DISEÑO DIGITAL

TERCERA EDICIÓN

M. Morris Mano
CALIFORNIA STATE UNIVERSITY, LOS ANGELES

TRADUCCIÓN

Roberto Escalona García
Ingeniero Químico
Universidad Nacional Autónoma de México

REVISIÓN TÉCNICA

Gonzalo Duchén Sánchez
Sección de Estudios de Postgrado e Investigación
Escuela Superior de Ingeniería Mecánica y Eléctrica
Unidad Culhuacán
Instituto Politécnico Nacional

PEARSON Educación®

México • Argentina • Brasil • Colombia • Costa Rica • Chile • Ecuador
España • Guatemala • Panamá • Perú • Puerto Rico • Uruguay • Venezuela

5 Lógica secuencial sincrónica

5-1 CIRCUITOS SECUENCIALES

Los circuitos digitales estudiados hasta ahora han sido combinacionales: sus salidas dependen exclusivamente de las entradas actuales. Aunque es probable que todos los sistemas digitales tengan circuitos combinacionales, casi todos los que se usan en la práctica también incluyen elementos de almacenamiento, que requieren que el sistema se describa en términos de *lógica secuencial*.

En la figura 5-1 se presenta un diagrama de bloques de un circuito secuencial. Consiste en un circuito combinacional al que se conectan elementos de almacenamiento para formar una trayectoria de retroalimentación. Los elementos de almacenamiento son dispositivos capaces de guardar información binaria. La información almacenada en estos elementos en cualquier momento dado define el *estado* del circuito secuencial en ese momento. El circuito secuencial recibe información binaria de entradas externas. Esas entradas, junto con el estado actual de los elementos de almacenamiento, determinan el valor binario de las salidas. También determinan la condición para cambiar el estado de los elementos de almacenamiento. El diagrama de bloques indica que las salidas de un circuito secuencial son función no sólo de las entradas, sino también del estado actual de los elementos de almacenamiento. El siguiente estado de los elementos de almacenamiento también es función de entradas externas y del estado actual. Así pues,

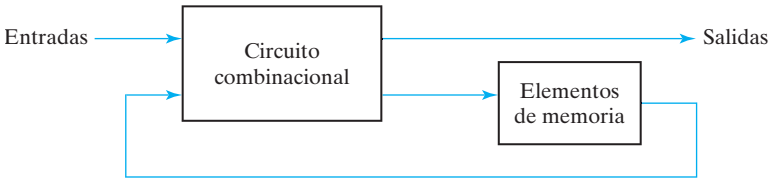


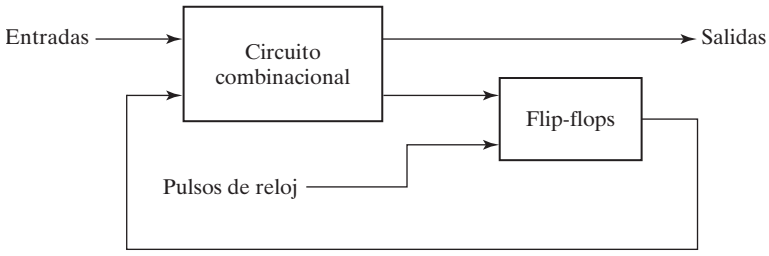
FIGURA 5-1
Diagrama de bloques de un circuito secuencial

un circuito secuencial se especifica con una sucesión temporal de entradas, salidas y estados internos.

Hay dos tipos principales de circuitos secuenciales, y su clasificación depende de los tiempos de sus señales. Un circuito secuencial *sincrónico* es un sistema cuyo comportamiento se define conociendo sus señales en instantes discretos. El comportamiento de un circuito secuencial *asincrónico* depende de las señales de entrada en cualquier instante dado y del orden en que cambian las entradas. Los elementos de almacenamiento que suelen usarse en los circuitos secuenciales asincrónicos son dispositivos de retardo de tiempo. La capacidad de almacenamiento de un dispositivo de retardo de tiempo se debe al tiempo que la señal tarda en propagarse por el dispositivo. En la práctica, el retardo interno de propagación de las compuertas lógicas tiene la suficiente duración como para producir el retardo requerido, de modo que podrían no ser necesarias unidades de retardo adicionales. En los sistemas asincrónicos tipo compuerta, los elementos de almacenamiento consisten en compuertas lógicas cuyo retardo de propagación hace posible el almacenamiento requerido. Así, un circuito secuencial asincrónico podría considerarse como un circuito combinacional con retroalimentación. Gracias a la retroalimentación entre compuertas lógicas, el circuito secuencial asincrónico podría volverse inestable ocasionalmente. El problema de inestabilidad impone muchas dificultades al diseñador. Los circuitos secuenciales asincrónicos se estudiarán en el capítulo 9.

Un circuito secuencial sincrónico utiliza señales que afectan a los elementos de almacenamiento únicamente en instantes discretos. La sincronización se logra con un dispositivo de temporización llamado *generador de reloj*, el cual produce un tren periódico de *pulsos de reloj*. Los pulsos de reloj se distribuyen por todo el sistema de modo que los elementos de almacenamiento sólo se vean afectados al llegar cada pulso. En la práctica, los pulsos de reloj se aplican con otras señales que especifican el cambio requerido en los elementos de almacenamiento. Los circuitos secuenciales sincrónicos que usan pulsos de reloj en las entradas de sus elementos de almacenamiento se denominan *circuitos secuenciales con reloj*, y son el tipo que se usa más comúnmente en la práctica. Casi nunca manifiestan problemas de estabilidad y es fácil dividir su temporización en pasos discretos independientes, cada uno de los cuales se puede considerar por separado.

