

Tema 3

SISTEMAS SECUENCIALES

3.1. CONCEPTOS BÁSICOS

Al igual que el Álgebra de Boole es el modelo matemático de la lógica combinacional, la Teoría de Autómatas (de número de estados finito) es el modelo matemático de la lógica secuencial. En lógica combinacional se representan todas aquellas funciones en las que, para conocer el valor de la salida en un determinado instante, solo hace falta conocer los valores de las entradas en ese instante, es decir, el circuito no tiene memoria y, por consiguiente, no es necesario definir su estado interno para poder predecir el valor de la salida una vez que se conoce la función y los valores de las entradas. Existen, no obstante, una serie de problemas que no pueden analizarse ni resolverse utilizando solo lógica combinacional. El ejemplo más sencillo de sistema cuya descripción es imposible sin definir estados internos es el que simula el comportamiento de un bolígrafo. Podemos admitir que es un sistema que posee una entrada con dos valores (pulsar o no pulsar), y que responde sacando o metiendo la punta. Sin embargo, esta respuesta depende del estado anterior (punta dentro o punta fuera). Si pulsamos estando la punta dentro, ésta sale. En cambio si estaba fuera, entra. Decimos entonces que el bolígrafo es un autómata de 2 estados.

La diferencia entre un circuito combinacional y un circuito secuencial estriba en que en estos últimos la salida es una función de las entradas y de un cierto estado del sistema. Un sistema secuencial se puede considerar como un sistema combinacional con unas propiedades añadidas de almacenamiento para retener el estado. El estado se constituirá por variables adicionales a las variables de entrada, llamadas *variables internas*. La salida del sistema dependerá de las variables internas además de las entradas, y a su vez el propio estado será actualizado en función del estado actual y de las entradas.

Un sistema secuencial cualquiera se representa por el siguiente conjunto:

$$M = \langle S, X, Y, F_1, F_2 \rangle$$

donde S es el conjunto de estados, X el conjunto de entradas, Y el conjunto de salidas, F_1 la función de transición (de estado siguiente) y F_2 la función de salida. Como ya hemos

visto en el primer tema, hay dos formas de representación de estas dos últimas funciones que dan lugar a dos modelos de sistemas secuenciales: los modelos Mealy y Moore. En ambos la función de estado siguiente depende del estado actual y de la entrada:

$$F_1 : S \times X \longrightarrow S$$

Difieren sin embargo en la función de salida. En el modelo Mealy, la salida es función del estado actual y de la entrada:

$$F_2 : S \times X \longrightarrow Y$$

Mientras que el modelo Moore la salida es solo función del estado actual

$$F_2 : S \longrightarrow Y$$

Ambos modelos son equivalentes. En general, cualquier sistema secuencial puede definirse mediante un modelo de Moore o de Mealy, y hay reglas concretas para traducir un tipo de autómatas a otro. Normalmente, el modelo de Moore emplea más estados internos que el de Mealy. En este tema veremos ejemplos de implementación de sistemas secuenciales según los dos modelos.

Son posibles dos alternativas a la hora de implementar el almacenamiento del estado y por lo tanto del sistema secuencial. Una de ellas es usar dispositivos que en sí mismos son sistemas secuenciales: los biestables. Un biestable presenta dos configuraciones estables y diferenciadas que es capaz de mantener bajo ciertas condiciones. Como son dos las configuraciones que presentan ($Q=0$ y $Q=1$), son dos los estados que pueden almacenar. Aumentando el número de biestables se aumenta el número de estados, de tal forma que con n biestables es posible representar 2^n estados. Entre los biestables existentes los más adecuados para la implementación de sistemas secuenciales son los biestables sincronizados a flancos, los cuales mantienen la configuración mientras no se active la señal de reloj. Los sistemas secuenciales contruidos con biestables sincronizados reciben el nombre de **síncronos**, ya que el cambio de estado se realiza en sincronismo con la señal de reloj.

La segunda posibilidad es utilizar retardos para almacenar los estados del sistema. En muchas ocasiones un lazo de realimentación introduce un retardo respecto a las líneas de entrada que puede utilizarse para almacenar el valor de una variable. En general con n retardos en líneas de realimentación pueden almacenarse 2^n estados. Este tipo de sistemas denominados **asíncronos** presentan un procedimiento de diseño menos general y con más problemas que el correspondiente al de los sistemas síncronos. En este curso nos centraremos exclusivamente en el estudio de sistemas secuenciales síncronos.

La forma clásica de representar un sistema secuencial es mediante una *Tabla de Estados* o mediante un *Diagrama de Estados*. La obtención de esta tabla o diagrama a partir de las especificaciones del sistema es el primer paso en el diseño de un sistema síncrono. Las dos son representaciones equivalentes y nos proporcionan la misma información sobre el sistema secuencial. En una Tabla de Estados se representan el estado siguiente y la

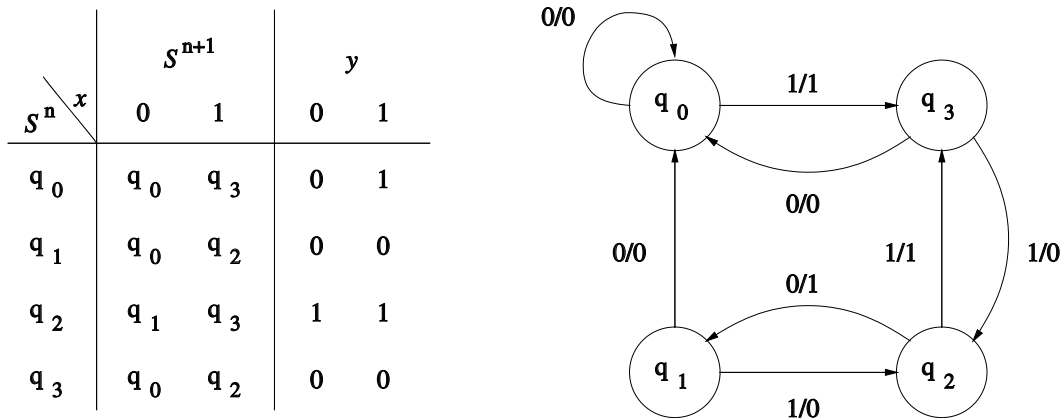


Figura 3.1: Tabla de estados y diagrama de estados.

salida en función del estado actual y de la entrada. Un Diagrama de Estados es una representación gráfica del sistema secuencial: se utiliza un círculo por cada uno de los estados y un arco por cada transición entre estados. En el modelo Mealy, encima de cada arco se indican las combinaciones de entrada que originan esa transición junto con la salida que se produce. Para el modelo Moore, en los arcos solo se ponen las combinaciones de entrada y el valor de la salida se coloca en el círculo correspondiente al estado asociado a ese valor. En la figura 3.1 mostramos un ejemplo de tabla de estados y diagrama de estados de un autómata Mealy.

3.2. BIESTABLES

3.2.1. Biestable RS

Un biestable, en su estructura más simple, se puede construir con dos puertas NOR realimentadas, tal y como se ilustra en la figura 3.2. A partir de este circuito se construyen formas más sofisticadas de biestables. La conexión cruzada de la salida de cada puerta a la entrada de la otra constituye el lazo de realimentación imprescindible en todo dispositivo de “memoria”. Esta celda básica cuenta con dos salidas (Q y \bar{Q}), y con dos entradas: SET (S) y RESET (R). Este tipo de biestable se conoce con el nombre de biestable RS.

Funcionamiento del biestable RS

Las dos entradas de un biestable RS van a realizar las siguientes acciones (opuestas) cuando son activadas:

- R (RESET): poner la salida a cero ($Q = 0, \bar{Q} = 1$)
- S (SET): poner la salida a uno ($Q = 1, \bar{Q} = 0$)

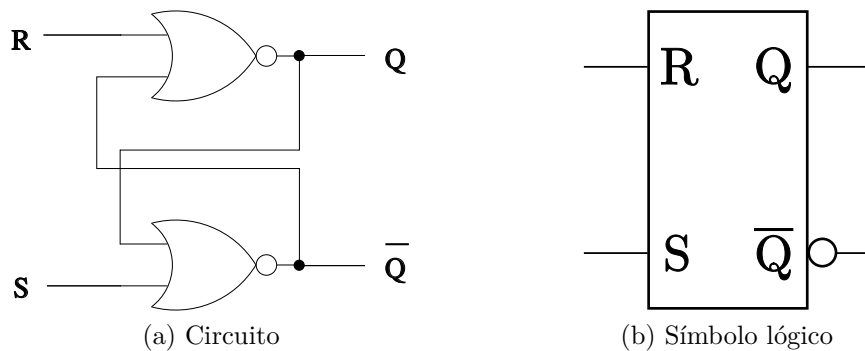


Figura 3.2: Biestable RS básico con puertas NOR.

Si ninguna de las entradas está activa, el biestable mantendrá las salidas en el valor previo. En el caso en que ambas entradas se activen simultáneamente Q y \bar{Q} tomarán el mismo valor, con lo que esta configuración normalmente no se utilizará.

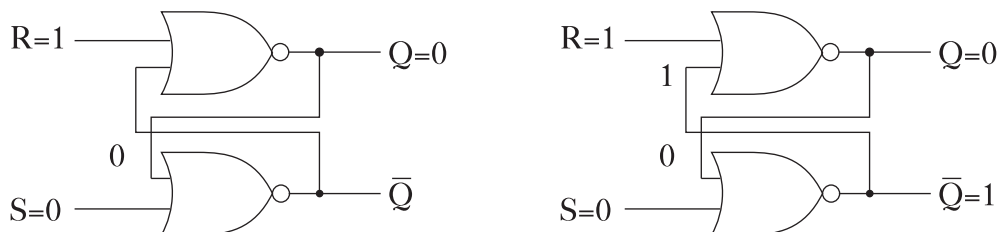
Biestable RS construido a partir de puertas NOR

En este caso las dos entradas R y S son activas a nivel alto (a 1), ya que, como se puede deducir rápidamente de la tabla de verdad de la puerta NOR, siempre que una de las entradas sea 1, la salida será siempre 0 (independientemente del valor de la segunda entrada):

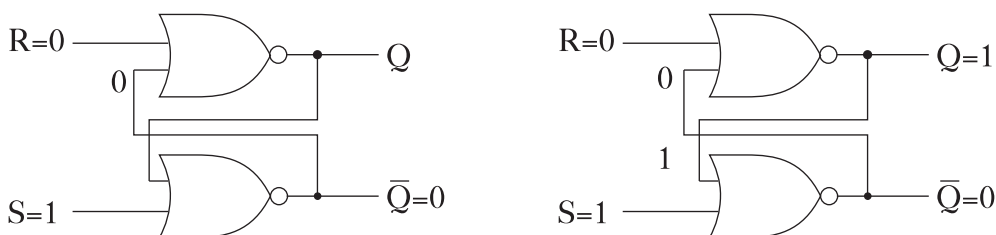
A	B	NOR
0	0	1
0	1	0
1	0	0
1	1	0

Esta característica es la que nos va a ayudar a analizar los circuitos con puertas NOR realimentadas. Así, se pueden dar los siguientes cuatro casos:

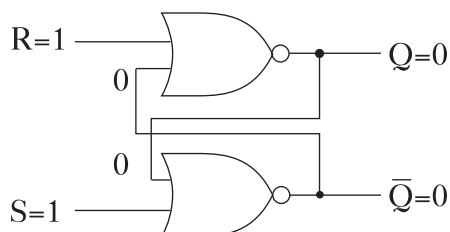
1. Si se activa el RESET ($R = 1, S = 0$) la salida será siempre cero ($Q = 0$). La otra puerta NOR tendrá entonces dos ceros como entradas, con lo que \bar{Q} será uno.



2. Si se activa el SET mientras el RESET está desactivada ($S = 1, R = 0$) entonces \bar{Q} siempre será cero (0). La otra puerta NOR tendrá, por lo tanto, dos ceros como entradas con lo que su salida (Q) será uno.

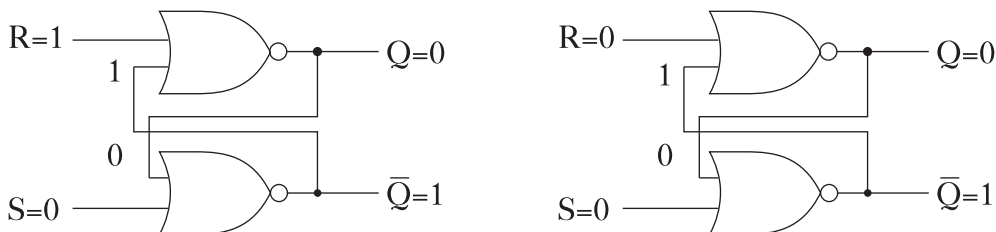


3. Si ambos, RESET y SET se activan ($R = 1, S = 1$) se hará $Q = 0$ y $\bar{Q} = 0$. Esta configuración normalmente no se utilizará, ya que, por norma general, nos interesará que Q y \bar{Q} sean siempre opuestas una a la otra. Es importante recalcar que el hecho de que las salidas se llamen Q y \bar{Q} no significa que una es la negada de la otra (y este caso, en donde activamos RESET y SET a la vez, es una buena prueba de ello).

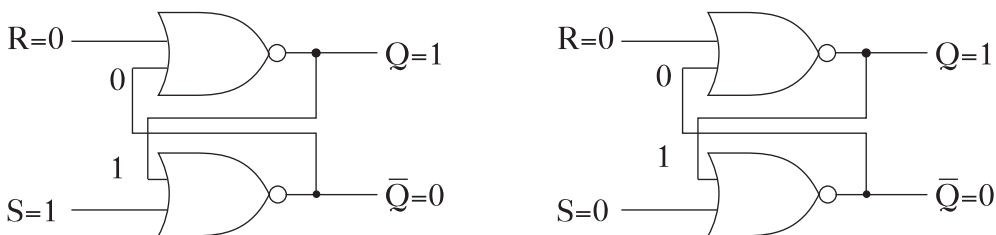


4. Supongamos que ninguna de ambas entradas sean activas ($R = 0, S = 0$). En esta situación el valor de las salidas de las puertas NOR no está determinada por estas entradas, sino por el valor de los lazos de realimentación (Q y \bar{Q}). Serán posibles dos configuraciones, que dependerán del valor previo de los lazos de realimentación y, por tanto, de todas las entradas anteriores. Como se puede ver, en ambos casos se mantendrá invariable la configuración inmediatamente anterior.

$$\{R = 1, S = 0\} \rightarrow \{R = 0, S = 0\}$$



$$\{R = 0, S = 1\} \rightarrow \{R = 0, S = 0\}$$



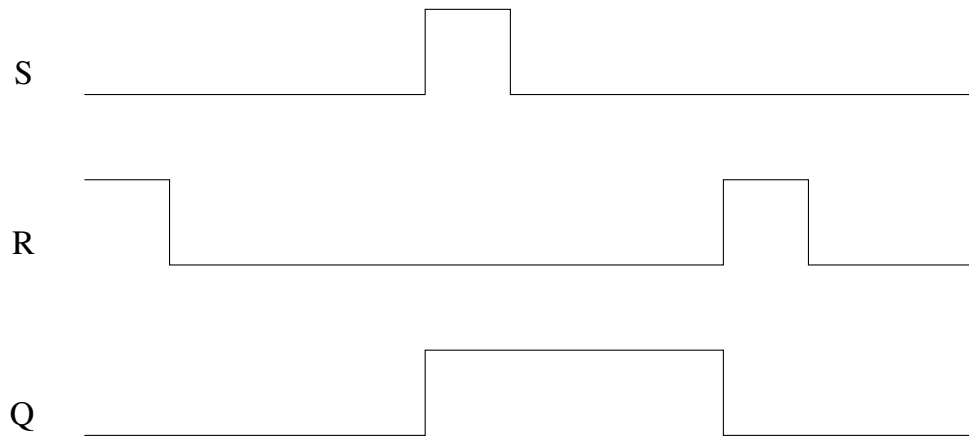


Figura 3.3: Ejemplo de funcionamiento de un biestable RS (NOR).

De esta forma, las tablas de transiciones de estado para un biestable RS (NOR) son:

R	S	Q^n	Q^{n+1}
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	No usado
1	1	1	No usado

R	S	Q^{n+1}
0	0	Q^n
0	1	1
1	0	0
1	1	No usado

Por tanto, cuando las entradas R y S están inactivas el biestable mantiene la configuración alcanzada en la operación previa, es decir, el estado siguiente (Q^{n+1}) será igual al estado actual (Q^n). De esta forma se guarda el bit de información previamente almacenado (1 si se ha actuado sobre el SET o 0 si se ha actuado sobre el RESET). En la figura 3.3 mostramos mediante un cronograma un ejemplo del funcionamiento de un biestable RS de puertas NOR.

