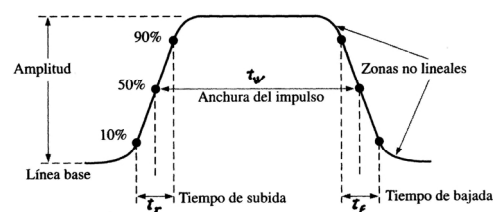


Fig. 3 Característica del pulso real

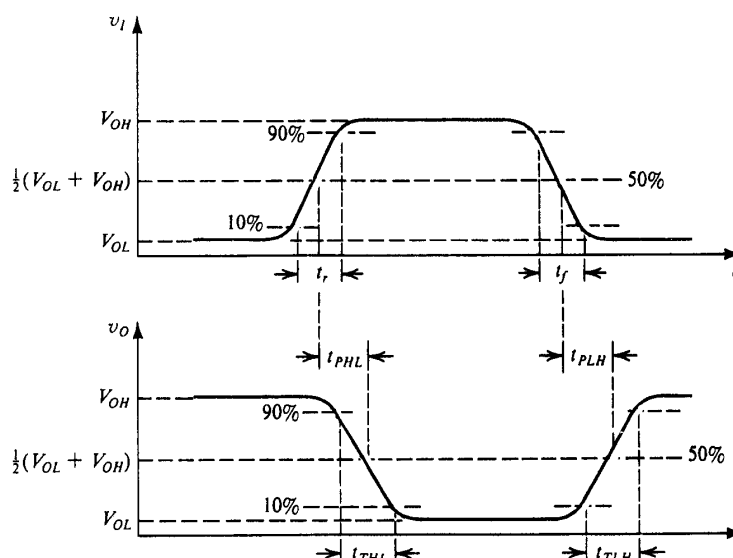


Fig.4 Definición del tiempo de propagación de un inversor

Características de las formas de onda: La mayoría de las señales que podemos encontrar en los sistemas digitales se compone de series de pulsos, algunas veces llamados también *trenes de pulsos* y pueden clasificarse en periódicas y no periódicas. Un tren de pulsos **periódico** es aquel que se repite a intervalos de tiempo fijos, este intervalo de tiempo fijo se denomina **periodo** (T). La frecuencia (f) es la velocidad a la que se repite y se mide en hercios (Hz). Un tren de pulsos no periódico, claro está, no se repite a intervalos de tiempo fijos, y puede estar compuesto por pulsos de distintos anchos y/o pulsos que tienen intervalos diferentes de tiempo de pulsos. Un ejemplo de cada tipo se muestra en la Figura 5.

La **frecuencia** (f) de un tren de pulsos (digital) es el inverso del periodo. Las relaciones entre frecuencias y periodo se pueden expresar de la siguiente manera:

$$f = \frac{1}{T}$$

$$T = \frac{1}{f}$$

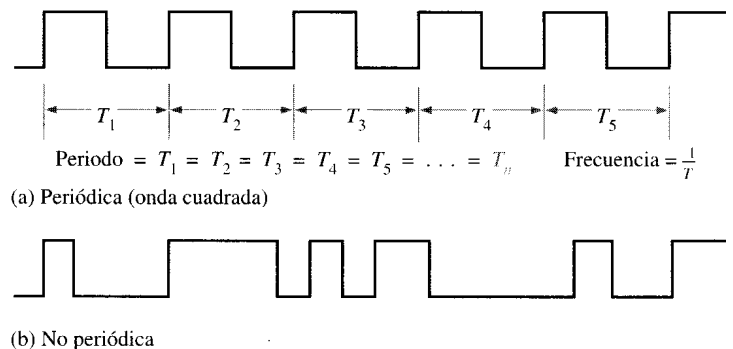


Fig. 5 Forma de ondas

Una característica importante de una señal digital periódica es su ciclo de trabajo. El **ciclo de trabajo** se define como la razón entre el ancho del pulso (t_w) y el periodo (T) y puede expresarse como un porcentaje.

$$\text{CicloDeTrabajo} = \left(\frac{t_w}{T} \right) 100\%$$

En la figura 6 se muestra una señal periódica donde cuyo ciclo de trabajo es de 10 ms.

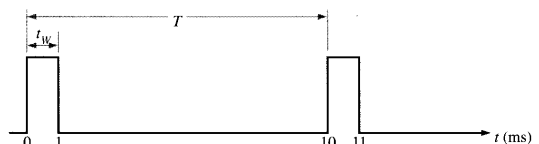


Fig. 6 Señal con ciclo de trabajo < 50%

Una señal digital contiene información binaria

La información binaria que manejan los sistemas digitales aparece en forma de señales que representan secuencias de bits. Cuando la señal está a nivel ALTO, se representa con un 1 binario, mientras que si la señal está a nivel BAJO, lo indica un 0 binario. Cada bit dentro de una secuencia ocupa un intervalo de tiempo definido, denominado **periodo del bit**.

El reloj. En los sistemas digitales, todas las señales se sincronizan con una señal de temporización básica denominada **reloj**. El reloj es una señal periódica en la que cada intervalo entre pulsos (el periodo) equivale a la duración del bit.

Un ejemplo de una señal de reloj se muestra en la Figura 7. Obsérvese que, en este caso, cada cambio de nivel de la señal A se produce en el flanco de bajada de dicha señal. Para cada pulso de la señal de reloj, la señal A puede estar a nivel ALTO o BAJO. Como hemos dicho, estos niveles de tensión altos o bajos representan una secuencia de bits. Un grupo de varios bits puede usarse como parte de una información binaria, igual que un número o una letra. La señal de reloj en sí misma no lleva información.

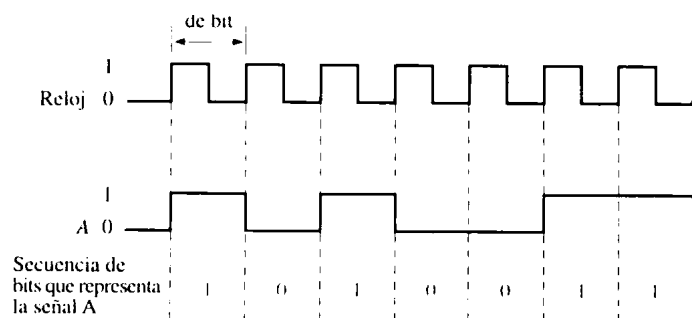


Fig. 7 Señal de reloj sincroniza otra señal que representa una secuencia de bits

Cronogramas o diagramas de tiempos

Un **diagrama de tiempos** es una gráfica de formas de onda digitales que muestra la relación temporal real entre dos o más señales, y cómo varía cada señal en relación con las demás. La Figura 8 es un ejemplo de un sencillo diagrama

de tiempos que muestra cómo se relacionan la señal de reloj y la señal A en el tiempo.

Examinando un diagrama de tiempos, se pueden determinar los estados (ALTO o BAJO), de cada una de las señales en cualquier instante de tiempo especificado, y el instante exacto en que cualquiera de las señales cambia de estado con respecto a las restantes. La Figura 8 es un ejemplo de un cronograma con cuatro señales. A partir de este diagrama podemos ver, por ejemplo, que las tres señales (A, B y C) están a nivel ALTO sólo durante el séptimo ciclo de reloj, y las tres pasan de nuevo a nivel BAJO cuando termina este mismo ciclo.

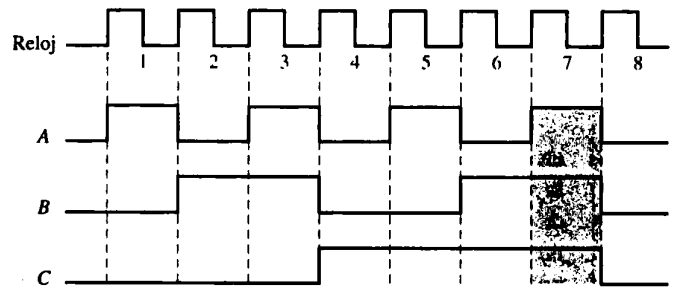


Fig. 8 Diagrama de tiempos

NÚMEROS, CÓDIGOS Y ARITMÉTICA BINARIA

NÚMEROS BINARIOS

Las computadoras y otros dispositivos digitales utilizan *números binarios*. El sistema de numeración binario, o de *base 2*, utiliza solamente los dígitos 0 y 1; los dígitos binarios se llaman bits. En los circuitos electrónicos de las computadoras el bit 0 habitualmente se representa por una tensión BAJA, mientras que el bit 1 corresponde a una tensión ALTA.

Las personas están acostumbradas a comprender el *sistema de numeración decimal*, o de *base 10*, que tiene 10 dígitos (0-9). Este sistema también tiene la característica de *valor por posición*; por ejemplo, la Figura 1(a) muestra que el número decimal 1327 es igual a 1000 más tres 100 más dos 10 más siete 1 ($1000 + 300 + 20 + 7 = 1327$).

El sistema de numeración binario también tiene la característica de valor por posición. El valor decimal de las cuatro primeras posiciones binarias se muestra en la Figura 1(b). El número binario 1001 (se pronuncia uno, cero, uno) se convierte a su equivalente decimal de 9. El bit del 1 del número binario de la Figura 2.1b se denomina *bit menos significativo* (LSB), mientras que el bit del 8 se denomina *bit más significativo* (MSB).

Potencias de 10	10^3	10^2	10^1	10^0
Valor de posición	1000	100	10	1

Decimal	1		3		2		7	
Decimal	1000	+	300	+	20	+	7	= 1327

Fig. 1(a) Valor de la posición en un número decimal

Potencias de 2	2^3	2^2	2^1	2^0
Valor de posición	8	4	2	1

MSB

1

0

0

1

LSB

Binario

Decimal

8

+

0

+

0

+

1

=

9

Fig. 1(b) Valor de la posición en un número binario

Los equivalentes binarios de los números decimales entre 0 y 15 se muestran en la Figura 1(c). Las personas que trabajan con computadoras memorizan como mínimo estos números binarios.

Convertir el número binario 10110110 (se pronuncia uno, cero, uno, uno, cero, uno, uno, cero) a su equivalente decimal. El procedimiento se muestra en la Figura 2. Por cada bit 1 del número binario se escribe debajo el valor de la posición decimal y después se suman los decimales ($128 + 32 + 16 + 4 + 2 = 182$), dando 182. Pequeños *subíndices* se utilizan para anotar la base (a veces denominada raíz) del número. El número 10110110_2 es por tanto un número binario, o en base 2 y el número 182_{10} es un número decimal o en base 10.

Convertir el número decimal 155 a binario. La Figura 3 muestra un procedimiento para hacer esta conversión. El número decimal 155 se divide primero por 2, dando un cociente de 77 y un resto de 1; el resto se convierte en el bit menos significativo (LSB) del número binario y se transfiere a esta posición en la Figura 3. El cociente (77) se transfiere como muestra la flecha y se convierte en el siguiente dividendo. Los cocientes se dividen repetidamente por 2 hasta que el cociente se hace 0 con un resto de 1. La Figura 3 muestra este procedimiento. La línea inferior muestra el resultado de la conversión: $155_{10} = 10011011_2$.

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

Fig. 1(c) equivalencia decimal-binaria

Potencias de 2	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0			
Valor de posición	128	64	32	16	8	4	2	1			
Binario	1	0	1	1	0	1	1	0			
Decimal	128	+	32	+	16	+	4	+	2	=	182

Fig. 2 Conversión binaria a decimal

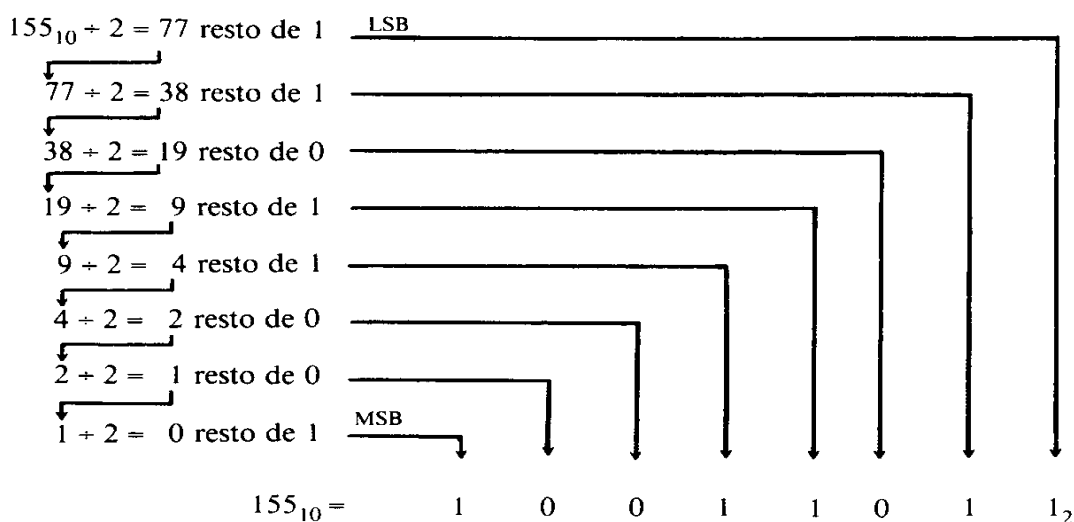


Fig. 3 Conversión decimal a binaria

NÚMEROS HEXADECIMALES

Una posición de la memoria de una microcomputadora puede contener el número binario 10011110. Esta larga cadena de ceros y unos es difícil de recordar y teclear. El número 10011110₂ puede convertirse en un número decimal. Una vez convertido éste es el número 158₁₀. Este proceso de conversión es demasiado largo. La mayoría de los sistemas de microcomputadoras utilizan la *notación hexadecimal* para simplificar la tarea de recordar y teclear números binarios como por ejemplo 10011110. El sistema de numeración hexadecimal, o de base 16, utiliza los 16 símbolos del 0 al 9, A, B, C, D, E y F. Lo equivalentes binarios, hexadecimales y decimales se muestran en la Figura 2.4.

Observar en la Figura 2.4 que cada símbolo hexadecimal representa una única combinación de 4 bits. El número binario 10011110 puede entonces ser representado como 9E en hexadecimal. Esto es, la parte 1001 del número binario es igual a 9, de acuerdo con la Figura 2.4, y la parte 1110 del número binario es igual a E en hexadecimal. Por tanto 10011110₂ es igual a 9E₁₆. Recordar que el subíndice indica la base del número.

Convertir el número binario 111010 en hexadecimal (hex). Comenzar por LSB y dividir el número binario en grupos de 4 bits cada uno, como indica la Figura 5(a). Entonces sustituir cada grupo de 4 bits por su dígito hex equivalente. El 1010₂ es igual a A en hex (ver Figura 4). El 0011₂ es igual a 3 en hex. Por tanto 111010₂ es igual a 3A₁₆.

Decimal	Hexadecimal	Binario			
		8	4	2	1
0	0	0	0	0	0
1	1	0	0	0	1
2	2	0	0	1	0
3	3	0	0	1	1
4	4	0	1	0	0
5	5	0	1	0	1
6	6	0	1	1	0
7	7	0	1	1	1
8	8	1	0	0	0
9	9	1	0	0	1
10	A	1	0	1	0
11	B	1	0	1	1
12	C	1	1	0	0
13	D	1	1	0	1
14	E	1	1	1	0
15	F	1	1	1	1

Fig. 4 Numeración decimal, hexadecimal y binaria.

Convertir el número hexadecimal 7F a su equivalente binario. La Figura 5b muestra que cada dígito hex es sustituido por su equivalente binario de 4 bits. En este ejemplo, el binario 0111 es sustituido por el hex 7 y 1111_2 sustituye a F_{16} . Por tanto $7F_{16}$ es igual a 1111111_2 .

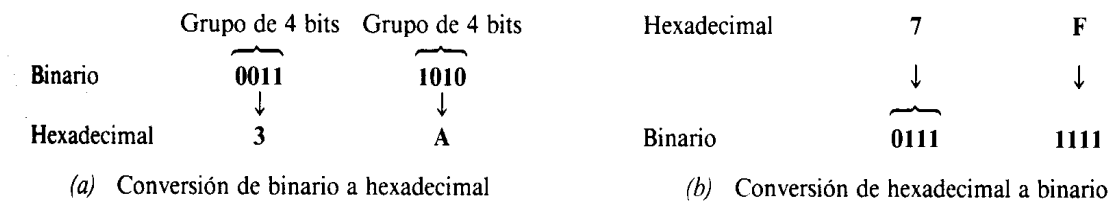


Fig.5 Conversión entre numeraciones

La notación hexadecimal es muy utilizada para representar números binarios. Las personas que utilizan la notación hexadecimal deben memorizar la tabla mostrada en la Figura 4.

Convertir el número hexadecimal 2C6E a su equivalente decimal. El procedimiento se muestra en la Figura 6 (a). los valores de posición para los cuatro primeros dígitos decimales son 4096, 256, 16, 1. El número hexadecimal contiene catorce (E_{16}) 1, seis 16, doce (C_{16}) 256 y dos 4096. Cada Valor de posición se multiplica y los productos se suman para obtener 11.374_{10} .

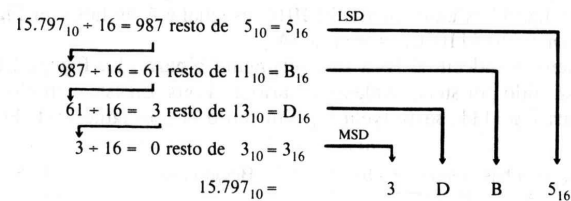
Potencias de 16	16^3	16^2	16^1	16^0
Valor de posición	4096	256	16	1

Hexadecimal	2	C	6	E
	↓	↓	↓	↓
	4096	256	16	1
	×2	×12	×6	×14
Decimal	8192	+ 3072	+ 96	+ 14

= 11.374_{10}

(a) Conversión de hexadecimal a decimal

Convertir el número decimal 15797 a su equivalente hexadecimal. El procedimiento se muestra en la Figura 6 b. La primera línea indica 15.797_{10} dividido por 16, dando un cociente de 987_{10} con un resto de 5_{10} . El resto se convierte entonces a su equivalente hexadecimal. Por tanto 5_{10} es igual a 5_{16} . El resto hexadecimal (5_{16}) se convierte en el dígito menos significativo (LSD) del número hexadecimal. El primer cociente (987) es el dividendo en la segunda línea y se divide por 16. El segundo cociente es 61 con un resto de 11_{10} , o B hexadecimal. La línea 3 muestra 61 dividido por 16, dando un cociente de 3 con un resto de 13_{10} , o D₁₆. La cuarta línea de la Figura 2.6b muestra el dividendo (3) dividido por 16, dando un cociente de 0 con un resto de 3_{10} , ó 3_{16} . Cuando el cociente se hace 0, como en la línea 4, se termina el cálculo. El 3_{16} es el dígito más significativo (MSD). El procedimiento mostrado en la Figura 6 b convierte el número decimal 15.797 en su equivalente hex de $3DB5_{16}$.



(b) Conversión de decimal a hexadecimal

Figura 6.

NÚMEROS BCD

Los número binarios puros se representan en notación hexadecimal para hacer más fácil la conversión. Sin embargo, la conversión binario a decimal es bastante difícil. En calculadoras, juegos e instrumentos digitales, donde son frecuentes las entradas y salidas del usuario en decimal, se utiliza un código especial para representar los números decimales. Este código se denomina BCD (*decimal-codificado-binario*). Las equivalencias entre decimal y BCD se muestran en la tabla de la Figura 7(a). Técnicamente, esta tabla detalla el *código* BCD 8421. La parte del nombre 8421 da el valor de la posición a los 4 bits del código BCD. También se utilizan otros códigos BCD, como por ejemplo el código BCD 5421 y el código de exceso 3.

Convertir el número decimal 3691 a su equivalente BCD 8421. El procedimiento se muestra en la Figura 7 b. Cada dígito decimal se

Decimal	BCD			
	8	4	2	1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

(a) Código BCD 8421

Los microprocesadores suman los números binarios puros. Sin embargo, muchos microprocesadores tienen instrucciones especiales para cambiar el resultado de las sumas a notación BCD. El número BCD se interpreta fácilmente entonces como número decimal utilizando los sencillos procedimientos mostrados en las Figura 7b y c.

(c) Conversión de BCD a decimal

regla 4, hay que tomar prestado un 1 de la siguiente posición más significativa (la posición del 32). El minuendo de la posición 32 es 0, por lo que entonces hay que tomar prestado el 1 de la posición 64. La posición del 32 toma un préstamo de la posición 64. Finalmente, la posición del 16 puede tomar prestado de la posición del 32. El minuendo de la posición del 16 es entonces 10_2 , y el sustraendo es 1 dando una diferencia de 1. La posición del 32 ahora muestra $1 - 1$ dando una diferencia de 0. La posición del 64 muestra $0 - 0$ dando una diferencia de 0. La posición del 128 muestra $0 - 0$ dando también una diferencia de 0. En resumen, el problema ejemplo de la Figura 2.9b muestra que el binario 00111001 restado de 01010101_2 da una diferencia de 00011100_2 . El problema también se muestra en forma decimal a la derecha.

Las reglas para la multiplicación binaria se muestran en la Figura 10 a. Las dos primeras reglas no necesitan explicación. El multiplicador es 1 en las dos últimas reglas. Cuando el multiplicador es 1 en la multiplicación binaria, el *multiplicando se copia* como producto. Cuando el multiplicador es 0, el producto siempre es 0.

Multiplicando	0	1	0	1
Multiplicador	$\times 0$	$\times 0$	$\times 1$	$\times 1$
Producto	$\overline{0}$	$\overline{0}$	$\overline{0}$	$\overline{1}$

Multiplicando	1101	13
Multiplicador	$\times 101$	$\times 5$
Primer producto parcial	$\overline{1101}$	$\overline{65}_{10}$
Segundo producto parcial	0000	
Tercer producto parcial	$\overline{1101}$	
Producto final	$\overline{1000001}_2$	

a) Reglas de la multiplicación

b) Problema de multiplicación binaria

Fig. 10

Multiplicar los números binarios 1101 y 101. Este problema ejemplo se muestra en la Figura 10b. Como en la multiplicación decimal, el multiplicando se multiplica primero por el dígito menos significativo (en este caso el bit del 1). El bit del 1 del multiplicador es 1, por tanto el multiplicando se copia como primer producto parcial. El bit del 2 del multiplicador es un 0, por tanto el segundo producto parcial es 0000. Observar que éste se *desplaza* una posición a la izquierda. El bit del 4 del multiplicador es 1, por tanto el multiplicando se copia como tercer producto parcial. Observar que 1101 se copia después del segundo desplazamiento a la izquierda. Los productos parciales primero, segundo y tercero se suman, dando el producto final de 1000001_2 . En resumen, la Figura 10b muestra que $1101_2 \times 101_2 = 1000001_2$ o que $13_{10} \times 5_{10} = 65_{10}$.

NOTACIÓN EN COMPLEMENTO A 2

Generalmente, en las computadoras se utilizan los números binarios. Sin embargo, a veces se utiliza un código especial denominado *notación en complemento a 2* cuando se necesitan *números con signo*. Este sistema simplifica la circuitería de la computadora.

Un registro o posición de almacenamiento en un microprocesador puede ser como el de la Figura 11 a. Este registro tiene espacio para datos de 8 bits. Las posiciones de los bits se numeran del 7 al 0. Los valores de las posiciones binarias se muestran en la parte inferior del registro. El bit 7 será el de la posición del 128, el bit 6 el de la posición del 64, etc.

La organización más frecuente de un registro de 8 bits utilizado para almacenar números con signo se muestra en las Figuras 11b y c. El bit 7 en ambos registros es el bit de signo. Este bit dice si el número es (+) positivo o (-) negativo. Un 0 en la posición del bit de signo significa que el número es positivo, mientras que un 1 indica que el número es negativo.

Si el número con signo es positivo como en la Figura 11b, las restantes posiciones de memoria (6-0) contienen un número binario de 7 bits. Por ejemplo, si el contenido del registro de la Figura 11b fuese 01000001, significa el decimal +65 (bit de signo positivo + $64 + 1$). Si el contenido del registro de la Figura 2.11b fuese 01111111, sería $+127_{10}$ (bit de

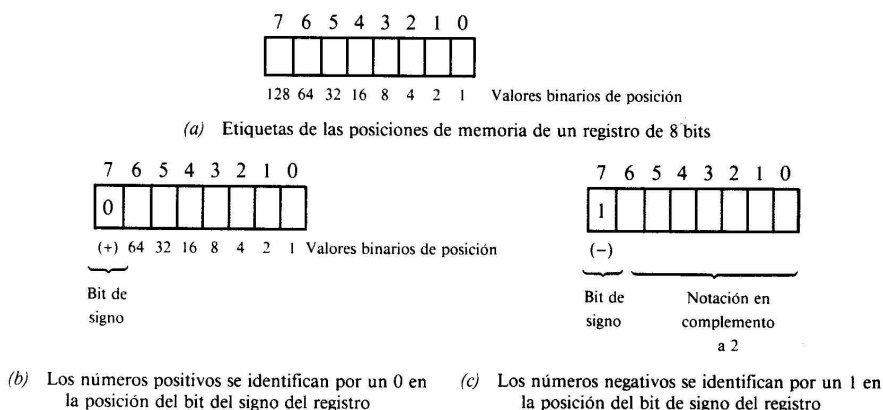


Figura 11.

signo positivo + 64 + 32 + 16 + 8 + 4 + 2 + 1). Este es el mayor número positivo que puede ser representado en este registro de 8 bits.

Si el número con signo es negativo como en la Figura 11c, el registro contendrá la forma en complemento a 2 de ese número. La tabla de la Figura 12 muestra la notación en complemento a 2 para números positivos y negativos. Observar que los números positivos tienen un 0 en el MSB, mientras que el resto de los números corresponden a un número binario. Los números negativos tienen un 1 en el MSB. Observar la línea +0 de la tabla de la Figura 12. La notación en complemento a 2 para +0 es 00000000. En la línea siguiente, observar que 11111111 es la notación en complemento a 2 de -1. Imaginar que el sistema es un odómetro que cuenta hacia atrás a medida que se avanza de 00000000 a 11111111.

¿Cuál sería la notación en complemento a 2 de -9?. Los pasos para hacer esta conversión se esbozan en la Figura 13 a y son los siguientes:

Paso 1: Listar el número decimal sin signo. Escribir 9 en este ejemplo.

Paso 2: convertir el número decimal a binario. Escribir el número binario 00001001 en este ejemplo.

Paso 3: complementar cada bit formando el complemento a 1. En este ejemplo escribir 11110110 como complemento a uno.

Paso 4: Sumar 1 al número en complemento a 1. En este ejemplo 1 a 11110110.