Los elementos de almacenamiento empleados en los circuitos secuenciales con reloj se llaman *flip-flops*. Un flip-flop es un dispositivo binario de almacenamiento que puede almacenar un bit de información. Un circuito secuencial podría usar muchos flip-flops para almacenar tantos bits como sea necesario. En la figura 5-2 se ilustra el diagrama de bloques de un circuito secuencial



a) Diagrama de bloques

b) Diagrama de temporización de los pulsos de reloj

FIGURA 5-2
Circuito secuencial sincrónico con reloj

sincrónico con reloj. Las salidas pueden provenir del circuito combinacional o de los flip-flops, o de ambos. Los flip-flops reciben sus entradas del circuito combinacional y también de una señal de reloj cuyos pulsos se presentan a intervalos fijos de tiempo, como se observa en el diagrama de temporización. El estado del flip-flop sólo puede cambiar durante una transición de pulso de reloj. Cuando el pulso de reloj no está activo, el ciclo de retroalimentación se rompe porque las salidas del flip-flop no pueden cambiar aunque cambie el valor de las salidas del circuito combinacional que alimenta sus entradas. Por tanto, la transición de un estado al siguiente se da únicamente a intervalos de tiempo preestablecidos, dictados por los pulsos de reloj.

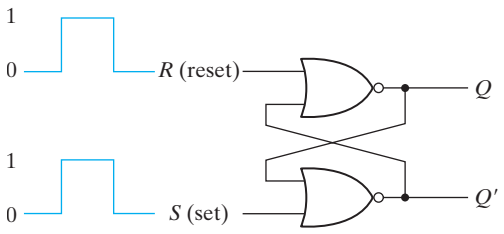
5-2 LATCHES

Un circuito flip-flop puede mantener un estado binario indefinidamente (en tanto se alimente electricidad al circuito), hasta que una señal de entrada le indique que debe cambiar de estado. Las principales diferencias entre los diversos tipos de flip-flops radican en el número de entradas que tienen y en la forma en que las entradas afectan el estado binario. Los tipos más básicos de flip-flops operan con niveles de señal y se llaman *latches*. Los latches que presentaremos aquí son los circuitos básicos con los que se construyen todos los flip-flops. Aunque los latches son útiles para almacenar información binaria y para diseñar circuitos secuenciales asincrónicos (véase la sección 9-3), no resultan prácticos en los circuitos secuenciales sincrónicos. En la sección que sigue presentaremos los tipos de flip-flops que se usan en los circuitos secuenciales.

Latch *SR*

El latch *SR* es un circuito con dos compuertas NOR acopladas en cruz o dos compuertas NAND acopladas en cruz. Tiene dos entradas, *S* (de *set*, establecer) y *R* (de *reset*, restablecer). El latch *SR* que se construye con dos compuertas NOR acopladas en cruz se aprecia en la figura 5-3. El latch tiene dos estados útiles. Cuando las salidas $Q = 1$ y $Q' = 0$, decimos que está en el *estado establecido*. Cuando $Q = 0$ y $Q' = 1$, está en el *estado restablecido*. Las salidas Q y Q' normalmente son una el complemento de la otra, pero si ambas entradas son 1 al mismo tiempo, se presenta un estado indefinido en el que ambas salidas son 0.

En condiciones normales, las dos entradas del latch permanecen en 0 a menos que se deba cambiar de estado. La aplicación momentánea de un 1 a la entrada *S* hace que el latch pase al estado establecido. La entrada *S* debe volver a 0 antes de cualquier otro cambio, para que no se presente el estado indefinido. Como se indica en la tabla de función de la figura 5-3b), dos condiciones de entrada hacen que el circuito esté en el estado establecido. La primera condición ($S = 1$,



a) Diagrama lógico

S	R	Q	Q'	
1	0	1	0	
0	0	1	0	(Después de $S = 1, R = 0$)
0	1	0	1	
0	0	0	1	(Después de $S = 0, R = 1$)
1	1	0	0	

b) Tabla de función

FIGURA 5-3
Latch *SR* con compuertas NOR

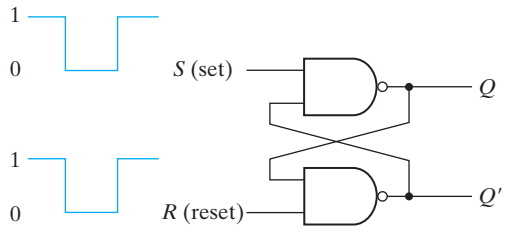
$R = 0$) es la acción que debe efectuar la entrada S para poner el circuito en el estado establecido. Cuando la entrada activa se quita de S, el circuito permanece en el mismo estado. Una vez que las dos entradas regresan a 0, será posible cambiar al estado restablecido aplicando momentáneamente un 1 a la entrada R. Luego puede quitarse el 1 de R y el circuito permanecerá en el estado restablecido. Así pues, cuando ambas entradas, S y R, son 0, el latch estará en el estado establecido o en el restablecido, dependiendo de cuál entrada fue 1 más recientemente.

Si se aplica un 1 a las dos entradas S y R del latch, ambas salidas cambian a 0. Esto produce un estado indefinido porque se hace imposible predecir cuál será el siguiente estado cuando ambas entradas vuelvan a 0, y también viola el requisito de que una salida sea el complemento de la otra. Durante el funcionamiento normal, esta condición se evita asegurándose de que no se aplique 1 a ambas entradas simultáneamente.