Biestable RS (NOR) sincronizado a nivel

El biestable básico, tal como se ha descrito hasta aquí, es un circuito secuencial asíncrono. En cualquier sistema digital que incluya un cierto número de puertas y elementos de este tipo prácticamente va a ser imposible garantizar que las señales R y S se presenten, exactamente, en los instantes de tiempo requeridos para realizar las operaciones lógicas, con lo que podemos perder fácilmente el control sobre el circuito. Esta dificultad puede soslayarse permitiendo cambios de estado en el biestable solo cuando lo indique un reloj externo que, usualmente, será común para todo el sistema secuencial. De este modo, las señales de salida se sincronizarán con el reloj, no dependiendo las transiciones del momento de llegada de las señales R y S , mejorando por tanto la coordinación.

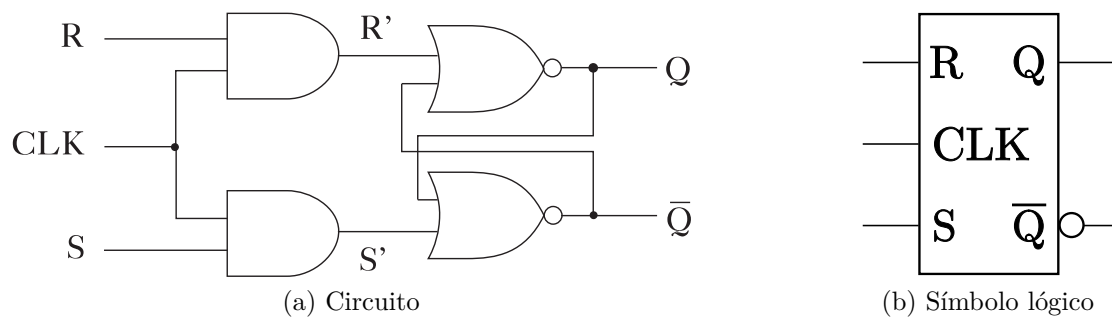


Figura 3.4: Biestable RS (NOR) sincronizado a nivel.

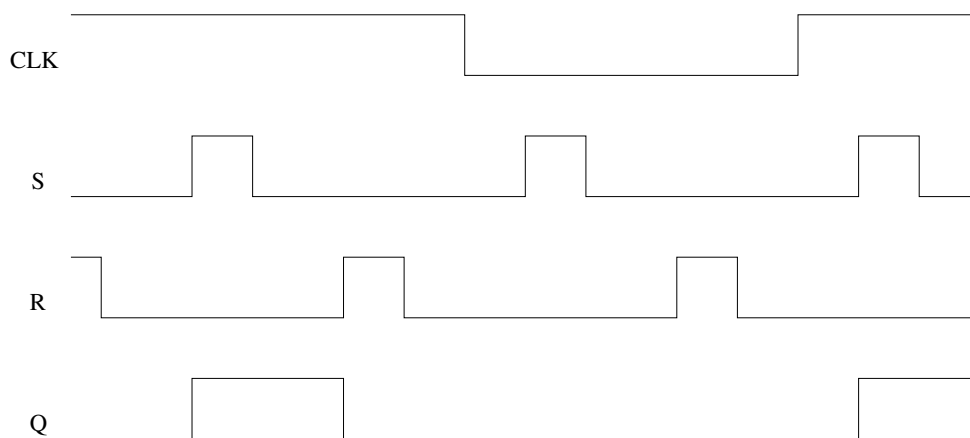


Figura 3.5: Ejemplo de funcionamiento de un biestable RS (NOR) síncrono.

El biestable RS sincronizado por **nivel** (figura 3.4) consta de una celda RS básica con puertas NOR, a la que se añaden dos puertas AND adicionales en la entrada. La señal de reloj que realiza la sincronización es introducida en cada puerta AND, en tanto que las señales R y S constituyen las otras entradas. De esta manera, las entradas a las puertas NOR (R' y S') solo serán activas cuando el reloj esté en alta, con lo que las entradas R y S seguirán determinando el estado final del biestable, pero en transiciones que únicamente podrán ocurrir cuando el reloj las permita.

Por tanto, la señal de reloj (CLK) que hemos introducido genera el siguiente comportamiento del biestable (ver figura 3.5):

- Cuando el reloj está inactivo ($CLK = 0$) tendremos $R' = S' = 0$ (figura 3.4a) y el biestable mantendrá su estado, independientemente de los valores de las señales R y S .
- Cuando el reloj está activo ($CLK = 1$) serán $R' = R$ y $S' = S$ (figura 3.4a), con lo cual el biestable tendrá un funcionamiento similar a un RS asíncrono.

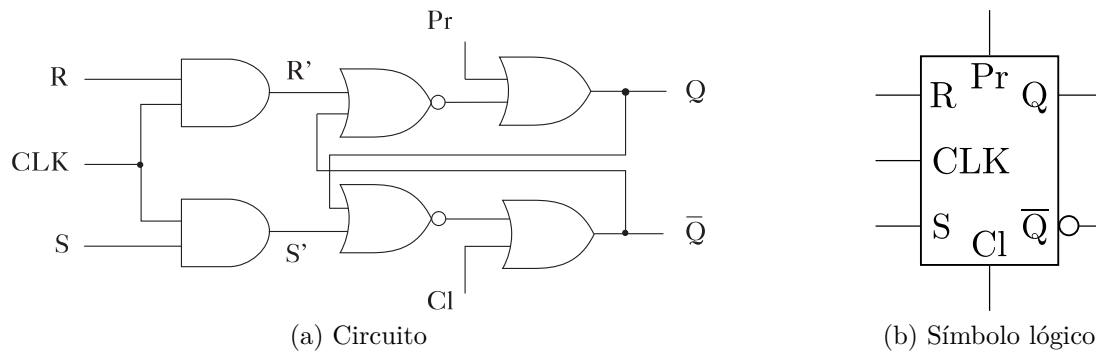


Figura 3.6: Biestable RS (NOR) sincronizado a nivel y con entradas Pr y Cl asíncronas.

Según esto, la tabla de transiciones de estado de un biestable RS síncrono es:

CLK	R	S	Q^{n+1}
0	x	x	Q^n
1	0	0	Q^n
1	0	1	1
1	1	0	0
1	1	1	No usado

Biestable RS (NOR) sincronizado a nivel con entradas asíncronas de Preset y Clear

En toda la descripción previa de los biestables RS sincronizados hemos partido de un cierto estado inicial de biestable a partir del cual se realizan las transiciones sincronizadas por la señal de reloj. En la práctica, es frecuentemente deseable disponer de los medios para fijar el estado del biestable a 0 ($Q = 0$) o a 1 ($Q = 1$) independientemente de sus entradas R , S o del reloj. Esto se consigue modificando el circuito del biestable en la forma ilustrada en la figura 3.6a. Las señales de Preset (Pr) y Clear (Cl) actúan de manera prioritaria e independiente de las otras líneas de entrada: si se activa Pr , Q pasará a 1, independientemente del resto de las señales; y si se activa Cl , Q pasará a 0.

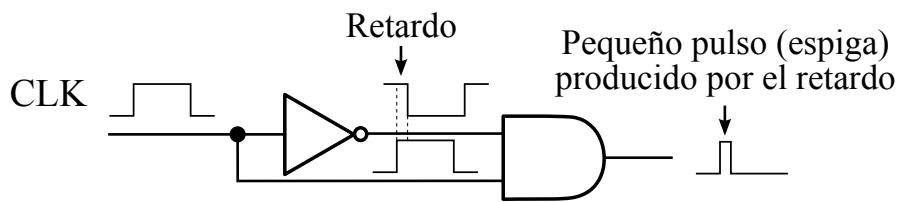


Figura 3.7: Detector de flanco positivo.

El comportamiento del biestable completo se puede ver en las siguientes tablas de transiciones de estado:

Pr	Cl	Q^{n+1}	Pr	Cl	CLK	R	S	Q^{n+1}
			0	1	x	x	x	0
			1	0	x	x	x	1
		Q^n	1	1	x	x	x	No permitido
0	0	0	0	0	0	x	x	Q^n
1	0	1	0	0	1	0	0	Q^n
1	1	No permitido	0	0	1	0	1	1
			0	0	1	1	0	0
			0	0	1	1	1	No usado

Resumiendo:

- Pr y S ponen a 1.
- Cl y R ponen a 0.
- S y R sincronizadas (solo tienen efecto cuando el reloj está activo).
- Pr y Cl asíncronas (tienen efecto siempre).
- En caso de contradicción tienen prioridad Pr y Cl .

Biestable RS disparado a flancos

En un biestable disparado por **flancos**, el cambio de estado solo se permite en las transiciones de la señal de reloj, o bien cuando la señal de reloj pasa del nivel 0 al nivel 1 (biestable disparado en flancos positivos), o bien cuando la señal de reloj pasa de 1 a 0 (biestable disparado en flancos negativos).

Hay varias formas de construir biestables disparados por flancos. Nosotros solo veremos una de ellas que utiliza el circuito detector de flancos de la figura 3.7. En este caso el circuito corresponde a un detector de flancos positivos. Para detectar un flanco negativo se puede usar este mismo detector de flancos positivos negando previamente la señal de reloj, o construir uno cambiando la puerta AND por una puerta NOR.

Con el detector de flancos colocado en la entrada de reloj obtenemos un biestable RS sincronizado a flanco positivo (figura 3.8a). Este circuito opera igual que el biestable RS

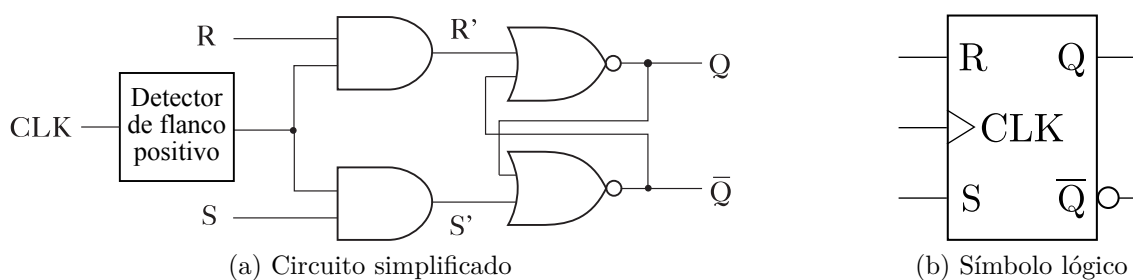


Figura 3.8: Biestable RS sincronizado a flanco positivo.

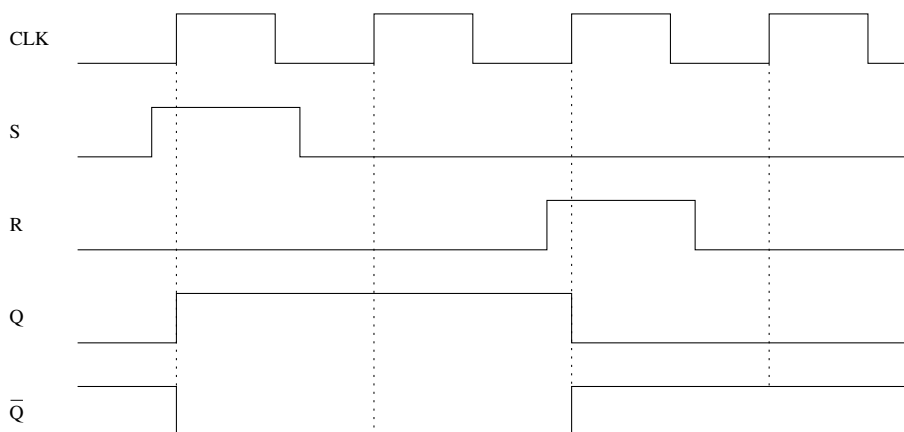


Figura 3.9: Ejemplo de funcionamiento de un biestable RS.

sincronizado a nivel que ya hemos visto, pero ahora cuando el reloj pasa a valor alto serán $R' = R$ y $S' = S$ durante un intervalo de tiempo muy pequeño (la duración del pulso debido al retardo, que es de unos nanosegundos). El resultado final es que el estado del biestable solo podrá cambiar en los flancos positivos de la señal de reloj (figura 3.9).

La tabla de transiciones de estado de un biestable RS sincronizado a flanco positivo sería:

CLK	R	S	Q^{n+1}
x	x	x	Q^n
↑	0	0	Q^n
↑	0	1	1
↑	1	0	0
↑	1	1	No usado

3.2.2. OTROS BIESTABLES

A continuación vamos a estudiar otros tres tipos de biestables disparados por flancos: D, JK y T. Aunque el biestable RS visto anteriormente no está disponible como circuito integrado, veremos que es la base para la construcción de estos otros tipos de biestables. En la figura 3.10 mostramos los símbolos lógicos de estos biestables.

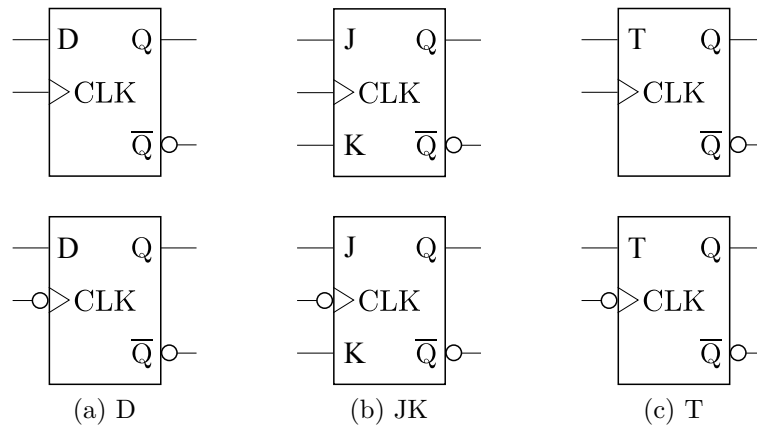


Figura 3.10: Símbolos lógicos de biestables: en la fila superior disparados por flanco positivo y en la fila inferior disparados por flanco negativo.

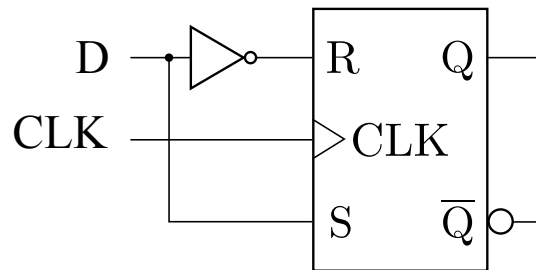


Figura 3.11: Biestable D sincronizado a flanco positivo.

Biestable D

El biestable tipo D (figura 3.11) es una modificación del biestable RS. La entrada D se aplica directamente a la entrada S , y su complemento a la entrada R . El nombre de biestable D viene como consecuencia de su capacidad de transferir “datos” desde la línea de entrada a la salida, siempre que los pulsos de reloj lo permitan. Cuando D es 0, se activa R y el biestable pasa a $Q = 0$; mientras que, cuando D es 1, se activa S y el biestable pasa a $Q = 1$. En ambos casos, la entrada se transmite a la salida, es decir, su tabla de transición será:

CLK	D	Q^{n+1}
x	x	Q^n
↑	0	0
↑	1	1

A partir de esta tabla podemos obtener la ecuación de transición de estados del biestable D que nos da el estado siguiente (Q^{n+1}) en función de D y el estado actual (Q^n):

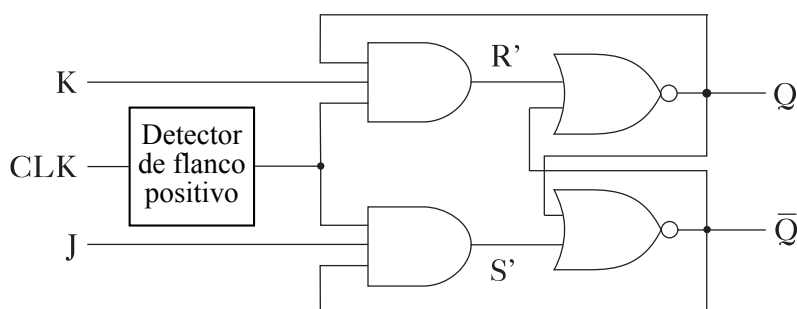


Figura 3.12: Biestable JK sincronizado a flanco.

		Q^{n+1}	
		0	1
D	Q^n		
	0		
	1	1	1

$$Q^{n+1} = D$$

Biestable JK

En el biestable RS vimos que existía un estado ambiguo como consecuencia de aplicar simultáneamente dos niveles activos a las líneas R y S . La ambigüedad surge como consecuencia de ser $Q = \overline{Q}$ a la salida, y por no conocer con certeza el estado del biestable resultante si ambas entradas se hacen inactivas simultáneamente. El biestable JK¹ es un refinamiento del RS en el que el estado indeterminado queda, en este caso, perfectamente definido. Las entradas J y K se comportan como las entradas S y R , respectivamente; sin embargo, cuando se activan simultáneamente, el biestable conmuta al estado complementario del que se encuentra.