Decimal	Representación de números con signo	
+127	0111 1111	Números positivos representados igual que en binario puro
⋮	⋮	
+8	0000 1000	
+7	0000 0111	
+6	0000 0110	
+5	0000 0101	
+4	0000 0100	
+3	0000 0011	
+2	0000 0010	
+1	0000 0001	
+0	0000 0000	
-1	1111 1111	Números negativos representados en forma de complemento a 2
-2	1111 1110	
-3	1111 1101	
-4	1111 1100	
-5	1111 1011	
-6	1111 1010	
-7	1111 1001	
-8	1111 1000	
⋮	⋮	
-128	1000 0000	

Fig.12 Números decimales con signo y notación en complemento a 2

Decimal	9	Paso 1	Escribir decimal
	↓	Paso 2	Convertir a binario
Binario	00001001	Paso 3	Complementar cada bit
	↓	Paso 4	Sumar + 1
Complemento a 1	11110110		
Complemento a 2	$\begin{array}{r} + \quad 1 \\ 11110110 \\ \hline \end{array} = -9$		

(a) Formación del complemento a 2 de un número negativo

Complemento a 2	11110000	Paso 1	Escribir complemento a 2
	↓	Paso 2	Complementar cada bit
Complemento a 1	00001111	Paso 3	Sumar + 1
Binario	$\begin{array}{r} + \quad 1 \\ 00001111 \\ \hline \end{array} = 16$		

Fig. 13 Ejemplos con números con complemento a 2

El resultado es la notación en complemento a 2 para el número decimal negativo. En este ejemplo, en la Figura 13 a, -9 es igual a 11110110 en forma de complemento a 2. Observar que el bit de signo (11110111) es 1, lo que significa que se trata de un número negativo.

¿Cuál es el equivalente decimal del número en complemento a 2 11110000? El procedimiento para hacer esta conversión se detalla en la Figura 13b. El procedimiento de complementar y sumar 1 es el mismo que se utiliza para convertir de binario a complemento a 2. El procedimiento de la Figura 13b muestra cómo se cambia cada bit del número en complemento a 2, formando el complemento a 1. Entonces se suma un 1 al número en complemento a 1, formando el número binario 00010000, que es igual a 16. Esto significa que la notación en complemento a 2 de 11110000 es igual a -16_{10} . El 16 debe ser negativo porque el bit de signo (MSB) del número en complemento a 2 es un 1.

ARITMÉTICA EN COMPLEMENTO A 2

Un microprocesador puede utilizar números en complemento a 2 porque puede complementar, incrementar (sumar +1 a un número) y sumar números binarios. Los microprocesadores no tienen circuitería para restar, en su lugar utilizan un sumador y números en complemento a 2 para realizar la sustracción.

Sumar los números decimales con signo +5 y +3. El proceso se muestra en la Figura 14 (a) en decimal y en la forma en complemento a 2. A partir de la tabla de la Figura 12 se encuentra que +5 es igual a 00000101 en su notación en complemento a 2, mientras que +3 es igual a 00000011. Los números en complemento a 2 00000101 y 00000011 se suman entonces igual que los números binarios regulares, obteniéndose la suma en complemento a 2 de 00001000. La suma 00001000 es igual a $+8_{10}$.

Primer sumando	(+5)	00000101
Segundo sumando	+(+3)	+0000011
Suma	(+8)	00001000

(a) Problema de suma en complemento a 2

Primer sumando	(+3)	00000011
Segundo sumando	+(-8)	+11111000
Suma	(-5)	11111011

(c) Problema de suma en complemento a 2

Primer sumando	(+7)	00000111
Segundo sumando	+(-3)	+1111101
Suma	(+4)	<u>1</u> 00000100

(b)

Descarta
overflow

Primer sumando	(-2)	11111110
Segundo sumando	+(-5)	+1111011
Suma	(-7)	<u>1</u> 11111001

(d)

Descarta
overflow

Fig. 14 Problemas de suma en complemento a dos con n° positivos y negativos.

Sumar los números decimales con signo +7 y -3. El procedimiento utilizando la suma en decimal y en complemento a 2 se muestra en la Figura 14b. a partir de la tabla de la Figura 12 se encuentra que +7 = 00000111 y -3 = 1111101 en la notación en complemento a 2. Los números en complemento a 2 se suman entonces (00000111 y 1111101) como si fuesen números binarios regulares, obteniéndose la suma de 100000100. El MSB es un arrastre de "overflow" del registro de 8 bits y se descarta. Por tanto, la suma en complemento a 2 es igual a 00000100, ó $+4_{10}$.

Sumar los números decimales con signo +3 y -8. El procedimiento se detalla en la Figura 14c. A partir de la tabla de la Figura 12 se encuentra +3 = 00000011 y -8 = 11111000 en notación en complemento a 2. Entonces se suman estos números en complemento a 2 como si fuesen números binarios regulares, obteniéndose una suma de 11111011. De nuevo de la Figura 12 se determina que la suma en complemento a 2 de 11111011 es igual a -5_{10} .

Sumar los decimales -2 y -5. El procedimiento se detalla en la Figura 14d. A partir de la tabla de la Figura 12 se encuentra que la notación en complemento a 2 de -2 = 11111110 y la de -5 = 1111011. Estos números se suman entonces como si fuesen números binarios regulares. La suma es 11111001. El MSB es un arrastre de "overflow" del registro de 8 bits y se descarta. La suma de los números en complemento a 2 (11111110 y 1111011) es entonces 11111001 (en complemento a 2). Utilizando la Figura 12 se determina que la suma en complemento a 2 de 11111001 es igual a -7_{10} .

Restar los números decimales con signo +5 de +8. El procedimiento se esboza en la Figura 15 a. El minuendo (+8) es igual a 00001000. El sustraendo es +5, ó 00000101. El 00000101 debe convertirse a su forma en

Minuendo	$(+8)$		00001000
Sustraendo	$- (+5)$	$= 00000101$	$\xrightarrow[\text{complemento a 2}]{\text{Convierte a}}$
Diferencia	$(+3)$		$\begin{array}{r} 00001000 \\ + 11111011 \\ \hline 10000011 \end{array}$

1

Descarta overflow

(a) Problema de resta en complemento a 2 utilizando suma

Minuendo	$(+2)$		00000010
Sustraendo	$- (+6)$	$= 00000110$	$\xrightarrow[\text{complemento a 2}]{\text{Convierte a}}$
Diferencia	(-4)		$\begin{array}{r} 00000010 \\ + 11111010 \\ \hline 11111100 \end{array}$

(b) Problema de resta en complemento a 2 utilizando suma

complemento a 2 (complementar y sumar 1), dando 11111011. El “minuendo” de 00001000 se suma al complemento a 2 del sustraendo 11111011 como si fuesen números binarios. La suma es igual a 100000011. El MSB es un arrastre de “overflow” del registro y se descarta, obteniéndose la suma de 00000011. A partir de la tabla de la Figura 12 se determina que la suma en complemento a 2 de 00000011 es igual a $+3_{10}$. Observar que al restar, el sustraendo se convierte a su forma en complemento a 2 y después se suma al minuendo. Utilizando la representación en complemento a 2 y un sumador, el microprocesador puede realizar la sustracción.

Restar el número decimal +6 de +2. El procedimiento se muestra en la Figura 15b. El minuendo +2 es igual 00000010. El sustraendo +6 es igual 00000110. Este sustraendo se convierte a su forma en complemento a 2 (complementar y sumar 1) dando 11111010. Los números (00000010 y 11111010) se suman como si fuesen números binarios dando una suma de 11111100. De la tabla de la Figura 2.12 se determina que la suma en complemento a 2 de 11111100 es igual a -4_{10} .

PUERTAS LÓGICAS

EL INVERSOR

El inversor (puerta NOT) realiza la operación denominada *inversión* o *complementación*. El inversor cambia un nivel lógico al nivel opuesto. En términos de bits, cambia un 1 por un 0, y un 0 por un 1.

En la Figura 1 se muestran los símbolos lógicos estándar del inversor.

El indicador de negación en los símbolos de las puertas, es un “círculo” (o) que indica inversión o complementación, cuando aparece en la entrada o en la salida de un elemento lógico, tal como muestra la Figura 1. Generalmente, las entradas se sitúan a la izquierda del símbolo lógico, y la salida a la derecha. Cuando en la entrada hay un círculo, quiere decir que el estado activo o verdadero de la entrada es 0. Cuando el círculo se sitúa en la salida significa que el estado activo o verdadero de la salida es 0. La ausencia de círculo en la entrada o en la salida significa que el estado activo o verdadero es 1.

Obsérvese que un cambio de posición del indicador de polaridad o de negación no implica un cambio en el modo de funcionamiento del inversor.

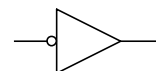
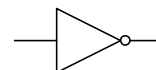


Fig. 1: Símbolos distintivos con indicadores de negación

Tabla de verdad del inversor

Cuando se aplica un nivel ALTO a la entrada de un inversor, en su salida se presenta un nivel BAJO. Cuando se aplica un nivel BAJO a la entrada, en su salida se presenta un nivel ALTO. En la Tabla 1 se resume esta operación. Esta tabla muestra la salida para cada posible entrada en términos de niveles y bits correspondientes. Una tabla tal como ésta se llama **tabla de verdad**.

Entrada	Salida
BAJO (0)	ALTO (1)
ALTO (1)	BAJO (0)

Tabla 1: Tabla de verdad del inversor

Funcionamiento del inversor

La Figura 2 muestra la salida de un inversor para un pulso de entrada, donde t_1 y t_2 indican los instantes de tiempo que corresponden a los pulsos de entrada y salida.

Cuando la entrada está a nivel BAJO, la salida está a nivel ALTO; y cuando la entrada está a nivel ALTO, la salida está a nivel BAJO, lo que da lugar a un pulso de salida invertido.

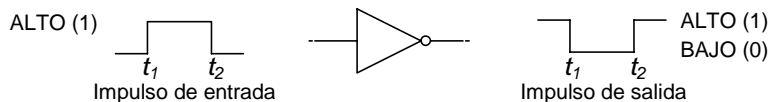


Fig. 2: Impulso de entrada en el inversor

Diagramas de tiempos

Un **diagrama de tiempos** o **cronograma** es básicamente una *gráfica que presenta de forma precisa las relaciones de dos o más formas de onda en función del tiempo*. Por ejemplo, la relación de tiempo del pulso de salida respecto al pulso de entrada de la Figura 2 puede representarse con un sencillo diagrama de tiempos, alineando los dos impulsos de modo que las ocurrencias de los flancos se presenten en los instantes de tiempo correctos. El flanco de subida del pulso de entrada y el flanco de bajada del pulso de salida se producen al mismo tiempo (idealmente). Igualmente, el flanco de bajada del pulso de entrada y el flanco de subida del pulso de salida se producen al mismo tiempo (idealmente). En la Figura 3 se muestra la relación de tiempos. Los diagramas de tiempos son muy útiles para ilustrar las relaciones de señales digitales de pulsos múltiples.

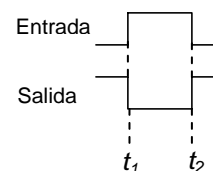


Fig. 3: Diagrama de tiempos para el caso de la Fig. 3.2

Expresión lógica del inversor

En el **álgebra booleana**, una variable se designa mediante una letra. El **complemento** de una variable se designa mediante una barra encima de la letra. Una variable puede tomar uno de dos valores, 1 ó 0. Si una variable dada es 1, su complemento es 0, y viceversa.

El modo de operación de un inversor (puerta NOT) puede expresarse del siguiente modo: si la variable de entrada se designa por A y la variable de salida por X, entonces

$$X = \bar{A}$$

Esta expresión establece que la salida es el complemento de la entrada, es decir, si A = 0, entonces X = 1; y si A = 1, entonces X = 0. La Figura 4 ilustra esto. La variable complementada \bar{A} se lee como "A negada".

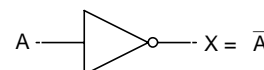


Fig. 4: El inversor complementa una variable de entrada

LA PUERTA AND

El término **puerta** se usa para describir un circuito que realiza una operación lógica básica. La puerta AND realiza la operación lógica conocida como multiplicación, tiene dos o más entradas y una única salida, como indica el símbolo lógico en la Figura 5. En el símbolo las entradas se sitúan a la izquierda y la salida a la derecha. Se muestra una puerta con dos entradas, pero una puerta AND puede tener cualquier número de entradas superior a éste.

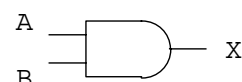


Fig. 5: Símbolo distintivo

Operación lógica de la puerta AND

La **puerta AND** genera una salida a nivel ALTO *sólo* cuando *todas* las entradas están a nivel ALTO. Cuando cualquiera de las entradas está a nivel BAJO, la salida se pone a nivel BAJO. Por tanto, el propósito básico de una puerta AND es determinar cuándo ciertas condiciones de entrada son simultáneamente verdaderas, como indican todas sus entradas estando a nivel ALTO, y producir una salida a nivel ALTO, para indicar que esas condiciones son verdaderas. Las entradas de la puerta AND de dos entradas de la Figura 5 se designan mediante A y B, y la salida con X, luego podemos establecer que el funcionamiento de la puerta es el siguiente:

En una puerta AND de dos entradas, la salida X es un nivel ALTO si A y B están a nivel ALTO; y X es un nivel BAJO si A es un nivel BAJO, o si B es un nivel BAJO, o si A y B está a nivel BAJO.

Tabla de verdad de la puerta AND

La operación lógica de una puerta puede expresarse mediante una tabla de verdad, en la que se enumeran todas las combinaciones de entrada con las correspondientes salidas, como muestra la Tabla 2 para una puerta AND de dos entradas. La tabla de verdad puede ampliarse para cualquier número de entradas. Aunque los términos nivel ALTO y nivel BAJO dan un sentido "físico" a los estados de entrada y salida, la tabla de verdad se presenta con 1s y 0s, ya que un nivel ALTO es equivalente a un 1, y un nivel BAJO es equivalente a 0 en lógica positiva. Para cualquier puerta AND, independientemente del número de entradas, la salida es un nivel ALTO *sólo* cuando *todas* las entradas están a nivel ALTO.

Entradas		Salida
A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

Tabla 2: Tabla de verdad de una puerta AND de dos entradas.

El número total de posibles combinaciones de entradas binarias a una puerta viene determinado por la siguiente fórmula:

$$N = 2^n$$

donde N son todas las posibles combinaciones, y n es el número de variables de entrada, luego

Para dos variables de entradas: $N = 2^2 = 4$ combinaciones

Para tres variables de entradas: $N = 2^3 = 8$ combinaciones

Para cuatro variables de entradas: $N = 2^4 = 16$ combinaciones

Funcionamiento con trenes de impulsos

En la mayoría de las aplicaciones, las entradas a una puerta no son niveles estacionarios, sino que son tensiones que cambian frecuentemente entre los

niveles lógicos ALTO y BAJO. Ahora vamos a ver el funcionamiento de las puertas AND con entradas que son señales digitales, teniendo en mente que una puerta AND obedece a su tabla de verdad, independientemente de que sus entradas sean niveles constantes o señales que varíen de un nivel a otro.

Al examinar el funcionamiento de una puerta AND con trenes de impulsos, nos fijaremos en los niveles de entrada para determinar el nivel de salida en cualquier instante dado. Por ejemplo, en la Figura 6, ambas entradas están a nivel ALTO durante el intervalo de tiempo t_1 , por lo que la salida en este intervalo estará a nivel ALTO. Durante el intervalo t_2 , la entrada A está a nivel BAJO (0) y la entrada B está a nivel ALTO (1), por lo que la salida se pondrá a nivel BAJO (0). De nuevo, durante el intervalo t_3 , ambas entradas están a nivel ALTO (1) y, por tanto, la salida está a nivel ALTO (1). Durante el intervalo t_4 , la entrada A está a nivel ALTO (1) y la B está a nivel BAJO (0), luego la salida está a nivel BAJO (0). Por último, durante el intervalo t_5 , la entrada A está a nivel BAJO (0) y la entrada B está a nivel BAJO (0) y, por tanto, la salida está a nivel BAJO (0). Como ya se sabe, un diagrama de las señales de entrada y de salida en función del tiempo se llama *diagrama de tiempos o cronograma*.

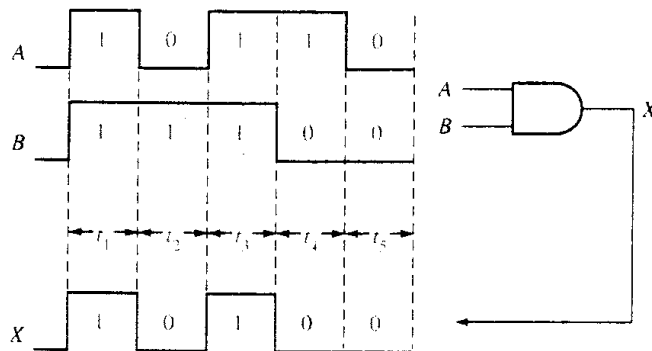


Fig. 6: Ejemplo de funcionamiento de una puerta AND con trenes de impulsos, y cronogramas que muestra las relaciones entre las entradas y la salida.

Expresiones lógicas para la puerta AND

La función lógica AND de dos variables se representa matemáticamente colocando un punto entre las dos variables, $A \cdot B$, o simplemente escribiendo las letras juntas sin el punto, AB . Normalmente, utilizaremos esta última notación, ya que es más cómoda de escribir.

La **multiplicación booleana** sigue las mismas reglas básicas que gobiernan la multiplicación binaria, y son las siguientes:

$$\begin{aligned} 0 \cdot 0 &= 0 \\ 0 \cdot 1 &= 0 \\ 1 \cdot 0 &= 0 \\ 1 \cdot 1 &= 1 \end{aligned}$$

La multiplicación booleana es lo mismo que la función AND

El funcionamiento de una puerta AND de dos entradas puede expresarse en forma de ecuación como sigue: si una variable de entrada es A y la otra variable es B, y la variable de salida es X, entonces la expresión booleana es

$$X = AB$$

La Figura 7(a) muestra la puerta con las variables de entrada y de salida indicadas.

Para extender la expresión AND a más de dos variables de entrada, simplemente utilice una nueva letra para cada variable de entrada. Por ejemplo, la función de una puerta AND de 3 entradas se puede expresar así: $X = ABC$, donde A, B y C son las variables de entrada. La expresión para una puerta AND de 4 entradas será $X = ABCD$, y así sucesivamente. Las partes (b) y (c) de la Figura 7 muestran, respectivamente, puertas AND de tres y cuatro variables de entrada.

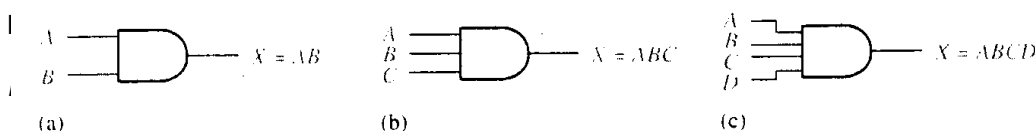


Fig. 7: Expresión booleana para puertas AND con dos, tres y cuatro entradas.

Una aplicación común de la puerta AND es **habilitar** (enable) o permitir el paso de una señal (tren de pulsos) de un punto a otro en determinados instantes, e inhibir o impedir el paso en otros instantes.

Un ejemplo sencillo de este particular uso de la puerta AND se muestra en la Figura 8, donde la puerta AND controla el paso de una señal (A) a un contador digital. El propósito de este circuito es medir la frecuencia de la señal A. El pulso de habilitación tiene un ancho de exactamente 1 s. Cuando la entrada de habilitación está a nivel ALTO, la señal A pasa a través de la puerta hasta el contador, y cuando la entrada de habilitación está a nivel BAJO, se impide el paso de la señal de entrada a través de la puerta.

Durante el intervalo de habilitación de 1 segundo, un cierto número de pulsos de la señal A pasa a través de la puerta AND hasta el contador. El número de pulsos que pasa durante 1 segundo es igual a la frecuencia de la señal. Por ejemplo, la Figura 8 muestra una señal en la que seis pulsos duran un segundo, lo que representa una frecuencia de 6 Hz. Si pasan 1.000 pulsos a través de la puerta en el intervalo de 1 segundo del pulso de habilitación, serán 1.000 pulsos/segundo, es decir una frecuencia de 1.000 Hz.

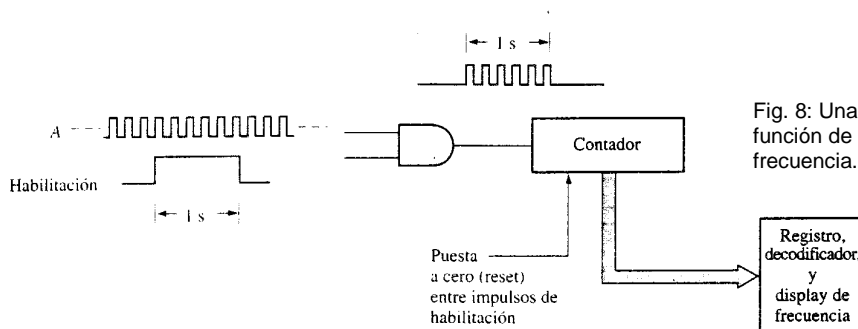


Fig. 8: Una puerta AND que realiza la función de habilitación para un contador de frecuencia.

El contador cuenta el número de pulsos por segundo y genera una salida binaria que pasa al circuito decodificador y al display, en el que se genera una lectura de la frecuencia. El pulso de habilitación se repite en determinados instantes, en los que se lleva a cabo un nuevo recuento por si la frecuencia ha variado, y el nuevo valor se presentará en el display. Entre los pulsos de habilitación, el contador se pone a cero para reiniciarse cada vez que se produce un pulso de habilitación. La frecuencia actual se almacena en un registro, de modo que el display no se vea afectado al poner a cero el contador.

LA PUERTA OR

La **puerta OR** tiene dos o más entradas y una salida, como indica el símbolo lógico estándar de la Figura 9, en la que se muestra la puerta OR con dos entradas. Una puerta OR puede tener cualquier número de entradas mayor o igual que dos.



Fig. 9: Símbolo distintivo.

La **puerta OR** genera un nivel ALTO a la salida cuando cualquiera de sus entradas está a nivel ALTO. La salida se pone a nivel BAJO sólo cuando todas las entradas están a nivel BAJO. Por tanto, el propósito de una puerta OR es determinar cuándo una o más de sus entradas están a nivel ALTO y generar una salida a nivel ALTO que indique esta condición. Las entradas de la puerta OR de dos entradas de la Figura 9 están etiquetadas como A y B, y la salida como X. Podemos establecer el funcionamiento de la puerta como sigue:

En una puerta OR, la salida X es un nivel ALTO si cualquiera de las entradas, A o B, o ambas, están a nivel ALTO; X es un nivel BAJO si ambas entradas, A y B, están a nivel BAJO.

Tabla de verdad de una puerta OR

En la Tabla 3 se describe el funcionamiento lógico de una puerta OR de dos entradas. Esta tabla de verdad puede extenderse a cualquier número de entradas e, independientemente del número de entradas, la salida es un nivel ALTO cuando una o más estradas están a nivel ALTO.

Entradas		Salida
A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

Tabla 3: Tabla de verdad para una puerta OR de dos entradas.

Funcionamiento con trenes de pulsos

Ahora vamos a ver el funcionamiento de una puerta OR con trenes de pulsos como entradas, teniendo en mente su modo de operación lógico. De nuevo, lo importante en el análisis del funcionamiento de la puerta con trenes de pulsos en las entradas es la relación de tiempos de todas las señales implicadas. Un ejemplo se muestra en la Figura 10.

Expresiones lógicas de la puerta OR

La función lógica OR de dos variables se representa matemáticamente mediante un signo + entre las dos variables, por ejemplo, $A + B$.

La suma en el álgebra de Boole implica variables cuyos valores son, bien el binario 1 o el binario 0. Las reglas básicas de la **suma booleana** son las siguientes:

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

La suma booleana es lo mismo que la función OR

Obsérvese que la suma booleana difiere de la suma binaria en el caso en que se suman dos 1s. En la suma booleana no existe acarreo.

El funcionamiento de una puerta OR de 2 entradas se puede expresar como sigue: si una variable de entrada es A y la otra variable de entrada es B, y la variable de salida es X, entonces la expresión booleana es

$$X = A + B$$

La Figura 11(a) muestra el símbolo lógico de la puerta indicando las variables de entradas y de salida.

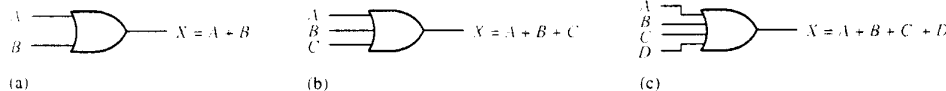


Fig. 11: Expresiones booleanas de las puertas OR con dos, tres y cuatro entradas.

Ejemplo de aplicación

En la Figura 12 se presenta parte de un sistema de alarma y detección de intrusión simplificado.

Este sistema se podría utilizar en una habitación de una casa: una habitación con dos ventanas y una puerta. Los sensores son interruptores magnéticos que producen un nivel de salida ALTO cuando se abre la puerta (o las ventanas) y una salida a nivel BAJO cuando se cierra. Mientras que las ventanas y la puerta están aseguradas, los interruptores están cerrados y las tres entradas de la puerta OR están a nivel BAJO. Cuando se abre una de las ventanas o la puerta, en la entrada correspondiente de la puerta OR se genera un nivel ALTO y la salida de la puerta lógica se pone a nivel ALTO. Entonces se activa el circuito de alarma para advertir de la intrusión.

LA PUERTA NAND

El término **NAND** es una contracción de NOT-AND, e implica una función AND con la salida complementada (negada). En la Figura 13 se muestra el símbolo lógico estándar para la puerta NAND de 2 entradas y su equivalente empleando los símbolos de la puerta AND seguida de un inversor.

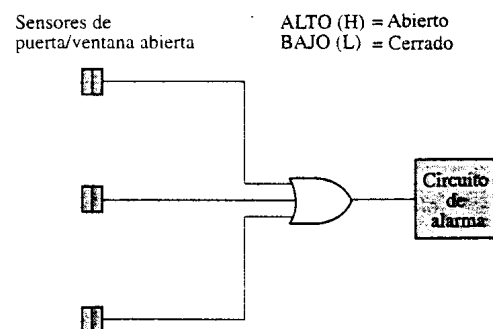


Fig. 12: Un sistema de detección de intrusión simplificado que utiliza una puerta OR.

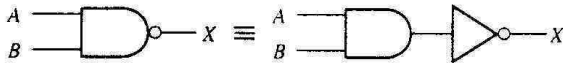


Fig. 13: Símbolos lógicos estándar de la puerta NAND

Operación lógica de la puerta NAND

La **puerta NAND** genera una salida a nivel BAJO sólo cuando todas las entradas están a nivel ALTO. Cuando cualquiera de las entradas está a nivel BAJO, la salida se pondrá a nivel ALTO. Para el caso concreto de la puerta NAND de dos entradas, como muestra la Figura 13, con la designación A y B para las entradas y X para la salida, el modo de operación se puede establecer como sigue:

En una puerta NAND de dos entradas, la salida X es un nivel BAJO si las entradas A y B están a nivel ALTO; X es un nivel ALTO si A o B están a nivel BAJO o si ambas, A y B, están a nivel BAJO.

Observe que esta operación, en términos de nivel de salida, es la opuesta a la operación lógica AND. En una puerta NAND, el nivel BAJO (0) es el nivel activo o verdadero de salida, como indica el círculo de la salida, y la Tabla 4 es la tabla de verdad, que resume la operación de la puerta NAND de dos entradas.

La expresión booleana para puerta NAND de dos entradas es:

$$X = \overline{AB}$$

Esta expresión significa que las dos variables de entrada, A y B, se multiplican (AND) primero y luego se complementan, tal y como indica la barra sobre la expresión lógica correspondiente a AND. Ésta es una descripción lógica en forma de ecuación de la operación de una puerta NAND con dos entradas.

Entradas		Salida
A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

Tabla 4: Tabla de verdad de la puerta NAND de 2 entradas.