El latch *SR* con dos compuertas NAND acopladas en cruz se muestra en la figura 5-4. Opera con ambas entradas normalmente en 1, a menos que sea preciso cambiar el estado del latch. La aplicación de 0 a la entrada S hace que la salida Q cambie a 1 y coloca al latch en el estado establecido. Cuando la salida S vuelve a 1, el circuito permanece en el estado establecido. Una vez que ambas entradas vuelven a 1, se permite cambiar el estado del latch aplicando un 0 a la entrada R. Esto hace que el circuito vuelva al estado restablecido y permanezca en él aun después de que ambas entradas vuelven a 1. La condición que no está definida en el caso del latch NAND es que ambas entradas sean 0 al mismo tiempo, así que debe evitarse esa combinación de entradas.

Si comparamos el latch NAND con el NOR, veremos que las señales de entrada del latch NAND requieren el complemento de los valores empleados para el latch NOR. Dado que el latch NAND requiere una señal 0 para cambiar su estado, también se le conoce como latch $S'R'$. Los apóstrofos (o testas sobre las letras) indican que las entradas deben estar en su forma complementada para activar el circuito.

Es posible modificar el funcionamiento del latch *SR* básico incluyendo una entrada de control adicional que determina cuándo puede cambiarse el estado del latch. En la figura 5-5 se muestra un latch *SR* con entrada de control. Consiste en el latch *SR* básico y dos compuertas NAND adicionales. La entrada de control C actúa como señal de habilitación para las otras dos entradas. La salida de las compuertas NAND permanecerán en el nivel 1 lógico en tanto la entrada de control permanezca en 0. Ésta es la condición latente del latch *SR*. Cuando la entrada de control cambia a 1, se permite que la información de la entrada S o R afecte al latch *SR*. Se alcanza el estado establecido con $S = 1, R = 0$ y $C = 1$. Para cambiar al estado restablecido, las entradas deben ser $S = 0, R = 1$ y $C = 1$. En ambos casos, cuando C regresa a 0, el cir-

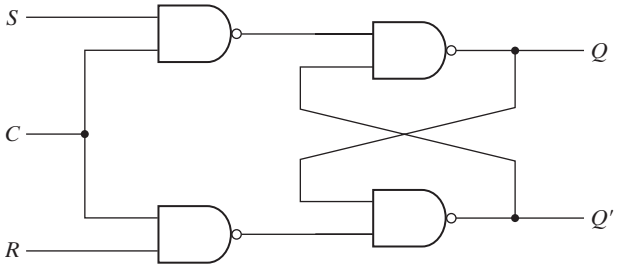


a) Diagrama lógico

S	R	Q	Q'	
1	0	0	1	
1	1	0	1	(Después de $S = 1, R = 0$)
0	1	1	0	
1	1	1	0	(Después de $S = 0, R = 1$)
0	0	1	1	

b) Tabla de función

FIGURA 5-4
Latch *SR* con compuertas NAND



a) Diagrama lógico

C	S	R	Siguiente estado de Q
0	X	X	Sin cambio
1	0	0	Sin cambio
1	0	1	$Q = 0$; estado restablecido
1	1	0	$Q = 1$; estado establecido
1	1	1	Indeterminado

b) Tabla de función

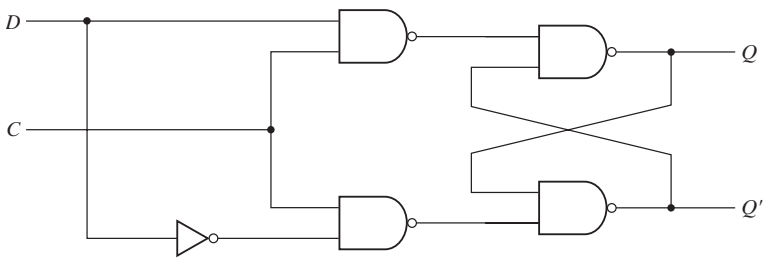
FIGURA 5-5
Latch *SR* con entrada de control

cuito permanece en su estado actual. La entrada de control inhabilita el circuito aplicando 0 a *C*, de modo que el estado de la salida no cambie sean cuales sean los valores de *S* y *R*. Además, cuando $C = 1$ y ambas entradas *S* y *R* son 0, el estado del circuito permanece sin cambio. Estas condiciones se presentan en la tabla de función que acompaña al diagrama.

Se presenta una condición indeterminada cuando las tres entradas son 1. Esta condición coloca ceros en ambas entradas del latch *SR* básico, lo que hace que éste pase al estado indefinido. Cuando la entrada de control vuelva a 0, no será posible determinar de forma concluyente el próximo estado, pues dependerá de cuál entrada, *S* o *R*, cambie primero a 0. Esta condición indeterminada dificulta el manejo de este circuito y casi nunca se usa en la práctica. No obstante, es un circuito importante porque otros latches y flip-flops se construyen con él.