En la figura 3.12 se muestra circuito simplificado de un biestable JK sincronizado a flanco positivo. Como se aprecia, existe un lazo de realimentación de las salidas hacia la puerta AND de entrada para evitar la inestabilidad del RS. Cuando las entradas J y K aparecen simultáneamente activas, la salida que en ese momento se encuentre a 1 hace que la salida de la puerta AND correspondiente se ponga a 1 (la otra permanecerá en 0), lo que produce el cambio de estado del biestable en cualquier caso.

El comportamiento de un biestable JK sincronizado a flanco positivo se puede resumir en la siguiente tabla:

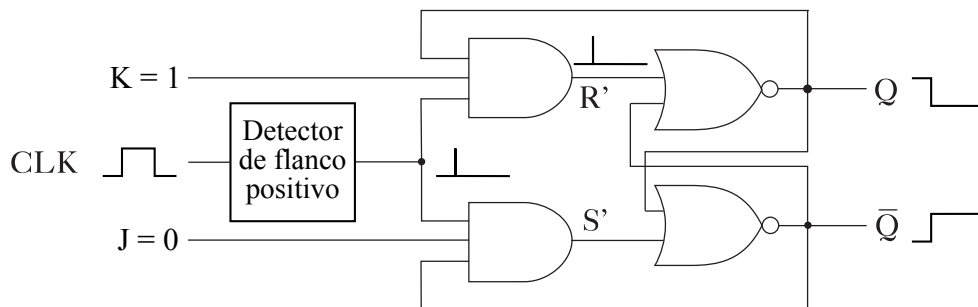
¹El nombre de biestable JK es en honor a su inventor Jack Kilby.

CLK	J	K	Q^{n+1}
x	x	x	Q^n
↑	0	0	Q^n
↑	0	1	0
↑	1	0	1
↑	1	1	$\overline{Q^n}$

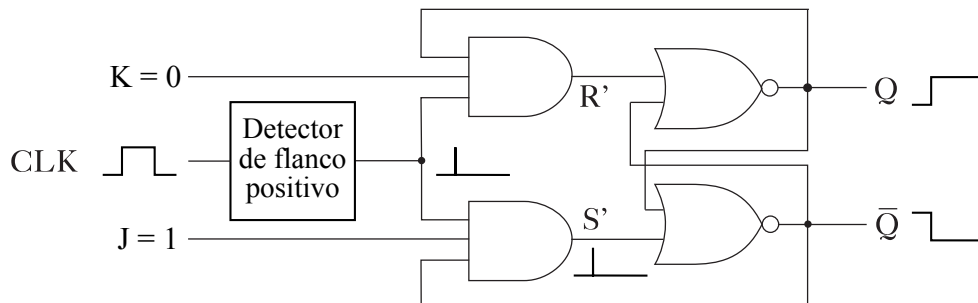
J → pone a 1
K → pone a 0

En la figura 3.12 puede observarse que mientras no aparezca un flanco en la señal de reloj se verifica que $R' = S' = 0$, con lo que el biestable mantiene el estado previamente almacenado, es decir, $Q^{n+1} = Q^n$. Veamos algunas transiciones debidas a la activación de J y K cuando aparece un flanco positivo del reloj. Obviamente, cuando J y K están desactivadas, es decir son cero, el biestable mantiene el estado actual, lo mismo que ocurría con el biestable RS.

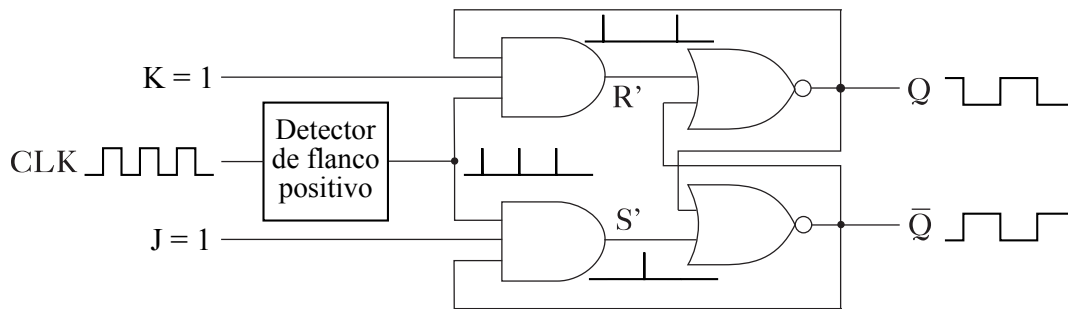
- Supongamos el biestable en $Q = 1$ y queremos ponerlo a 0. Para ello necesitamos activar (poner a 1) la entrada K . El comportamiento del biestable podemos observarlo en la siguiente figura:



- Supongamos el biestable en $Q = 0$ y queremos ponerlo a 1. Para ello necesitamos activar (poner a 1) la entrada J , como se muestra en esta otra figura:



- Por último veamos que ocurre cuando cuando J y K están activas simultáneamente (es decir, $J = K = 1$). Suponemos que inicialmente el biestable tiene almacenado el estado $Q = 1$.



En este último caso, se puede observar que el biestable conmuta de estado en cada flanco positivo de reloj, pasando alternativamente por las dos condiciones de salida estables. Este fenómeno se debe a la realimentación de las salidas del biestable a las puertas AND de entrada, que es precisamente lo que nos permite conmutar de estado.

Las tablas de transiciones del biestable JK (completa y reducida) son las siguientes:

J	K	Q^n	Q^{n+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

J	K	Q^{n+1}
0	0	Q^n
0	1	0
1	0	1
1	1	$\overline{Q^n}$

A partir de ellas podemos obtener la ecuación de transición para el biestable JK:

		Q^{n+1}			
		00	01	11	10
J	0		1		
	1	1	1		1

$$Q^{n+1} = J\overline{Q^n} + \overline{K}Q^n$$

Biestable T

El biestable tipo T es una versión simplificada del biestable JK. Tal y como se observa en la figura 3.13, se obtiene directamente del JK conectando juntas las entradas J y K . La designación "T" para este biestable es consecuencia de la característica de cambio de estado de este biestable (*toggle*). Cuando $T = 1$, entonces $J = K = 1$ y el biestable cambiará de estado. Cuando $T = 0$, entonces $J = K = 0$ y el biestable permanece en el estado en el que se encontraba. Sus tablas de transición son:

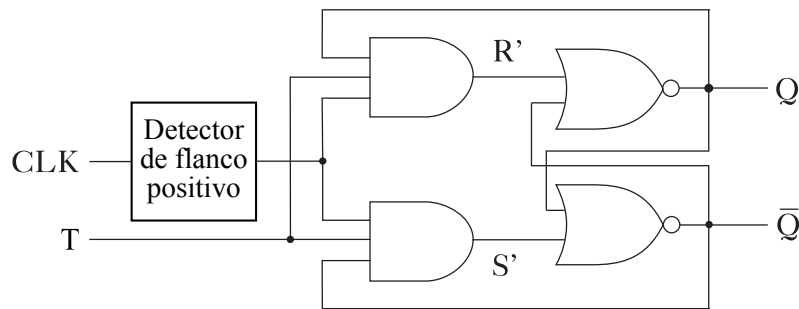


Figura 3.13: Biestable tipo T sincronizado a flanco positivo.

CLK	T	Q^{n+1}	T	Q^n	Q^{n+1}
x	x	Q^n	0	0	0
↑	0	Q^n	0	1	1
↑	1	$\overline{Q^n}$	1	0	1
			1	1	0

La ecuación del estado siguiente (Q^{n+1}) de un biestable T en función de sus entradas actuales (T y Q^n) es:

		Q^{n+1}	
		0	1
T	0		1
	1	1	

$$Q^{n+1} = T \overline{Q^n} + \overline{T} Q^n$$

3.3. SÍNTESIS DE CIRCUITOS SECUENCIALES SÍNCRONOS

Para la síntesis de los circuitos secuenciales síncronos vamos a emplear cualquiera de los biestables sincronizados a flancos que hemos visto. Es indiferente emplear un biestable u otro con sincronismo a flanco positivo o a flanco negativo, pero sí es obligatorio que todos los biestables utilizados tengan el mismo tipo de sincronismo. Los pasos fundamentales para el diseño de un sistema secuencial síncrono o autómata síncrono son los siguientes:

1. Definición del sistema, es decir, obtención de la tabla de estados y del diagrama a partir de las especificaciones del circuito.
2. Asignación binaria, es decir, asignación de valores binarios a cada uno de los estados, entradas y salidas.

3. Construcción de la tabla de transición a partir de la tabla de estados original, sustituyendo todas las variables por sus valores binarios.
4. Calcular las tablas de transición de todas las entradas de los biestables.
5. Obtención de las ecuaciones de estado siguiente y de salida.
6. Implementación del circuito.

A continuación vamos a mostrar dos ejemplos de diseño de autómatas síncronos, correspondiendo el primero a un sistema secuencial tipo Moore y el segundo a uno Mealy.

3.3.1. Ejemplo de diseño de un autómata tipo Moore

Vamos a diseñar un circuito que reconozca una subsecuencia particular de 3 bits sin solapamiento, en este caso 101, y que produzca una salida 1 cuando tal secuencia se haya producido.

Suponemos un sistema con una entrada x que va recibiendo los bits uno a uno en cada ciclo de reloj. Cuando una subsecuencia de entradas sea 101 el sistema ha de producir una salida $y = 1$ en el siguiente ciclo de reloj al correspondiente al último 1 de entrada. En los demás instantes la salida es $y = 0$. Por ejemplo, para la siguiente secuencia de entrada, la salida es la siguiente:

$$\begin{aligned} x &\longrightarrow 01110111110111\dots \\ y &\longrightarrow 00000010000010\dots \end{aligned}$$

Vamos a considerar que la subsecuencia 101 ha de producirse sin solapamiento, es decir, que en dos subsecuencias 101 solapadas, solo es válida la primera:

$$\begin{aligned} x &\longrightarrow 10101110101010\dots \\ y &\longrightarrow 00010000010001\dots \end{aligned}$$

El primer paso es la definición del sistema y para ello tenemos que ver los estados que se precisan. Obviamente no es lo mismo tener una entrada 1 cuando las entradas anteriores han sido 10 que cuando han sido cualquier otro valor, pues en el primer caso la salida en el siguiente ciclo será 1, mientras que en el segundo será 0. El sistema por tanto ha de recordar las entradas anteriores. Ello se consigue llevando al sistema a estados diferentes. Por tanto, hemos de tener un estado, q_0 , tal que si el sistema se encuentra en él es porque hasta el momento no se ha recibido como entrada ningún bit de la subsecuencia; hemos de tener otro estado, q_1 , que nos indique que hemos recibido como entrada el primer 1 de la subsecuencia; otro estado, q_2 , que recuerde que las dos últimas entradas han sido 10. Podemos también incluir un estado (q_3) que indique que las tres últimas entradas han sido 101.

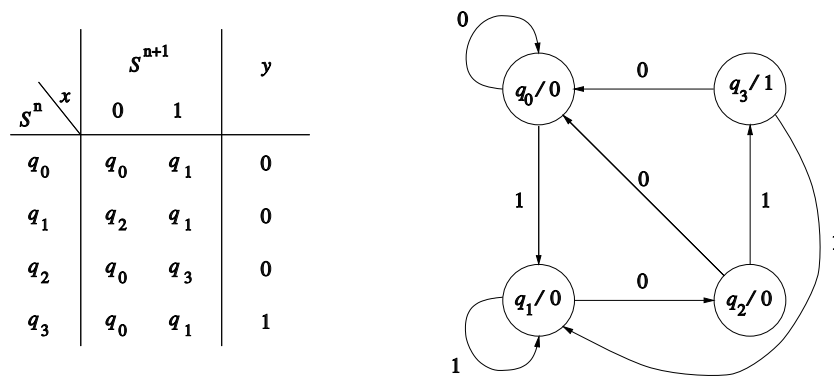


Figura 3.14: Tabla y diagrama de estados del sistema que reconoce la subsecuencia 101 sin solapamiento.

Estados	Significado
q_0	No hemos recibido ningún elemento de la subsecuencia
q_1	Hemos recibido el primer 1 de la subsecuencia
q_2	Hemos recibido en las dos últimas entradas 10
q_3	Hemos recibido en las tres últimas entradas 101

Veamos como construir el diagrama o la tabla de estados. Si estamos en el estado q_0 y recibimos como entrada 0, seguiremos en el estado q_0 ; en cambio, si recibimos un 1 pasaremos al estado q_1 . Si estamos en el estado q_1 y recibimos un 0 pasaremos al estado q_2 ; si recibimos un 1 pasaremos (nos quedaremos) en q_1 . Si estamos en el estado q_2 y recibimos un 1, hemos completado la serie de entradas 101 y pasaremos al estado q_3 ; si recibimos un 0 tendremos hasta el momento 100 que no nos sirve para nada y pasaremos al estado q_0 . Desde el estado q_3 (una vez conseguida la subsecuencia 101) empezará de nuevo el proceso y pasaremos a q_0 si la entrada es 0 ó a q_1 si la entrada es 1. En cuanto a la función de salida, todos los estados darán un valor 0, excepto q_3 que corresponde a haber recibido la subsecuencia y, por tanto, dará salida 1. El diagrama de estados y la tabla de estados se muestran en la figura 3.14

El segundo paso es la asignación binaria. Como tanto las entradas como las salidas son ya valores binarios, nos quedan por asignar solo los estados. En este caso escogemos estos valores binarios:

$Q_1 Q_0$	Estado
00	q_0
01	q_1
11	q_2
10	q_3

La tabla de transición que nos queda, con esta asignación, es:

	$Q_1^n Q_0^n \backslash x$	$Q_1^{n+1} Q_0^{n+1}$		y
		0	1	
q_0	00	00	01	0
q_1	01	11	01	0
q_2	11	00	10	0
q_3	10	00	01	1

El cuarto paso es la implementación de las variables de estado usando biestables y la obtención de las tablas de transición de estos. En este caso hemos escogido biestables JK, que son los más empleados en este tipo de diseños debido a que suelen proporcionar ecuaciones más simples por el gran número de indiferencias que aparecen en las tablas. El cálculo de los valores de las entradas J y K de los biestables lo haremos a partir de la tabla de transición del JK:

J	K	Q^{n+1}	$Q^{n+1} = J\overline{Q^n} + \overline{K}Q^n$	Q^n	Q^{n+1}	J	K
0	0	Q^n	$0 = J \cdot 1 + \overline{K} \cdot 0$	0	0	0	—
0	1	0	$1 = J \cdot 1 + \overline{K} \cdot 0$	0	1	1	—
1	0	1	$0 = J \cdot 0 + \overline{K} \cdot 1$	1	0	—	1
1	1	$\overline{Q^n}$	$1 = J \cdot 0 + \overline{K} \cdot 1$	1	1	—	0

Sabiendo el valor del estado actual y el del estado al que queremos llegar, podemos calcular los valores necesarios de J y K con la tabla de la derecha. Por ejemplo, para $Q_1^n Q_0^n = 00$ y $x = 0$, Q_1^{n+1} será 0 y, por tanto, han de ser $J_1 = 0$ y $K_1 = \text{—}$. Aplicando este método obtenemos las tablas de las funciones J y K de los dos biestables que se pueden ver en la figura 3.15, donde hemos señalado los implicantes correspondientes. También mostramos la tabla para la salida y .

El quinto paso es la obtención de las ecuaciones de estado siguiente y de salida. Las expresiones resultantes son las siguientes:

$$\begin{aligned} J_1 &= Q_0^n \overline{x} & K_1 &= \overline{Q_0^n} + \overline{x} \\ J_0 &= x & K_0 &= Q_1^n \\ y &= Q_1^n \overline{Q_0^n} \end{aligned}$$

En la figura 3.16 vemos la implementación del diseño. Para finalizar, podríamos comprobar algunos casos:

- Supongamos que el sistema se encuentra en el estado q_0 ($Q_1^n Q_0^n = 00$) y que la entrada es $x = 0$. En este caso la salida es $y = 0$, $J_1 = 0$, $K_1 = 1$, $J_0 = 0$ y $K_0 = 1$, con lo cual el sistema se mantendrá en el mismo estado q_0 .
- Supongamos que el sistema se encuentra en el estado q_2 ($Q_1^n Q_0^n = 11$) y que la entrada es $x = 1$. En este caso la salida es $y = 0$, $J_1 = 0$, $K_1 = 0$, $J_0 = 1$ y $K_0 = 1$, con lo cual el sistema pasará al estado q_3 ($Q_1^{n+1} Q_0^{n+1} = 10$).
- Supongamos que el sistema se encuentra en el estado q_3 ($Q_1^n Q_0^n = 10$) y que la entrada es $x = 1$. En este caso la salida es $y = 1$, $J_1 = 0$, $K_1 = 1$, $J_0 = 1$ y $K_0 = 0$, con lo cual el sistema pasará al estado q_1 ($Q_1^{n+1} Q_0^{n+1} = 01$).