LA PUERTA NOR

El término **NOR** es una contracción de NOT-OR e implica una función OR con la salida invertida (complementada). En la Figura 14 se muestra el símbolo lógico estándar para la puerta NOR de 2 entradas y su equivalente empleando los símbolos de la puerta OR seguida de un inversor.

Operación lógica de la puerta NOR

La **puerta NOR** genera una salida a nivel BAJO cuando cualquiera de sus entradas está a nivel ALTO. Cuando todas sus entradas estén a nivel BAJO, la salida se pondrá a nivel ALTO. Para el caso concreto de la puerta NOR de dos entradas, que se muestra en la Figura 14, con la designación A y B para las entradas y X para la salida, el modo de operación se puede establecer como sigue:

En una puerta NOR de dos entradas: la salida X es un nivel BAJO si cualquiera de sus entradas A o B está a nivel ALTO, o si ambas entradas A y B están a nivel ALTO; X es un nivel ALTO si A y B están a nivel BAJO.

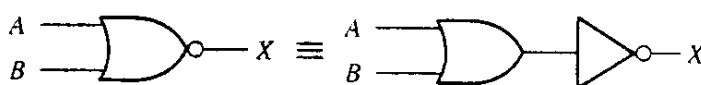


Fig. 14: Símbolo lógico estándar para la puerta NOR.

Esta operación genera un nivel de salida opuesto al que genera la puerta OR. En una puerta NOR, el nivel BAJO es el nivel activo o verdadero de salida, como indica el círculo de la salida. La Tabla 5 es la Tabla de verdad para la puerta NOR de dos entradas.

Entradas		Salida
A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

Tabla 5: Tabla de verdad de la puerta NOR de 2 entradas.

La expresión booleana para la salida de una puerta NOR de dos entradas se expresa así:

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

Esta ecuación indica que las dos variables de entrada primero se suman (operación OR) y luego se complementan, tal y como indica la barra sobre la expresión lógica OR.

PUERTAS OR-EXCLUSIVA Y NOR-EXCLUSIVA

La puerta OR-exclusiva

En la Figura 15 se muestra el símbolo estándar para la puerta OR-exclusiva (XOR) y abajo su tabla de verdad. La puerta XOR tiene sólo dos entradas.

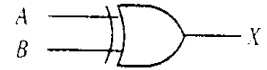


Fig. 15: Símbolo lógico estándar de la puerta OR-exclusiva.

La salida de una **puerta OR-exclusiva** se pone a nivel ALTO sólo cuando las dos entradas están a niveles lógicos opuestos. Esta operación se puede expresar, en función de dos entradas A y B y una salida X, del siguiente modo:

Entradas		Salida
A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

En una puerta OR-exclusiva, la salida X es un nivel ALTO si la entrada A está a nivel BAJO y la entrada B está a nivel ALTO; o si la entrada A está a nivel ALTO y la entrada B está a nivel BAJO; X es un nivel BAJO si tanto A como B están a nivel ALTO o BAJO.

La puerta NOR-exclusiva

En la Figura 16 se presenta el símbolo estándar de la **puerta NOR-exclusiva** (XNOR). Al igual que la puerta XOR, la puerta XNOR sólo tiene dos entradas. El círculo en la salida del símbolo de la puerta XNOR indica que su salida es la opuesta a la de la puerta XOR. Cuando dos niveles lógicos de entrada son opuestos, la salida de la puerta NOR-exclusiva es un nivel BAJO. La operación se puede expresar del siguiente modo (A y B son las entradas, y X es la salida).



Fig. 16: Símbolo lógico estándar para la puerta NOR-exclusiva.

Entradas		Salida
A	B	X
0	0	1
0	1	0
1	0	0
1	1	1

En una puerta NOR-exclusiva, la salida X es un nivel BAJO si la entrada A está a nivel BAJO y la entrada B está a nivel ALTO, o si A está a nivel ALTO y B está a nivel BAJO; X es un nivel ALTO si A y B están ambas a nivel ALTO o BAJO.

Tabla 6: Tabla de verdad de la puerta NOR-exclusiva.

La operación lógica se resume en la Tabla 6. Obsérvese que la salida es un nivel ALTO cuando las dos entradas están al mismo nivel.

PUERTAS LÓGICAS INTEGRADAS

Existen varias tecnologías de circuitos integrados digitales que se usan para implementar las puertas lógicas básica. Dos de esta tecnologías, CMOS y TTL, son las más extendidas y otra de ellas, la ECL, se usa para aplicaciones más especializadas. Las operaciones lógicas NOT, AND, OR, NAND, NOR y OR-exclusiva son las mismas, independientemente de la tecnología de circuitos integrados que se utilice; es decir, una puerta AND tiene la misma función lógica se implemente con la tecnología CMOS, TTL o ECL.

CONSIDERACIONES SOBRE CARACTERÍSTICAS DE LAS PUERTAS

CORRIENTES DE ENTRADA Y DE SALIDA FAN-OUT

La corriente de entrada que tiene una puerta depende de cómo está implementada y en el caso de las puertas integradas de la familia lógica a la que pertenecen. En algunos casos también dependen de que la entrada esté alta o baja, y también varía entre los diversos tipos de compuerta. Por ejemplo para una puerta de la familia TTL (*Transistor Transistor Logic*), la máxima corriente de entrada para una señal de entrada de 1 lógico es de $40\text{ }\mu\text{A}$ y para una entrada de 0 lógico es de $-1,6\text{ mA}$. Como las corrientes se toman convencionalmente positivas cuando van hacia el interior de un dispositivo, el signo negativo indica que esta corriente fluye hacia el exterior del dispositivo.

La especificación para la antedicha familia establece que cuando un 1 lógico es salida, el circuito entrega una corriente de salida de por lo menos $-400\text{ }\mu\text{A}$; y cuando un 0 lógico es salida, absorbe por lo menos 16 mA . Una vez más el signo negativo indica que esta corriente fluye hacia el exterior del dispositivo. Dichas corrientes máximas garantizan que no se sobrepasen los niveles de tensión correspondientes a los niveles lógicos 1 y 0. Teniendo en cuenta las especificaciones de las corrientes de entrada antes señaladas, se puede apreciar que a la salida de una puerta TTL podemos conectar hasta 10 entrada de puertas TTL sin sobrepasar las especificaciones de salida de las mismas. Decimos que la cargabilidad de una puerta TTL es 10 entradas estándar TTL o que tiene un **fanout** de 10, como se ilustra en la figura 1.

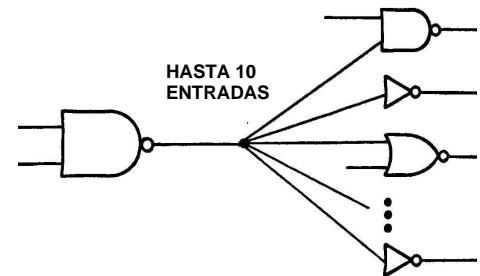


Fig. 1 “Fanout” de una puerta TTL

DISPOSITIVOS EN COLECTOR ABIERTO

Hay situaciones en que es necesario conectar la salida de varias compuertas entre sí, un ejemplo es el caso en que varios dispositivos a su debido tiempo deben colocar un dato en una línea para transmitírselo a un dado receptor. No se deben conectar entre si las salidas de compuertas normales ya que las mismas se pueden dañarán.

Recordemos que la salida de una puerta la podemos representar como una conexión a un par de llaves que actúan de tal forma que cuando una está abierta la otra esta cerrada y viceversa, el esquema se muestra en la figura 2 . Cual de las dos llaves es la que está abierta depende de los estados de la entrada y de la lógica de la compuerta que se trate.

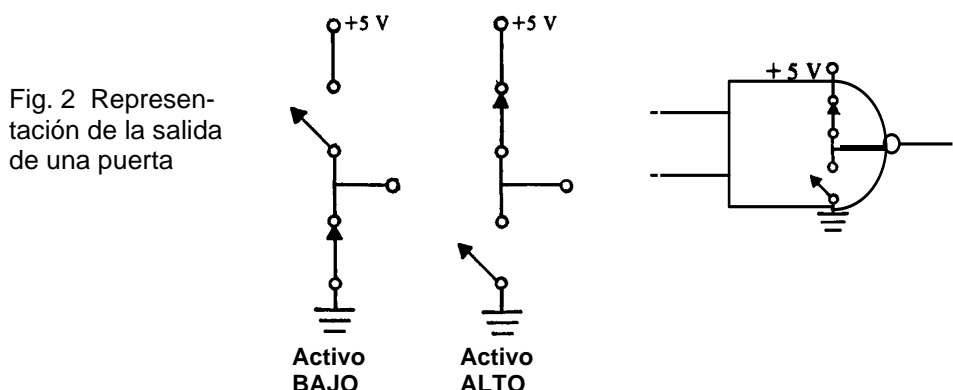


Fig. 2 Representación de la salida de una puerta

Si dos salidas S y T se conectan entre si, y sus niveles lógicos son distintos competirán por fijar su nivel. La Figura 3 muestra esta situación y podemos ver que las llaves cortocircuitan la alimentación. Las llaves están representado los transistores de salida de la puerta que en tales circunstancias pueden quemarse.

Una solución a este problema es el uso de las puertas de *colector abierto* cuya salida usa un solo transistor que actúa como una llave como se muestra en la figura 4.

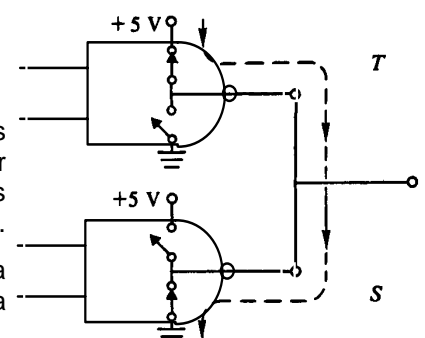


Fig. 3 Salidas conectada entre si

Supongamos que varias puertas NAND de colector abierto se conectan entre si, como se muestra en la figura 5.

Si la salida de alguna de las compuertas va a nivel bajo, su llave conecta la salida a tierra y no habrá problemas de que una salida se quemase dado que no existe puerta que fuerce la salida a tensión de alimentación positiva. Para que la línea de salida pueda ir a un valor alto la salida se debe conectar con un resistor a la tensión positiva. Dicha resistor se la conoce como “*resistencia de pull-up*”.

Como se ve en la figura TTT si alguna de las salida de las puertas va a bajo la salida será baja, se trata de una puerta NOR y esta implementación se llama OR cableada.

Las puertas de colector abierto además de ser útiles para implementar lógica cableada y buses también se usan para alimentar diodos emisores de luz (LEDs) rele y otros dispositivos.

Fig. 5 Tres puertas colector abierto con salidas OR cableada

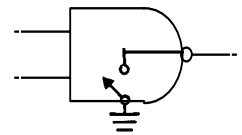
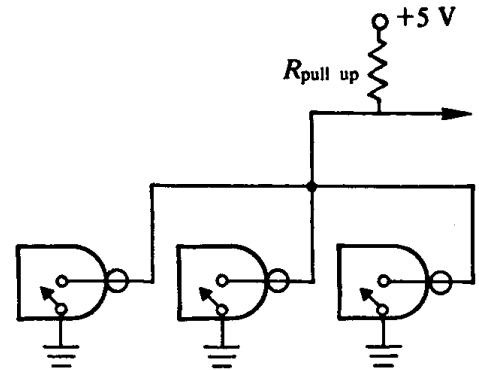


Fig. 4 Esquema de la puerta de colector abierto



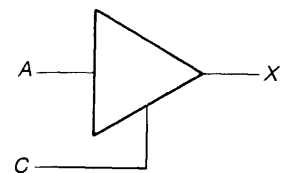
DISPOSITIVOS DE TRES ESTADOS

Aunque los dispositivos de colector abierto pueden usarse para buses actualmente son mas usados los dispositivos con salida de tres estados. Los dispositivos de tres estados se conocen como **tri-state**. Este término es una marca registrada de la National Semiconductor Corporation.

Las compuertas convencionales tienen dos estados de salida posibles: 0 y 1. En algunas circunstancias resulta conveniente contar con un *tercer estado* que corresponda a una condición de **alta impedancia (Hi Z)**, cuando se permite que la salida flote. En estas circunstancias el voltaje de salida estará determinado por cualquier circuito externo que se le conecte. Los circuitos con esta propiedad reciben el nombre de **compuertas lógicas de tres estados**. La salida de la compuerta se “habilita” o “deshabilita” mediante una entrada de control, que por lo general recibe el símbolo C en las compuertas sencillas. En los circuitos más complejos esta señal de control a menudo se conoce como línea de **habilitación de salida**.

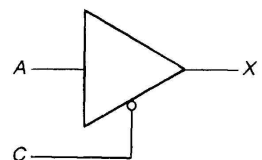
La figura 6 muestra los símbolos lógicos usados para representar la función de tres estados. La figura 6 (a) muestra un buffer no inversor (también los hay inversores) con una entrada de control activa a nivel alto (es decir, la salida se habilita si C = 1); la figura 6 (b) muestra el símbolo para una compuerta similar con una entrada de control activa a nivel bajo (la salida se habilita si C = 0).

El circuito de salida de una compuerta de tres estados lo podemos representar con el símil de llaves empleado anteriormente al que se agrega una tercera llave que conecta o no la salida lógica de acuerdo con la señal de control, Fig. 7. Esto permite que la salida flote, independiente de las otras entradas de la compuerta.



$$X = A \text{ si } C = 1$$

(a) Control activo a nivel alto



$$X = A \text{ si } C = 0$$

(b) Control activo a nivel bajo

Fig. 6 Símbolo lógico de buffer de tres estados

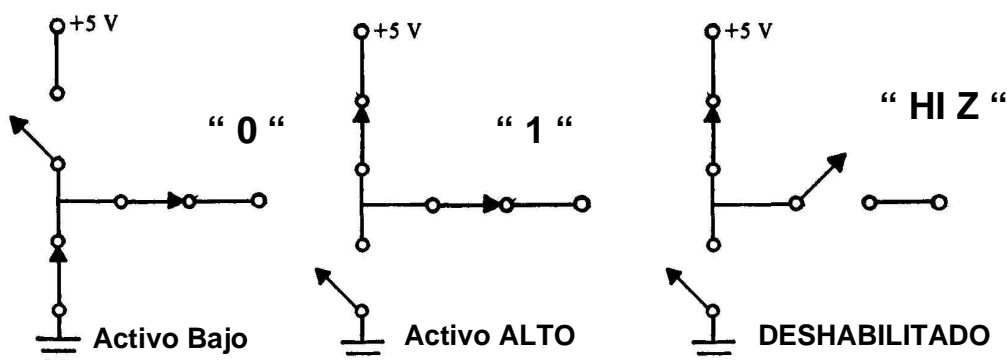


Fig. 7 Ilustración de salida de tres estados

Los dispositivos de tres estados permiten que varias fuentes generadoras de datos compartan una sola línea de comunicación, siempre y cuando un solo dispositivo "hable" en la línea a la vez. La figura 8 muestra un ejemplo de cómo se puede hacer. Tres bits de entrada SSRC2 a SSRC0, selecciona una de las ocho fuentes de datos que pueden transmitir por una sola línea, SDATO haciendo uso de un decodificador de 3 a 8 74LS138 haciendo que solo una de las ocho líneas SEL se active por vez habilitando el correspondiente registro de tres estados.

La mayoría de las aplicaciones de comunicación usan más de una línea de datos, la reunión de varias líneas constituyen un bus. Por ejemplo es común que un sistema posea un bus de 8 líneas que transportan ocho bits de información en forma simultanea. Los dispositivos de tres estados se pueden usar en la creación de sistemas de buses en los que las salidas de varios dispositivos estén conectadas entre sí. Cada dispositivo puede entonces colocar datos sobre el bus siempre y cuando se habilite la salida de un solo dispositivo a la vez. En ocasiones el bus debe transportar información no solo de una serie de transmisores a uno o mas receptores de datos en una sola dirección sino que es necesario que la información se propague en ambos sentidos. Ello se puede solucionar con un tranceptor formado por pares de registros de tres estados conectados en direcciones opuestas de modo que los datos puedan transferirse en cualquier dirección.

En la figura 9 se muestra el diagrama lógico y su símbolo de un tranceptor octal de tres estados. La entrada DIR determina la dirección de transferencia. De A a B, DIR=0 o de B a A DIR=1. El buffer de tres estado solo está habilitado para la dirección seleccionada si la entrada /G está activada.

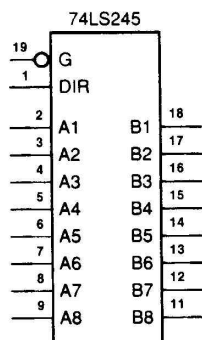


Fig. 9 Tranceptor octal de tres estados

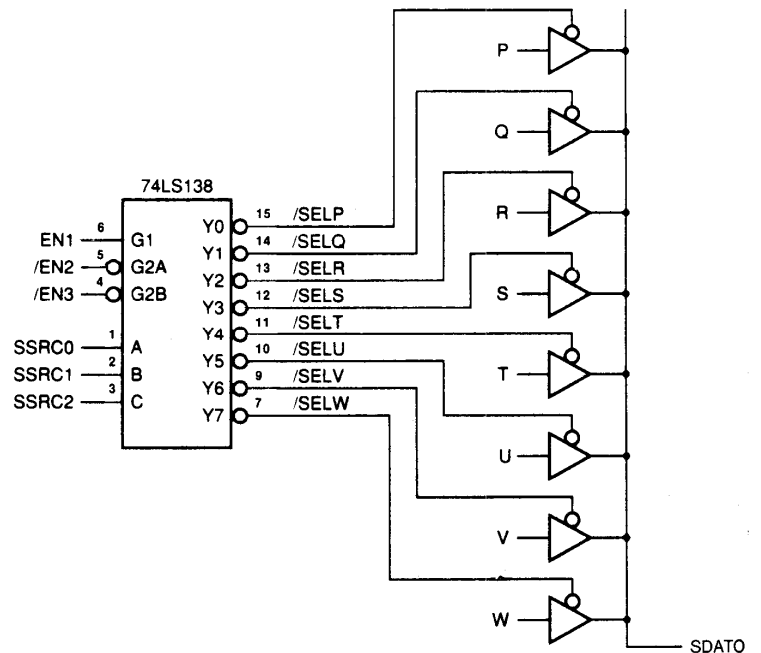
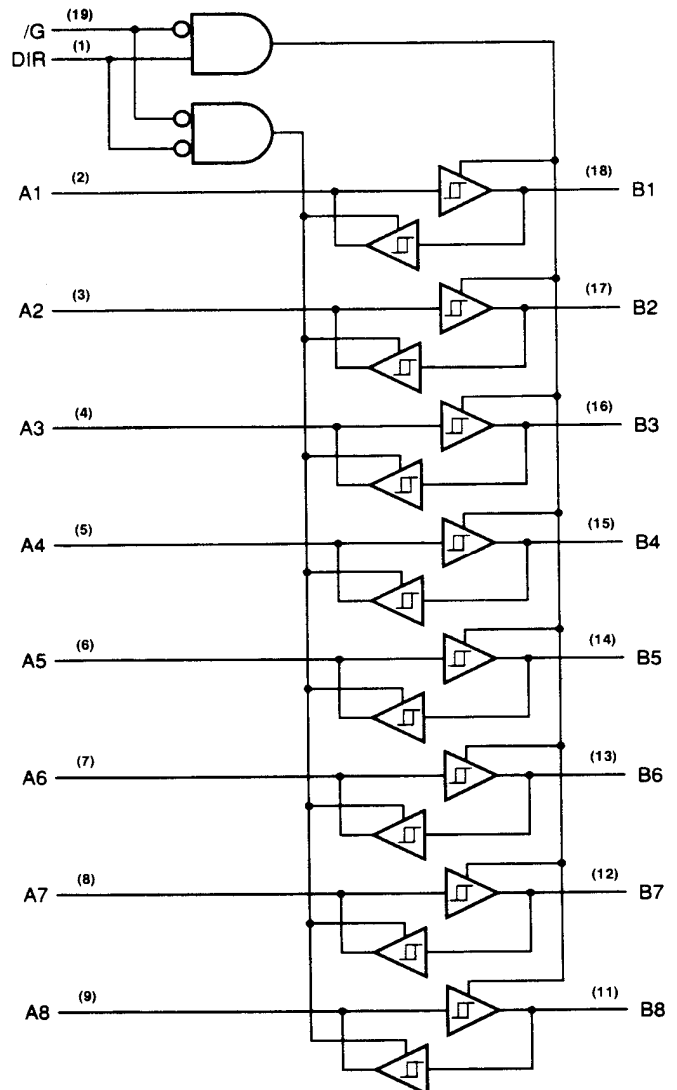


Fig. 8 Ocho generadores comparten una línea común



NIVELES LÓGICOS E INMUNIDAD AL RUIDO

Al tratar brevemente los niveles lógicos vimos que existen dos rangos de tensión, uno para definir el cero lógico y otro para definir el uno. Esos dos niveles están dados por cuatro especificaciones diferentes para los mismos: $V_{H(máx)}$, $V_{H(mín)}$, $V_{L(máx)}$ y $V_{L(mín)}$.

Para las distintas familias lógicas los rangos de tensión para los niveles lógicos son específicos y en general distintos para la entrada que para la salida de las compuertas. Las especificaciones para los rangos de entrada se simboliza con V_{IH} para el nivel alto de entrada y V_{IL} para el nivel bajo de entrada. Para los niveles alto y bajo de salida se simbolizan con V_{OH} V_{OL} respectivamente.

En la figura 10 se muestran los niveles lógicos definidos y garantizados para puertas de la familia TTL. Se puede observar que dentro de las bandas se ha representado una señal con un ruido superpuesto. Si el ruido es excesivo a la entrada de una puerta y sobrepasa los límites $V_{IH(MAX)}$ o decrece de $V_{IH(MIN)}$; la puerta lo puede tomar como un cambio de nivel y responder en consecuencia.

Para que los circuitos lógicos no se vean afectados adversamente por el ruido deben tener cierta **inmunidad al ruido**, que es la capacidad de tolerar fluctuaciones de tensión en su entrada sin que cambie la salida.

Si en la figura 10 el ruido hace que la tensión del nivel ALTO caiga por debajo de 2 volt el funcionamiento no es predecible y puede ser que la puerta lo tome como un nivel bajo. De forma similar una tensión superior a 0.8 volt en el nivel Bajo puede ser interpretado como ALTO produciendo un error.

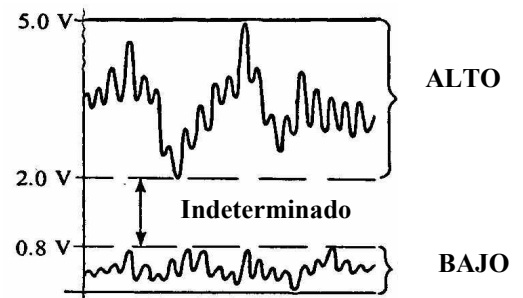
La medida de la inmunidad al ruido de un circuito se denomina **margen de ruido**. Se especifican dos valores de margen de ruido: margen de ruido de nivel alto V_{NH} y margen de ruido de nivel bajo V_{NL} que se definen con las siguientes ecuaciones

$$V_{NH} = V_{OH(MIN)} - V_{IH(MIN)}$$

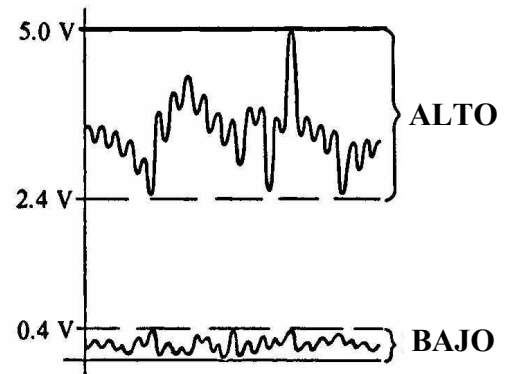
$$V_{NL} = V_{IL(MAX)} - V_{OL(MAX)}$$

Los márgenes de ruido se ilustran en la figura 11 para la familia TTL siendo distinto para otras familias lógicas, por ejemplo en la CMOS los márgenes son mayores.

Cuando la tolerancia al ruido sea una consideración crítica, CMOS es una elección obvia pues ofrece una inmunidad al ruido alta. Es común que estas aplicaciones usen una tensión de alimentación de 15 V para mejorar más aún la inmunidad al ruido, aunque esto afecta de manera adversa el consumo de potencia.



a) Niveles de entrada requeridos



b) Nivel de salida garantizado para 10 cargas TTL

Fig. 10 Niveles para TTL a) de entrada, b) de salida

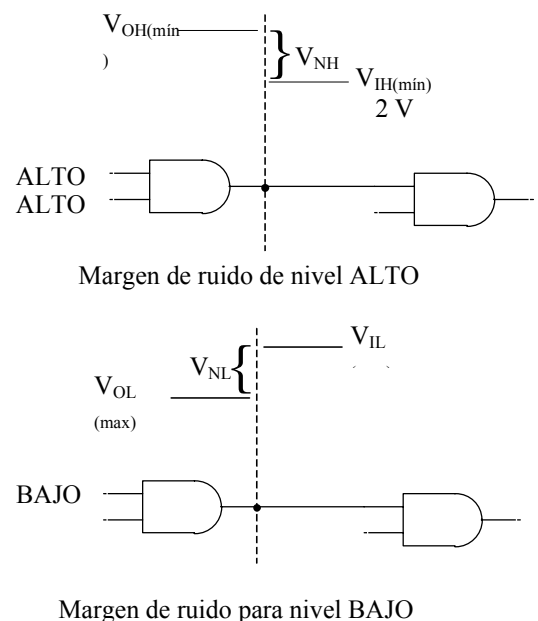


Fig. 11 Ilustración de los márgenes de ruido en TTL

ENTRADAS DE DISPARO DE SCHMITT

Cuando las señales de entrada a una puerta lógica varían lentamente la salida puede realizar algunas transiciones cuando la entrada está en la región indefinida entre los niveles alto y bajo. Estos problemas relacionados con las entradas de variación lenta se pueden disminuir mediante compuertas lógicas con **histéresis**. Uno de los circuitos más comunes con esta característica es el circuito **disparador de Schmitt (Schmitt Trigger)**, que produce un solo cambio en el estado lógico aun para una entrada ruidosa de variación lenta. La característica

de transferencia de un inversor Schmitt Trigger de la familia TTL se muestra en la figura 12.

De la característica de transferencia se puede ver que la salida sólo cambia de alto a bajo si la entrada supera el voltaje de umbral superior V_{TH} , y sólo cambia de bajo a alto si la entrada es inferior al voltaje de umbral inferior V_{TL} . Esto evita que pequeñas cantidades de ruido conmuten la salida en forma repetitiva entre los dos estados de salida. Para el caso ilustrado la histéresis típica es del orden de los 800 mV. Está determinada por una realimentación positiva y su valor fijado por resistores internos.

Se pueden encontrar dos tipos de puertas lógicas del disparador de Schmitt: el inversor y compuertas NAND cuyos símbolos se muestran en la figura 13 que lo distinguen de la los símbolos lógicos correspondientes por el lazo de histéresis dibujado en su interior.