Latch *D*

Una forma de eliminar la condición indeseable del estado indeterminado en el latch *SR* es garantizar que las entradas *S* y *R* nunca sean 1 al mismo tiempo. Esto se hace en el latch *D* que se ilustra en la figura 5-6. Este latch sólo tiene dos entradas: *D* (datos) y *C* (control). La entrada *D* pasa directamente a la entrada *S* y su complemento se aplica a la entrada *R*. En tanto la entrada de control esté en 0, el latch *SR* acoplado en cruz tendrá ambas entradas en el nivel 1 y el circuito no podrá cambiar de estado sea cual sea el valor de *D*. La entrada *D* se muestrea cuando $C = 1$. Si $D = 1$, la salida *Q* pasará a 1, colocando el circuito en el estado establecido. Si $D = 0$, la salida *Q* pasará a 0, colocando el circuito en el estado restablecido.



a) Diagrama lógico

C	D	Siguiente estado de Q
0	X	Sin cambio
1	0	$Q = 0$; estado restablecido
1	1	$Q = 1$; estado establecido

b) Tabla de función

FIGURA 5-6
Latch *D*

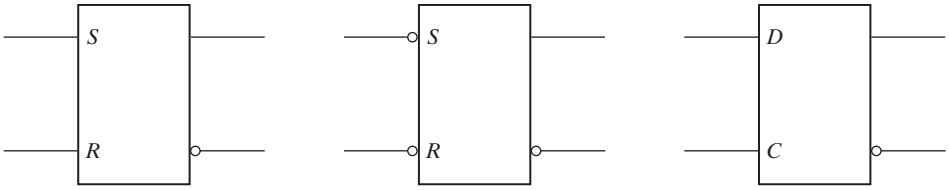


FIGURA 5-7
Símbolos gráficos de latches

El latch *D* se llama así por su capacidad para almacenar datos en su interior. Es apropiado para usarse como almacenamiento temporal de información binaria entre una unidad y su entorno. La información binaria presente en la entrada de datos del latch *D* se transfiere a la salida *Q* cuando se habilita la entrada de control. La salida seguirá los cambios en la entrada de datos en tanto esté habilitada la entrada de control. Esta situación crea un camino de la entrada *D* a la salida, y es por ello que el circuito se conoce como latch *transparente*. Cuando se inhabilita la entrada de control, la información binaria que estaba presente en la entrada de datos en el momento en que se presentó la transición se conservará en la salida *Q* hasta que se habilite otra vez la entrada de control.

En la figura 5-7 se presentan los símbolos gráficos para los distintos latches. Los latches se representan con un rectángulo cuyas entradas están a la izquierda, y sus salidas, a la derecha. Una salida indica la salida normal, y la otra (con burbuja), su complemento. En el símbolo gráfico del latch *SR*, las entradas *S* y *R* se indican dentro del rectángulo. En el caso de un latch de compuertas NAND, se añaden burbujas a las entradas para indicar que el establecimiento y el restablecimiento se efectúan con la señal de 0 lógico. En el símbolo gráfico del latch *D* las entradas *D* y *C* se indican dentro del rectángulo.

5-3 FLIP-FLOPS

El estado de un latch o flip-flop se conmuta con un cambio en la entrada de control. Este cambio momentáneo se denomina *disparo* y decimos que la transición que causa dispara el flip-flop. El latch *D* con pulsos en su entrada de control es básicamente un flip-flop que se dispara cada vez que el pulso alcanza el nivel de 1 lógico. En tanto la entrada de pulso se mantenga en este nivel, cualquier cambio en la entrada de datos hará que cambie la salida y el estado del latch.

Como se aprecia en el diagrama de bloques de la figura 5-2, un circuito secuencial tiene una trayectoria de retroalimentación de las salidas de los flip-flops a la entrada del circuito combinacional. Por tanto, las entradas de los flip-flops se derivan en parte de las salidas de esos mismos flip-flops y de otros. Cuando se usan latches como elementos de almacenamiento, surge una dificultad grave. Las transiciones de estado de los latches se inician tan pronto como el pulso de reloj cambia al nivel de 1 lógico. El nuevo estado del latch aparece en la salida mientras el pulso aún está activo. Esta salida se conecta a las entradas de los latches a través del circuito combinacional. Si las entradas aplicadas a los latches cambian mientras el pulso de reloj todavía está en el nivel de 1 lógico, los latches responderán a nuevos valores y podría presentarse un nuevo estado de salida. El resultado es una situación impredecible, ya que el estado de los latches podría seguir cambiando durante todo el tiempo que el pulso de reloj se mantiene en el

a) Respuesta al nivel positivo

b) Respuesta al borde positivo

c) Respuesta al borde negativo

FIGURA 5-8
Respuesta al reloj en un latch y un flip-flop

estado activo. Debido a este funcionamiento poco confiable, no es posible aplicar directamente, ni a través de lógica combinacional, la salida de un latch a la entrada del mismo latch o de otro, cuando todos los latches se disparan con la misma fuente de reloj.

Los circuitos de flip-flop se construyen de tal manera que funcionan correctamente cuando forman parte de un circuito secuencial que utiliza un solo reloj. El problema del latch es que responde a un cambio en el *nivel* de un pulso de reloj. Como se observa en la figura 5-8a), una respuesta de nivel positivo en la entrada de control permite cambios en la salida cuando la entrada *D* cambia mientras el reloj se mantiene en el nivel de 1 lógico. La clave para que el flip-flop funcione correctamente es dispararlo únicamente durante una *transición* de la señal. Un pulso de reloj sufre dos transiciones: de 0 a 1 y de 1 a 0 al regresar. Como se aprecia en la figura 5-8, la transición positiva se define como el borde (o flanco) positivo, y la negativa, como el borde negativo. Hay dos formas de modificar un latch para formar un flip-flop. Una consiste en utilizar dos latches en una configuración especial que aísla la salida del flip-flop para que no se vea afectada mientras su entrada está cambiando. Otra consiste en producir un flip-flop que se dispare únicamente durante una transición de señal (de 0 a 1 o de 1 a 0) y quede inhabilitado durante el resto del pulso de reloj. Ahora mostraremos la implementación de ambos tipos de flip-flop.