$Q_1^n Q_0^n \backslash x$	Q_1^{n+1}		J_1		K_1	
	0	1	0	1	0	1
00	0	0			—	—
01	1	0	1		—	—
11	0	1	—	—	1	
10	0	0	—	—	1	1

$Q_1^n Q_0^n \backslash x$	Q_0^{n+1}		J_0		K_0	
	0	1	0	1	0	1
00	0	1		1	—	—
01	1	1	—	—		
11	0	0	—	—	1	1
10	0	1		1	—	—

$Q_1^n Q_0^n$	y
00	0
01	0
11	0
10	1

$Q_1^n \backslash Q_0^n$	0	1
0		
1	1	

Figura 3.15: Tablas de las funciones de estado siguiente y salida del ejemplo.

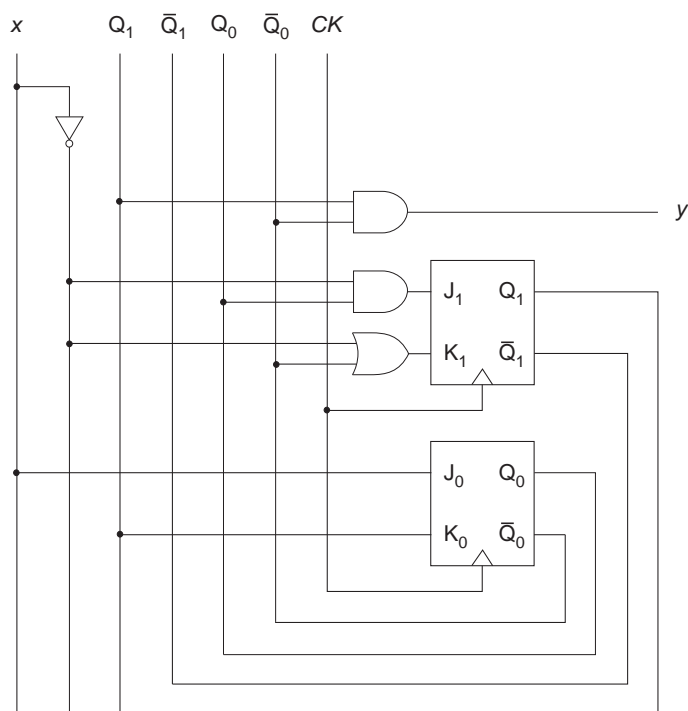


Figura 3.16: Implementación del autómata Moore del ejemplo.

3.3.2. Ejemplo de diseño de un autómata tipo Mealy

Vamos a diseñar un circuito que compare entradas serie de cuatro bits en dos canales y que nos proporcione una salida 1 cuando los cuatro bits coincidan. Por tanto, tendremos dos entradas x_1 y x_0 que van aceptando los bits uno en cada ciclo de reloj y cuando se llega al cuarto bit el sistema produce una salida $y = 1$ si los cuatro bits recibidos en los dos canales coinciden. La salida es $y = 0$ en los restantes casos. Una vez completada la lectura de los cuatro bits se comienza con otros cuatro y así sucesivamente. Por ejemplo:

$$\begin{array}{lcl}
 x_1 & \longrightarrow & 0111 \quad 0101 \quad 1110 \quad \dots \\
 x_0 & \longrightarrow & 0101 \quad 0101 \quad 1000 \quad \dots \\
 y & \longrightarrow & 0000 \quad 0001 \quad 0000 \quad \dots
 \end{array}$$

Comenzando con la definición del sistema, precisamos por un lado estados que nos recuerden el número de bits que hemos recibido 1, 2 y 3 (se pueden incluir estados que recuerden el cuarto bit recibido, pero no es necesario) y, por otro lado, estados que nos indiquen si hasta el momento los bits recibidos por ambos canales coinciden o no. Tenemos, por tanto, los siguientes estados:

$S^n \setminus x_1 x_0$	S^{n+1}				y			
	00	01	11	10	00	01	11	10
q_0	q_1	q_2	q_1	q_2	0	0	0	0
q_1	q_3	q_4	q_3	q_4	0	0	0	0
q_2	q_4	q_4	q_4	q_4	0	0	0	0
q_3	q_5	q_6	q_5	q_6	0	0	0	0
q_4	q_6	q_6	q_6	q_6	0	0	0	0
q_5	q_0	q_0	q_0	q_0	1	0	1	0
q_6	q_0	q_0	q_0	q_0	0	0	0	0

Cuadro 3.1: Tabla de estados del sistema comparador de 4 bits.

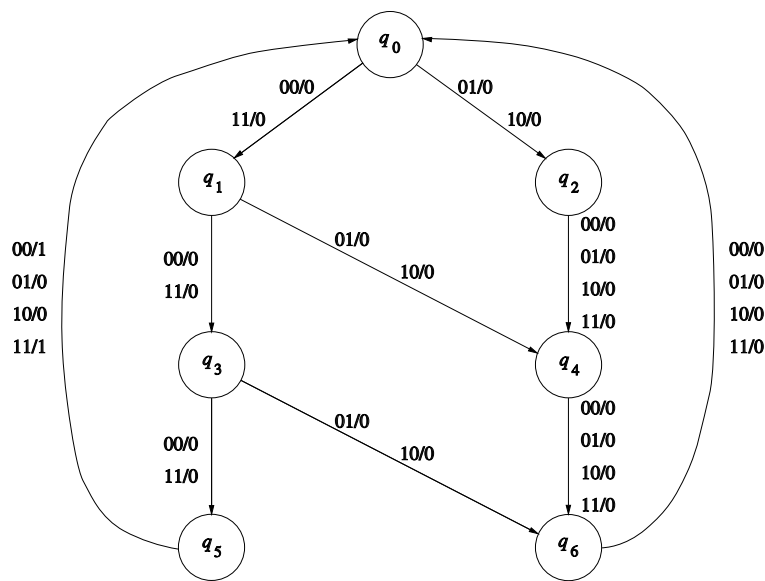


Figura 3.17: Diagrama de estados del sistema comparador de 4 bits.

Estados	Significado
q_0	Estado inicial.
q_1	Se ha recibido el primer bit. Iguales en ambos canales.
q_2	Se ha recibido el primer bit. Distintos.
q_3	Se han recibido dos bits. Los dos coinciden en ambos canales.
q_4	Se han recibido dos bits. Distintos.
q_5	Se han recibido tres bits. Los tres coinciden en ambos canales.
q_6	Se han recibido tres bits. Distintos

Una vez que hemos obtenido la tabla de estados (cuadro 3.1) el siguiente paso es la asignación binaria. De nuevo en este caso solo necesitamos asignar valores binarios a los estados. Teniendo en cuenta que necesitamos tres bits para representar siete estados, hemos elegido la siguiente asignación:

$Q_2^n Q_1^n Q_0^n$	Estado
000	q_0
001	–
010	q_1
011	q_5
100	q_4
101	q_3
110	q_2
111	q_6

Con estos valores obtenemos la siguiente tabla de transición:

$Q_2^n Q_1^n Q_0^n \setminus x_1 x_0$	$Q_2^{n+1} Q_1^{n+1} Q_0^{n+1}$				y			
	00	01	11	10	00	01	11	10
q_0	010	110	010	110	0	0	0	0
–	–	–	–	–	–	–	–	–
q_5	000	000	000	000	1	0	1	0
q_1	101	100	101	100	0	0	0	0
q_2	100	100	100	100	0	0	0	0
q_6	000	000	000	000	0	0	0	0
q_3	011	111	011	111	0	0	0	0
q_4	111	111	111	111	0	0	0	0

Necesitamos tres biestables para implementar las variables binarias de estado. Hemos escogido de nuevo biestables JK y en la figura 3.18 mostramos las tablas de las entradas J y K. En la figura 3.19 vemos la tabla de la función de salida. A partir de estas tablas obtenemos las siguientes expresiones:

$$\begin{aligned}
 J_2 &= Q_1^n \overline{Q_0^n} + \overline{Q_1^n} \overline{x_1} x_0 + \overline{Q_1^n} x_1 \overline{x_0} & K_2 &= Q_1^n Q_0^n + Q_0^n \overline{x_1} \overline{x_0} + Q_0^n x_1 x_0 \\
 J_1 &= 1 & K_1 &= 1 \\
 J_0 &= Q_2^n \overline{Q_1^n} + \overline{Q_2^n} Q_1^n \overline{x_1} \overline{x_0} + \overline{Q_2^n} Q_1^n x_1 x_0 & K_0 &= Q_1^n \\
 y &= \overline{Q_2^n} Q_0^n \overline{x_1} \overline{x_0} + \overline{Q_2^n} Q_0^n x_1 x_0
 \end{aligned}$$

Finalmente, en la figura 3.20 mostramos la implementación del diseño realizado.

3.4. CONTADORES

Un contador es un circuito secuencial cuya función es seguir una cuenta o conjunto predeterminado de estados como consecuencia de la aplicación de un tren de pulsos (reloj) en una de sus entradas. Los contadores son circuitos construidos a base de biestables sincronizados a flancos y de puertas lógicas para realizar la conexión entre ellos. Las puertas lógicas en un contador se conectan de forma que fuercen a los biestables a seguir la secuencia prescrita de estados.

$q_2^n q_1^n q_0^n$	$x_1 x_0$				Q_2^{n+1}				J_2				K_2			
	00	01	11	10	00	01	11	10	00	01	11	10	00	01	11	10
000	0	1	0	1	0	1	0	1	—	—	—	—	—	—	—	—
001	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
011	0	0	0	0	0	0	0	0	—	—	—	—	—	—	—	—
010	1	1	1	1	1	1	1	1	—	—	—	—	—	—	—	—
110	1	1	1	1	—	—	—	—	0	0	0	0	0	0	0	0
111	0	0	0	0	—	—	—	—	1	1	1	1	1	1	1	1
101	0	1	0	1	—	—	—	—	1	0	1	0	1	0	1	0
100	1	1	1	1	—	—	—	—	0	0	0	0	0	0	0	0

$q_2^n q_1^n q_0^n$	$x_1 x_0$				Q_1^{n+1}				J_1				K_1			
	00	01	11	10	00	01	11	10	00	01	11	10	00	01	11	10
000	1	1	1	1	1	1	1	1	—	—	—	—	—	—	—	—
001	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
011	0	0	0	0	—	—	—	—	1	1	1	1	1	1	1	1
010	0	0	0	0	—	—	—	—	1	1	1	1	1	1	1	1
110	0	0	0	0	—	—	—	—	1	1	1	1	1	1	1	1
111	0	0	0	0	—	—	—	—	1	1	1	1	1	1	1	1
101	1	1	1	1	1	1	1	1	—	—	—	—	—	—	—	—
100	1	1	1	1	1	1	1	1	—	—	—	—	—	—	—	—

$q_2^n q_1^n q_0^n$	$x_1 x_0$				Q_0^{n+1}				J_0				K_0			
	00	01	11	10	00	01	11	10	00	01	11	10	00	01	11	10
000	0	0	0	0	0	0	0	0	—	—	—	—	—	—	—	—
001	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
011	0	0	0	0	—	—	—	—	1	1	1	1	1	1	1	1
010	1	0	1	0	1	0	1	0	—	—	—	—	—	—	—	—
110	0	0	0	0	0	0	0	0	—	—	—	—	—	—	—	—
111	0	0	0	0	—	—	—	—	1	1	1	1	1	1	1	1
101	1	1	1	1	—	—	—	—	0	0	0	0	0	0	0	0
100	1	1	1	1	1	1	1	1	—	—	—	—	—	—	—	—

Figura 3.18: Mapas de las funciones de estado siguiente del sistema comparador.

$Q_2^n Q_1^n Q_0^n$	$x_1 x_0$			
	00	01	11	10
000	0	0	0	0
001	—	—	—	—
011	1	0	1	0
010	0	0	0	0
110	0	0	0	0
111	0	0	0	0
101	0	0	0	0
100	0	0	0	0

Figura 3.19: Mapa de la función de salida del sistema comparador.

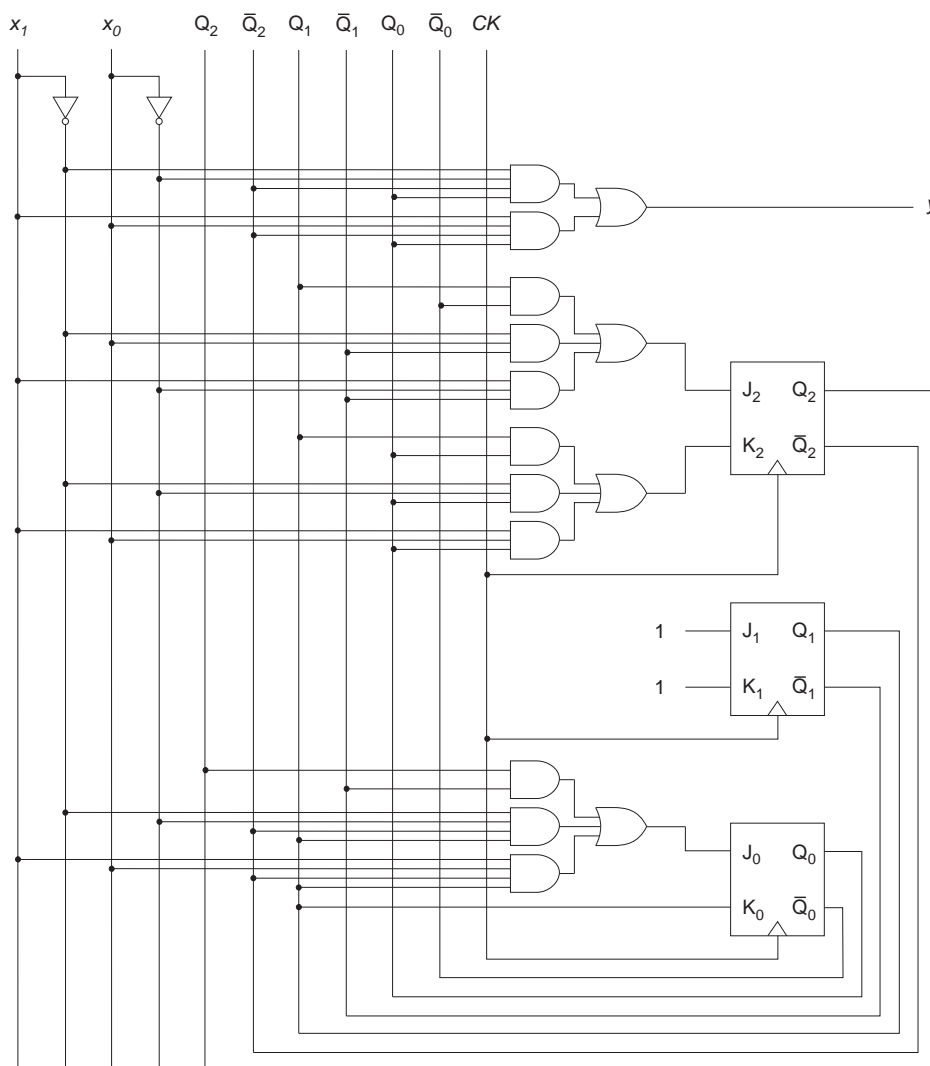


Figura 3.20: Implementación del autómata Mealy del ejemplo.

Dado que cada biestable es capaz de almacenar 2 estados, un sistema de n biestables podrá codificar 2^n estados (números de cuenta) diferentes. El paso del estado o número de cuenta actual al estado siguiente se realiza en sincronismo con la señal de reloj, de tal forma que el contador va avanzando por la secuencia de estados hasta que termina de completar un ciclo, es decir, una secuencia de conteo particular. Una vez terminado el ciclo, vuelve al estado de partida y comienza un nuevo ciclo. Como cada estado solo posee un estado siguiente, es fácil deducir que todos los estados que recorre un contador en un ciclo o secuencia de conteo son diferentes. Si el número de estados diferentes que recorre es k , se habla de un contador módulo k . El número máximo de estados posibles es 2^n , por lo que $k \leq 2^n$. Un contador se denomina binario si $k = 2^n$.