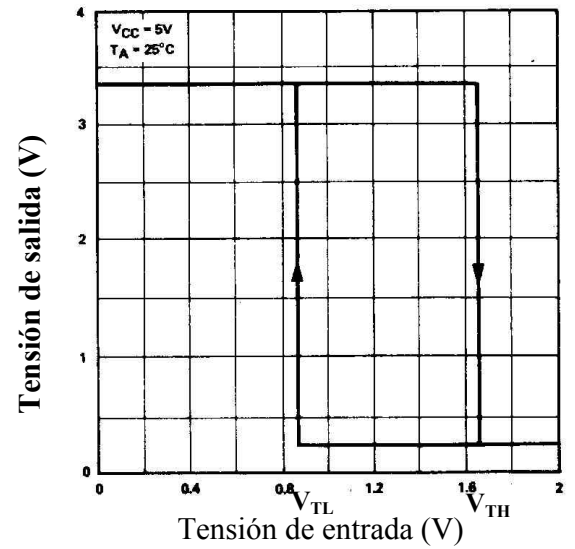


Fig.12 Curva de transferencia de un inversor TTL Schmitt trigger



Fig.13 Símbolos Lógicos de puertas Schmitt Trigger

ELEMENTOS ALGEBRA DE BOOLE

El álgebra de Boole son las matemáticas de los sistemas digitales.

Es indispensable tener unos conocimientos básicos del álgebra booleana para estudiar y analizar los circuitos lógicos. Anteriormente se han introducido las operaciones y expresiones booleanas para las puertas NOT, AND, OR, NAND y NOR. Haremos una revisión y ampliación de las mismas.

Los términos *variable*, *complemento* y *literal* son términos ampliamente utilizados en el álgebra de Boole. Una **variable** es un símbolo (en general se usa una letra mayúscula) que se utiliza para representar magnitudes lógicas. Cualquier variable puede tener un valor 1 o 0. El **complemento** es el inverso de la variable y se indica mediante una barra encima de la misma. Por ejemplo, el complemento de la variable A es \bar{A} . Si $A = 1$, entonces $\bar{A} = 0$. El complemento de la variable A se lee como "A negada" o "complementada". Algunas veces también se utiliza un apóstrofe para indicar el complemento de una variable en lugar de la barra. Por ejemplo, el complemento de B puede escribirse B'. Un **literal** se define como una variable o el complemento de una variable.

Adición booleana

Como vimos, la **suma booleana** es equivalente a la operación OR y sus reglas son:

$$0 + 0 = 0; \quad 0 + 1 = 1; \quad 1 + 0 = 1; \quad 1 + 1 = 1$$

En el álgebra de Boole, un **término suma** es una suma de literales. En los circuitos lógicos, un término suma se produce mediante la operación OR, sin que exista ninguna operación AND en la expresión. Algunos ejemplos son: $A + B$, $A + \bar{B}$, $A + B + \bar{C}$ y $\bar{A} + B + C + \bar{D}$.

Un término suma es igual a 1 cuando uno o más de los literales es un 1. Un término suma es igual a 0 si y sólo si cada uno de los literales es 0.

Multiplicación booleana

Vimos también que la **multiplicación booleana** es equivalente a la operación AND y sus reglas básicas son:

$$0 \bullet 0 = 0; \quad 0 \bullet 1 = 0; \quad 1 \bullet 0 = 0; \quad 1 \bullet 1 = 1$$

En el álgebra de Boole, un **término producto** es el producto de literales. En los circuitos lógicos, un término producto se produce mediante la operación AND, sin que aparezca ninguna operación OR en la expresión. Algunos ejemplos de términos producto son: AB , $A\bar{B}$, ABC y $A\bar{B}C\bar{D}$.

Un término producto es igual a 1 si y sólo si cada uno de los literales es 1. Un término producto es igual a 0 si uno o más de sus literales es 0.

LEYES Y REGLAS DEL ÁLGEBRA DE BOOLE

Las leyes básicas del álgebra de Boole (las **leyes conmutativas** de la suma y la multiplicación, y las **leyes asociativas** de la suma y la multiplicación y la **ley distributiva**) son las mismas que las del álgebra ordinaria. Cada una de las leyes se ilustra con dos o tres variables, pero el número de variables no está limitado a esta cantidad.

Leyes conmutativas. La ley conmutativa de la suma para dos variables se escribe como sigue:

$$A + B = B + A$$

Esta ley establece que el orden en que se aplica a las variable la operación OR es indiferente. Recuerde que cuando se aplica a los circuitos lógicos la terminología del álgebra de Boole, la adición y la operación OR son una misma cosa.

La ley conmutativa de la multiplicación para dos variables es:

$$AB = BA$$

Esta ley establece que el orden en que se aplica a las variables la operación AND es indiferente.

Leyes asociativas. La ley asociativa de la adición para tres variables se escribe, en forma algebraica, de la siguiente manera:

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

Esta ley establece que al aplicar la operación OR a más de dos variables, el resultado es el mismo independientemente de la forma en que se agrupen las variables. En la Figura 4.3 podemos ver esta ley aplicada a las puertas OR.

La ley asociativa de la multiplicación para tres variables se escribe de la siguiente manera:

$$A(BC) = (AB)C$$

Esta ley establece que al aplicar la operación AND a más de dos variables, el resultado es el mismo independientemente de la forma en que se agrupen las variables.

Ley distributiva. La ley distributiva para tres variables se escribe de la siguiente manera:

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

Esta ley establece que aplicar la operación OR a dos o más variables y luego aplicar la operación AND al resultado de esta operación y a otra variable aislada, es equivalente a aplicar la operación AND a la variable aislada con cada uno de los sumandos y luego aplicar la operación OR a los productos resultantes. La ley distributiva también expresa el proceso de sacar factor común, en el que la variable común A se saca como factor de los productos parciales, como, por ejemplo en $AB + AC = A(B + c)$.

Reglas del álgebra booleana

En el listado que sigue se enumera las doce reglas básicas, muy útiles, para la manipulación y simplificación de **expresiones booleanas**. Las nueve primeras reglas responden en términos de su aplicación a las puertas lógicas. Las reglas 10, 11 y 12 pueden obtenerse a partir de las regla más sencillas y de las leyes anteriormente explicadas.

- 1.- $1 + A = 1$
- 2.- $0 + A = A$
- 3.- $A + A = A$
- 4.- $A + \bar{A} = 1$
- 5.- $1 \cdot A = A$
- 6.- $0 \cdot A = 0$
- 7.- $A \cdot A = A$
- 8.- $A \cdot \bar{A} = 0$
- 9.- $\overline{\bar{A}} = A$
- 10.- $A + AB = A$
- 11.- $A + \bar{A}B = A + B$
- 12.- $(A + B)(A + C) = A + BC$

TEOREMAS DE DEMORGAN

DeMorgan, propuso dos teoremas que constituyen una parte muy importante del álgebra de Boole. En términos prácticos, los teoremas de DeMorgan nos demuestran la equivalencia entre las puertas NAND y negativa-OR, y las puertas NOR y negativa-AND, que se han tratado.

El primer teorema de DeMorgan se enuncia de la siguiente forma:

El complemento de un producto de variables es igual a la suma de los complementos variables.

O dicho de otra manera:

El complemento de dos o más variables a las que se aplica la operación AND es equivalente a aplicar la operación OR a los complementos de cada variable.

La fórmula para expresar este teorema para dos variables es:

$$\overline{XY} = \bar{X} + \bar{Y}$$

El segundo teorema de DeMorgan se enuncia así:

El complemento de una suma de variables es igual al producto de los complementos de las variables.

O dicho de otra manera:

El complemento de dos o más variables a las que se aplica la operación OR es equivalente a aplicar la operación AND a los complementos de cada variable.

La fórmula para expresar este teorema es:

$$\overline{X + Y} = \overline{X} \overline{Y}$$

Como se ha comentado, los teoremas de DeMorgan se aplican también a expresiones en las que existen más de dos variables.

COMBINACIONES DE PUERTAS LÓGICAS

Los sistemas digitales están formados por combinaciones de puertas lógicas, las cuales pueden ser descritas por una tabla de verdad, expresiones booleanas o diagramas de símbolos lógicos.

Se puede presentar el caso que tenemos un sistema muy elaborado de puertas lógicas y deseamos conocer cómo es su comportamiento. Para ello construimos la tabla de verdad del diagrama, realizamos una *síntesis* del sistema. En la Figura 1 a se muestra un diagrama de dos entradas constituido por dos inversores, dos puertas AND de dos entradas y una OR de dos entradas. En la Figura 1 b se muestra la tabla de verdad que incluye las puertas intermedias P y Q para facilitar su construcción. Vemos que el resultado coincide con el de la puerta XOR. Este procedimiento puede ser aplicado a combinaciones complejas de puertas analizando todas las combinaciones posibles de entrada.

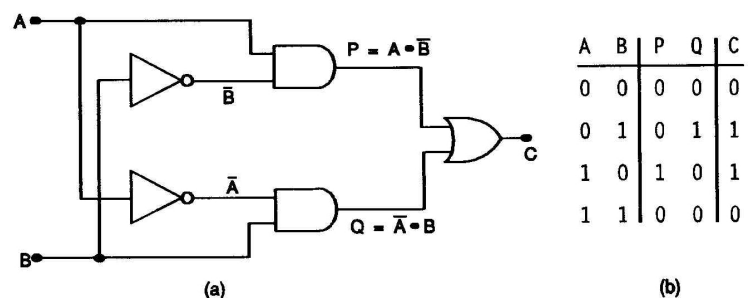


Fig. 1 Puerta OR-EXCLUSIVA realizada con puertas

Por otra parte, frecuentemente es necesario usar el procedimiento inverso: construir el diagrama de puertas que cumpla con una dada tabla de verdad. Supongamos que se requiere que la salida sea verdadera (igual a uno) para las combinaciones binarias de entrada 0110, 1010 y 1110, pero no para cualquier otra. Agudizando el ingenio siempre podemos construir un arreglo de compuertas que responde a lo requerido. Caso contrario podemos recurrir a la "fuerza bruta", método que se ilustra en la Figura 2. Las cuatro entradas A, B, C y D están conectadas a tres puertas AND con inversores de ser necesario. La salida de cada AND responde a una de las tres condiciones especificadas.

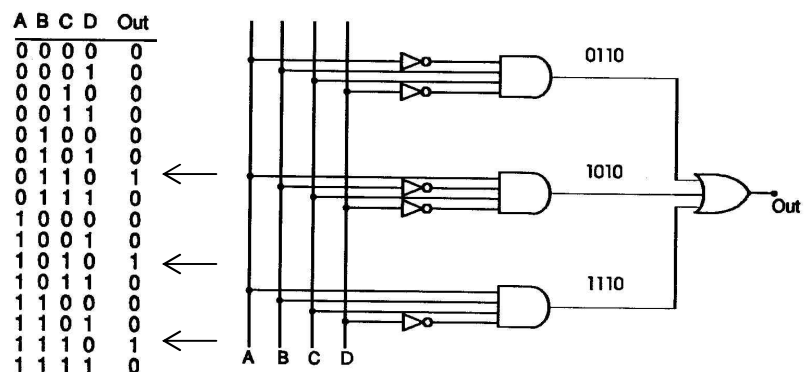
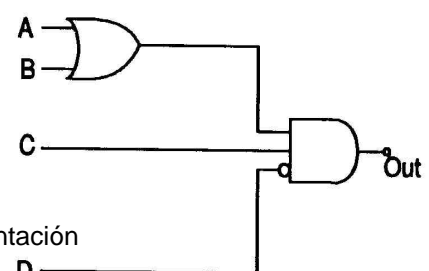


Fig.2 Implementación a "fuerza bruta" de la tabla de verdad

Las salidas están combinadas en una OR de tres entradas, lo que a su salida genera la respuesta correcta. Una solución más eficiente se muestra en la Figura 3 que con menos puertas responde a la misma tabla de verdad.

Planteado un sistema lógico siempre podemos expresarlo como una tabla de verdad y ésta como una ecuación booleana, o viceversa, las reglas y leyes nos permiten manipular las expresiones booleanas para simplificar la implementación con compuertas.

Fig. 3 implementación simplificada



SIMPLIFICACIÓN MEDIANTE EL ALGEBRA DE BOOLE

Muchas veces, a la hora de aplicar el álgebra booleana, hay que reducir una expresión a su forma más simple o cambiarla a una forma más conveniente para conseguir una implementación más eficiente. El método que se va a tratar en presente ejemplo utiliza las reglas, leyes y teoremas del álgebra de Boole para manipular y simplificar una expresión. En general este método requiere un profundo conocimiento del álgebra booleana y una considerable experiencia en su aplicación, por no mencionar también un poquito de ingenio y destreza.

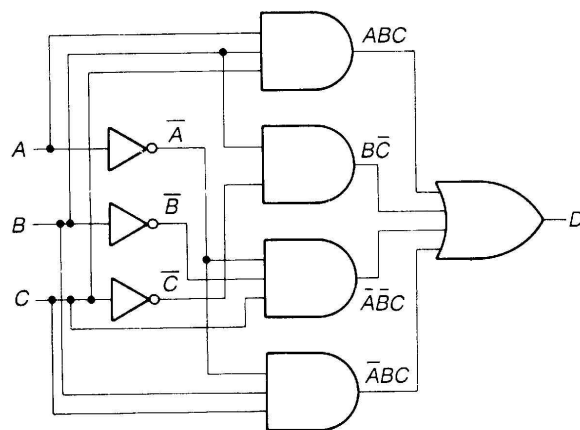
Una expresión booleana simplificada emplea el menor número posible de puertas en la implementación de una determinada expresión. A continuación, se presenta un ejemplo de simplificación descriptos paso a paso.

Ejemplo:

Considérese la expresión:

$$D = \overline{A}\overline{B}C + B\overline{C} + \overline{A}BC + ABC$$

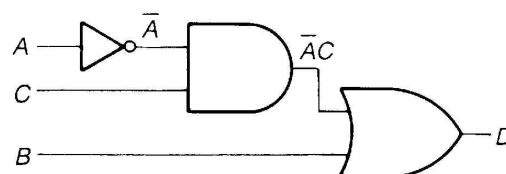
Esto se puede implantar directamente como



En forma alternativa, se puede reducir mediante el uso del álgebra booleana, como se indica utilizando las identidades o leyes antes presentadas.

$$\begin{aligned}
 D &= \overline{A}\overline{B}C + B\overline{C} + \overline{A}BC + ABC \\
 &= \overline{A}\overline{B}C + B\overline{C} + BC(\overline{A} + A) && \text{Ley distributiva} \\
 &= \overline{A}\overline{B}C + B\overline{C} + BC && \text{función OR} \\
 &= \overline{A}\overline{B}C + B(\overline{C} + C) && \text{Ley distributiva} \\
 &= \overline{A}\overline{B}C + B && \text{función OR} \\
 &= \overline{A}C + B && \text{regla n}^\circ 11
 \end{aligned}$$

Esto se puede implantar con sólo tres compuertas:



Resulta claro a partir del ejemplo anterior, que esta forma de simplificación algebraica puede reducir en gran medida la complejidad de expresiones booleanas, así como el costo de la puesta en práctica. Sin embargo, no siempre está claro que el proceso de simplificación siempre sea directo. La manipulación de este tipo con frecuencia es un inspirado trabajo de adivinación y presenta el problema de que uno nunca está seguro de si se obtuvo una solución óptima.

Es importante entender el proceso de simplificación algebraica pues constituye una técnica útil para funciones sencillas. Sin embargo, para expresiones más complejas normalmente se emplean métodos más poderosos.

MAPAS DE KARNAUGH

Un mapa de Karnaugh proporciona un método sistemático de simplificación de expresiones booleanas y, si se aplica adecuadamente, genera las expresiones suma de productos y producto de sumas más simples posibles. Como hemos visto, la efectividad de la simplificación algebraica depende de nuestra familiaridad con las leyes, reglas y teoremas del álgebra booleana y de nuestra habilidad a la hora de aplicarlas. Por otro lado, el mapa de Karnaugh es básicamente una “receta” para la simplificación.

Un mapa de Karnaugh es similar a una tabla de verdad, ya que muestra todos los posibles valores de variables de entrada y la salida resultante para cada valor. En vez de estar organizada en filas y columnas como una tabla de verdad, el mapa de Karnaugh es una secuencia de **celdas** en la que cada celda representa un valor binario de las variables de entrada. Las celdas se disponen de manera que la simplificación de una determinada expresión consiste en agrupar adecuadamente las celdas. Los mapas de Karnaugh pueden utilizarse para expresiones de dos, tres, cuatro y cinco variables. Existe otro método, que también está fuera de nuestro propósito, denominado método de Quine-McClusky, que puede usarse para un número de variables mayor.

LÓGICA COMBINACIONAL

INTRODUCCIÓN:

A continuación se presentan distintos tipos de circuitos lógicos combinacionales MSI (Medium Scale Integration, integración a media escala), incluyendo sumadores, comparadores, decodificadores, codificadores, convertidores de código, multiplexores (selectores de datos) y demultiplexores. También se incluyen algunos ejemplos de aplicación de estos dispositivos para ilustrar cómo pueden usarse en situaciones prácticas.

SUMADORES

Los sumadores son muy importantes no solamente en las computadoras, sino en muchos tipos de sistemas digitales en los que se procesan datos numéricos. Comprender el funcionamiento de un sumador es fundamental en el estudio de los sistemas digitales. Estudiaremos el semisumador y el sumador completo.

El semisumador

Recordemos las reglas básicas de la suma binaria:

$$\begin{aligned} 0 + 0 &= 0 \\ 0 + 1 &= 1 \\ 1 + 0 &= 1 \\ 1 + 1 &= 10 \end{aligned}$$

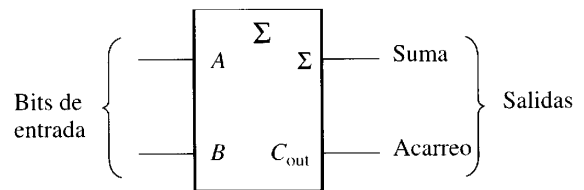


Fig. 1 Símbolo lógico del semisumador

Todas estas operaciones se realizan mediante un circuito lógico denominado *semisumador*.

Un semisumador admite dos dígitos binarios en sus entradas y genera dos dígitos binarios en sus salidas: un bit de suma y un bit de acarreo.

Los semisumadores se representan mediante el símbolo lógico de la Figura 1.

Lógica del semisumador. A partir del funcionamiento lógico de un semisumador expuesto en la Tabla, las expresiones correspondientes a la suma y al acarreo de salida se pueden obtener como funciones de las entradas. Obsérvese que la salida de acarreo (C_{out}) es 1 sólo cuando A y B son 1; por lo tanto, C_{out} puede expresarse como una operación AND de las variables de entrada.

$$C_{out} = AB$$

Obsérvese ahora que la salida correspondiente a la suma (Σ) es uno sólo si las variables A y B son distintas. Por tanto, la suma puede expresarse como una operación OR-exclusiva de las variables de entrada.

$$\Sigma = A \oplus B$$

A partir de las ecuaciones anteriores, se puede desarrollar la implementación lógica del funcionamiento de un semisumador. La salida de acarreo se produce mediante una puerta AND, siendo A y B sus dos entradas, y la salida de la suma se obtiene mediante una puerta OR-exclusiva, como muestra la Figura 2.

A	B	C_{out}	Σ
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Σ = suma

C_{out} = acarreo de salida

A y B = variables de entrada (operandos)

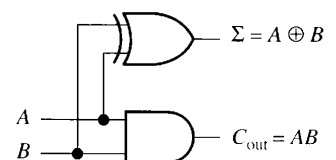


Fig. 2 Semisumador

Lógica del sumador completo.

Un sumador completo suma dos bits de entrada y el bit de acarreo de entrada, fig 3. A partir del semisumador, ya conocemos que la suma de los dos bits de entrada A y B consiste en la operación OR-exclusiva entre estas dos variables, $A \oplus B$. Para sumar el acarreo de entrada (C_{in}) a los bits de entrada, hay que volver a aplicar la operación OR-exclusiva, obteniéndose la siguiente ecuación de salida para el sumador completo:

$$\Sigma = (A \oplus B) \oplus C_{in}$$

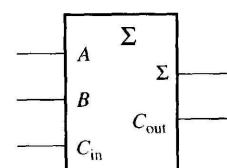
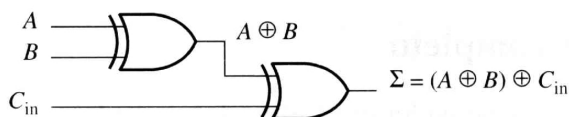
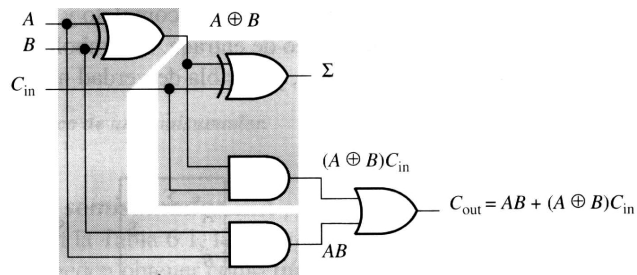


Fig. 3 Símbolo lógico del sumador



(a) Lógica necesaria para realizar la suma de tres bits

Fig. 4 Sumador completo



(b) Circuito lógico de un sumador completo

Esto significa que, para implementar la función de un sumador completo, se pueden utilizar dos puertas OR-exclusiva. La primera tiene que generar el término $A \oplus B$, y la segunda toma como entradas la salida de la primera puerta XOR y el acarreo de entrada, como se muestra en la Figura 4 (a).

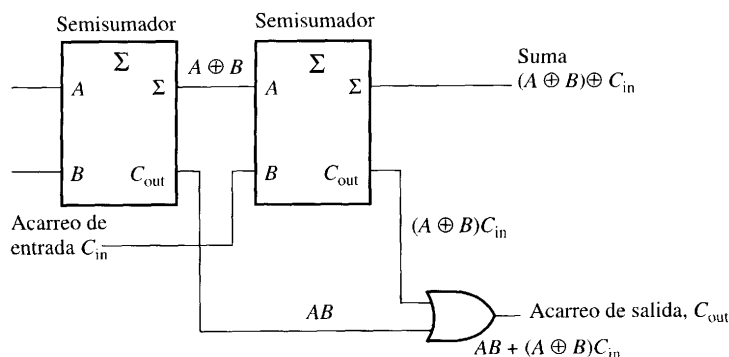
El acarreo de salida es 1 cuando las dos entradas de la primera puerta XOR son 1, o cuando las dos entradas de la segunda puerta XOR son 1. Esto se puede comprobar analizando la Tabla de la derecha. El acarreo de salida del sumador completo se obtiene a partir del producto lógico (AND) de las entradas A y B, y del producto lógico (AND) de $A \oplus B$ y de C_{in} , sumando (OR) después ambos términos resultantes, como se muestra en la siguiente ecuación. Esta función, una vez implementada, se combina con la de la suma lógica para constituir un circuito sumador completo, como se muestra en la Figura 4 (b).

A	B	C_{in}	C_{out}	Σ
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

C_{in} = acarreo de entrada, algunas veces se designa por CI
 C_{out} = acarreo de salida, algunas veces se designa por CO
 Σ = suma
A y B = variables de entrada (operandos)

$$C_{out} = AB + (A \oplus B) C_{in}$$

Obsérvese que, en la Figura 4 (b), existen dos semisumadores conectados, como se muestra en el diagrama de bloques de la Figura 5 (a), cuyos acarreos de salida se aplican a una puerta OR. El símbolo lógico mostrado en la Figura 5 (b) será el que normalmente empleemos para representar un sumador completo.



(a) Dos semisumadores formando un sumador completo

Fig. 5 Sumador completo construido con dos semisumadores

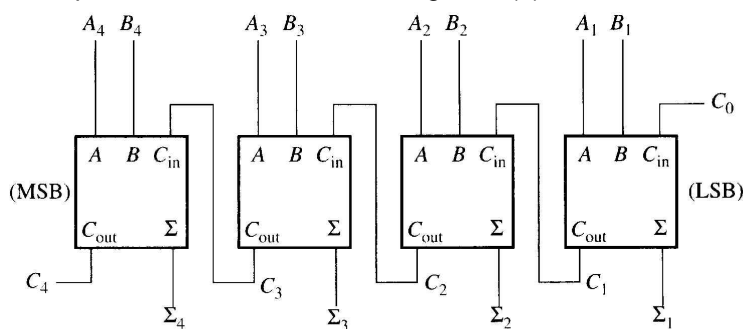
SUMADORES BINARIOS EN PARALELO

Para formar un sumador binario en paralelo se conectan dos o más sumadores completos.

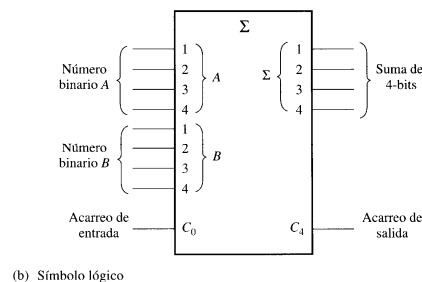
Sumadores en paralelo de cuatro bits

Un grupo de cuatro bits se denomina **nibble**. Un sumador básico en paralelo de 4 bits se implementa mediante cuatro sumadores completos, como se muestra en la Figura 6 (a). De nuevo, los bits menos significativos (A_1 y B_1) de cada número que se suma, se introducen en el sumador completo que está más a la derecha; los bits de orden más alto se introducen sucesivamente en los siguientes sumadores, aplicando los bits más significativos de cada número (A_4 y B_4) al sumador que está más a la izquierda. La salida de acarreo de cada sumador se conecta a la entrada de acarreo del siguiente sumador de orden superior. Estos acarreos se denominan acarreos internos.

En la mayoría de las hojas de características suministradas por los fabricantes, se denomina C_0 al acarreo de entrada del sumador del bit menos significativo; C_4 , en el caso de cuatro bits, sería el acarreo de salida del sumador del bit más significativo; Σ_1 (LSB) hasta Σ_4 (LSB) son las sumas de salida. El símbolo lógico correspondiente se muestra en la Figura 6 (b).



(a) Diagrama de bloques



(b) Símbolo lógico

Fig. 6 Sumador en paralelo de un nibble

En función del método utilizado para manipular los acarreos¹ en un sumador paralelo, existen dos tipos: el sumador de *acarreo serie* y el sumador de *acarreo anticipado*.

Un sumador de **acarreo serie** es aquel en el que la salida de acarreo de cada sumador completo se conecta a la entrada de acarreo de la siguiente etapa de orden inmediatamente superior (una etapa es un sumador completo). La suma y el acarreo de salida de cualquier etapa no se pueden generar hasta que tiene lugar el acarreo de entrada, lo que da lugar a un retardo temporal en el proceso de adición. El retardo de propagación del acarreo para cada sumador completo es el tiempo transcurrido desde la aplicación del acarreo de entrada hasta que se produce el acarreo de salida, suponiendo que las entradas A y B ya existen.

Un método que permite acelerar el proceso de adición eliminando este retardo del acarreo serie es la adición con **acarreo anticipado**. El sumador con acarreo anticipado anticipa el acarreo de salida de cada etapa y, en función de los bits de entrada de ambos sumandos, genera el acarreo de salida bien mediante la generación de acarreo o la propagación de acarreo.