Flip-flop *D* disparado por borde (o flanco)

En la figura 5-9 se representa la construcción de un flip-flop *D* con dos latches *D* y un inversor. El primer latch es el amo, y el segundo, el esclavo. El circuito muestrea la entrada *D* y cambia su salida *Q* únicamente en el borde negativo del reloj controlador (designado por *CLK* [*clock*]). Cuando el reloj es 0, la salida del inversor es 1. El latch esclavo queda habilitado y su salida *Q* es igual a la salida *Y* del amo. El latch amo queda inhabilitado porque $CLK = 0$.

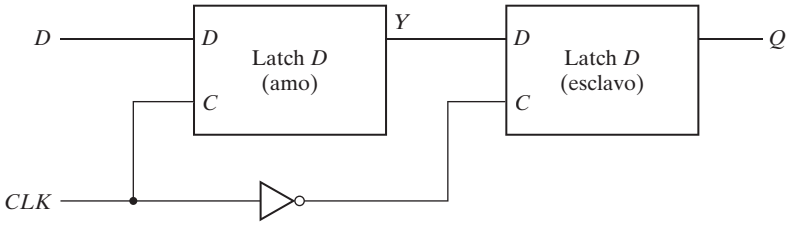


FIGURA 5-9
Flip-flop *D* amo-esclavo

Cuando el pulso de entrada cambia al nivel de 1 lógico, el dato de la entrada *D* externa se transfiere al amo. El esclavo, empero, estará inhabilitado en tanto el reloj permanezca en el nivel 1, porque su entrada *C* es 0. Cualquier cambio en la entrada hará que cambie la salida *Y* del amo, pero no afectará la salida del esclavo. Cuando el pulso vuelva a 0, el amo quedará inhabilitado y aislado de la entrada *D*. Al mismo tiempo, el esclavo estará habilitado y el valor *Y* se transferirá a la salida *Q* del flip-flop. Así, la salida del flip-flop sólo puede cambiar durante la transición del reloj de 1 a 0.

El comportamiento del flip-flop amo-esclavo que acabamos de describir implica que la salida sólo puede cambiar durante el borde negativo del reloj. También es posible diseñar el circuito de modo que la salida del flip-flop cambie en el borde positivo del reloj. Esto sucede en un flip-flop que tiene un inversor adicional entre la terminal *CLK* y la unión entre el otro inversor y la entrada *C* del latch amo. Un flip-flop semejante se dispara con un pulso negativo, de modo que el borde negativo del reloj afecte al amo, y el positivo, al esclavo y a la terminal de salida.

Una construcción más eficiente de un flip-flop *D* disparado por borde utiliza tres latches *SR*, como se muestra en la figura 5-10. Dos latches responden a las entradas externas *D* (datos) y

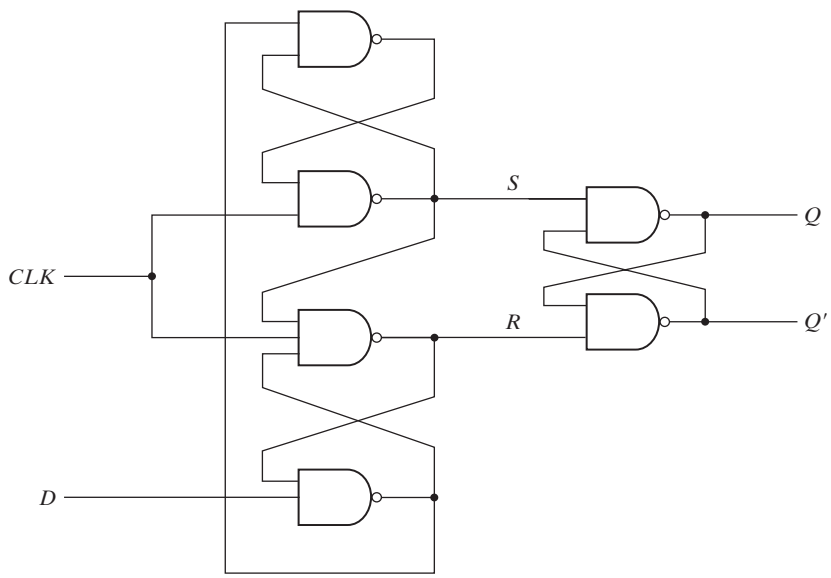


FIGURA 5-10
Flip-flop tipo *D* disparado por borde positivo

CLK (reloj). El tercer latch proporciona las salidas para el flip-flop. Las entradas *S* y *R* del latch de salida se mantienen en el nivel 1 lógico cuando $CLK = 0$. Esto hace que la salida permanezca en su estado actual. La entrada *D* podría ser 0 o 1. Si $D = 0$ cuando *CLK* pasa a 1, *R* cambia a 0. Esto hace que el flip-flop pase al estado restablecido, de modo que $Q = 0$. Si hay un cambio en la entrada *D* mientras $CLK = 1$, la terminal *R* permanecerá en 0. Así, el flip-flop queda bloqueado y no responde a más cambios en la entrada. Cuando el reloj vuelve a 0, *R* cambia a 1 y hace que el latch de salida pase a la condición latente sin cambiar su salida. Asimismo, si $D = 1$ cuando *CLK* pasa de 0 a 1, *S* cambiará a 0. Esto hace que el circuito pase al estado establecido y que $Q = 1$. Cualquier cambio en *D* mientras $CLK = 1$ no afectará la salida.