3.4.1. Contadores binarios

Un contador, como todo circuito secuencial, se puede implementar como un autómata, aunque es un autómata muy especial, puesto que no posee entradas (cada estado solo posee un único estado siguiente al que accede cuando existe una variación o flanco en la señal de reloj), y no posee salidas (la salida de un contador es su propio estado interno, es decir, el valor almacenado en sus biestables). De esta manera podemos utilizar el método de diseño de sistemas secuenciales que ya hemos estudiado. Veamos un ejemplo y diseñemos un contador binario ascendente módulo 8, es decir, un contador que siga la secuencia $\{\dots, 0, 1, 2, 3, 4, 5, 6, 7, \dots\}$. La tabla de transiciones de estado del contador será:

Q_2^n	Q_1^n	Q_0^n	Q_2^{n+1}	Q_1^{n+1}	Q_0^{n+1}
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	1	0	1
1	0	1	1	1	0
1	1	0	1	1	1
1	1	1	0	0	0

Si utilizamos biestables JK, la tabla de las entradas J y K de cada biestable es la siguiente:

Q_2^n	Q_1^n	Q_0^n	Q_2^{n+1}	Q_1^{n+1}	Q_0^{n+1}	J_2	K_2	J_1	K_1	J_0	K_0
0	0	0	0	0	1	0	-	0	-	1	-
0	0	1	0	1	0	0	-	1	-	-	1
0	1	1	1	0	0	1	-	-	1	-	1
0	1	0	0	1	1	0	-	-	0	1	-
1	1	0	1	1	1	-	0	-	0	1	-
1	1	1	0	0	0	-	1	-	1	-	1
1	0	1	1	1	0	-	0	1	-	-	1
1	0	0	1	0	1	-	0	0	-	1	-

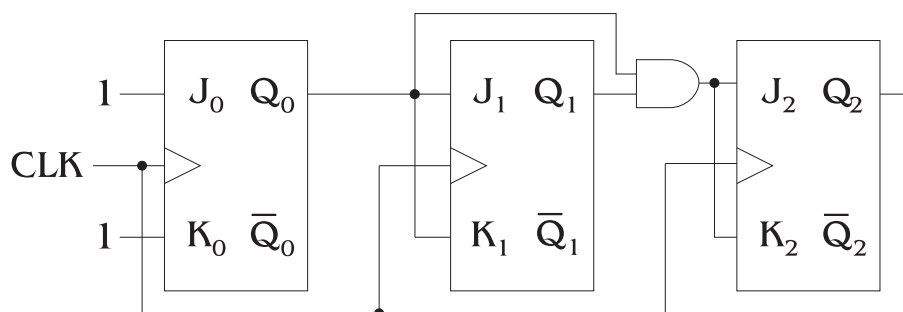


Figura 3.21: Contador binario módulo 8.

Solo nos resta encontrar las expresiones de las entradas J y K de cada biestable en función del estado actual. Para ello podemos minimizar las funciones de la tabla anterior utilizando diagramas de Karnaugh:

		J_2			
		Q_0^n	Q_1^n	Q_2^n	Q_0^n
Q_2^n	0	0	0	1	0
Q_2^n	1	-	-	-	-

		J_1			
		Q_0^n	Q_1^n	Q_2^n	Q_0^n
Q_2^n	0	0	1	-	-
Q_2^n	1	0	1	-	-

		J_0			
		Q_0^n	Q_1^n	Q_2^n	Q_0^n
Q_2^n	0	1	-	-	1
Q_2^n	1	1	-	-	1

		K_2			
		Q_0^n	Q_1^n	Q_2^n	Q_0^n
Q_2^n	0	-	-	-	-
Q_2^n	1	0	0	1	0

		K_1			
		Q_0^n	Q_1^n	Q_2^n	Q_0^n
Q_2^n	0	-	-	1	0
Q_2^n	1	-	-	1	0

		K_0			
		Q_0^n	Q_1^n	Q_2^n	Q_0^n
Q_2^n	0	-	1	1	-
Q_2^n	1	-	1	1	-

Estas ecuaciones se pueden generalizar para secuencias de conteo mayores y para un número mayor de biestables de la siguiente forma:

$$\begin{aligned}
 T_0 &= J_0 = K_0 = 1 \\
 T_1 &= J_1 = K_1 = Q_0^n \\
 T_2 &= J_2 = K_2 = Q_1^n Q_0^n \\
 T_3 &= J_3 = K_3 = Q_2^n Q_1^n Q_0^n \\
 T_4 &= J_4 = K_4 = Q_3^n Q_2^n Q_1^n Q_0^n \\
 &\dots
 \end{aligned}$$

El circuito resultante para este contador se puede ver en la figura 3.21.

Se puede plantear el problema inverso de tomar como partida el esquema de un contador ya diseñado y obtener su secuencia de conteo. Leyendo las conexiones del contador obtenemos las expresiones de las Js y las Ks de cada biestable en función del estado actual del contador (Q^n). Partiendo de la cuenta 000 (u otros valores concretos de las Q^n) y

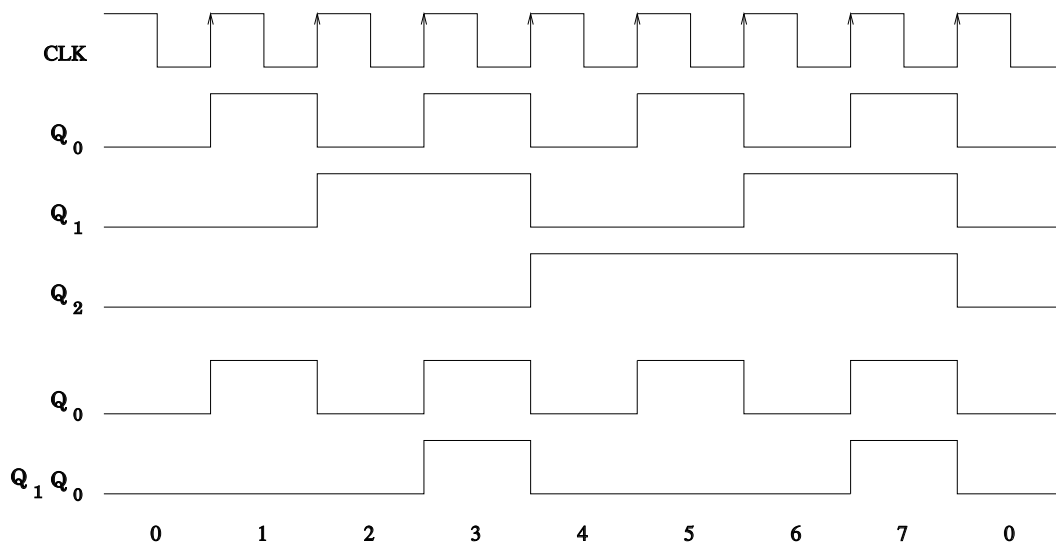


Figura 3.22: Cronograma del contador binario módulo 8.

usando las ecuaciones anteriores obtenemos los valores de las J_i y K_i . Dados estos valores se determina a partir de la Tabla de Verdad del JK los nuevos valores de los Q^{n+1} , que son los correspondientes a la cuenta siguiente. Usando estos nuevos valores de cuenta (valores de Q^n) se recalculan los nuevos valores de J_i y K_i y así sucesivamente hasta completar la secuencia de conteo, es decir, hasta que se repite un estado.

Podemos ver el funcionamiento de este proceso aplicándolo al contador que hemos diseñado anteriormente (ver figura 3.21). Así obtenemos las funciones de onda de la figura 3.22.

3.4.2. Contadores no binarios

Hasta ahora hemos construido contadores binarios, esto, es con n biestables la secuencia que se ha implementado es $\{\dots, 0, 1, 2, \dots, 2^n-1, \dots\}$. Sin embargo, cabe la posibilidad de generar secuencias de conteo que no sigan el orden natural o que el número de cuentas sea menor que 2^n . En general, un contador que realice k cuentas distintas recibe el nombre de contador módulo k .

El método de diseño que hemos expuesto en el apartado anterior para contadores binarios es un método general que se puede aplicar para cualquier tipo de cuenta y se puede utilizar cualquier tipo de biestable (JK, RS, T y D) para su implementación.

Como ejemplo de esta afirmación, construyamos a partir de biestables tipo T un contador que siga la secuencia $\{\dots, 0, 7, 5, 3, 4, 2, \dots\}$. El primer paso es obtener la tabla de transiciones de estado del biestable T:

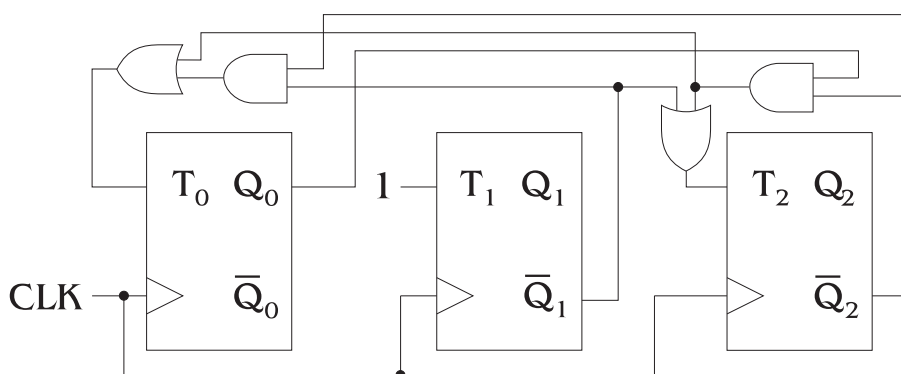


Figura 3.23: Contador de secuencia $\{\dots, 0, 7, 5, 3, 4, 2, \dots\}$.

T	Q^{n+1}	$Q^{n+1} = T\bar{Q}^n + \bar{T}Q^n$	Q^{n+1}	Q^n	T
0	Q^n	$0 = T1 + \bar{T}0$	0	0	0
1	\bar{Q}^n	$1 = T1 + \bar{T}0$	0	1	1
		$0 = T0 + \bar{T}1$	1	0	1
		$1 = T0 + \bar{T}1$	1	1	0

El segundo paso, y último, es la obtención de las expresiones de las T_i a partir de la cuenta actual y de la cuenta siguiente. En las cuentas no usadas ponemos indiferencias. El resultado final será:

Q_2^n	Q_1^n	Q_0^n	Q_2^{n+1}	Q_1^{n+1}	Q_0^{n+1}	T_2	T_1	T_0
0	0	0	1	1	1	1	1	1
0	0	1	-	-	-	-	-	-
0	1	1	1	0	0	1	1	1
0	1	0	0	0	0	0	1	0
1	1	0	-	-	-	-	-	-
1	1	1	1	0	1	0	1	0
1	0	1	0	1	1	1	1	0
1	0	0	0	1	0	1	1	0

Para obtener las expresiones de las T_i en función de las Q_i^n minimizamos mediante diagramas de Karnaugh.

T_2	T_1	T_0																																													
<table border="1" style="border-collapse: collapse; margin: auto;"> <tr> <td style="padding: 5px;">$Q_2^n \backslash Q_1^n Q_0^n$</td> <td style="padding: 5px;">00</td> <td style="padding: 5px;">01</td> <td style="padding: 5px;">11</td> <td style="padding: 5px;">10</td> </tr> <tr> <td style="padding: 5px;">0</td> <td style="padding: 5px;">1</td> <td style="padding: 5px;">-</td> <td style="padding: 5px;">1</td> <td style="padding: 5px;">0</td> </tr> <tr> <td style="padding: 5px;">1</td> <td style="padding: 5px;">1</td> <td style="padding: 5px;">1</td> <td style="padding: 5px;">0</td> <td style="padding: 5px;">-</td> </tr> </table>	$Q_2^n \backslash Q_1^n Q_0^n$	00	01	11	10	0	1	-	1	0	1	1	1	0	-	<table border="1" style="border-collapse: collapse; margin: auto;"> <tr> <td style="padding: 5px;">$Q_2^n \backslash Q_1^n Q_0^n$</td> <td style="padding: 5px;">00</td> <td style="padding: 5px;">01</td> <td style="padding: 5px;">11</td> <td style="padding: 5px;">10</td> </tr> <tr> <td style="padding: 5px;">0</td> <td style="padding: 5px;">1</td> <td style="padding: 5px;">-</td> <td style="padding: 5px;">1</td> <td style="padding: 5px;">1</td> </tr> <tr> <td style="padding: 5px;">1</td> <td style="padding: 5px;">1</td> <td style="padding: 5px;">1</td> <td style="padding: 5px;">-</td> <td style="padding: 5px;">1</td> </tr> </table>	$Q_2^n \backslash Q_1^n Q_0^n$	00	01	11	10	0	1	-	1	1	1	1	1	-	1	<table border="1" style="border-collapse: collapse; margin: auto;"> <tr> <td style="padding: 5px;">$Q_2^n \backslash Q_1^n Q_0^n$</td> <td style="padding: 5px;">00</td> <td style="padding: 5px;">01</td> <td style="padding: 5px;">11</td> <td style="padding: 5px;">10</td> </tr> <tr> <td style="padding: 5px;">0</td> <td style="padding: 5px;">1</td> <td style="padding: 5px;">-</td> <td style="padding: 5px;">1</td> <td style="padding: 5px;">0</td> </tr> <tr> <td style="padding: 5px;">1</td> <td style="padding: 5px;">0</td> <td style="padding: 5px;">0</td> <td style="padding: 5px;">0</td> <td style="padding: 5px;">-</td> </tr> </table>	$Q_2^n \backslash Q_1^n Q_0^n$	00	01	11	10	0	1	-	1	0	1	0	0	0	-
$Q_2^n \backslash Q_1^n Q_0^n$	00	01	11	10																																											
0	1	-	1	0																																											
1	1	1	0	-																																											
$Q_2^n \backslash Q_1^n Q_0^n$	00	01	11	10																																											
0	1	-	1	1																																											
1	1	1	-	1																																											
$Q_2^n \backslash Q_1^n Q_0^n$	00	01	11	10																																											
0	1	-	1	0																																											
1	0	0	0	-																																											

Es decir, $T_2 = \bar{Q}_1^n + \bar{Q}_2^n Q_0^n$, $T_1 = 1$ y $T_0 = \bar{Q}_2^n \bar{Q}_1^n + \bar{Q}_2^n Q_0^n$. El circuito resultante se puede ver en la figura 3.23 y un cronograma de sus salidas está en la figura 3.24.

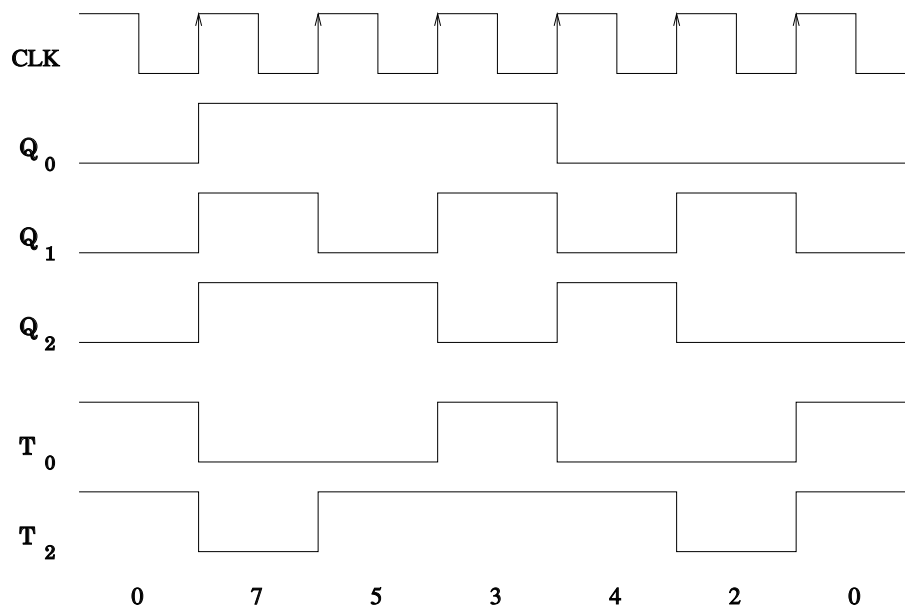


Figura 3.24: Cronograma de las salidas del contador de secuencia $\{\dots, 0, 7, 5, 3, 4, 2, \dots\}$.

Una vez construido el contador podemos comprobar que la secuencia de conteo que realmente sigue es la que hemos diseñado. Cabe preguntarse que pasará si el contador cae, debido a un mal funcionamiento o comportamiento anómalo de sus componentes físicos, en alguna de las cuentas no utilizadas, en este caso 1 y 6:

1. Si el contador cae en $Q_2^n Q_1^n Q_0^n = 001$ (cuenta 1), sustituyendo en las ecuaciones tendremos que $T_2 = 1$, $T_1 = 1$ y $T_0 = 1$ con lo cual el contador pasará al estado $Q_2^{n+1} Q_1^{n+1} Q_0^{n+1} = 110$ (cuenta 6).
2. Si el contador cae en $Q_2^n Q_1^n Q_0^n = 110$ (cuenta 6), sustituyendo en las ecuaciones obtenemos $T_2 = 0$, $T_1 = 1$ y $T_0 = 0$ y el siguiente estado será $Q_2^{n+1} Q_1^{n+1} Q_0^{n+1} = 100$ (cuenta 4), que ya forma parte de la secuencia, con lo que el contador seguirá contando normalmente.

Una consecuencia de poner indiferencias en los estados (cuentas) no utilizados es que, si por casualidad el contador cae en uno de esos estados, desconocemos, a priori, el estado siguiente al que pasará. Cabe la posibilidad de que el contador produzca un ciclo por el que nunca llegue a un estado utilizado o válido. En el ejemplo anterior, esto podría ocurrir si el contador se quedara en los ciclos $\{\dots, 1, 6, \dots\}$, $\{\dots, 1, \dots\}$, o $\{\dots, 6, \dots\}$. Un contador cuyos estados no utilizados tienen esta propiedad (de generar ciclos de estados o cuentas no válidas) se dice que puede bloquearse. En todo diseño de un contador no binario siempre es necesario determinar el estado siguiente de cada estado no utilizado para comprobar que no se bloquea.