La **generación de acarreo** tiene lugar cuando el sumador completo genera internamente un acarreo de salida. Sólo cuando ambos bits de entrada son 1 se genera un acarreo. El acarreo de salida, C_g , se expresa como la función AND de los 2 bits de entrada, A y B.

$$C_g = AB$$

La **propagación de acarreo** tiene lugar cuando el acarreo de entrada se transmite como acarreo de salida. Un acarreo de entrada puede ser propagado por el sumador completo cuando uno o ambos bits de entrada son igual a 1. El acarreo propagado, C_p , se expresa como la función OR de los bits de entrada.

$$C_p = A + B$$

Tabla de verdad de un sumador en paralelo de 4 bits

La Tabla es la tabla de verdad de un sumador de 4 bits. En algunas hojas de características, las tablas de verdad se denominan *tablas de función* o *tablas de verdad funcionales*. El subíndice n representa los bits del sumador y puede ser igual a 1, 2, 3 ó 4 para un sumador de 4 bits. C_{n-1} es el acarreo del sumador previo. Los acarreos C_1 , C_2 y C_3 se generan internamente. C_0 es un acarreo de entrada externo y C_4 es una salida.

C_{n-1}	A_n	B_n	Σ_n	C_n
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

COMPARADORES

La función básica de un comparador consiste en comparar las magnitudes de dos cantidades binarias para determinar su relación. En su forma más sencilla, un circuito comparador determina si dos números son iguales.

Igualdad

Como ya vimos, la puerta OR-exclusiva se puede emplear como un comparador básico, ya que su salida es 1 si sus dos bits de entrada son diferentes y 0 si son iguales.

Para comparar números binarios de dos bits, se necesita una puerta OR-exclusiva adicional. Los dos bits menos significativos (LSB) de ambos números se comparan mediante la puerta G_1 y los dos más significativos (MSB) son comparados mediante la puerta G_2 como se muestra en la Figura 7. Si los dos números son iguales, sus correspondientes bits también lo son, y la salida de cada puerta OR-exclusiva será 0. Si los correspondientes conjuntos de bits no son idénticos, la salida de la puerta OR-exclusiva será un 1.

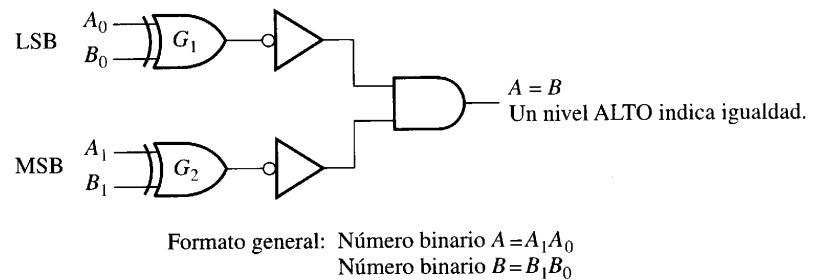


Fig. 7 Comparador de 2 números de 2 bits

Para obtener un único resultado de salida que indique la igualdad o desigualdad entre los dos números, se pueden usar dos inversores y una puerta AND, como muestra la Figura 7. La salida de cada puerta OR-exclusiva se invierte y se aplica a la entrada de la puerta AND. Cuando los bits de entrada de cada OR-exclusiva son iguales, lo que quiere decir que los bits de ambos números son iguales, las entradas de la puerta AND son 1, por lo que el resultado a su salida también será 1. Cuando los dos números no son iguales, al menos uno o ambos conjuntos de bits será distinto, lo que da lugar a, al menos, un 0 en una de las entradas de la puerta AND, y el resultado a su salida será 0. Por tanto, la salida de la puerta AND indica la igualdad (1) o desigualdad (0) entre dos números.

Desigualdad

Además de disponer de una salida que indica si los dos números son iguales, muchos circuitos integrados comparadores tienen salidas adicionales que indican cuál de los dos números que se comparan es el mayor. Esto significa que existe una salida que indica cuándo el número A es mayor que el número B ($A > B$) y otra salida que indica cuándo A es menor que B ($A < B$), como se muestra en el símbolo lógico del comparador de cuatro bits de la Figura 8.

Para determinar una desigualdad entre los números binarios A y B, en primer lugar se examina el bit de mayor orden de cada número. Las posibles condiciones son las siguientes:

- 1.- Si $A_3 = 1$ y $B_3 = 0$, entonces A es mayor que B.
- 2.- Si $A_3 = 0$ y $B_3 = 1$, entonces A es menor que B.
- 3.- Si $A_3 = B_3$, entonces tenemos que examinar los siguientes bits de orden inmediatamente inferior.

Estas tres posiciones son válidas para cada posición que ocupen los bits dentro del número. El procedimiento general consiste en comprobar una desigualdad en cualquier posición, comenzando por los bits más significativos (MSB). Cuando se encuentra una desigualdad entre bits con posiciones de orden menor debe ignorarse, ya que podrían indicar una relación entre los números completamente opuesta. *La relación de más alto orden es la que tiene prioridad.*

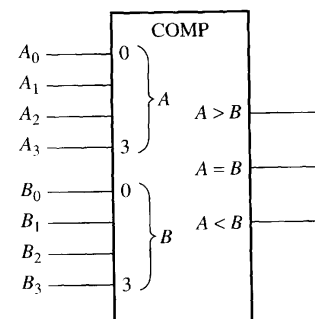


Fig.8 comparador de 4 bits

DECODIFICADORES

La función básica de un decodificador es detectar la presencia de una determinada combinación de bits (código) en sus entradas y señalar la presencia de este código mediante un cierto nivel de salida. En su forma más general, un decodificador posee, líneas de entrada para gestionar n bits, y en una de las 2^n líneas de salida indica la presencia de una o más combinaciones de n bits.

El decodificador binario básico

Supongamos que necesitamos determinar cuándo aparece el número binario 1001 en las entradas de un circuito digital. Se puede utilizar una puerta AND como elemento básico de decodificación, ya que produce una salida a nivel ALTO

sólo cuando todas sus entradas están a nivel ALTO. Por tanto, debe asegurarse de que todas las entradas de la puerta AND estén a nivel ALTO cuando se introduce el número 1001, lo cual se puede conseguir invirtiendo las dos entradas centrales (cuyos bits son 0), como se muestra en la Figura 9.

La ecuación lógica para el **decodificador** de la Figura 9 (a) se desarrolla como se ilustra en la Figura 9(b). Se debe comprobar que la salida es siempre 0 excepto cuando se aplican las entradas $A_0 = 1$, $A_1 = 0$, $A_2 = 0$ y $A_3 = 1$. A_0 es el bit menos significativo y A_3 el más significativo. En general cuando se representa un número binario o cualquier otro código de pesos, el bit menos significativo siempre es el situado más a la derecha cuando el número se escribe en sentido horizontal, y el de más arriba cuando se escribe en vertical, a menos que se indique lo contrario.

Si se utiliza una puerta NAND en lugar de una AND, como se muestra en la Figura 9, una salida a nivel BAJO indicará la presencia del código binario adecuado, que en este caso es 1001.

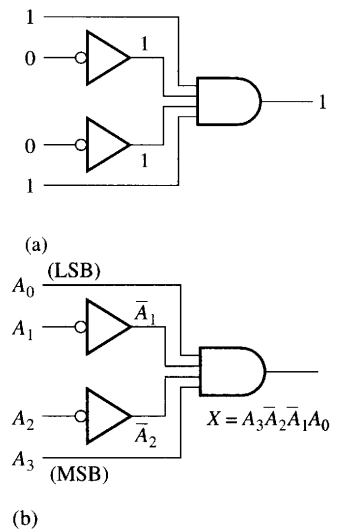


Fig.9 decodificador del código 1001

El decodificador de 4 bits

Para poder decodificar todas las posibles combinaciones de cuatro bits, se necesitan 16 puertas de decodificación ($2^4 = 16$). Este tipo de decodificador se denomina comúnmente *decodificador de 4 líneas a 16 líneas*, ya que para cualquier código dado en las entradas, sólo se activa una de las dieciséis posibles salidas. En la Tabla se muestra una lista de los 16 códigos binarios y sus correspondientes funciones de decodificación (en este caso la decodificación es por nivel bajo).

Dígito decimal	Entradas binarias				Función de decodificación	Salidas															
	A ₃	A ₂	A ₁	A ₀		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	$\overline{A_3}\overline{A_2}\overline{A_1}\overline{A_0}$	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	0	0	0	1	$\overline{A_3}\overline{A_2}\overline{A_1}A_0$	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	0	0	1	0	$\overline{A_3}\overline{A_2}A_1\overline{A_0}$	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
3	0	0	1	1	$\overline{A_3}\overline{A_2}A_1A_0$	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1
4	0	1	0	0	$\overline{A_3}A_2\overline{A_1}\overline{A_0}$	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1
5	0	1	0	1	$\overline{A_3}A_2\overline{A_1}A_0$	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1
6	0	1	1	0	$\overline{A_3}A_2A_1\overline{A_0}$	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1
7	0	1	1	1	$\overline{A_3}A_2A_1A_0$	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1
8	1	0	0	0	$A_3\overline{A_2}\overline{A_1}\overline{A_0}$	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1
9	1	0	0	1	$A_3\overline{A_2}\overline{A_1}A_0$	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1
10	1	0	1	0	$A_3\overline{A_2}A_1\overline{A_0}$	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
11	1	0	1	1	$A_3\overline{A_2}A_1A_0$	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1
12	1	1	0	0	$A_3A_2\overline{A_1}\overline{A_0}$	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1
13	1	1	0	1	$A_3A_2\overline{A_1}A_0$	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
14	1	1	1	0	$A_3A_2A_1\overline{A_0}$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1
15	1	1	1	1	$A_3A_2A_1A_0$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

Si se necesita una salida activa a nivel BAJO para cada número decodificado, el decodificador completo se puede implementar mediante puertas NAND e inversores. Para decodificar cada uno de los 16 códigos binarios se requieren 16 puertas NAND (las puertas AND se pueden usar para producir salidas activas a nivel ALTO).

El decodificador BCD a 7 segmentos

Este tipo de decodificador acepta código BCD en sus entradas y proporciona salidas capaces de excitar un display de 7 segmentos para indicar un dígito decimal. En la Figura 10 se muestra el diagrama lógico de un decodificador básico de 7 segmentos conectado a un display.

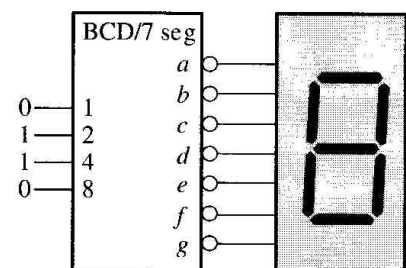


Fig.10 decodificador del código BCD a 7 segmentos

CODIFICADORES

Un codificador es un circuito lógico combinacional que, esencialmente, realiza la función “inversa” del decodificador. Un codificador permite que se introduzca en una de sus entradas un nivel activo que representa un dígito, como puede ser un dígito decimal u octal, y lo convierte en una salida codificada, como BCD o binario. Los codificadores se pueden diseñar también para codificar símbolos diversos y caracteres alfabéticos. El proceso de conversión de símbolos comunes o números a un formato codificado recibe el nombre de codificación.

Codificador decimal-BCD

Este tipo de *decodificador* posee diez entradas, una para cada dígito decimal, y cuatro salidas que corresponden al código BCD, como se muestra en la Figura 11. Este es un codificador básico de 10 líneas a 4 líneas.

El código BCD (8421) se muestra en la Tabla. A partir de esta tabla podemos determinar la relación entre cada bit BCD y los dígitos decimales, con el fin de analizar la lógica. Por ejemplo, el bit más significativo del código BCD, A_3 , es siempre un 1 para los dígitos decimales 8 o 9. La Expresión OR para el bit A_3 en función de los dígitos decimales puede por tanto escribirse como:

$$A_3 = 8 + 9$$

El bit A_2 es siempre un 1 para los dígitos decimales 4, 5, 6 o 7 y puede expresarse como una función OR de la manera siguiente:

$$A_2 = 4 + 5 + 6 + 7$$

El bit A_1 es siempre un 1 para los dígitos decimales 2, 3, 6 o 7 y puede expresarse como:

$$A_1 = 2 + 3 + 6 + 7$$

Finalmente, A_0 es siempre un 1 para los dígitos 1, 3, 5, 7 o 9. La expresión para A_0 es:

$$A_0 = 1 + 3 + 5 + 7 + 9$$

Ahora vamos a implementar el circuito lógico necesario para codificar en código BCD cada dígito decimal, utilizando las expresiones lógicas que se acaban de desarrollar. Consiste simplemente en aplicar la operación OR a los dígitos decimales de entrada apropiados, para así formar cada salida BCD. La lógica del codificador que resulta de estas expresiones se muestra en la Figura 12.

El funcionamiento básico del circuito de la Figura 12 es el siguiente: cuando aparece un nivel ALTO en una de las líneas de entrada correspondientes a los dígitos decimales, se generan los niveles apropiados en las cuatro líneas BCD de salida. Por ejemplo, si la línea de entrada 9 está a nivel ALTO (suponiendo que todas las demás entradas están a nivel BAJO), esta condición producirá un nivel ALTO en las salidas A_0 y A_3 , y un nivel BAJO en A_1 y A_2 que es el código BCD (1001) del número decimal 9.

Codificador MSI decimal-BCD

El 74HC147 es un *codificador con prioridad* con entradas activas a nivel BAJO (0) para los dígitos decimales del 1 al 9, y salidas BCD activas a nivel BAJO, como se indica en el símbolo lógico de la Figura 13. Una salida BCD cero se consigue cuando ninguna de las entradas está activa

Ejemplo de aplicación

El típico ejemplo de aplicación es un codificador de teclado. Por ejemplo, los diez dígitos decimales del teclado de una computadora tienen que codificarse para poder ser procesados por el circuito lógico. Cuando se pulsa una de las teclas, el dígito decimal se codifica a su correspondiente código BCD. La Figura 14 muestra la disposición de un sencillo codificador de teclado que utiliza un codificador con prioridad 74HC147. Las teclas se representan mediante diez pulsadores, conectados cada uno de ellos a una **resistencia de pull-up** (resistencia de conexión a

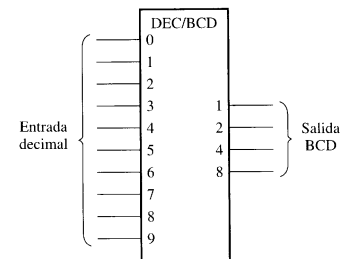


Fig.11 codificador decimal a BCD

Dígito decimal	Código BCD			
	A_3	A_2	A_1	A_0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

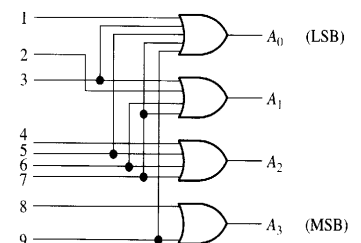


Fig.12 Lógica del codificador decimal a BCD

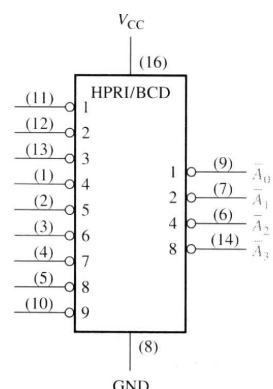
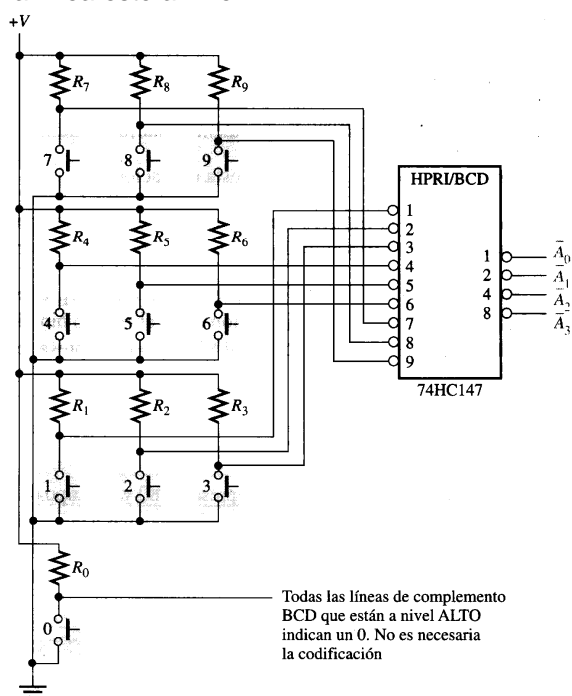


Fig.13 Diagrama lógico del 74HC147

la alimentación $+V_{CC}$). Las resistencias pull-up aseguran que la línea esté a nivel ALTO cuando no haya ninguna tecla pulsada. Cuando se pulsa una tecla, la línea se conecta a tierra y se aplica un nivel BAJO a la correspondiente entrada del codificador. La tecla cero no está conectada, ya que la salida BCD es cero cuando ninguna de las otras teclas está pulsada.

La salida complementada BCD del codificador se conecta a un dispositivo de almacenamiento, de forma que los sucesivos códigos BCD se almacenan hasta que se haya introducido el número completo. Próximamente veremos los métodos de almacenamiento de números BCD y de datos binarios.

Fig.14 Esquema simplificado de un teclado con salida codificada



MULTIPLEXORES (SELECTORES DE DATOS)

Un multiplexor (MUX) es un dispositivo que permite dirigir la información digital procedente de diversas fuentes a una única línea para ser transmitida a través de dicha línea a un destino común. El multiplexor básico posee varias líneas de entrada que permiten conmutar los datos digitales provenientes de cualquier entrada hacia la línea de salida. A los multiplexores también se les conoce como selectores de datos.

El símbolo lógico de un *multiplexor* (**MUX**) de cuatro entradas se muestra en la figura 15. Obsérvese que dispone de dos líneas de selección de datos, dado que con dos bits se puede seleccionar cualquiera de las cuatro líneas de entrada de datos.

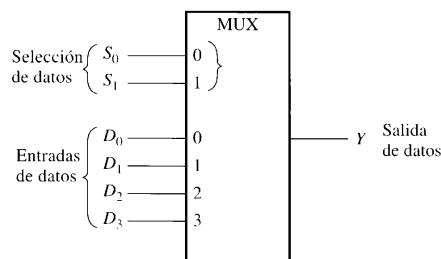


Fig.15 Símbolo lógico de un selector o multiplexor.

En la Figura 15, un código binario de dos bits en las entradas de selección de datos (S) va a permitir que los datos de la entrada seleccionada pasen a la salida de datos. Si aplicamos un 0 binario ($S_1 = 0$ y $S_0 = 0$) a las líneas de selección de datos, los datos de la entrada D_0 aparecerán en la línea de datos de salida. Si aplicamos un 1 binario ($S_1 = 0$ y $S_0 = 1$), obtendremos en la salida los datos de D_1 . Si aplicamos un 3 binario ($S_1 = 1$ y $S_0 = 1$), los datos D_3 serán conmutados a la línea de salida. El resumen del funcionamiento se puede ver en la Tabla.

Entradas de selección de datos		Entrada seleccionada
S_1	S_0	
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

Ejemplos de aplicación

Display multiplexor de 7 segmentos. La Figura 16 muestra un método simplificado de multiplexación de números BCD para un display de 7 segmentos. En este ejemplo, se visualizan en el display de 7 segmentos números de 2 dígitos, mediante el uso de un único decodificador BCD a 7 segmentos. Este método básico de multiplexación puede ampliarse para visualizar números con cualquier cantidad de dígitos. Su funcionamiento básico es el siguiente:

Se aplican dos dígitos BCD ($A_3 A_2 A_1 A_0$ y $B_3 B_2 B_1 B_0$) a las entradas de un multiplexor. Se aplica una señal cuadrada a la línea de selección de datos de forma que cuando está a nivel BAJO, los bits de A ($A_3 A_2 A_1 A_0$) pasan a las entradas del decodificador BCD a 7 segmentos 74LS48. El nivel BAJO en la entrada de selección de datos genera un nivel BAJO en la entrada 1 del decodificador de 2 líneas a 4 líneas 74LS139, activando su salida 0 y habilitando el display del dígito A, al conectar su terminal común a masa. El dígito A se encuentra ahora encendido, mientras que el B está apagado.

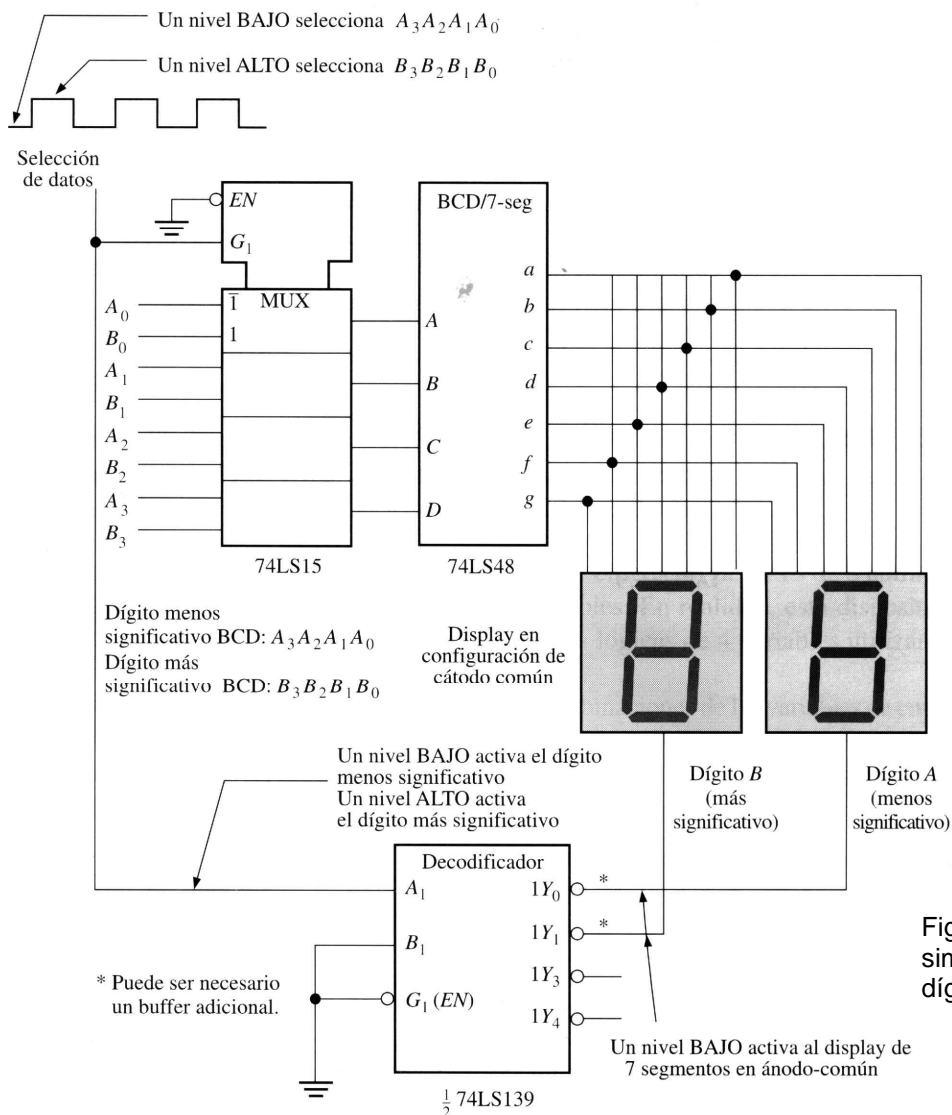


Fig. 16 Multiplexado simplificado de dos dígitos

Cuando la línea de selección de datos pasa a nivel ALTO, los bits de B ($B_3 B_2 B_1 B_0$) pasan a las entradas del decodificador BCD a 7 segmentos. Ahora se activa la salida 1 del decodificador 74LS139, encendiendo el display del dígito B, que pasa a visualizarse, mientras que el A se encuentra apagado. El ciclo se repite a la frecuencia de la señal cuadrada que se aplica a la entrada de selección de datos. Esta frecuencia tiene que ser lo suficientemente alta (unos 30 Hz) para evitar el parpadeo en los displays cuando se multiplexa la presentación de los dígitos.

DEMULPLEXORES

Un demultiplexor (DEMUX) básicamente realiza la función contraria a la del multiplexor. Toma datos de una línea y los distribuye a un determinado número de líneas de salida. Por este motivo, el demultiplexor se conoce también como distribuidor de datos. Como veremos, los decodificadores pueden utilizarse también como demultiplexores.

La Figura 17 muestra un circuito *demultiplexor* (DEMUX) de una línea a 4 líneas. La línea de entrada de datos está conectada a todas las puertas AND. Las dos líneas de selección de datos activan únicamente una puerta cada vez y los datos que aparecen en la línea de entrada de datos pasarán a través de la puerta seleccionada hasta la línea de salida de datos asociada.

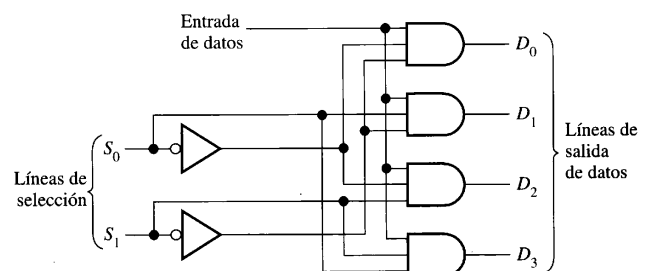


Fig. 17 Demultiplexor de 1 línea a 4 líneas

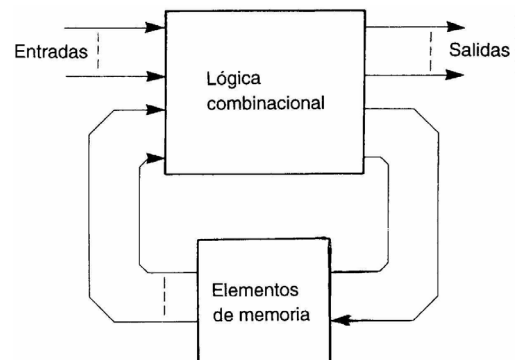
LÓGICA SECUENCIAL

INTRODUCCIÓN:

Hemos visto que en la lógica combinacional las salidas están determinadas sólo por los estados existentes de las entradas. En la **lógica secuencial**, sin embargo, las salidas están determinadas no sólo por las entradas existentes sino también por la secuencia de entradas anteriores que condujeron al estado existente. En otras palabras, el circuito tiene la característica de **memoria**.

La Figura muestra un sistema secuencial generalizado compuesto por lógica combinacional con algún tipo de memoria dentro de un camino de realimentación. Los elementos de memoria son dispositivos que pueden almacenar información binaria; la salida en cualquier momento está determinada por las entradas presentes y por la información almacenada en los elementos de memoria. La información almacenada dentro de la memoria determina el **estado** del circuito en un tiempo cualquiera, mientras que el siguiente estado del sistema se encuentra determinado por las entradas existentes y la secuencia de entradas que les precedieron.

Los circuitos secuenciales se pueden dividir en **síncronos** y **asíncronos**. En los **sistemas síncronos**, las entradas, las salidas y los estados internos se muestrean en instantes de tiempo definidos, y son controlados por una señal de **reloj**. En estos sistemas es común que muchos circuitos sean controlados por una sola señal de reloj de modo tal que su funcionamiento esté sincronizado. En los **sistemas asíncronos** los circuitos responden a cambios en las entradas en cualquier momento. Los efectos de los cambios de las entradas se propagan entonces a través del sistema y cada circuito agrega su propio retardo. Entonces, en los sistemas secuenciales asíncronos se pueden actualizar los estados internos y las variables de salida en cualquier momento.