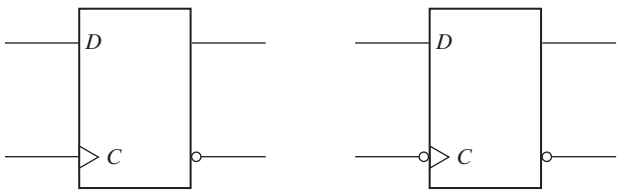
En síntesis, cuando el reloj de entrada del flip-flop disparado por borde positivo efectúa una transición positiva, el valor de *D* se transfiere a *Q*. Una transición negativa de 1 a 0 no afecta la salida, y tampoco lo hace cuando *CLK* está en el nivel estable de 1 lógico o 0 lógico. Por tanto, este tipo de flip-flop responde a la transición de 0 a 1 y a ninguna otra cosa.

Los tiempos de la respuesta de un flip-flop a los datos de entrada y al reloj se deben tomar en consideración al usar flip-flop disparados por borde. Hay un tiempo mínimo, llamado *tiempo de preparación* (*setup* en inglés), durante el cual la entrada *D* se debe mantener en un valor constante antes de que se presente la transición de reloj. Asimismo, hay un tiempo mínimo, llamado *tiempo de retención* (*hold* en inglés), durante el cual la entrada *D* no debe cambiar después de la aplicación de la transición positiva del reloj. El retardo de propagación del flip-flop se define como el intervalo de tiempo entre el borde disparador y la estabilización de la salida en un nuevo estado. Éstos y otros parámetros se especifican en los libros de datos de los fabricantes de familias lógicas específicas.

El símbolo gráfico para el flip-flop *D* disparado por flanco aparece en la figura 5-11. Es similar al utilizado para el latch *D*, excepto por el símbolo triangular antes de la letra *C* que designa una entrada *dinámica*. El *indicador dinámico* denota el hecho de que el flip-flop responde a la transición de borde del reloj. Una burbuja afuera del rectángulo, junto al indicador dinámico, denota un borde negativo para disparar el circuito. La ausencia de la burbuja denota una respuesta al borde positivo.

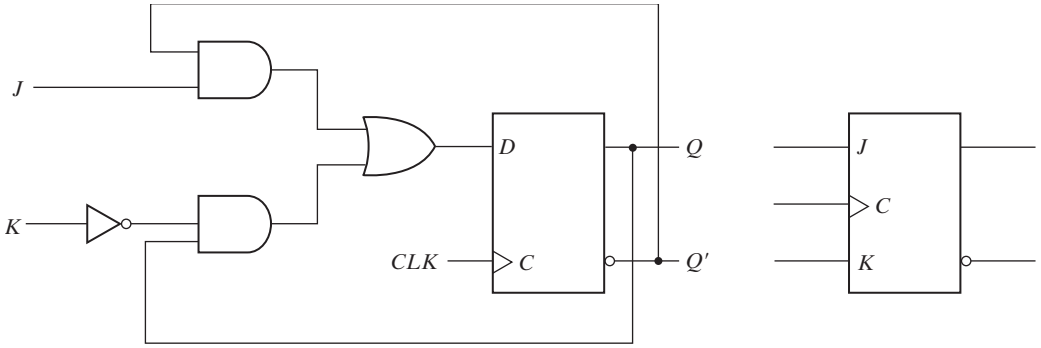
Otros flip-flops

Los circuitos de integración a muy grande escala contienen miles de compuertas en un paquete. Los circuitos se construyen interconectando las diversas compuertas para crear un sistema digital. Cada flip-flop se construye interconectando compuertas. El flip-flop más económico y eficiente construido de esta manera es el flip-flop *D* disparado por borde, porque es el que



a) Borde positivo b) Borde negativo

FIGURA 5-11
Símbolo gráfico para el flip-flop *D* disparado por borde



a) Diagrama de circuito b) Símbolo gráfico

FIGURA 5-12
Flip-Flop *JK*

menos compuertas requiere. Es posible construir otros tipos de flip-flops utilizando el flip-flop *D* y lógica externa. Dos flip-flops ampliamente utilizados en el diseño de sistemas digitales son los flip-flops *JK* y *T*.

Hay tres operaciones que pueden efectuarse con un flip-flop: establecerlo en 1, restablecerlo a 0 y complementar su salida. El flip-flop *JK* realiza las tres operaciones. El diagrama de circuito de un flip-flop *JK* construido con un flip-flop *D* y compuertas se reproduce en la figura 5-12a). La entrada *J* establece el flip-flop en 1, la entrada *K* lo restablece a 0 y, cuando ambas entradas están habilitadas, la salida se complementa. Esto se verifica investigando el circuito aplicado a la entrada *D:*

$$D = JQ' + K'Q$$

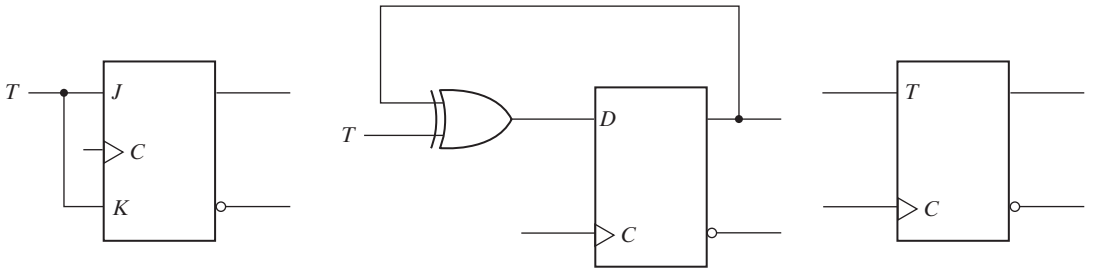
Cuando $J = 1$ y $K = 0, D = Q' + Q = 1$, así que el siguiente borde del reloj establece la salida en 1. Cuando $J = 0$ y $K = 1$, $D = 0$, así que el siguiente borde del reloj restablece la salida a 0. Cuando $J = 1$ y $K = 1$, $D = Q'$, y el siguiente borde del reloj complementa la salida. Cuando $J = K = 0, D = Q$, y el borde de reloj no altera la salida. El símbolo gráfico del flip-flop *JK* se indica en la figura 5-12b). Es similar al del flip-flop *D*, excepto que ahora las entradas están marcadas con *J* y *K*.