Una alternativa es la no utilización de indiferencias en el proceso de diseño del contador, colocando en las cuentas no utilizadas estados siguientes concretos (no necesariamente el mismo). Ello complicará el diseño, ya que eliminará indiferencias, pero garantiza un

correcto funcionamiento del contador, aún en el caso de que, por cualquier causa, llegue a alcanzar un estado o número de cuenta no válido.

3.4.3. Contadores con carga paralela

Los contadores utilizados en sistemas digitales a menudo requieren la capacidad de almacenar un número binario inicial antes de la operación de conteo. Esta transferencia de un número de cuenta determinado al contador recibe el nombre de carga en paralelo.

En este apartado vamos a ver como realizar una carga en paralelo síncrona (en sincronismo con la señal de reloj). En el caso de una carga síncrona las únicas entradas sobre las que se pueden actuar son J y K, ya que son las únicas que son síncronas. En este apartado solo consideraremos biestables JK, pero el tratamiento es similar para los otros tipos de biestables.

Un contador con carga en paralelo, además de la señal de reloj, necesita una señal adicional que le indique cuando debe contar y cuando debe cargar el estado o principio de una nueva cuenta o ciclo. A esta señal se le suele denominar LOAD y su función es seleccionar, mediante MUXes 2 a 1, las entradas adecuadas de los biestables (J y K) en función de la tabla siguiente:

LOAD	Valores de J y K para
0	contar
1	realizar una carga en paralelo

Cuando $LOAD = 0$ se seleccionan los valores de J y K obtenidos en el proceso de diseño del contador (el contador cuenta). Cuando $LOAD = 1$ necesitamos poner en J_i y en K_i los valores adecuados para que el biestable i -ésimo cargue el valor X_i (el bit i -ésimo del estado que queremos cargar) en el siguiente pulso de reloj. Es decir, si $X_i = 0$ significa que en el biestable i -ésimo debemos poner un cero, y cuando $X_i = 1$ debemos poner un uno, independientemente del valor almacenado. Para ello tendremos que poner las entradas que se muestran en la siguiente tabla:

X_i	J_i	K_i
0	0	1
1	1	0

O lo que es lo mismo, $J_i = X_i$ y $K_i = \overline{X_i}$. Es decir, en la entrada J_0 del primer biestable el MUX 2 a 1 escoge entre 1 y X_0 , mientras que en la entrada K_0 otro MUX escoge entre 1 y $\overline{X_0}$. Algo similar ocurre en el resto de los biestables. El circuito completo de un contador binario módulo 8 con carga en paralelo se muestra en la figura 3.25.

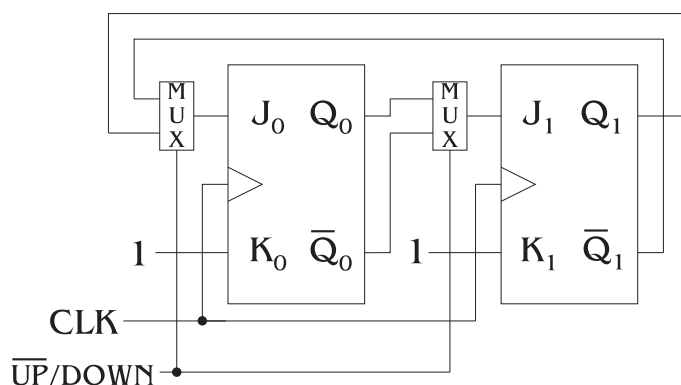


Figura 3.26: Contador reversible $\{\dots, 0, 1, 2, \dots / \dots, 2, 1, 0, \dots\}$.

$\overline{UP/DOWN}$. Si $\overline{UP/DOWN} = 0$ la cuenta será ascendente (\overline{UP}) y si $\overline{UP/DOWN} = 1$ la cuenta será descendente (\overline{DOWN}). El circuito final es el de la figura 3.26.

3.4.5. Diseño de contadores a partir de contadores comerciales

Existen varios tipos de contadores en circuitos integrados que cuentan en BCD ($\{\dots, 0, 1, 2, \dots, 9, 0, \dots\}$) o en binario ($\{\dots, 0, 1, 2, \dots, 15, 0, \dots\}$). Estos contadores pueden ser reversibles o no, tener la posibilidad de carga en paralelo (LOAD), borrado (CLEAR) y conteo (COUNT) (cuando el conteo es 0, el contador está parado y mantiene el valor de la cuenta). Lo más usual es que el borrado se realice mediante la activación de los Clear de los biestables (entradas asíncronas) y que, por tanto, sea una señal asíncrona. La carga en paralelo suele ser síncrona.

Además del valor de la cuenta, suelen tener como salida una señal de ACARREO, que se pone a 1 durante la última cuenta del contador (en la cuenta 9 si el contador es BCD o en la 15 si es binario) si está activada la señal de conteo, siendo 0 en los demás instantes.

En la figura 3.27 se muestra el diagrama interno de un contador binario de 4 bits con carga en paralelo síncrona, borrado asíncrono y señal de conteo.

Este contador se puede utilizar como bloque fundamental para diseñar otros contadores a partir de él. Por ejemplo, se puede construir un contador módulo 6 de 4 formas distintas (ver figura 3.28), según se use la carga en paralelo síncrona o el borrado asíncrono.

(a) Se detecta el estado 0110 (6) y se impide la cuenta de este estado actuando sobre el CLEAR (señal asíncrona), el cual pone inmediatamente a cero el contador.

(b) Se detecta el estado 0101 (5) y se actúa sobre LOAD. La carga en paralelo (del 0), al ser síncrona, se realizará en el ciclo siguiente (en el siguiente flanco de reloj), por lo cual este estado se contará.

(c) Detectando el estado 1111 (15) mediante el acarreo y actuando sobre la carga en paralelo (cargando la cuenta 10).

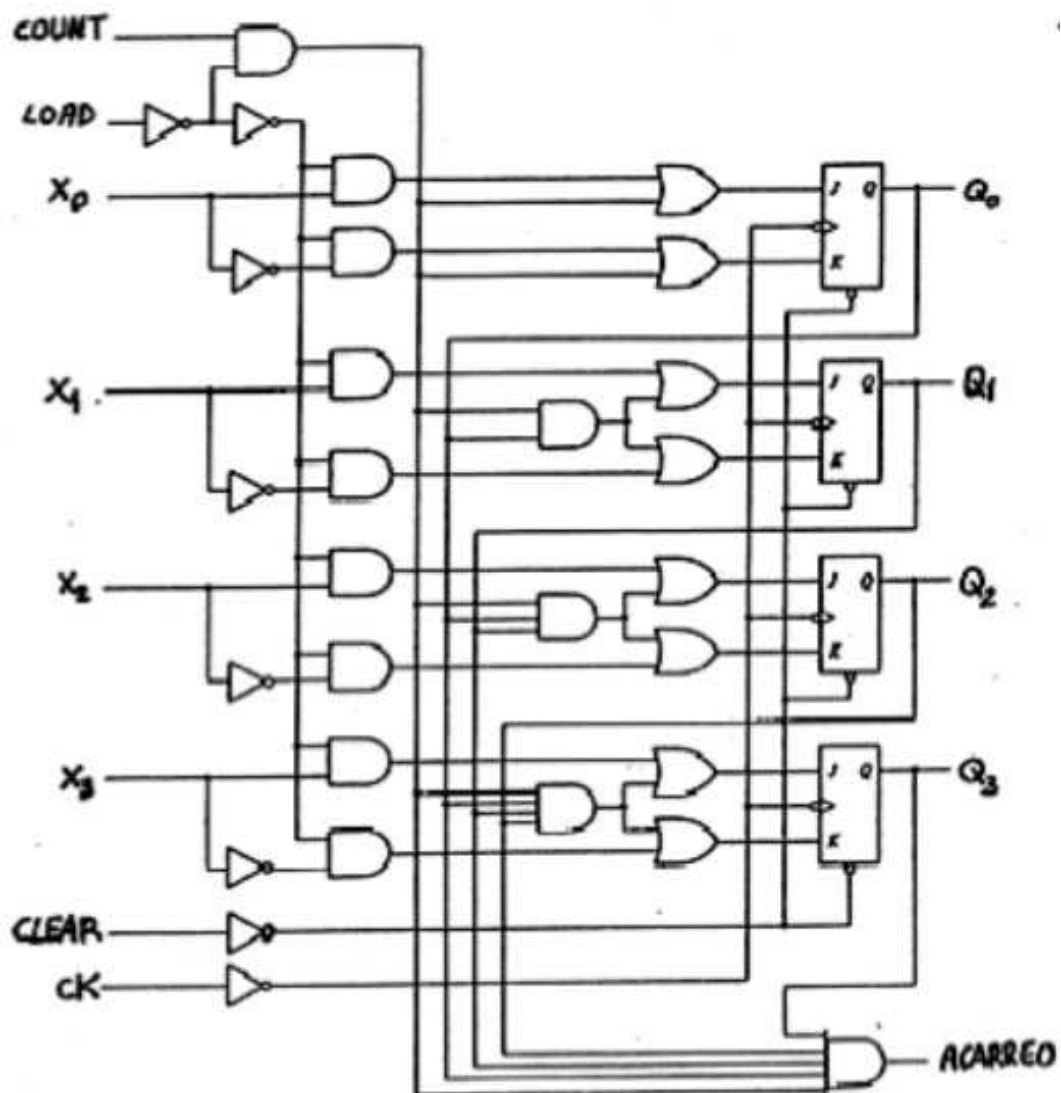


Figura 3.27: Contador binario de 4 bits con carga paralela síncrona, borrado asíncrono y señal de conteo.

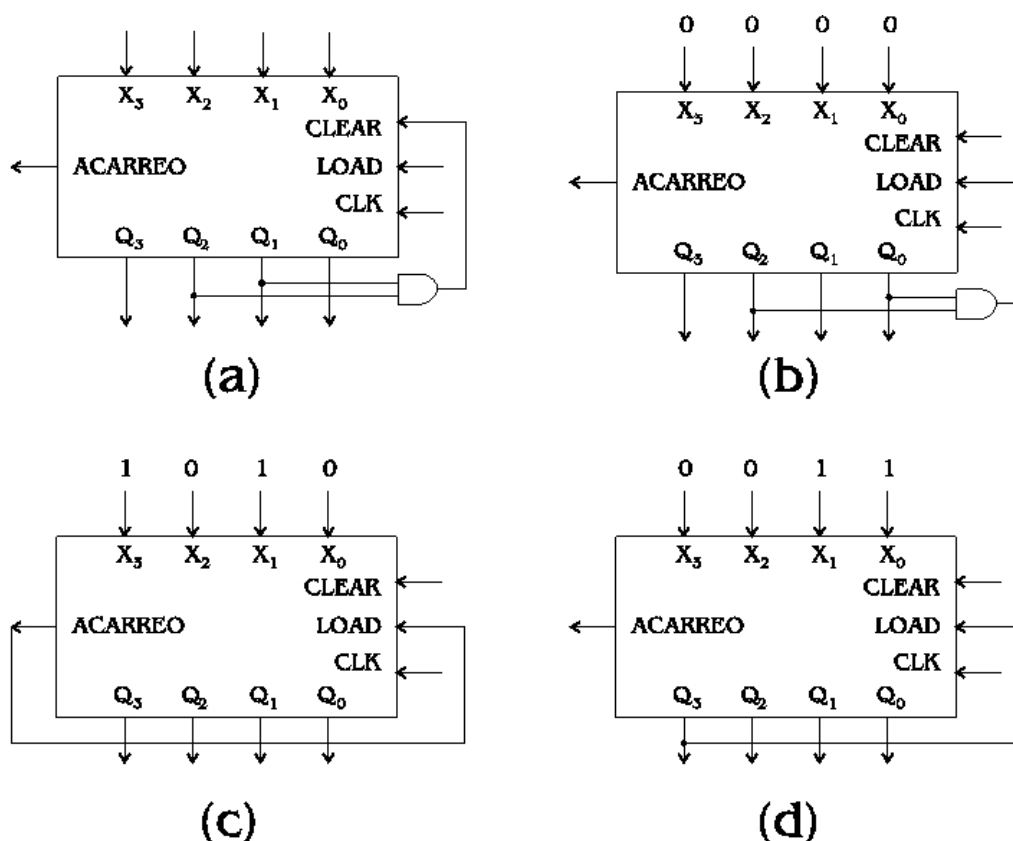


Figura 3.28: Cuatro formas de construir un contador módulo 6 a partir del contador de la figura 3.27.

(d) Detectando el estado 1000 (8) mediante Q_3 y actuando sobre la carga en paralelo para cargar el estado 3.

También se puede usar este contador u otros contadores comerciales para construir contadores de un número mayor de estados. Para ello los contadores se pueden conectar en cascada, es decir, la salida del último estado de un contador permite el cambio de estado del siguiente contador. Para esta conexión en cascada podemos usar la salida de acarreo o bien mediante una conexión asíncrona, (figura 3.29a) o bien mediante una conexión síncrona (figura 3.29b).

3.5. REGISTROS

Un registro tiene como función primordial el almacenar información. La diferencia entre un registro y un biestable es que éste solo puede almacenar un bit, mientras que un registro es capaz de almacenar n bits. Un registro consta, básicamente, de un conjunto de celdas de almacenamiento binarias (generalmente constituidas por biestables tipo D disparados a flancos) más un conjunto de puertas encargadas de realizar su conexión. A continuación vamos a ver algunos tipos de registros.

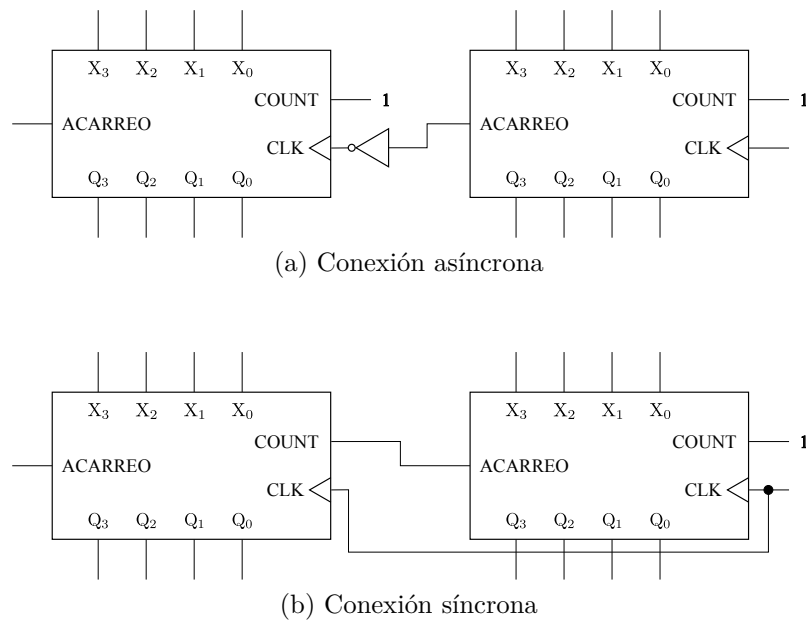


Figura 3.29: Contador módulo 256 construido con conexión en cascada de dos contadores módulo 16.

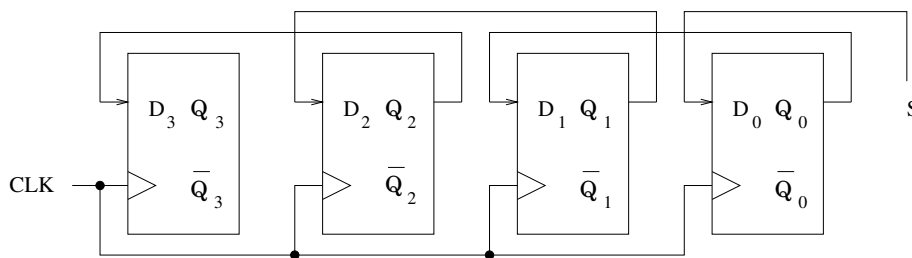


Figura 3.30: Registro de desplazamiento a la izquierda.

3.5.1. Registros de desplazamiento

La forma más sencilla de construir un registro de desplazamiento a la izquierda de n bits es conectar n biestables D uniendo la salida de cada uno a la entrada del siguiente en la forma indicada en la figura 3.30, es decir, $D_i = Q_{i-1}$. El registro solo tendrá una entrada externa, la del biestable menos significativo, D_0 o S , por la que entrarán los datos al registro (uno en cada ciclo de reloj).