Aunque muchos circuitos electrónicos son de naturaleza secuencial, los bloques secuenciales más comunes son los diversos tipos de **multivibrador**. Esta clasificación cubre circuitos de muchas formas diferentes que se caracterizan por tener dos salidas que son la inversa una de la otra, y ninguna, una o más entradas. Las salidas reciben de ordinario etiquetas Q y \bar{Q} . Tener sólo estas dos salidas significa que los circuitos tendrán sólo dos estados de salida posibles, específicamente $Q = 1, \bar{Q} = 0$ y $Q = 0, \bar{Q} = 1$. Los diferentes tipos de multivibrador se definen mediante el comportamiento de los circuitos en estos dos estados. Hay tres tipos básicos posibles:

- **Multivibrador biestables**, en los que ambos estados de salida son estables. Cuando se encuentra en un estado, el circuito permanecerá en él hasta que una señal de entrada haga que cambie de estado. Hay varios tipos de multivibrador biestable, pero por desgracia no hay un acuerdo general sobre los nombres que deben usarse para estas clases de dispositivo. Algunos ingenieros se refieren a todos los dispositivos biestables como flip-flops, mientras que otros usan el término latch para los dispositivos sensibles por nivel y flip-flops para los dispositivos disparados por flanco y disparados por pulsos (El significado de estos términos se explicará un poco más adelante). Aquí adoptaremos la segunda terminología, pues da información adicional acerca de la forma del dispositivo.

- **Multivibradores monoestables**, en los que un estado es estable y el otro es metaestable (o casi estable). El circuito permanecerá en su estado estable hasta que resulte afectado por una señal de entrada apropiada, momento en el cambiará a su estado metaestable; permanecerá en su estado metaestable durante un período fijo de tiempo (determinado por los parámetros del circuito) y luego volverá en forma automática a su estado estable. El circuito se comporta como un generador de un solo impulso. Cuando se le dispara, entra en su estado metaestable, ocasionando que las salidas cambien durante un período fijo de tiempo. Este circuito se conoce como de **un solo disparo**.

- **Multivibradores astables**, en los que ambos estados son metaestables. El circuito permanece en cada estado durante un período fijo de tiempo (determinado por los parámetros del circuito) antes de conmutar a su otro estado. Esto produce un circuito que oscila en forma continua de un estado a otro, es decir, un oscilador digital.

LATCHES

El latch (cerrojo) es un tipo de dispositivo de almacenamiento temporal de dos estados (biestable), que se suele agrupar en una categoría diferente a la de los flip-flops.

Básicamente, los latches son similares a los flip-flops, ya que son también dispositivos de dos estados que pueden permanecer en cualquiera de sus dos estados gracias a su capacidad de realimentación, lo que consiste en conectar (realimentar) cada una de las salidas a la entrada opuesta. La diferencia principal entre ambos tipos de dispositivos está en los niveles de entrada empleado para cambiar de estado.

El latch S-R (SET-RESET)

Un **latch** es un tipo de dispositivo lógico **biestable** o **multivibrador**. Un latch S-R (Set-Reset) con entrada activa a nivel ALTO se compone de dos puertas NOR acopladas tal como se muestra en la Figura 1 (a); un latch \bar{S} - \bar{R} con entrada activa a nivel BAJO está formado por dos puertas NAND conectadas tal como se muestra en la Figura 1 (b). Obsérvese que la salida de cada puerta se conecta a la entrada de la puerta opuesta. Esto origina la **realimentación** (feedback) regenerativa (o positiva) característica de todos los latches y flip-flops.

Los símbolos lógicos para ambos tipos de latches, con entradas activas a nivel ALTO y a nivel BAJO, se muestran en la Figura 2

Para explicar el funcionamiento del latch, usaremos el latch de puerta NOR de la figura 1. Tenemos un circuito con dos entradas R y S y dos salidas llamadas Q y \bar{Q} . Si una de las entradas de una compuerta NOR está en cero la puerta se comporta como un inversor. Si las dos entradas R y S están a 0 las puertas NOR se comportan como simples inversores donde están dos puertas NOT encadenadas, fig 1 (c).

Si la salida del inversor A es igual a 1 como está conectada a la entrada del B la salida de este será $\bar{Q} = 0$. A su vez la salida de la puerta B está conectada a la entrada de la A. Que hace que su salida sea 1 condición que habíamos supuesto. En otras palabras el circuito es estable con $Q = 1$ y $\bar{Q} = 0$.

Alternativamente si Q es igual a 0 conducirá al estado estable en que $Q = 0$ y $\bar{Q} = 1$. Por lo tanto el circuito tiene dos estados estables en que una de la salida de la puerta es el complemento de la otra, se trata de un multivibrador biestable. En este circuito no se puede predecir cual de los dos estados adquirirá cuando se lo alimenta pero si que permanecerá con el inicial.

En el latch con ambas entradas R y S en 0 el circuito permanecerá en un estado y no cambiará. Podemos llamar a esta condición modo de *memoria* del circuito.

Si ahora ponemos R a uno mientras mantenemos $S = 0$, Q se pondrá a 0 (independientemente del estado previo) condición de RESET del circuito y consecuentemente será $\bar{Q} = 1$. Si R retorna a cero, el circuito entra en el modo de memoria y permanecerá en ese estado.

En forma similar si ahora ponemos S a uno mientras mantenemos $R = 0$, Q se pondrá a 1 (independientemente del estado previo) condición de SET del circuito y consecuentemente será $\bar{Q} = 0$. Nuevamente si S retorna a cero, el circuito entra en el modo de memoria y permanecerá en ese estado.

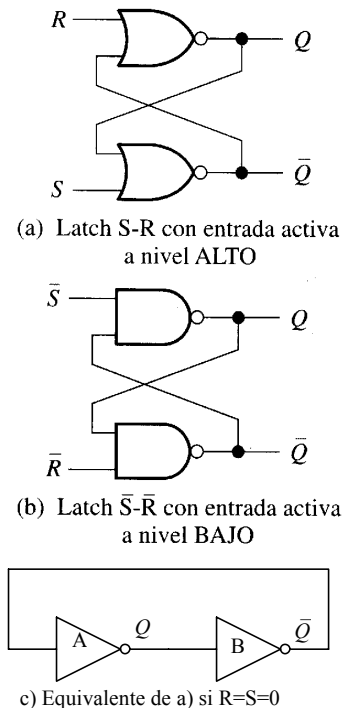


Fig. 1 Latch S-R

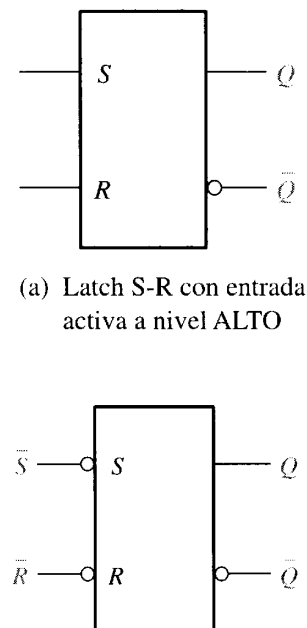


Fig. 2 Símbolos lógicos

S	R	Q_n	\bar{Q}_n	
0	0	Q_{n-1}	\bar{Q}_{n-1}	Sin cambio
0	1	0	1	RESET
1	0	1	0	SET
1	1	0	0	Estado ambiguo

\bar{S}	\bar{R}	Q_n	\bar{Q}_n	
0	0	0	0	Estado ambiguo
0	1	1	0	SET
1	0	0	1	RESET
1	1	Q_{n-1}	\bar{Q}_{n-1}	Sin cambio

Fig. 3 Tablas de verdad correspondientes a los latch de las fig. 1 a) y 1 b)

Debemos notar que la condición $R = S = 1$ ambas salidas estarán en cero. Bajo estas circunstancias el circuito no puede permanecer mucho tiempo ya que sus

salidas son complementarias y el circuito no funcionaría como biestable. Debido a ello esta condición de entrada está generalmente prohibida.

Podemos representar el comportamiento del Latch S-R haciendo uso de la tabla de verdad, la cual suele denominarse también tabla de transiciones por indicar las transiciones entre distintos estados.

En la figura 4 se ven las formas de onda de entrada y de salida correspondientes al latch de la fig. 1 b) si se aplican a sus entradas las formas de onda \bar{S} y \bar{R} .

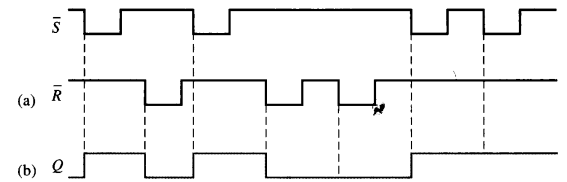


Fig. 4 forma de ondas del latch \bar{S} - \bar{R}

Ejemplo de aplicación

El latch como eliminador del rebote de los contactos. Un buen ejemplo de aplicación de un latch \bar{S} - \bar{R} consiste en la eliminación del “rebote” producido por los contactos de un interruptor mecánico. Cuando el polo de un interruptor choca con el contacto de cierre del interruptor, vibra o rebota varias veces hasta que, finalmente, se consigue contacto firme. Aunque estos rebotes son mínimos, producen unos picos de tensión que pueden ser inadmisibles en un sistema digital. Esta situación se observa en la Figura 5 (a).

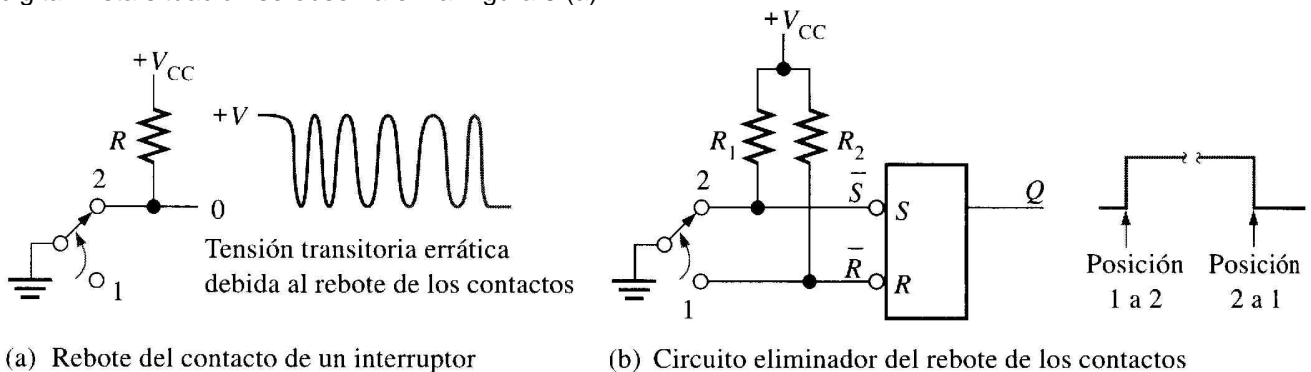


Fig. 5 Eliminador de rebote de los contactos

Se puede utilizar un latch \bar{S} - \bar{R} para eliminar los efectos de los rebotes del interruptor, como se muestra en la Figura 5 (b). El interruptor se encuentra normalmente en la posición 1, manteniendo la entrada \bar{R} a nivel BAJO y al latch en estado RESET. Cuando el interruptor pasa a la posición 2, \bar{R} pasa a nivel ALTO debido a la resistencia del pull-up conectada a V_{CC} y \bar{S} pasa a nivel BAJO cuando se produce el primer contacto. Aunque \bar{S} permanece a nivel BAJO durante un breve espacio de tiempo antes de que el interruptor rebote, este tiempo es suficiente para activar (SET) el rebote del interruptor, no va a afectar el latch, y éste permanecerá en el estado SET. Téngase en cuenta que la salida Q del latch proporciona una transición limpia del nivel BAJO al nivel ALTO, por lo que se eliminan los picos de tensión causados por el rebote de los contactos. De forma similar se produce una transición limpia de nivel ALTO a nivel BAJO cuando el interruptor vuelve a la posición 1.

EL LATCH S-R CON ENTRADA DE HABILITACIÓN

El diagrama y el símbolo lógico de un latch con entrada de habilitación se muestran en la Figura 6. Las entradas S y R controlan el estado al que va a cambiar el latch cuando se aplica un nivel ALTO a la entrada de habilitación (EN, enable). El latch no cambia de estado hasta que la entrada EN está a nivel ALTO el estado de las entradas S y R. En este circuito, el estado no válido del latch se produce cuando las dos entradas S y R están simultáneamente a nivel ALTO.

Si se aplican las señales de entrada al latch S-R que posee entrada de habilitación a la que se le aplica la onda **EN** mostrada en la Figura 7 (a) la correspondiente forma de onda de salida Q se muestra en la Figura 7 (b).

Siempre que S está a nivel ALTO y R a nivel BAJO, un nivel ALTO en la entrada **EN** hace que el latch se ponga en estado SET. Siempre que S está a nivel BAJO y R a nivel ALTO, un nivel ALTO en la entrada EN hace que el latch se ponga en estado RESET.

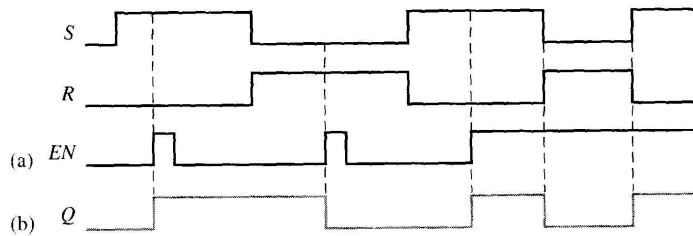
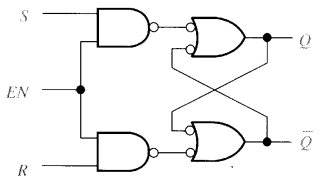
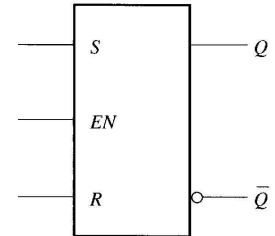


Fig. 7 Formas de onda del latch con habilitación



(a) Diagrama lógico



(b) Símbolo lógico

Fig. 6 Latch S-R con habilitación

EL LATCH D CON ENTRADA DE HABILITACIÓN

Existe otro tipo de latch con entrada de habilitación que se denomina latch D. Se diferencia del latch S-R en que sólo tiene una entrada, además de la de habilitación, **EN**. Esta entrada recibe el nombre de entrada de datos (**D**). La Figura 8 muestra el diagrama y símbolo lógico de este tipo de latch. Cuando la entrada D está a nivel ALTO y la entrada EN también, el latch se pone en estado SET. Cuando la entrada D está a nivel BAJO y la entrada EN está a nivel ALTO, el latch se pone en estado RESET. Dicho de otra manera, la salida Q es igual a la entrada D cuando la entrada de habilitación EN está a nivel ALTO.

Si se aplican las entradas que se muestran en la Figura 9 a un latch D con entrada de habilitación que, inicialmente, está en estado RESET se obtiene la forma de onda de salida Q, mostrada en (b) la Figura9 (b)

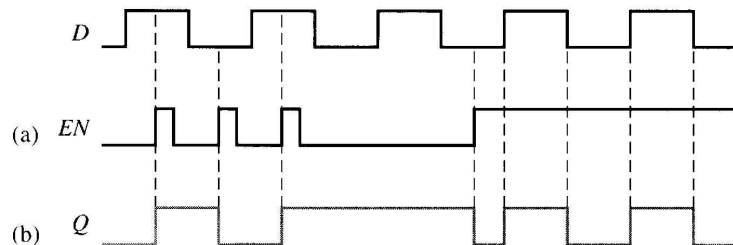
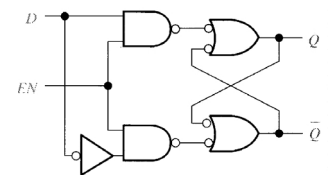
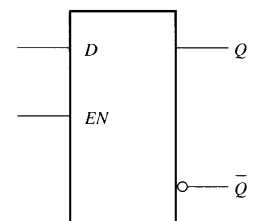


Fig. 9 Formas de onda



(a) Diagrama lógico



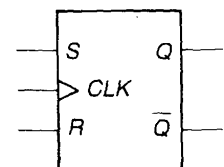
(b) Símbolo lógico

Fig. 8 Latch D

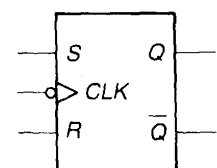
Siempre que D y EN estén a nivel ALTO, la salida Q será un nivel ALTO. Siempre que D sea un nivel BAJO y EN esté a nivel ALTO, Q se pondrá a nivel BAJO. Cuando EN está a nivel BAJO, el estado del latch no se ve afectado por la entrada D.

BIESTABLES DISPARADOS POR FLANCO O FLIP-FLOP

En muchas situaciones es necesario sincronizar el funcionamiento de muchos circuitos diferentes, y resulta de utilidad controlar en forma precisa el momento en que un circuito cambiará de estado. Algunos dispositivos biestables están contruidos de manera que sólo cambian de estado ante la aplicación de una señal de **disparo**. Esta señal de disparo se define con el flanco que sube o baja de una señal de entrada llamada **reloj**. Estos dispositivos reciben el nombre de **biestables disparados por flanco** o, más comunmente, **flip-flop**; se dividen en aquellos que son disparados por el flanco de subida de la señal de reloj (y se les llama dispositivos disparados por flanco positivo) y los que son disparados por el flanco de bajada del reloj (dispositivos disparados por flanco negativo).



(a) Disparado por flanco positivo



(b) Disparado por flanco negativo

Fig. 10 Símbolos lógicos del flip flop S-R disparado por flanco

Existen flip-flops de muchas formas diferentes, de los cuales tomaremos en cuenta una selección. Los símbolos lógicos que se usan para los flip-flops se asemejan a los de los latches con compuerta, excepto en que la entrada de habilitación se reemplaza con una entrada de reloj. La línea de reloj se muestra en forma convencional por medio de un triángulo; se usa un círculo de inversión para mostrar un dispositivo disparado por flanco negativo. La figura 10 incluye ejemplos de estos símbolos para flip-flops S-R.

Flip-flop S-R disparado por flanco

El funcionamiento del flip-flop S-R se asemeja al del latch S-R excepto en que el circuito sólo responde a sus entradas en el flanco de subida o de bajada (dependiendo del dispositivo) de la señal de entrada de reloj. Se usan muchas formas de dispositivos disparados por flanco. La figura 11(a) muestra un ejemplo de un circuito disparado por flanco positivo. El funcionamiento de este circuito es

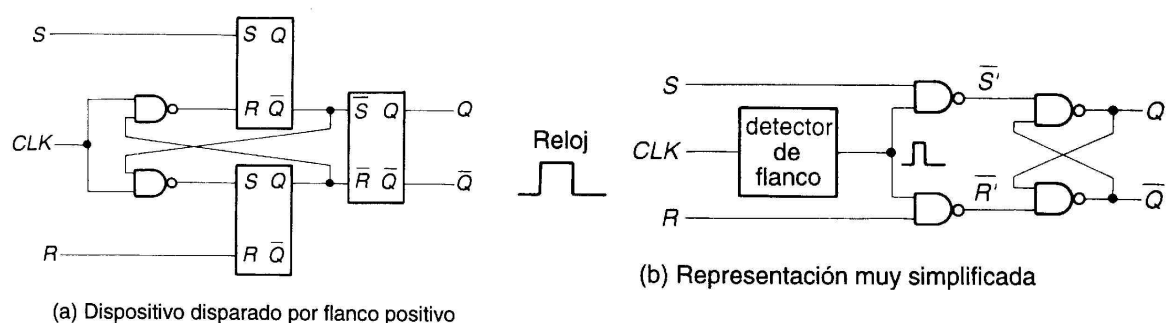


Fig. 11 Flip Flop S-R disparado por flancos

complejo y depende en parte de los retardos dentro del circuito. Quizá resulte de mayor utilidad considerar un modelo conceptual de este circuito, como el que aparece en la figura 11 (b). Este circuito no es una representación verdadera del funcionamiento del circuito anterior, pero proporciona un panorama de sus características.

El circuito de la figura 11 (b) es similar al del latch S-R con compuerta pero tiene circuitos adicionales para la detección del flanco. Una transición apropiada de la entrada de reloj genera un pulso corto que habilita brevemente la red de compuerta, permitiendo que el circuito responda a las entradas S y R. En ausencia de las transiciones de reloj, el circuito permanece en su modo de memoria y las salidas no cambian. Cuando se detecta un flanco apropiado, el circuito responde a las señales presentes en las entradas S y R.

El funcionamiento de un flip-flop S-R disparado por flanco positivo se puede describir mediante la tabla de transición mostrada a la derecha.

S	R	CLK	Q_n	\bar{Q}_n	
0	0	↑	Q_{n-1}	\bar{Q}_{n-1}	Sin cambio
0	1	↑	0	1	RESET
1	0	↑	1	0	SET
1	1	↑	0	0	Estado ambiguo

↑ = transición de reloj con flanco positivo

Como el circuito sólo responde cuando ocurre una transición de reloj apropiada, sólo se muestra su funcionamiento en esos momentos. El reloj no actúa como una entrada ordinaria, y por lo tanto la tabla sólo tiene cuatro filas en lugar de ocho.

La tabla de transición muestra que mientras S y R son 0, el circuito permanece en su estado presente sin importar la señal que haya en la entrada de reloj. Si R está alta y S baja cuando ocurre una transición de reloj con flanco positivo, el circuito se PONE A CERO (RESET). Si S está alta mientras R está baja cuando ocurre una transición de reloj con flanco positivo, el circuito se PONE A UNO (SET). Al igual que con otras formas biestables S-R, si tanto S como R están activas, las salidas son ambiguas.

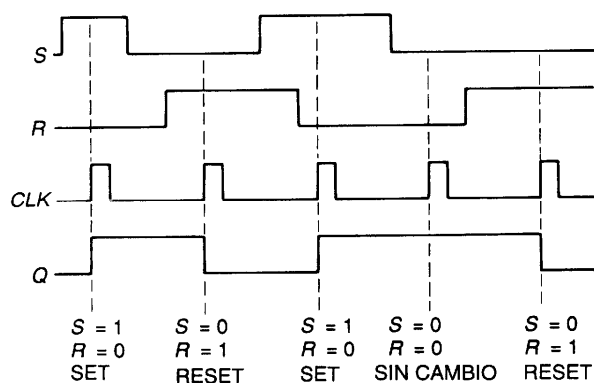


Fig.12 Forma de onda de un flip flop S-R disparado por flanco positivo

La tabla de transición para un dispositivo disparado por flanco negativo es idéntica a la que acabamos de dar, excepto que en este caso es el flanco negativo el que actúa como disparador. Esto aparecería en la tabla como una flecha hacia abajo (\downarrow).

La figura 12 muestra el funcionamiento de un flip-flop S-R disparado por flanco positivo. Notará el lector que en un momento durante las formas de onda que aparecen en esta figura, tanto S como R están activas (altas). Esto no ocasiona problemas pues el circuito sólo responde a las estradas en el flanco de subida del reloj y en estos momentos sólo una de las entradas está activa en todos los casos.

FLIP-FLOP D DISPARADO POR FLANCO

El flip-flop D es tan sólo una versión disparada por flanco del latch D antes descrito. El símbolo lógico y circuito equivalente de esta compuerta aparecen en la figura 13.

Una vez más es posible describir el funcionamiento del flip-flop D mediante el uso de una tabla de transición. Como la compuerta tiene una sola entrada (además del reloj), la tabla sólo tiene dos filas.

D	CLK	Q_n	\bar{Q}_n	
0	\uparrow	0	1	RESET
1	\uparrow	1	0	SET

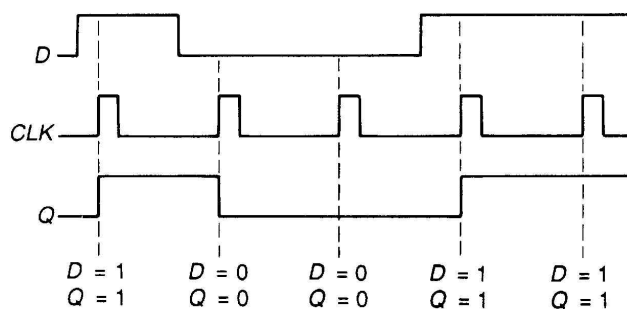
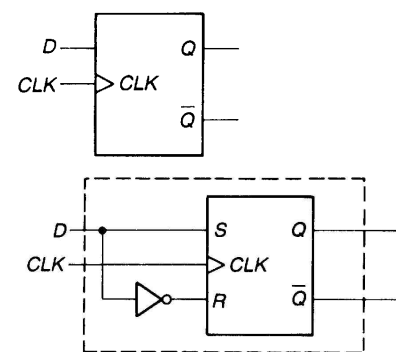


Fig.14 Forma de onda de un flip flop D disparado por flanco positivo



(b) Circuito equivalente.

Fig. 13 Flip flop D

FLIP-FLOP J-K DISPARADO POR FLANCO

El flip-flop S-R tiene muchos usos pero padece de una combinación de entrada ambigua cuando tanto S como R se activan de manera simultánea. El flip-flop J-K resuelve esta deficiencia mediante la definición de la operación que ha de efectuarse, sobre transiciones de reloj activas, cuando ambas entradas están activas.

Con el dispositivo S-R ya tenemos combinaciones de entrada que corresponden a SET, RESET, y Sin cambio. La operación adicional obvia para incrementar la flexibilidad es BÁSCULA (TOGGLE). La tabla de transición que aparece a continuación muestra las características de un flip-flop J-K disparado por flanco positivo. Se puede ver que la entrada J corresponde a la entrada S de un flip-flop S-R y que K corresponde a la entrada R.

J	K	CLK	Q_n	\bar{Q}_n	
0	0	\uparrow	Q_{n-1}	\bar{Q}_{n-1}	Sin cambio
0	1	\uparrow	0	1	RESET
1	0	\uparrow	1	0	SET
1	1	\uparrow	\bar{Q}_{n-1}	Q_{n-1}	BÁSCULA

La función del flip-flop J-K se logra utilizando circuitos similares a los del flip-flop S-R excepto en que las señales de salida Q y \bar{Q} se realimentan para modificar el funcionamiento de la red de compuerta. Este circuito aparece en la figura 15.

La compuerta se basa en un biestable S-R disparado por flanco positivo con entradas S y R. Las entradas para este circuito provienen de dos compuertas AND, A1 y A2. Para que cualquiera de estas compuertas AND produzca la salida de 1 lógico requerida para activar la entrada biestable correspondiente, ambas

entradas deben estar en 1. A partir del circuito es evidente que A2 sólo puede producir una salida alta cuando Q está alta, y que A1 sólo puede producir una salida alta cuando \bar{Q} está alta. Consideremos ahora el funcionamiento del circuito ante diferentes combinaciones de condiciones de entrada y estados.

Si tanto J como K están en cero cuando ocurre un disparo del reloj, tanto S como R estarán bajas sin importar el estado de salida del circuito. Este es el modo de memoria del biestable y por lo tanto no ocurrirá cambio alguno.

Si $J = 1$ y $K = 0$ cuando ocurre un disparo del reloj, la acción que se emprenda dependerá del estado de salida existente. Si $Q = 0$ y $\bar{Q} = 1$, ambas entradas a A1 estarán altas y S estará alta, de modo que Q se colocará en 1. Si Q ya está en 1, \bar{Q} estará en 0, deshabilitando a A1 y manteniendo baja a S. Entonces, con esta combinación de entradas, si Q está en 0 se PONE en 1 y si ya está en 1 sólo permanece en 1.

Si $J = 0$ y $K = 1$ cuando ocurre un disparo del reloj, con base en la simetría del circuito está claro que si Q se encuentra en 1 se PONE a 0 y que si ya está en 0 simplemente se quedará en ese estado.

Si tanto J como K están en 1 cuando ocurre un disparo del reloj, la acción depende una vez más del estado de salida existente. Si Q está en 1, se habilitará A2 y se deshabilitará A1 de manera que R se active pero S no. Esto traerá como resultado que Q cambie de 1 a 0. Sin embargo, si Q ya está en 0, se habilitará A1 y se deshabilitará A2 de manera que S esté activada, y R no. Esto tendrá como resultado el cambio de Q de 0 a 1. Entonces, en cualquier caso Q BÁSCULA (TOGGLE) entre los dos estados de salida.

La figura 16 ilustra el funcionamiento de un flip-flop J-K disparado por flanco positivo en respuesta a una serie de combinaciones de entrada. También existen dispositivos disparados por flanco negativos.

La gran versatilidad del flip-flop J-K lo convierte quizá en el tipo de elemento biestable que más se usa. Existe un gran número de modos de funcionamiento diferentes, incluyendo su uso para reproducir las funciones de otros tipos de flip-flop. Esto se ilustra en la figura 17, que muestra varias configuraciones comunes.

La figura 17(a) indica que puede usarse un flip-flop J-K como reemplazo directo para un dispositivo S-R pues su funcionamiento es idéntico para todas las entradas permisibles de este último. También se puede usar un biestable J-K para producir la función de un flip-flop D, como muestra la figura 17 (b). En la figura 13 vimos este circuito, que usaba un flip-flop S-R. Como con esta configuración S y R no se pueden activar en forma simultánea, un flip-flop J-K producirá un efecto idéntico. La figura 17 (c) muestra un flip-flop J-K configurado con sus entradas J y K unidas para formar una sola entrada T.

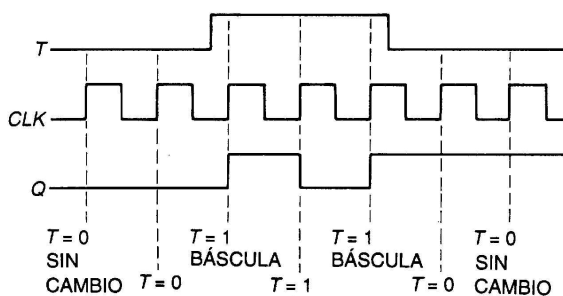
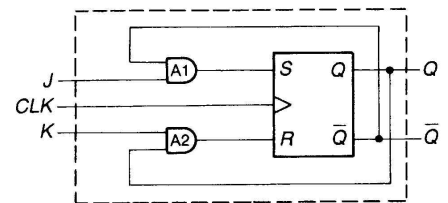
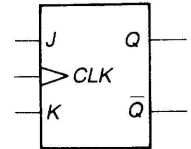


Fig. 18 Formas de onda del flip flop T

de biestable; en la siguiente sección veremos aplicaciones de este dispositivo.



(a) Circuito lógico simplificado



(b) Símbolo lógico

Fig. 15 Flip flop J-K disparado por flanco positivo

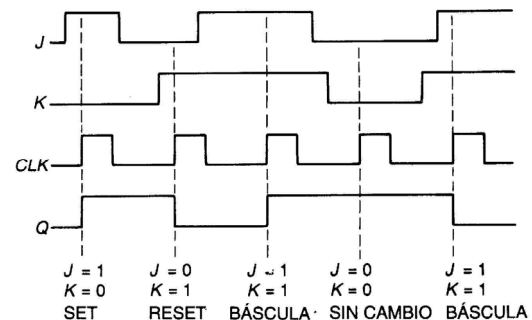
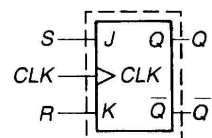
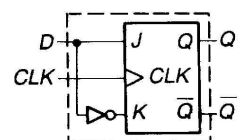


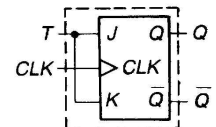
Fig. 16 Forma de onda del flip flop J-K disparado por flanco positivo



(a) Flip-flop S-R



(b) Flip-flop D



(c) Flip-flop T

Fig. 17 Flip flop J-K para implementar otros flip flops

ENTRADAS ASÍNCRONAS

Del análisis anterior sabemos que las entradas de control de los diversos flip-flops (es decir S, R, J, K, D y T) sólo tienen efecto en el momento de una transición apropiada de la señal de reloj (CLK). Por lo tanto, nos referimos a estas entradas de control como **síncronas**, pues su funcionamiento está sincronizado por la entrada de reloj.

En muchas aplicaciones resulta provechoso poder poner a "1" (PRESET) o "0" (CLEAR) la salida Q en cualquier momento, independientemente del reloj. Por lo tanto, algunos dispositivos tienen entradas adicionales para efectuar estas funciones. Éstas reciben el nombre de **entradas asíncronas** pues no están controladas por el estado del reloj. Por desgracia, los fabricantes de circuitos integrados no pueden acordar nombres comunes para estas entradas y se les puede llamar PRESET y CLEAR, DC SET y DC CLEAR, SET y RESET, o DIRECT SET y DIRECT CLEAR. Aquí usaremos los nombres PRESET (PRE) y CLEAR (CLR). Como ocurre con las entradas de control, estas líneas pueden ser activas a nivel alto o activas a nivel bajo, aunque es más común que sean activas a nivel bajo. La figura 19 (a) ilustra cómo se pueden incorporar estas entradas en el circuito de un flip-flop J-K, mientras que la figura 19 (b) muestra cómo se representan en un diagrama lógico.

Las entradas PRESET y CLEAR pueden anular las otras entradas al circuito, actuando de manera similar a las entradas S y R en un latch S-R. Es evidente que, como ocurre con el latch S-R, es necesario asegurarse que ambas entradas asíncronas no estén activas en forma simultánea, pues esto haría que ambas salidas fueran 1 y por lo tanto no serían complementarias. La figura 20 ilustra el funcionamiento de estas entradas en un flip-flop J-K disparado por flanco positivo.

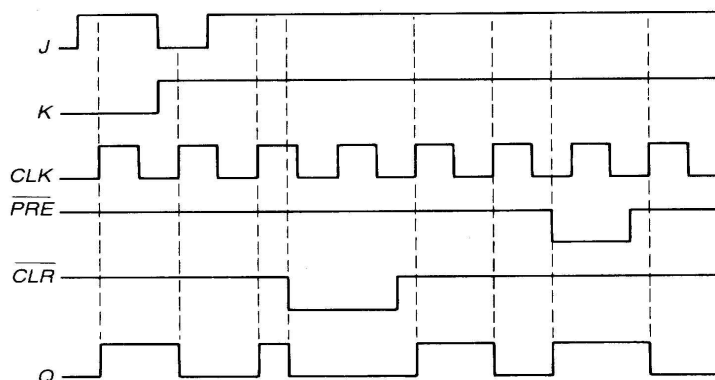
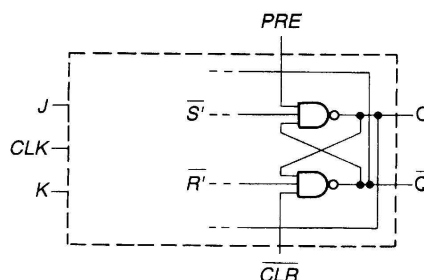
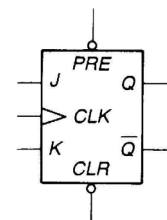


Fig 20 Formas de onda del flip flop J-K disparado por flancos con preset y Clear activos en bajo



(a) Circuito lógico simplificado



(b) Símbolo lógico

Fig 19 J-K con preset y Clear activos en bajo

FLIP-FLOP MAESTRO-ESCLAVO

Otra clase de flip-flop son los maestro-esclavo disparados por pulso, que han sido reemplazados progresivamente por los dispositivos disparados por flanco. Aunque los flip-flops maestro-esclavo están prácticamente obsoletos, todavía es posible encontrar este tipo de flip-flop en algunos equipos. Los datos se introducen en el flip-flop con el flanco anterior del impulso del reloj, pero la salida no refleja el estado de la entrada hasta que llega el flanco posterior. El flip-flop maestro-esclavo disparado por pulso no permite variar los datos mientras el impulso del reloj se encuentre activo.

En la Figura 20 se muestra un flip-flop **maestro-esclavo** J-K. La tabla de verdad es la misma que la de los flip-flops J-K disparados por flanco, excepto en la manera en que se sincroniza con la señal de reloj. Internamente es, sin embargo, bastante diferente.

Este tipo de flip-flop se compone de dos secciones: el maestro y el esclavo. El maestro es básicamente un latch con entrada de habilitación y

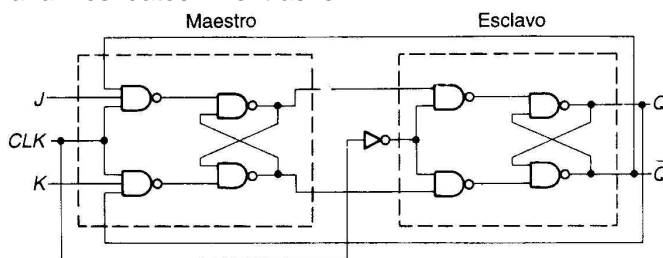







Fig 20 Diagrama del flip flop J-K maestro esclavo

el esclavo es lo mismo, excepto en que está sincronizado con el impulso invertido del reloj y se controla mediante las salidas del maestro en lugar de por las entradas externas J-K.

El maestro va a asumir el estado determinado por las entradas J y K durante el flanco anterior (positivo) del impulso de reloj. El estado del maestro se transfiere entonces al esclavo durante el flanco posterior (negativo) del impulso de reloj ya que las salidas del maestro se aplican a las entradas del esclavo y se invierte el impulso de reloj en el esclavo. En el flanco de bajada del impulso de reloj, el estado del esclavo aparece en las salidas Q y \bar{Q} . La salida Q se realimenta a la entrada de la puerta G_2 y la salida \bar{Q} se realimenta a la entrada de la puerta G_1 para producir el modo de basculación característico cuando $J = 1$ y $K = 1$. El funcionamiento lógico se resume en la Tabla de la figura 21. Una limitación del funcionamiento maestro-esclavo es que las entradas (J y K) no pueden cambiar mientras el impulso de reloj está activo, porque el estado del latch maestro puede cambiar durante dicho intervalo de tiempo.

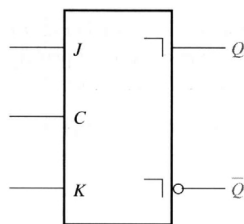
Los símbolos lógicos del flip-flop J-K maestro-esclavo se muestran en la Figura 22. La clave para identificar un flip-flop disparado por pulso (maestro-esclavo) mediante su símbolo lógico es el símbolo de salida pospuesta ANSI/IEEE ($\bar{\square}$) situado en las salidas. Este símbolo significa que la salida no refleja los datos de la entrada J-K, hasta que llega el flanco de reloj (bien positivo o negativo) siguiente al flanco disparador. Téngase en cuenta que no hay ningún indicador dinámico de entrada (\triangleright) en la entrada del reloj (c), como ocurre en un flip-flop disparado por flanco.

Entradas			Salidas		Observaciones
J	K	CLK	Q	\bar{Q}	
0	0		Q_0	\bar{Q}_0	No cambio
0	1		0	1	RESET
1	0		1	0	SET
1	1		\bar{Q}_0	Q_0	Basculación

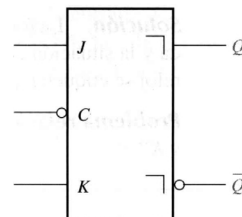
 = pulso de reloj

Q_0 = nivel de salida anterior al pulso de reloj

Fig. 21 Tabla de verdad del flip flop J-K maestro esclavo



- (a) Reloj activo a nivel ALTO: los datos se sincronizan con el flanco positivo del impulso de reloj y se transfieren a la salida en el siguiente flanco negativo



- (b) Reloj activo a nivel BAJO: los datos se sincronizan con el flanco negativo del impulso de reloj y se transfieren a la salida en el siguiente flanco positivo

Fig.22 Símbolo lógico del flip flop J-K maestro esclavo disparado por pulso

MONOESTABLES O CIRCUITOS DE UN SOLO DISPARO

En la figura 1 consideramos un biestable formado a partir de dos compuertas NOR conectadas en un anillo. Consideremos ahora el circuito de la figura 23.

Supongamos en principio que la señal de entrada T está en 0. La resistencia R , conectada a una tensión igual a 1 lógico, ocasionará que el capacitor C se cargue o descargue de modo que V_1 sea igual a esta tensión. Esto hará que Q sea 0 y como ambas entradas a la compuerta son 1 y 0, \bar{Q} será 1. Como \bar{Q} y V_1 se encuentran en una tensión que corresponde a 1 lógico, no habrá corriente a través del capacitor. Esta condición es **estable** y el circuito permanecerá en este estado por tiempo indefinido a menos que la entrada cambie.

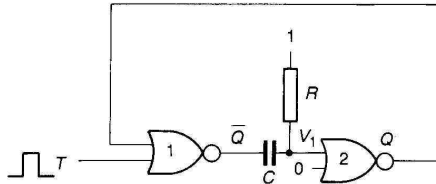
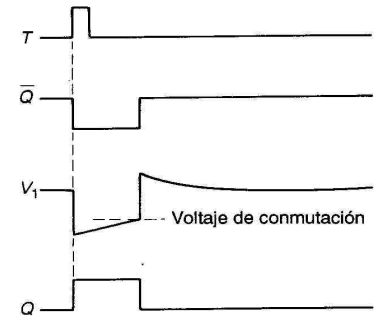


Fig. 23 Monoestable, circuito lógico y diagrama de tiempos



Consideremos ahora que sucede si la entrada T se hace alta durante poco tiempo. Cuando T se va a 1, \bar{Q} se conmuta al nivel bajo. Como la tensión en los extremos del capacitor no puede cambiar de manera instantánea, esto hará que V_1 vaya a bajo, lo que a su vez hará alta a Q . Mientras Q esté alta, \bar{Q} se mantendrá baja aún cuando T regrese a 0. Entonces, la aplicación de un pulso positivo a T conmuta el circuito a un estado en el que Q es 1 y éste permanece ahí aún después de que T vuelva a 0. Sin embargo, este estado no es estable, sino sólo **metaestable**. Mientras se encuentra en este estado, \bar{Q} se halla en 0 y por lo tanto existe una tensión a través de la combinación en serie de R y C . Esto produce una corriente a través de la resistencia que carga el capacitor, lo que trae como resultado que V_1 aumente en forma exponencial hacia la tensión que representa al 1 lógico. En algún punto se hace lo suficientemente grande como para que la entrada de la compuerta 2 lo interprete como 1 lógico. En este punto Q volverá a cero, \bar{Q} se irá a 1 y la tensión en los extremos del capacitor llevará a V_1 por encima del 1 lógico. Esta tensión se reducirá en forma gradual conforme se descarga el capacitor a través de R . El circuito se encuentra de nuevo en estado estable.

El circuito tiene entonces un estado estable y un estado metaestable. Se le puede forzar a que entre en su estado metaestable mediante la aplicación de una señal de entrada apropiada. Entonces permanecerá en ese estado durante un período fijo de tiempo determinado por los valores de R y C y de las tensiones de conmutación dentro del circuito. Por lo tanto, el circuito recibe el nombre de **monoestable** o **circuito de un solo disparo**. La etiqueta T que se da a la señal de entrada significa **entrada de disparo**.

Aunque es posible construir monoestables a partir de simples compuertas lógicas, es más común usar circuitos integrados dedicados. Éstos usan técnicas de circuito más complejas que los que describimos antes, pero tienen características similares. El símbolo lógico para un monoestable aparece en la figura 24.

Los monoestables de circuito integrado contienen todos los componentes activos que se requieren para formar el dispositivo dentro de un solo encapsulado. Por lo general, todo lo que se requiere es la adición de una sola resistencia y un capacitor para determinar la duración de los pulsos. Los monoestables se pueden dividir en dos tipos: monoestables *no redispables* y monoestables *redispables*.

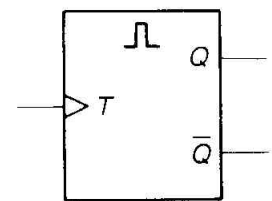


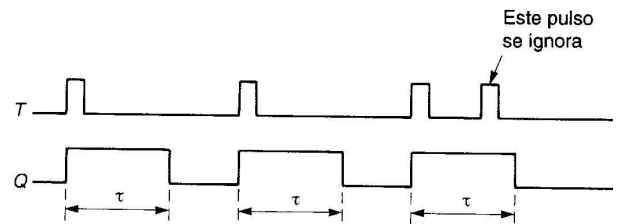
Fig. 24 Símbolo lógico de un monoestable

MONOESTABLES NO REDISPABLES

Los monoestables no-redispables ignoran cualquier pulso de disparo que ocurra mientras el circuito se encuentra en su estado metaestable. En otras

palabras, la entrada está deshabilita durante el pulso de salida. Esto se ilustra en la figura 25.

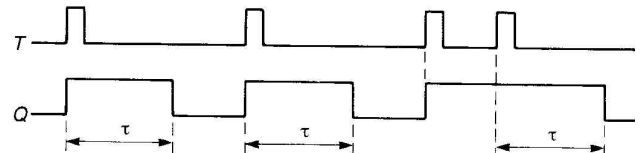
Fig. 25 Diagrama de tiempos de un monoestable no redispensible



MONOESTABLES REDISPENSABLES

Los monoestables redispensables resultan afectados por la entrada de disparo durante el pulso de salida. Esto tiene como resultado que el pulso de salida se extienda como lo muestra la figura 10.38.

Fig. 26 Diagrama de tiempos de un monoestable redispensible



ASTABLES

Al comparar los circuitos de las figuras 1 y 23 resulta evidente que la adición de una combinación de resistencia-capacitor transforma un estado estable en un estado metaestable. Quizá entonces no sea sorprendente que si agregamos una segunda combinación de resistencia-capacitor podamos generar un circuito con dos estados metaestables. Este circuito no requiere señal de entrada alguna y por lo tanto podemos reemplazar las compuertas NOR por inversores, como la figura 1 (c). El circuito resultante aparece en la figura 27

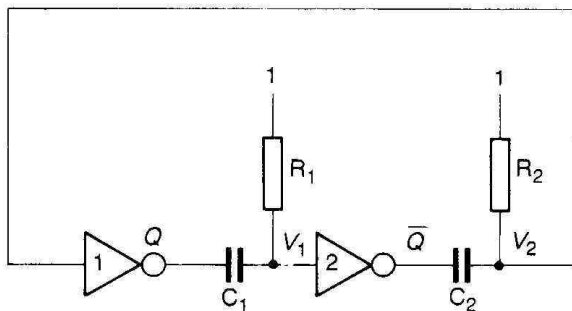
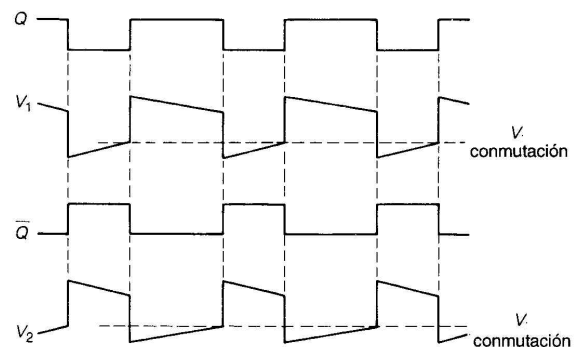


Fig. 27 Multivibrador astable, circuito y formas de onda



Supongamos que en principio C_1 y C_2 están descargados y que Q es 0. Como no hay tensión en los terminales de C_1 , V_1 será igual a Q , es decir, 0. Esto ocasionará que la salida de la segunda compuerta esté en 1. Como C_2 se encuentra descargado, este nivel lógico se aplicará a la entrada de la compuerta 1. Esto, a su vez, generará un 0 lógico en Q , lo que es consistente con nuestra suposición original.

Este estado es metaestable pues ahora C_1 se cargará hasta que V_1 sea mayor que la tensión de conmutación de la compuerta 2. En este momento \bar{Q} se irá a 0, haciendo que V_2 baje y que Q se vaya a 1. Ahora C_1 se descargará y C_2 se cargará hasta que V_2 sea mayor que la tensión de conmutación de la compuerta 1, con lo que el circuito cambiará de estado una vez más y el ciclo volverá a empezar. Así, el circuito oscilará en forma continua de un estado a otro con Q y \bar{Q} produciendo formas de onda de pulsos regulares. La cantidad de tiempo que permanezcan en cada estado está determinada por los valores de las resistencias y capacitores, así como por las tensiones de conmutación. Si las compuertas son idénticas y los productos C_1R_1 y C_2R_2 son iguales, el circuito producirá una **onda cuadrada**. La figura 27 ilustra las formas de onda producidas dentro del circuito.

Como sucede con los monoestables, los astables por lo general se usan en forma de circuito integrado. También existen circuitos que se pueden configurar para realizar una gama de funciones, como el **timer 555**. Este circuito integrado

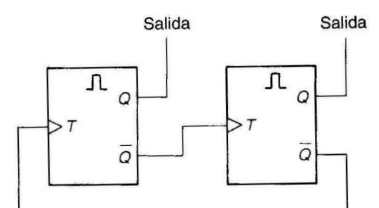


Fig. 28 Astable formado con dos monoestables

se puede usar como astable mediante dos monoestables conectados en un anillo, como se muestra en la figura 28. En este caso el flanco posterior del pulso generado por el primer monoestable activa el segundo. El flanco posterior de este pulso activa a su vez al primero y el ciclo se repite.

REGISTRO DE MEMORIA

En casi todos los campos de la electrónica digital son muy utilizados los registros de una clase u otra. Uno de los tipos de registro de uso más común es el que se utiliza para almacenar palabras de datos dentro de los computadores y las calculadoras. Estos registros se pueden usar directamente dentro de los cálculos, como en el caso del *acumulador* que hay dentro de un procesador o calculadora, o se pueden usar para aplicaciones generales de memoria en las que se utiliza miles, quizá millones, de registros para almacenar programas y datos.

Al analizar Los latches D, además de usarse para almacenar un bit de información frecuentemente se usan en grupos para almacenar un conjunto de bits o *palabra* de información. Es común combinar cierto número de latches dentro de un solo circuito integrado para obtener cuatro bits (un latch cuádruple) de puede guardar un *nibble* de información, u ocho bits (un latch octal) que almacena un byte de información en un solo chip. En la figura 29 se ilustra un latch octal en el que se puede muestrear y almacenar ocho bits mediante el uso de una sola entrada de habilitación. Cuando la entrada de habilitación está en alto las salidas Y_0 - Y_7 son iguales a las entradas X_0 - X_7 . Cuando la entrada de habilitación está en bajo, las salidas retienen o almacenan el valor que tenían las respectivas entradas en el momento de transición de la habilitación de alto a bajo. Esta técnica es usada para almacenar palabras de datos en paralelo en computadoras y otras áreas de electrónica digital.

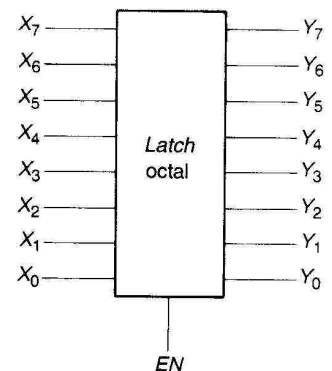


Fig. 29 Latch octal

También los *flip-flops* D se usan con este propósito, como lo ilustra la figura 30, que muestra un registro de memoria de 4 bits formado con *flip-flops* D maestro/esclavo.

Se usa una amplia gama de técnicas de circuito para el almacenamiento de información digital.

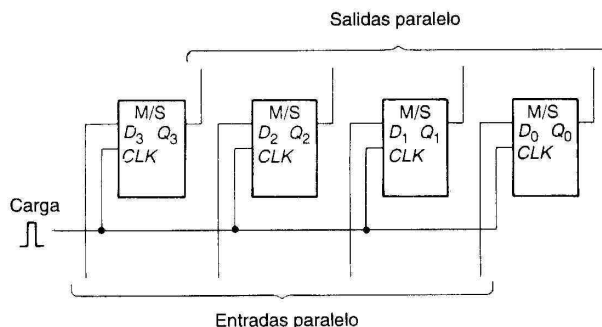


Fig. 30 Registro de memoria de cuatro bits

REGISTROS DE DESPLAZAMIENTO

Los registros de desplazamiento se usan para transformar palabras de información en paralelo a una sucesión de bits sobre una sola línea. Esto se conoce como **datos en serie**. Un registro de desplazamiento también se puede usar para tomar una sucesión de datos en serie y generar una palabra de datos en paralelo a partir de ellos. Este proceso se ilustra en la figura 31.

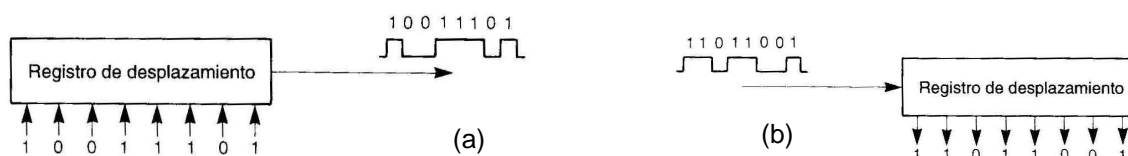


Fig.31 REGISTROS DE DESPLAZAMIENTO (a) conversión paralelo a serie (b) conversión serie a paralelo

Una aplicación de esta técnica es en la comunicación a larga distancia, en la que se pueden transformar palabras de información paralelas a forma serie para enviarlas a través de una sola línea, en lugar de necesitar varias líneas paralelas.

Considere el funcionamiento del circuito de la figura 32 (a). El circuito consiste en cuatro *flip-flops* D maestro/esclavo conectados en serie. Se aplica al circuito una secuencia de pulsos de desplazamientos regulares, que forman la señal de reloj para cada etapa.