El registro que hemos diseñado realiza un desplazamiento de los bits hacia la izquierda, pero si realizamos a la inversa la conexión de los biestables podemos construir un registro de desplazamiento a la derecha. La salida de cada uno de los biestables se conectará a la entrada del situado a su derecha, es decir, $D_i = Q_{i+1}$. La entrada de los datos se realiza por el biestable más significativo, D_{n-1} o S , tal y como se ve en la figura 3.31. En cada ciclo de reloj van entrando nuevos bits y el contenido del registro se va desplazando a cada ciclo de reloj de biestable en biestable, como se observa en la figura 3.32. Destacar que en este tipo de circuitos por todos los biestables circula la misma información, con

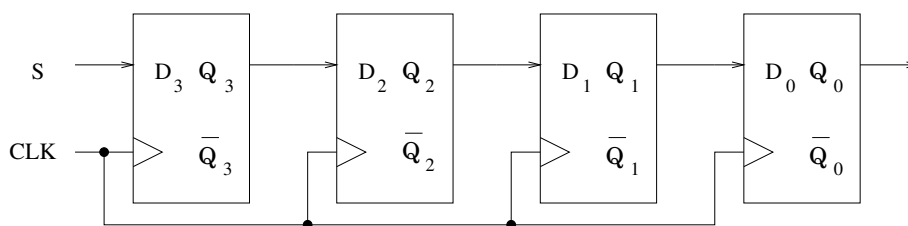


Figura 3.31: Registro de desplazamiento a la derecha.

		Q_3	Q_2	Q_1	Q_0
ciclo 0	d_0	-	-	-	-
ciclo 1	d_1	d_0	-	-	-
ciclo 2	d_2	d_1	d_0	-	-
ciclo 3	d_3	d_2	d_1	d_0	-
ciclo 4		d_3	d_2	d_1	d_0

Figura 3.32: Funcionamiento de un registro de desplazamiento hacia la derecha.

un retardo de un ciclo de reloj respecto de su biestable vecino.

3.5.2. Registro bidireccional con carga en paralelo y borrado

Incluyendo los dos tipos de conexiones entre los biestables D podemos construir un registro con posibilidad de ser desplazado tanto hacia la derecha como hacia la izquierda. La selección de una u otra operación se realizará mediante MUXes en la entrada D de cada biestable. La carga en paralelo y el borrado se pueden añadir usando MUXes 4 a 1, conectando a una de las entradas un dato procedente del exterior (para la carga en paralelo) o un "0" para el borrado. Ambas operaciones, realizadas de esta forma, son síncronas.

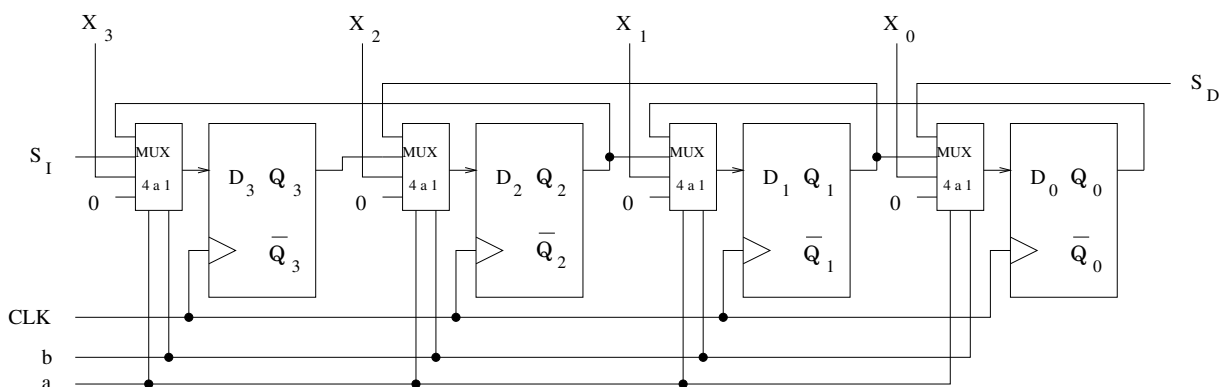


Figura 3.33: Registro bidireccional con carga en paralelo y borrado.

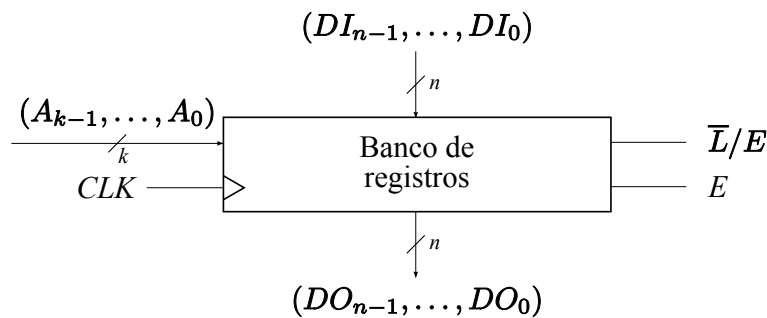


Figura 3.34: Banco de registros.

Selección del MUX		Acción
a	b	
0	0	Desplazamiento a la izquierda
0	1	Desplazamiento a la derecha
1	0	Carga en paralelo síncrona
1	1	Borrado síncrono

La disposición de los MUXes, sus entradas y sus conexiones se pueden ver en la figura 3.33.

3.5.3. Bancos de registros

En ocasiones, resulta conveniente agrupar en un solo circuito un conjunto de registros tales que el procesamiento de la información que almacenan solo afectará a uno de ellos, a lo sumo, en cada ciclo de reloj. Un conjunto de registros organizado de este modo responde al esquema de la figura 3.34, que representa un *banco de registros*, formado por 2^k registros de n bits cada uno. Las k entradas (A_{k-1}, \dots, A_0) , denominadas *líneas de dirección*, permiten seleccionar el registro que se va a leer o escribir. La entrada \bar{L}/E permite seleccionar el tipo de operación a realizar: $\bar{L}/E = 0$ significa lectura, mientras que $\bar{L}/E = 1$ significa escritura. En las operaciones de lectura el contenido del registro seleccionado aparecerá sobre las salidas (DO_{n-1}, \dots, DO_0) , denominadas *líneas de salida de datos*. Análogamente, en las operaciones de escritura el dato a escribir debe introducirse por las entradas (DI_{n-1}, \dots, DI_0) , denominadas *líneas de entrada de datos*, y el dato se almacenará en el registro seleccionado en coincidencia con el flanco activo de la señal de reloj. Finalmente la señal E actúa como una señal de habilitación, de tal forma que si toma valor 0 entonces el funcionamiento del dispositivo queda inhibido.

Una posible implementación de un banco de 4 registros de 4 bits aparece en la figura 3.35, donde las dos líneas de dirección, A_1 y A_0 , están conectadas a la entrada de un decodificador 2 a 4, habilitado por la señal de *strobe* E . La escritura en cada registro está controlada por una de las salidas del decodificador, de tal forma que la señal de carga del registro i solo se activará si $\bar{L}/E = 1$, $E = 1$, y la entrada del decodificador toma el valor i . En este caso será el valor presente en las líneas de entradas de datos, DI_{3-0} , el que se

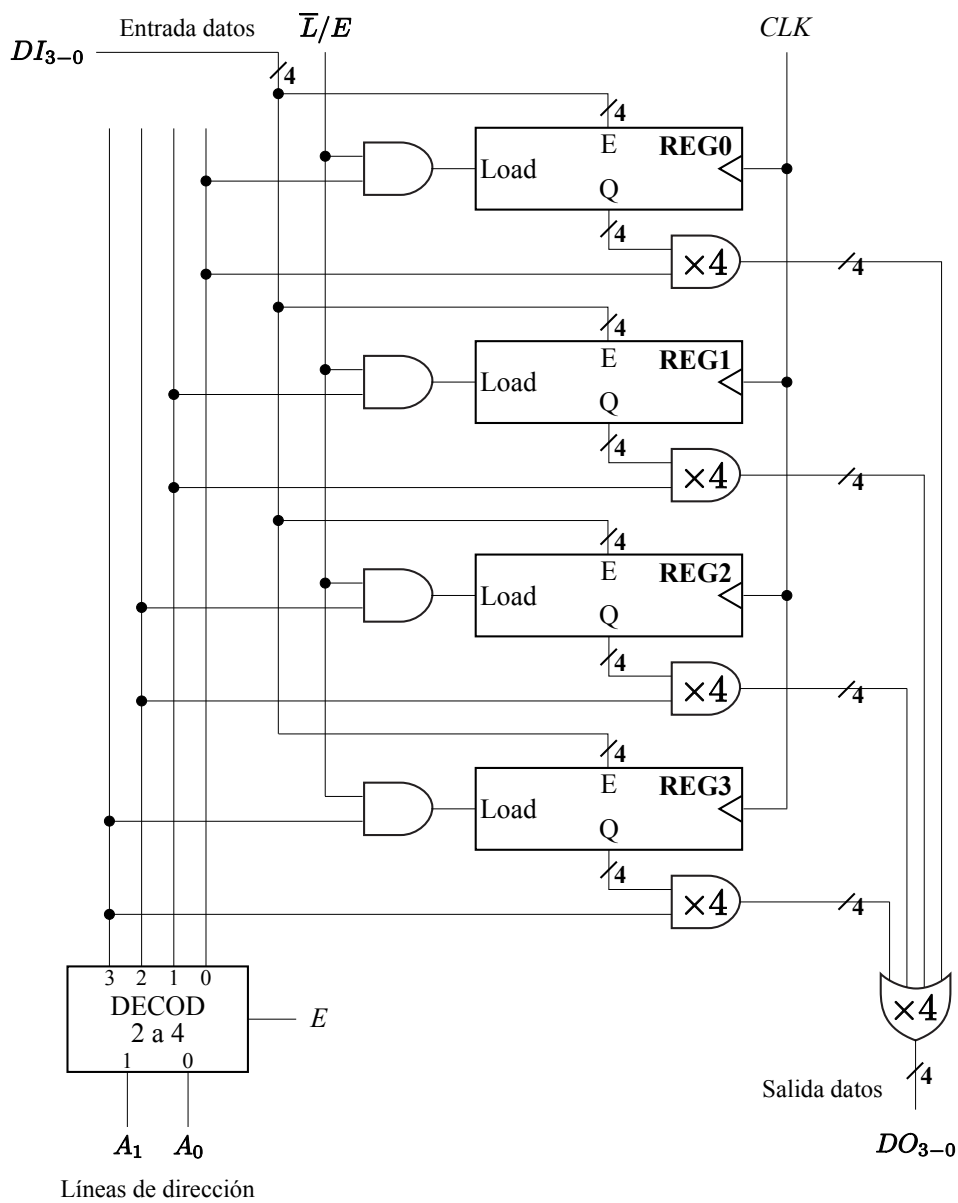


Figura 3.35: Implementación de un banco de 4 registros de 4 bits.

cargará en el registro. Por lo que respecta a la lectura, las salidas de los cuatro registros pasan a través de un conjunto de puertas AND, de tal forma que todas ellas quedan transformadas en vectores de ceros, excepto las correspondientes al registro seleccionado, que a través de un conjunto de puertas OR alcanza las líneas de salida de datos, DO_{3-0} .

3.5.4. Aplicaciones de los registros

Como ya sabemos, desplazar un número binario i veces a la izquierda equivale a multiplicarlo por 2^i . Igualmente, desplazar un número binario hacia la derecha i posiciones equivale a dividirlo entre 2^i (quedándonos únicamente con la parte entera de dicha división). Además de multiplicar y dividir números entre potencias de 2, los registros de desplazamiento se utilizan, entre otras cosas, para la conversión de datos paralelo/serie y serie/paralelo (imprescindible en las comunicaciones serie). A continuación vamos a ver como se emplean los registros de desplazamiento en tales aplicaciones.

La manipulación de datos constituidos por varios bits puede realizarse de dos formas distintas: en modo serie o en modo paralelo. Se dice que un sistema digital opera en modo serie cuando la información se transfiere y manipula bit a bit. Por ejemplo, cuando el contenido de un registro se transfiere a otro desplazando los bits de un registro al siguiente, un bit en cada ciclo de reloj. Para prevenir la pérdida de datos, se hace recircular el contenido del primero de los registros. Si el registro es de n bits, serán necesarios n ciclos de reloj para la transferencia.

En el modo paralelo, la información se transfiere y manipula con todos sus bits a la vez. Por ejemplo, en una transferencia entre dos registros en modo paralelo todos los bits se transfieren del primer registro al segundo en un solo ciclo de reloj, mediante una carga en paralelo. Si los datos son de n bits, los registros han de tener n salidas y n entradas.

Conversión de datos paralelo a serie

Los datos entran en el registro en un ciclo de reloj, todos los bits a la vez, mediante una carga en paralelo. Los bits de salida se toman en la salida Q del último biestable D (el menos significativo) uno en cada ciclo de reloj, a la vez que se va desplazando el registro hacia la derecha.

Conversión de datos serie a paralelo

Los datos entran por la entrada serie del registro, un bit en cada ciclo de reloj. Una vez completada la carga de todos los bits, se toman como líneas de salida las Q de cada uno de los biestables D (se leen todos los bits a la vez).

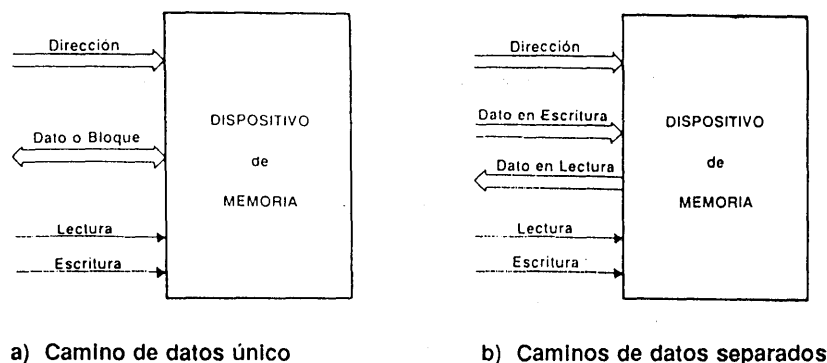


Figura 3.36: Interfaz externo de una memoria.

3.6. MEMORIAS

La memoria de un procesador digital es la parte encargada de almacenar la información que este necesita para su funcionamiento. Esta información puede consistir en valores utilizados en cálculos, resultados de operaciones, código de programas, etc. En general se clasifica la información que se puede almacenar en la memoria de un procesador en dos tipos: *datos* e *instrucciones*. Las operaciones que permiten las memorias son básicamente la *lectura* (o recuperación) y la *escritura* (o almacenamiento) de información en posiciones específicas. En ambas operaciones la posición de memoria que está siendo accedida es seleccionada por medio de su *dirección*.

Desde un punto de vista externo, la memoria puede verse como una caja negra (figura 3.36) cuyo interfaz con el resto del sistema está formado por varias líneas de comunicación. Estas líneas pueden ser básicamente de tres tipos:

1. *Líneas de datos*, utilizadas para el intercambio de la información almacenada en la memoria con el resto del sistema. Existen memorias en las que las mismas líneas de datos son compartidas tanto para almacenar como para recuperar la información (memorias de camino de datos único), y memorias en las que cada operación se realiza mediante unas líneas de datos dedicadas (memorias con caminos de datos separados).
2. *Líneas de direcciones*, utilizadas para seleccionar la posición de la memoria a la que se desea acceder.
3. *Líneas de control*, utilizadas para indicar el tipo de operación a realizar (lectura o escritura) en la posición escogida.

La *capacidad de almacenamiento* de una memoria puede expresarse en unidades de bits, bytes o palabras, siendo el byte la unidad utilizada con mayor frecuencia. Debido a

Abreviatura	Denominación	Equivalencia	Valor aproximado
KB	Kilobyte	1KB = 2^{10} bytes	1000 bytes
MB	Megabyte	1MB = 2^{20} bytes	1 millón de bytes
GB	Gigabyte	1GB = 2^{30} bytes	1000 millones de bytes
TB	Terabyte	1TB = 2^{40} bytes	1 billón de bytes
PB	Petabyte	1PB = 2^{50} bytes	1000 billones de bytes

Cuadro 3.2: Unidades utilizadas para expresar la capacidad de almacenamiento.

la creciente capacidad de las memorias actuales los múltiplos de un byte indicados en el cuadro 3.2 son de uso común.

3.6.1. Estructura general de una memoria

Desde el punto de vista de la implementación, para que un dispositivo pueda ser utilizado como memoria debe disponer de los siguientes componentes:

1. Un *medio o soporte*, en el que se almacene la información utilizando, en la mayoría de los casos, dos estados diferentes de energía.
2. Un *transductor de escritura*, que genere la energía necesaria para forzar un estado determinado en el soporte.
3. Un *transductor de lectura*, que permite detectar el estado almacenado en el soporte.
4. Un *mecanismo de direccionamiento*, que permita escoger en el soporte la ubicación a leer o escribir.

El medio o soporte

El soporte debe cumplir las siguientes condiciones para poder almacenar información:

1. Debe diferenciar al menos dos estados estables caracterizados por una magnitud física (p.e. momento magnético, carga eléctrica, corriente, marca óptica, etc.).
2. Ha de ser posible pasar de un estado a otro mediante la aplicación de una señal externa.
3. Ha de ser posible detectar el estado existente en un instante determinado.