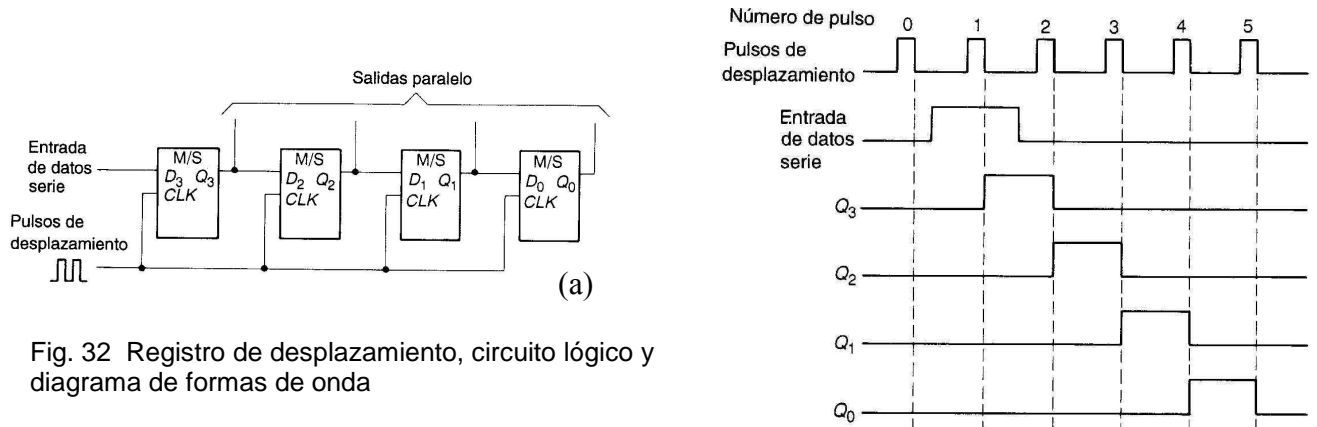


Fig. 32 Registro de desplazamiento, circuito lógico y diagrama de formas de onda

Si damos por supuesto que en principio la entrada de datos serie está en 0 y que la salida de cada *flip-flop* es cero, los repetidos pulsos de desplazamiento no tendrán efecto alguno sobre el circuito. Si ahora la entrada de datos serie se va a 1 durante un pulso de reloj y luego vuelve a cero, esta entrada se moverá unidireccionalmente a lo largo del registro, como muestra la figura 32 (b). Durante el pulso número 1 la entrada de datos es 1, por lo que D_3 está alta justo antes del flanco descendente de la señal de reloj. Entonces, cuando el reloj cae a 0, Q_3 se pone a 1. Ninguno de los otros *flip-flops* resulta afectado por la transición, pues ésta ocurre conforme el reloj se hace bajo. Justo antes del fin del pulso número 2, Q_3 y D_2 están altas pero D_3 , D_1 y D_0 están bajas. Entonces, después del pulso de reloj Q_2 se hace alta y Q_3 se pone a 0, mientras que Q_1 y Q_0 no resultan afectadas. Ante pulsos de reloj sucesivos cada salida a su vez se va a alto durante un ciclo de reloj y luego vuelve a cero. Después del pulso número 5, el registro está de nuevo en su estado original.

De lo anterior debe resultar evidente que cualquier patrón de “1” y “0” dentro del registro se mueve un bit hacia la derecha al final de cada pulso de desplazamiento. Tenemos por lo tanto un **registro de desplazamiento**. Un patrón de cuatro bits de datos serie se desplaza en el registro después de 4 pulsos de desplazamiento y aparece como una palabra paralela en las salidas Q_3 - Q_0 . Añadiendo más *flip-flops* se pueden formar registros de la longitud deseada, permitiendo el uso de palabras más largas. Por lo tanto, el registro efectúa una **conversión de serie a paralelo**.

Para efectuar una **conversión de paralelo a serie** es necesario modificar el circuito de la figura 32 (a) de manera que los datos en paralelo se puedan cargar en el registro de desplazamiento. Ya hemos visto en la figura 30 cómo se puede usar una serie de *flip-flops* D como un registro de memoria. Lo que ahora requerimos son circuitos que permitan que un conjunto de *flip-flops* se carguen en paralelo desde un registro de memoria y sean capaces de entregar datos en serie. Este circuito aparece en la figura 33.

La figura 33 muestra el subsistema que se requiere para llevar a la práctica esta función. Se trata de un **selector de datos** cuya función es permitir que una señal de control, SELECT, determine cuál de las entradas A y B se conectará a la salida X. Si SELECT es 1, A' siempre será igual a A pero B' siempre será 0. Entonces X será igual a A. Si SELECT es 0, A' será igual a 0 y B' será igual a B, lo que hace que X sea igual a B. Entonces cuando SELECT está alta $X = A$ y cuando está baja $X = B$. Ya nos hemos topado con la convención de identificar las señales activas a nivel bajo colocando una “barra” sobre su nombre simbólico. Aquí

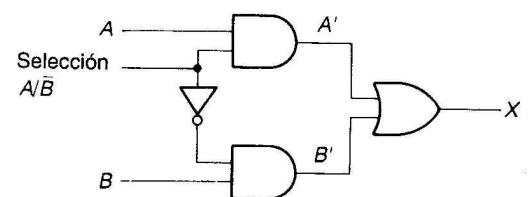


Fig. 33 Selector de datos

describimos la línea SELECT mediante el símbolo A/\overline{B} , que indica que cuando la línea está alta se selecciona A y cuando está baja se selecciona B.

La figura 34 muestra un registro de desplazamiento de carga paralelo de 4 bits. El funcionamiento del circuito está determinado por una sola señal de control, que se invierte para alimentar tres selectores de datos. Cuando la señal de control está alta, la salida Q de cada etapa se conecta a la entrada D de la siguiente y el circuito equivale eléctricamente al circuito de la figura 32. Cada vez que se aplica un pulso de reloj, el contenido del registro se mueve un lugar hacia la derecha y aparece como información de salida serie en Q_0 . Cuando la señal de control está baja, la entrada D de cada etapa se conecta a las líneas paralelas de entrada. La aplicación de un pulso de reloj ocasionará que se cargue en el registro el patrón de "1" y "0" de estas entradas. Por lo tanto, tenemos un registro que se puede usar para mover o cargar información bajo el control de una sola entrada. Esta entrada se puede nombrar *desplaza/carga* (*shift/load*).

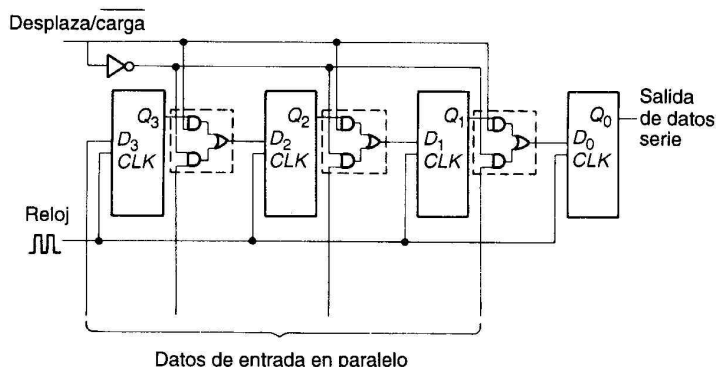


Fig. 34 Diagrama lógico del registro de desplazamiento de 4 bits

El circuito de la figura 34 se puede usar para conversión tanto de serie a paralelo como de paralelo a serie por medio de la aplicación apropiada de las señales. También es posible, mediante una ligera modificación del circuito, construir un registro que desplace en cualquier dirección. Esto se logra por medio de un circuito selector de datos para determinar si la entrada a una etapa particular proviene de la salida de la etapa de la derecha o de la izquierda.

Aunque es bastante factible construir estos circuitos a partir de compuertas sencillas estándar, es más común el uso de circuitos integrados especializados que proporcionen todos los componentes de un registro de cambios dentro de un solo encapsulado. Los dispositivos típicos proporcionan registro de 4 u 8 bits con una diversidad de características. Se pueden producir registros más largos conectando varios dispositivos en serie. Esto se logra llevando la salida serie de un dispositivo a la entrada serie del siguiente.

Aplicaciones de los registros de desplazamiento

Uno de los usos más comunes de los registros de desplazamiento se encuentra en los sistemas de **comunicación serie**. Esto implica la conversión de datos en paralelo a forma serie en el *transmisor*, transportando la información serie a través de cierta distancia y luego convirtiéndola de nuevo a forma paralelo en el *receptor*. El proceso se ilustra en la figura 35.

En el corazón del transmisor se encuentra un registro de desplazamiento que carga los datos de entrada en paralelo y luego los saca en forma serie a una velocidad determinada por una señal de reloj local. La sucesión de datos serie se transmite entonces al receptor a través de alguna forma de *canal de transmisión*. Este canal puede tener la forma de un cable, una señal de radio, una serie de pulsos de luz láser o algún otro medio de información. En el receptor, un segundo registro de desplazamiento carga la información serie y la saca en forma paralela. Para que pueda cargar la información debe recibir no sólo los datos serie, sino también la señal de reloj que le permita sincronizarse con el transmisor.

La principal ventaja de este método de transmisión consiste en que requiere menos líneas para comunicar la información: sólo dos líneas (una para los datos y otra para el reloj), en lugar de una línea por cada bit de la información paralela. Las técnicas serie también se usan mucho para la comunicación a larga distancia. Así mismo, se utilizan en aplicaciones de corto alcance, en ocasiones

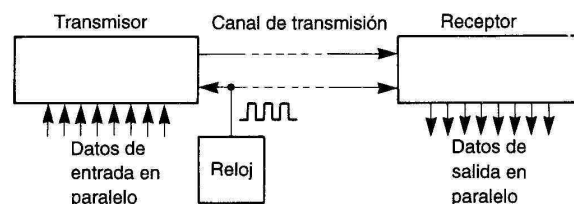


Fig. 35 esquema de comunicación serie

de unos cuantos centímetros. En algunos sistemas la necesidad de transmitir la señal de reloj junto con la información se elimina generando (o recuperando) la señal de reloj en el receptor. Esto reduce a una la cantidad de líneas que se requiere.

CONTADORES ASÍNCRONOS

CONTADORES DE RIZO (*Ripple counters*)

El circuito de la figura 36; consiste en cuatro *flip-flops* J-K disparados por flanco negativo configurados como *flip-flops* de báscula por medio de la conexión de ambas entradas J y K al 1 lógico. La salida Q de cada etapa forma la entrada de reloj para la siguiente.

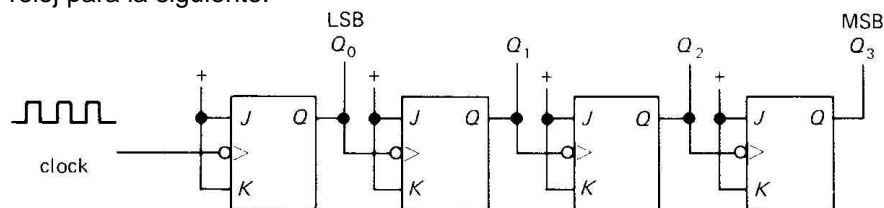


Fig. 36 Contador asíncrono

La figura 37 muestra las formas de onda resultantes dentro del circuito cuando se aplica una señal de reloj de onda cuadrada. Se puede ver que Q_0 cambia en cada flanco descendente del reloj, produciendo una forma de onda cuadrada de frecuencia de la mita que la del reloj. Q_1 cambia en cada flanco descendente de Q_0 y por lo tanto produce una forma de onda cuadrada de frecuencia mitad que la de Q_0 , un cuarto de la frecuencia del reloj. De manera similar, cada etapa subsecuente divide la señal de reloj por un factor de dos, produciendo frecuencias cada vez más bajas. Este circuito se puede considerar como un **divisor de frecuencias** y cada etapa representa un divisor de **frecuencias por la mitad**.

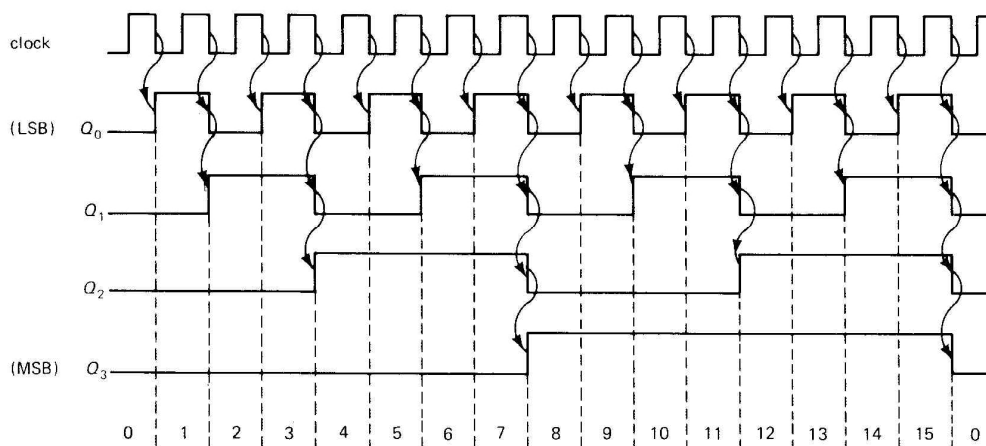


Fig. 37 Formas de onda de contador asíncrono.

Los divisores de frecuencia se pueden construir con cualquier número de etapas para brindar una proporción de división adecuada. Por ejemplo, los relojes digitales a menudo utilizan estos divisores con 15 etapas para dividir las oscilaciones de un cristal de cuarzo de 32 768 Hz para proporcionar una señal de 1 Hz que active un motor pasa a paso (para los relojes con una carátula analógica que usa manecillas) o un indicador digital.

También resulta interesante ver la secuencia de "1" y "0" que aparece en las salidas de los *flip-flops*. La Tabla muestra los valores de las salidas para cada pulso de reloj.

Se puede ver que el patrón de las salidas representa el código del número de pulsos aplicados al circuito. Por lo tanto, el circuito representa un *contador*. En este circuito en particular, los efectos de la entrada se propagan a lo largo de la serie de *flip-flops* y las salidas cambian de manera secuencial a lo largo de la línea. Es por ello que a esta forma de contador se le llama **contador de rizo** (ripple counter).

Nº pulsos	Q_3	Q_2	Q_1	Q_0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1
16	0	0	0	0
17	0	0	0	1
18	0	0	1	0
19	0	0	1	1
20	0	1	0	0

De la Tabla se puede ver que el circuito cuenta hasta el equivalente binario de 15 y luego comienza de nuevo desde cero. Por lo tanto, el contador toma 16 valores distintos. Este tipo de contador se conoce como **contador de módulo 16** o, en ocasiones, tan sólo **contador mod-16**. Usando más o menos etapas podemos construir una gama de circuitos que cuenten módulo 2^n , en los que n es el número de *flip-flops* que se use. Estos contadores cuentan desde 0 hasta 2^n-1 y luego repiten.

CONTADORES DE MÓDULO N

Hemos visto que variando el número de etapas podemos modificar nuestro contador de rizo simple para producir circuitos que cuenten hasta números diferentes antes de volver a comenzar desde cero. Sin embargo, la variación del número de etapas sólo nos permite elegir valores para los módulos del contador que sean potencias de dos. En muchas aplicaciones deseamos contar hasta ciertos números que quizá no satisfagan este requisito. Por lo tanto, necesitamos un método para construir un contador como **contador de módulo N**.

Para generar un contador con un módulo de N , sólo necesitamos asegurar que en el pulso del reloj después de que el contador llegue a $N-1$, éste vuelva a cero.

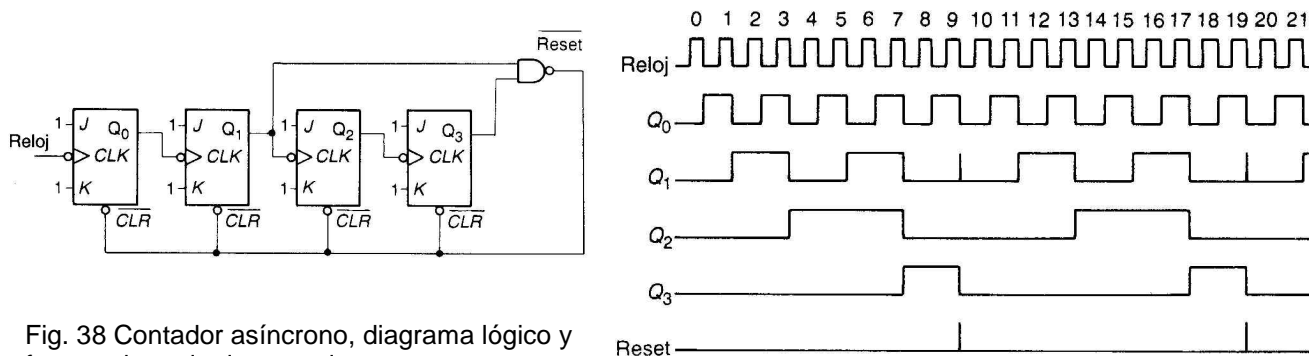


Fig. 38 Contador asíncrono, diagrama lógico y formas de onda de contador .

En la figura 37 aparece un ejemplo de este tipo de circuito para un valor de $N = 10$. Los contadores que cuentan módulo 10 se conocen como **contadores de década**. Los contadores de década que producen números binarios en el intervalo de 0000 a 1001 a menudo reciben el nombre de **contadores de decimal codificado en binario**, o simplemente **contadores BCD** (binary coded decimal).

El circuito del contador de década es similar al del contador asíncrono simple de la figura 36, pero tiene un circuito de *reset* adicional para poner a cero todos los *flip-flops* cuando hayan llegado a la cuenta de 10. La operación de puesta a cero se logra usando *flip-flops* que tienen entradas **CLEAR** (puesta a cero). La señal aplicada a esta entrada proviene de circuitos que detectan una cuenta de 10. Como el 10 es el 1010 binario, se puede usar una compuerta NAND de dos entradas para detectar la primera ocasión en que los bits 1 y 3 están altos (recuerde que los bits empiezan desde el bit 0). Tan pronto como se detecta esta situación, la línea **RESET** pasa a bajo, poniendo el contador a cero, lo que también hace que **RESET** vuelva a 1. Entonces el contador continúa a partir de cero al igual que antes. Como los circuitos de puesta a cero actúan con gran rapidez, los efectos del pico produciendo sobre Q_1 por lo general son despreciables.

Se puede construir un contador de módulo N para cualquier valor de N mediante la construcción de un contador con n etapas, donde $2^n > N$, y luego añadiendo un circuito de puesta a cero que detecte la cuenta de N .

CONTADORES DESCENDENTES

En algunas aplicaciones es necesario tener un contador que cuente hacia abajo en lugar de hacerlo hacia arriba. Esto se puede lograr por medio de un circuito similar al del contador de rizo, tomando la señal del reloj para las etapas siguientes de \bar{Q} en lugar de Q . Esto se ilustra en la figura 38, en la que se debe notar que las salidas se siguen tomando de $Q_0 - Q_3$ y no de $\bar{Q}_0 - \bar{Q}_3$.

cómo podría pensarse. La tabla Fig. 40 muestra la secuencia de salida para el contador descendente de cuatro etapas. Esta técnica se puede aplicar a contadores de cualquier módulo.

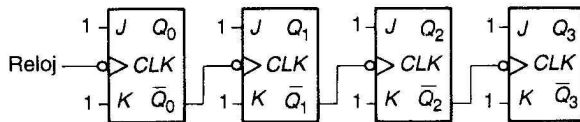
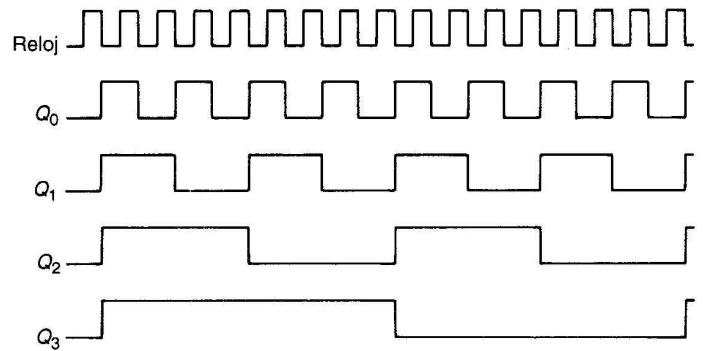


Fig. 39 Contador asíncrono descendente, diagrama lógico y formas de onda de contador



CONTADORES ASCENDENTES/DESCENDENTES

Combinando los circuitos de las figuras 36 y 38 por medio del **selector de datos** de la figura 33 es posible construir un contador que pueda contar hacia arriba o hacia abajo. Este circuito aparece en la figura 40.

La dirección de la cuenta está controlada por la señal *ascendente / descendente*. Cuando esta línea se encuentra alta, se usa la salida Q de cada etapa a fin de proporcionar el reloj para la siguiente etapa y el circuito cuenta hacia arriba. Cuando la línea de control está baja, la salida \bar{Q} se conecta a la siguiente etapa y el circuito cuenta hacia abajo.

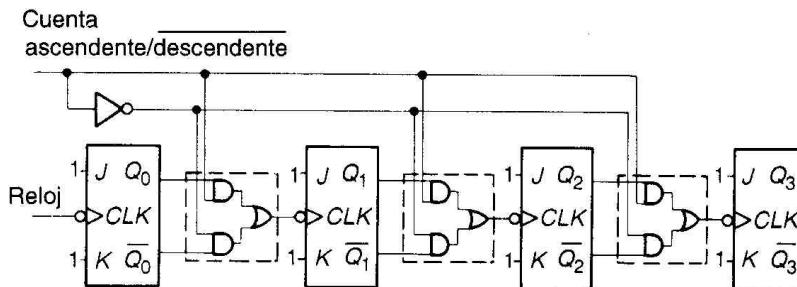


Fig. 41 Contador asíncrono ascendente/descendente

Reloj	Q_3	Q_2	Q_1	Q_0	Cuenta
0	0	0	0	0	0
1	1	1	1	1	15
2	1	1	1	0	14
3	1	1	0	1	13
4	1	1	0	0	12
5	1	0	1	1	11
6	1	0	1	0	10
7	1	0	0	1	9
8	1	0	0	0	8
9	0	1	1	1	7
10	0	1	1	0	6
11	0	1	0	1	5
12	0	1	0	0	4
13	0	0	1	1	3
14	0	0	1	0	2
15	0	0	0	1	1
16	0	0	0	0	0
17	1	1	1	1	15
18	1	1	1	0	14
19	1	1	0	1	13
20	1	1	0	0	12

Fig. 40 Tabla de salidas del Contador descendente.

RETARDO EN LA PROPAGACIÓN EN LOS CONTADORES DE RIZO

Aunque los contadores de rizo son muy fáciles de construir, tienen una desventaja importante que se hace evidente en particular cuando se requiere el funcionamiento a alta velocidad. Como la salida de un *flip-flop* está activada por el cambio de estado de la etapa previa, los retardos producidos por cada *flip-flop* se suman a lo largo de la cadena. Cada *flip-flop* tarda un tiempo finito para responder a los cambios en sus entradas. Esto se llama **tiempo de retardo en la propagación** o t_{pd} . En un contador de rizo de n etapas se requerirá $n \times t_{pd}$ para que el contador responda. Si se toma una lectura del contador durante este tiempo, el valor será confuso pues algunas etapas habrán cambiado, mientras que otras no. Esto produce un límite fundamental a la máxima frecuencia de reloj que se puede usar con el contador. Si se reciben pulsos de reloj en la primera etapa antes de que la última haya respondido, el contador no leerá el valor correcto en ningún momento.

CONTADORES SÍNCRONOS

Una solución a los problemas del retardo en la propagación dentro de los contadores de rizo es el uso de técnicas síncronas. Esto se logra conectando

todos los *flip-flops* que hay dentro de un contador a una señal de reloj común para que todos cambien de estado al mismo tiempo. Esto asegura que poco tiempo después de que haya cambiado la señal de reloj se pueda leer el contador con la confianza de que todas las etapas habrán respondido.

Es evidente que si todas las etapas de un contador se conectan al mismo reloj, entonces se debe usar algún método para determinar cuáles etapas cambian de estado y cuáles permanecen iguales. Para investigar esto, consideremos las salidas que se requieren de un contador de cuatro etapas, como muestra la tabla de la figura 41.

Al ver esta secuencia podemos determinar reglas que definan cuándo cambian las diversas salidas. Podemos entonces producir un contador que consista en varios *flip-flops* J-K disparados por flanco, con circuitos que hagan valer estas reglas. Como todos los *flip-flops* se disparan en forma simultánea, responderán a los valores de sus entradas de control tienen justo antes del flanco del reloj. El pequeño retardo causado por los circuitos que se utilicen para generar estas señales garantizará que se encuentren estables mientras responden los *flip-flops*.

Q_0 Esta salida es la más fácil de definir pues cambia con cada pulso del reloj. Esto se puede producir con facilidad configurando el *flip-flop* 0 para que bascule con cada pulso del reloj, mediante la conexión tanto de J como de K a 1.

Q_1 Esta salida cambia de estado después de cada periodo de reloj en el que Q_0 esté alta. Esto se puede lograr conectando tanto J como K del *flip-flop* 1 a Q_0 . Cuando Q_0 se encuentre alta, esta etapa basculará; cuando esté baja, permanecerá sin cambio.

Q_2 Esta salida cambia después de cada periodo de reloj cuando tanto Q_0 como Q_1 estén altas. Esto se puede lograr uniendo las tres entradas mediante la función AND y aplicando la señal resultante a las entradas J y K del *flip-flop* 2.

Q_3 Esta salida cambia después de cada periodo de reloj cuando Q_0 , Q_1 y Q_2 estén altas. Esto se puede lograr uniendo las tres entradas mediante la función AND y aplicando una señal que se aplique a las entradas J y K del *flip-flop* 3

El circuito resultante aparece en la figura 42. Es evidente que esta técnica se puede extender para producir contadores síncronos de cualquier longitud.

reloj	Q_3	Q_2	Q_1	Q_0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1
16	0	0	0	0
17	0	0	0	1
18	0	0	1	0
19	0	0	1	1
20	0	1	0	0

Fig.42 Tabla de salidas del Contador ascendente de 4 etapas.

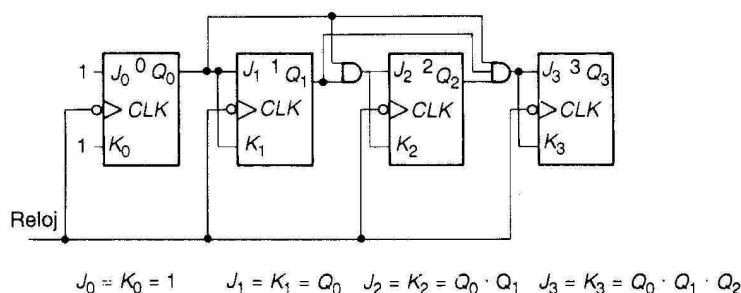


Fig 43 Contador síncrono de cuatro etapas

Conforme aumenta el número de elementos en el contador, crece el número de señales que se debe unir mediante la función AND en cada etapa. Este problema se puede remediar haciendo notar que las señales requeridas por las entradas J y K de una etapa son tan sólo las requeridas por la etapa anterior unidas mediante la función AND con la salida de la etapa anterior. Esto permite simplificar el circuito, como se muestra en la figura 44. Este circuito se puede extender agregando tantas etapas idénticas como sea necesario.

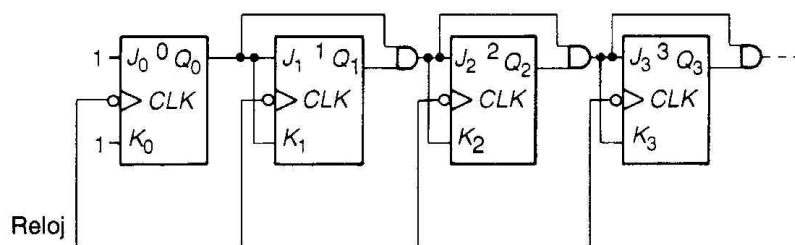


Fig 44 Contador síncrono de cuatro etapas en cascada

Los contadores síncronos se pueden configurar para producir contadores ascendentes, descendentes, ascendentes/descendentes y de módulo N de manera similar a los contadores de rizo. Tienen la ventaja de que todas las salidas cambian al mismo tiempo, permitiendo la lectura confiable del contador poco tiempo después; y tienen la desventaja de presentar una mayor complejidad en comparación con los contadores de rizo, pero se pueden usar a frecuencias de reloj más altas.