Los soportes pueden ser *discretos* o *continuos*, en los primeros cada bit de información es almacenado en un dispositivo físico individual (p.e. memorias de semiconductores), mientras que en los segundos los bits son almacenados en secuencia sobre un medio continuo

(p.e. cintas magnéticas). En general los soportes discretos son más caros, pero sus transductores son más sencillos y el mecanismo de direccionamiento es más rápido y simple.

Otra característica de los soportes es el tiempo que la información permanece grabada en los mismos. De acuerdo a esta característica podemos clasificarlos en las siguientes categorías:

1. *Duraderos*, denominados también medios *no volátiles*, en ellos la información permanece mientras no se realice una nueva operación de escritura (p.e. discos magnéticos).
2. *Volátiles*, la información desaparece si se interrumpe la energía de alimentación del soporte (p.e. memorias RAM).
3. *Con refresco*, la información se degrada con el paso del tiempo. En estos soportes es necesario "refrescar" periódicamente la información almacenada (p.e. memorias de condensadores).
4. *De lectura destructiva*, la lectura de la información implica su destrucción. Para mantener la información es necesario volver a escribirla después de leerla (p.e. memorias de ferritas).
5. *Permanentes*, denominados también medios de *solo lectura*, en ellos la información es siempre la misma y no puede borrarse (p.e. discos CD-ROM, memorias ROM de semiconductores). Algunos soportes permiten un número limitado de borrados y reescrituras mediante algún proceso especial (p.e. memorias EPROM y EEPROM).

Los transductores

Todo dispositivo de memoria debe disponer de transductores de lectura y escritura que permitan detectar y forzar respectivamente el estado del soporte. Estos transductores son en general elementos caros, por lo que durante el diseño es necesario alcanzar un compromiso entre su coste y la velocidad de lectura/escritura del dispositivo de memoria.

Los transductores pueden estar físicamente unidos al soporte (p.e. en las memorias de semiconductores) o ser independientes de él (p.e. cabezales de lectura/escritura en cintas o discos magnéticos).

El mecanismo de direccionamiento

Su función es la de seleccionar una posición específica en el soporte. En función del tipo de mecanismo utilizado podemos clasificar las memorias en las categorías siguientes:

1. Memorias de *direccionamiento cableado*, en ellas los transductores son fijos y el mecanismo de direccionamiento está implícitamente implementado en la conexión física entre los transductores y el soporte. La posición es escogida mediante la activación

de las señales que seleccionan el transductor correspondiente (p.e. memorias de semiconductores). A este tipo de memorias se les denomina también memorias de *acceso aleatorio* o *directo* debido a que en ellas el tiempo de acceso es independiente de la posición accedida.

2. Memorias de *direccionamiento por bloques*, que tienen transductores independientes del soporte y en las que el mecanismo de direccionamiento utiliza algún tipo de información añadida a los datos e interpretada por la unidad de control del dispositivo para situar el transductor frente la posición correcta en el soporte (p.e. cintas y discos magnéticos).
3. Memorias *asociativas*, en las que el direccionamiento se hace en función del contenido de la memoria y no de su ubicación (p.e. en el direccionamiento de algunas memorias *caché*).

3.6.2. Tipos de memorias

En lo que respecta a los contenidos de esta asignatura, distinguiremos entre los siguientes tipos de memorias:

1. Memorias de *acceso secuencial*, que se caracterizan por que la información es almacenada y recuperada secuencialmente. En estas memorias, si la última posición accedida es la n -ésima no podrá accederse a la $(n + k)$ -ésima posición hasta k ciclos después. La implementación típica de este tipo de memorias es mediante la utilización de registros de desplazamiento o bien mediante emulación software en memoria RAM. Dependiendo de la forma en que se almacene y recupere la información podemos distinguir dos tipos de configuraciones:
 - a) *Memorias FIFO* (First-in First-out) o colas, en las que los valores se recuperan en el mismo orden en el que fueron almacenados.
 - b) *Memorias LIFO* (Last-in First-out) o pilas, en las que los valores se recuperan en orden inverso al que fueron almacenados.
2. Memorias de *semiconductores*, que son memorias implementadas mediante dispositivos semiconductores que incorporan un mecanismo de direccionamiento cableado y, por lo tanto, son estáticas y de acceso aleatorio. Distinguiremos dos categorías dentro de este tipo de memorias:
 - a) Memorias de *solo lectura* (ROM), utilizadas habitualmente para almacenar los programas de arranque de los ordenadores. Son memorias no volátiles y de lectura no destructiva. Existen distintas variantes de estas memorias que proporcionan características adicionales:
 - 1) Memorias ROM: Los datos contenidos en ellas son implementados en fábrica.

- 2) Memorias PROM (Programmable ROM): son memorias programables por el usuario mediante pulsos de corriente. Al igual que en las ROMs básicas, la programación solo se puede realizar una vez, quedando la información permanentemente almacenada en la memoria sin posibilidad de modificación.
 - 3) Memorias EPROM (Erasable PROM): estas memorias permiten múltiples reprogramaciones. La información es almacenada al igual que en las PROM por el usuario y puede, además, ser borrada mediante radiaciones ultravioleta. El borrado se realiza en todas las posiciones de memoria a la vez.
 - 4) Memorias EEPROM (Electrically-Erasable PROM): se diferencian de las anteriores en que tanto la programación como el borrado se realizan eléctricamente y pueden ser selectivos a nivel de palabra, es decir, pueden reprogramarse posiciones específicas de la memoria.
 - 5) Memorias Flash: que también permiten el borrado selectivo aunque a diferencia de las EEPROM este se realiza a nivel de bloque y no de palabra. Este tipo de memoria se está popularizando en la actualidad y tiene capacidades que van desde los 4Mb hasta varios Gb con un ancho de palabra de 8 ó 16 bits.
- b) Memorias de *lectura-escritura* (RAM), utilizadas como memoria principal en los ordenadores. Son memorias volátiles de lectura no destructiva. Dependiendo del mecanismo de almacenamiento utilizado para cada posición de la memoria, pueden clasificarse las memorias RAM en dos categorías:
- 1) *Estáticas*: en las que se utilizan biestables para almacenar la información.
 - 2) *Dinámicas* o con refresco: en las que se utilizan condensadores para mantener el estado almacenado en cada posición de la memoria. Debido a la pérdida de carga inherente a los condensadores es necesario "refrescarlos" periódicamente para mantener correctamente los datos almacenados. Este tipo de implementación permite mayores densidades de integración y, por tanto, mayores capacidades de almacenamiento. Sin embargo, la necesidad de refresco produce que sean más lentas que las memorias RAM estáticas.

La estructura típica de este tipo de memorias (figura 3.37) está formada por una matriz de memoria, un decodificador de direcciones, un conjunto de transductores (1 por cada bit del tamaño de palabra almacenado) y un elemento de control. En general estas memorias tienen las siguientes características:

- a) *Bus de direcciones*, que puede ser *completo*, de tamaño m para 2^m direcciones posibles, o *multiplexado*, se reciben primero $m/2$ bits de la dirección y después el resto.
- b) *Bus de datos*, de tipo triestado para permitir la conexión directa de los buses de varias memorias entre si.
- c) *Señales de control*, que modifican diferentes aspectos del funcionamiento de la memoria. Algunas de las más habituales son:

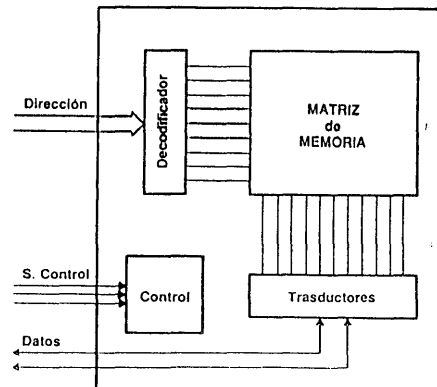


Figura 3.37: Diagrama de bloques de una memoria de semiconductores.

- 1) \overline{CS} o \overline{CE} , para la activación del chip.
 - 2) \overline{OE} , activa la salida triestado.
 - 3) \overline{WE} , selecciona una operación de escritura.
 - 4) \overline{RAS} y \overline{CAS} , se usan con buses de direcciones multiplexados para indicar que parte de la dirección se está enviando.
- d) *Ancho de palabra*, que es normalmente de 4, 8, 16 o 32 bits.
- e) *Capacidad*, sus valores habituales van desde 4KB hasta 512MB, aunque es una característica que seguirá incrementándose a medida que aumente el nivel de integración de los circuitos integrados. Además, como se puede ver en la figura 3.38, pueden conectarse varios chips de memoria entre si para aumentar el número de palabras almacenadas, la longitud de palabra, o ambas cosas a la vez.

La utilización correcta de este tipo de memorias requiere tener en cuenta sus requisitos de temporización. En la figura 3.39 se muestra el cronograma del ciclo de lectura en una memoria RAM. Durante este ciclo se supone que la señal de control \overline{WE} está activada. Los tiempos implicados en la operación son los siguientes:

- a) el mínimo *tiempo de ciclo* (t_c) necesario para completar una operación de lectura o escritura correctamente.
- b) el *tiempo de acceso* (t_a), que es el tiempo que transcurre desde que se aplica una dirección en el bus de direcciones hasta que está disponible en la salida el dato almacenado en ella.
- c) el *tiempo de deselección* (t_d), que es el tiempo que el valor almacenado permanece en la salida después de cambiar la dirección aplicada en el bus de direcciones.

El ciclo de escritura se muestra en la figura 3.40. Ahora se aplica un pulso de duración (t_w) a la señal de control \overline{WE} para almacenar, en la dirección indicada, el

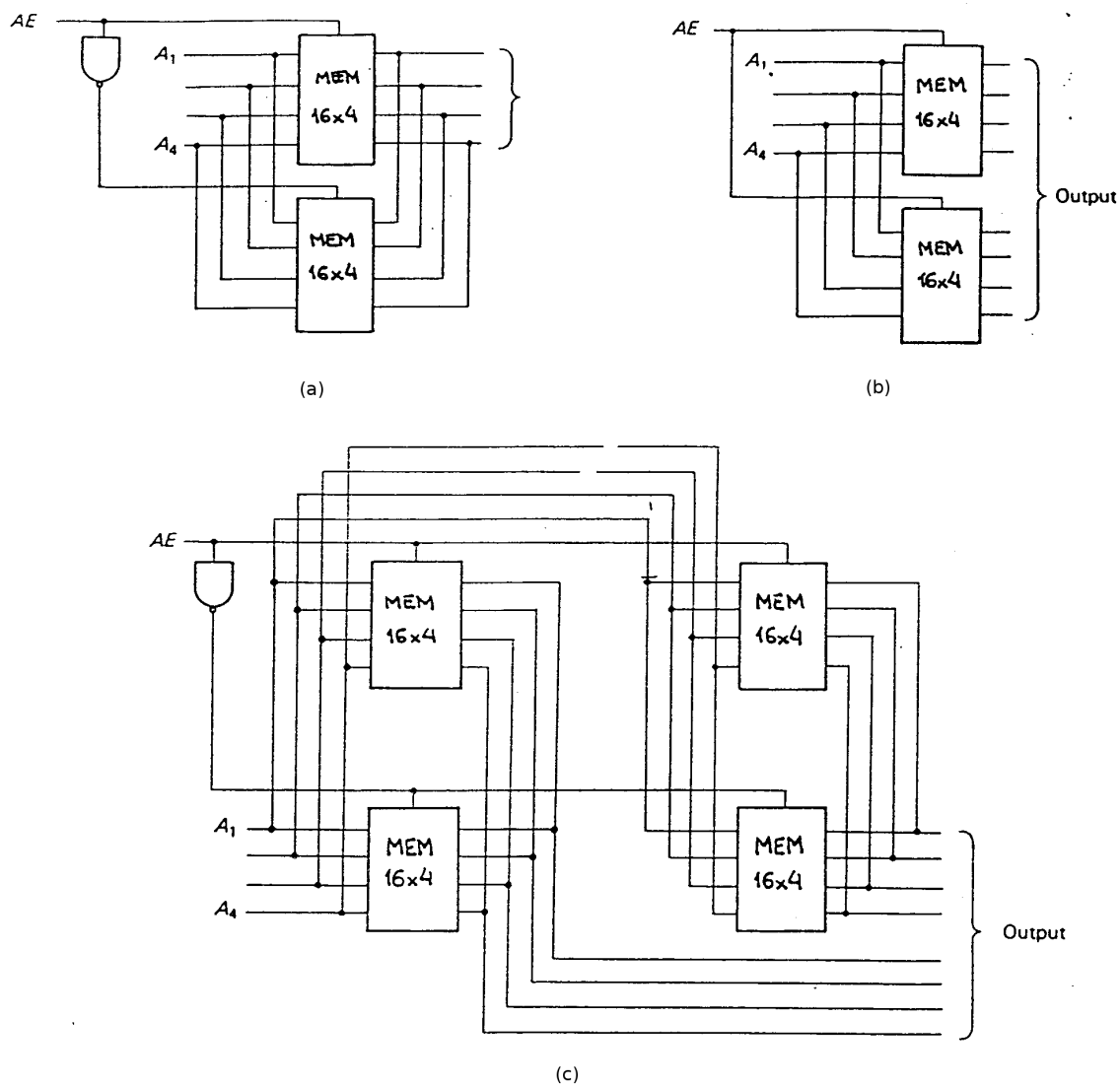


Figura 3.38: Conexión en cascada de varios circuitos de memoria: (a) memoria 32x4 compuesta por dos módulos 16x4; (b) memoria 16x8 compuesta por dos módulos 16x4; (c) memoria 32x8 compuesta por 4 módulos 16x4.

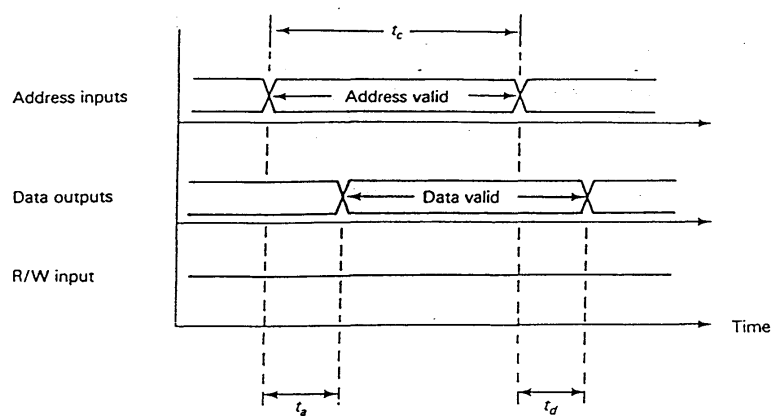


Figura 3.39: Ciclo de lectura en una memoria RAM.

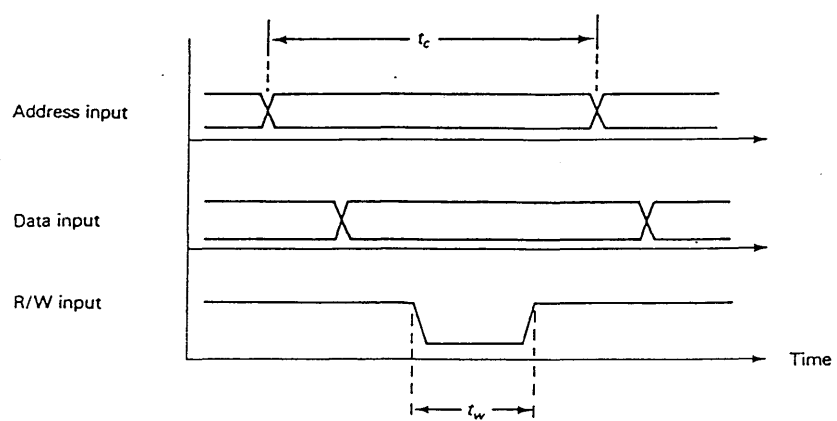


Figura 3.40: Ciclo de escritura en una memoria RAM.

valor presente en la entrada de datos. Tanto la dirección como el valor de entrada deben ser estables antes de la aplicación del pulso de escritura. Nótese que en las memorias ROM únicamente tiene sentido hablar de ciclo de lectura.

3. Memorias *asociativas*, en las que como ya se ha comentado el direccionamiento se hace en función del contenido almacenado en la memoria, es decir, las memorias asociativas tienen la capacidad de acceder a una palabra almacenada utilizando como dirección el valor de dicha palabra o un subcampo de la misma. El acceso se realiza mediante la comparación en paralelo de la clave de búsqueda con todas las palabras almacenadas en la memoria. Debido a esta capacidad a estas memorias también se las denomina *memorias direccionables por contenido (CAM)*, *memorias de búsqueda paralela* o *memorias multiacceso*, y son utilizadas en múltiples aplicaciones como la implementación de memorias *caché* asociativas, el diseño de los mecanismos de control de la memoria virtual, la manipulación de la información almacenada en grandes bases de datos, el tratamiento de señales de radar, el procesamiento de imágenes, la inteligencia artificial o la implementación de procesadores asociativos (STARAN, PEPE), entre otras. Su principal desventaja respecto a las memorias de acceso aleatorio es su elevado coste de implementación.