El flip-flop *T* (*toggle*) es un flip-flop complementador y se puede implementar con un flip-flop *JK* si se conectan entre sí las entradas *J* y *K*, como se observa en la figura 5-13a). Cuando $T = 0$ ($J = K = 0$), un borde de reloj no modifica la salida. Cuando $T = 1$ ($J = K = 1$), un borde de reloj complementa la salida. El flip-flop complementador es útil para diseñar contadores binarios.

El flip-flop *T* se puede construir con un flip-flop *D* y una compuerta OR exclusivo, como se indica en la figura 5-13b). La expresión para la entrada *D* es

$$D = T \oplus Q = TQ' + T'Q$$

Cuando $T = 0$, entonces $D = Q$, y la salida no cambia. Cuando $T = 1$, entonces $D = Q'$ y la salida se complementa. El símbolo gráfico de este flip-flop tiene una *T* en la entrada.



a) Con un flip-flop *JK* b) Con un flip-flop *D* c) Símbolo gráfico

FIGURA 5-13
Flip-Flop *T*

Tablas características

Una tabla característica define las propiedades lógicas de un flip-flop describiendo su funcionamiento en forma tabular. En la tabla 5-1 se presentan las tablas características de tres tipos de flip-flops. Definen el siguiente estado en función de las entradas y del estado actual. $Q(t)$ se refiere al estado actual antes de la aplicación de un borde de reloj. $Q(t + 1)$ es el siguiente estado, un periodo de reloj después. Observe que la entrada de borde de reloj no se incluye en la tabla característica, pero se supone implícitamente entre el tiempo t y el tiempo $t + 1$.

La tabla característica del flip-flop *JK* revela que el siguiente estado es igual al estado actual cuando las entradas J y K son ambas 0. Esto se expresa como $Q(t + 1) = Q(t)$, e indica que el reloj no produce ningún cambio de estado. Cuando $K = 1$ y $J = 0$, el reloj restablece el flip-flop y $Q(t + 1) = 0$. Con $J = 1$ y $K = 0$, el flip-flop se establece y $Q(t + 1) = 1$.

Tabla 5-1
Tablas características de flip-flops

Flip-Flop *JK*

J	K	$Q(t + 1)$	
0	0	$Q(t)$	Sin cambio
0	1	0	Restablecer
1	0	1	Establecer
1	1	$Q'(t)$	Complementar

Flip-Flop *D*

D	$Q(t + 1)$	
0	0	Restablecer
1	1	Establecer

Flip-Flop *T*

T	$Q(t + 1)$	
0	$Q(t)$	Sin cambio
1	$Q'(t)$	Complementar

Cuando tanto J como K son 1, el siguiente estado cambia al complemento del estado actual, lo que se expresa como $Q(t + 1) = Q'(t)$.

El siguiente estado de un flip-flop D depende únicamente de la entrada D y es independiente del estado actual. Esto se expresa como $Q(t + 1) = D$, y significa que el valor del siguiente estado será igual al valor de D. Cabe señalar que el flip-flop D no tiene una condición de "sin cambio". Esta condición se obtiene inhabilitando el reloj o dejando el reloj y conectando la salida de vuelta a la entrada D cuando el estado del flip-flop no debe cambiar.

La tabla característica del flip-flop T tiene sólo dos condiciones. Cuando $T = 0$, el borde de reloj no cambia el estado. Cuando $T = 1$, el borde de reloj complementa el estado del flip-flop.

Ecuaciones características

Las propiedades lógicas de un flip-flop descritas en la tabla característica también se pueden expresar algebraicamente con una ecuación característica. En el caso del flip-flop D, tenemos la ecuación característica

$$Q(t + 1) = D$$

Esto nos dice que el siguiente estado de la salida será igual al valor de la entrada D en el estado actual. La ecuación característica para el flip-flop JK se deduce de la tabla característica o del circuito de la figura 5-12. Se obtiene

$$Q(t + 1) = JQ' + K'Q$$

donde Q es el valor de la salida del flip-flop antes de la aplicación de un borde de reloj. La ecuación característica del flip-flop T se obtiene del circuito de la figura 5-13:

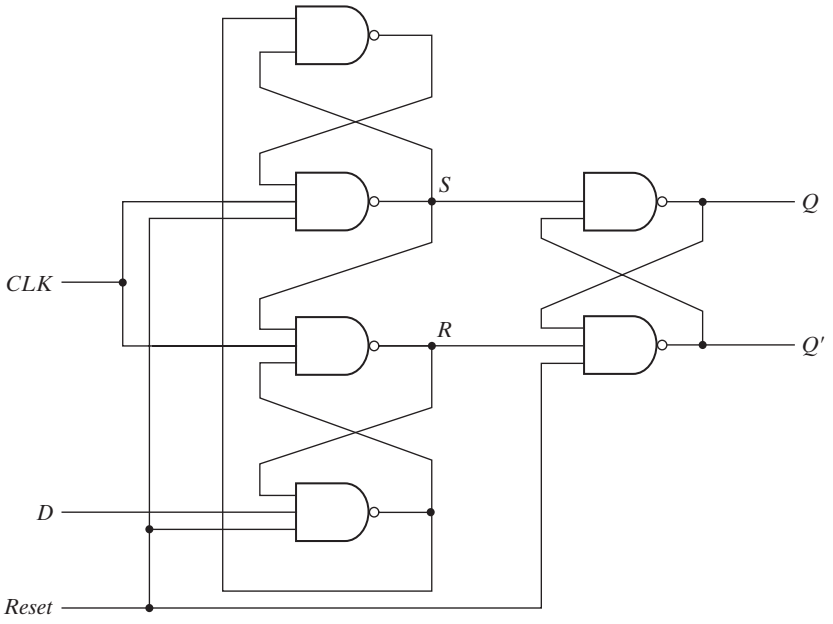
$$Q(t + 1) = T \oplus Q = TQ' + T'Q$$

Entradas directas

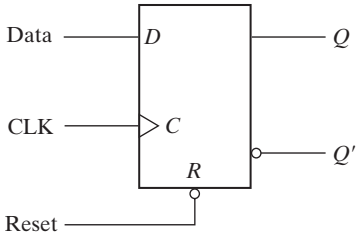
Algunos flip-flop tienen entradas asincrónicas que sirven para forzar el flip-flop a un estado dado independientemente del reloj. La entrada que pone el flip-flop en 1 se llama *preestablecimiento* (*preset*) o *establecimiento directo*. La entrada que pone en 0 el flip-flop se llama *borrado* (*clear*) o *restablecimiento directo*. Cuando se enciende un sistema digital, se desconoce el estado de los flip-flops. Las entradas directas sirven para poner todos los flip-flops del sistema en un estado inicial conocido antes del funcionamiento con reloj.

En la figura 5-14 se ilustra un flip-flop D disparado por flanco positivo, con restablecimiento asincrónico. El diagrama de circuito es igual al de la figura 5-10, excepto por la entrada de restablecimiento (*reset*) adicional conectada a tres compuertas NAND. Cuando la entrada de restablecimiento es 0, hace que Q' se mantenga en 1; esto, a su vez, pone en 0 la salida Q, con lo que el flip-flop se restablece. Otras dos conexiones de la entrada de restablecimiento garantizan que la entrada S del tercer latch SR se mantenga en 1 lógico mientras la entrada de restablecimiento está en 0, sean cuales sean los valores de D y de CLK.

El símbolo gráfico del flip-flop D con restablecimiento directo tiene una entrada adicional que se marca con R. La burbuja en la entrada indica que el restablecimiento está activo en el nivel de 0 lógico. Los flip-flops con establecimiento directo utilizan el símbolo S para la entrada de establecimiento asincrónico.



a) Diagrama de circuito



b) Símbolo gráfico

R	C	D	Q	Q′
0	X	X	0	1
1	↑	0	0	1
1	↑	1	1	0

c) Tabla de función

FIGURA 5-14
Flip-flop *D* con restablecimiento asincrónico

La tabla de función especifica la operación del circuito. Cuando $R = 0$, la salida se restablece a 0. Este estado es independiente de los valores de D o de C. El funcionamiento normal con reloj sólo podrá iniciarse después de que la entrada de restablecimiento cambie a 1 lógico. El reloj en C lleva una flecha hacia arriba para indicar que el flip-flop se dispara con el borde positivo del reloj. El valor en D se transfiere a Q con cada señal de reloj de borde positivo, siempre que $R = 1$.

5-4 ANÁLISIS DE CIRCUITOS SECUENCIALES CON RELOJ

El comportamiento de un circuito secuencial con reloj está determinado por las entradas, las salidas y el estado de sus flip-flops. Las salidas y el siguiente estado son función de las entradas y del estado actual. El análisis de un circuito secuencial consiste en obtener una tabla o diagrama para la sucesión temporal de entradas, salidas y estados internos. También es posible escribir expresiones booleanas que describan el comportamiento del circuito secuencial. Tales expresiones deberán incluir la sucesión temporal necesaria, sea directa o indirectamente.

Un diagrama lógico se reconoce como circuito secuencial con reloj si incluye flip-flops con entradas de reloj. Los flip-flops pueden ser de cualquier tipo, y el diagrama lógico podría incluir o no compuertas de circuitos combinacionales. En esta sección se incluye una representación algebraica para especificar la condición del siguiente estado en términos del estado actual y de las entradas. Luego se presentará una tabla de estados y un diagrama de estados para describir el comportamiento del circuito secuencial. Mostraremos otra representación algebraica para especificar el diagrama lógico de los circuitos secuenciales. Se incluyen también ejemplos específicos para ilustrar los diversos procedimientos.

Ecuaciones de estado

El comportamiento de los circuitos secuenciales con reloj se describe algebraicamente con ecuaciones de estado. Una *ecuación de estado* (también llamada *ecuación de transición*) especifica el siguiente estado en función del estado actual y las entradas. Consideremos el circuito secuencial de la figura 5-15. Consta de dos flip-flops D, A y B, una entrada x y una salida y. Puesto que la entrada D de un flip-flop determina el valor del siguiente estado, podemos escribir un conjunto de ecuaciones de estado para el circuito:

$$A(t + 1) = A(t)x(t) + B(t)x(t)$$
$$B(t + 1) = A'(t)x(t)$$

Una ecuación de estado es una expresión algebraica que especifica la condición para una transición de estado de un flip-flop. El miembro izquierdo de la ecuación, donde aparece $(t + 1)$, denota el siguiente estado del flip-flop, un borde de reloj después. El miembro derecho de la ecuación es una expresión booleana que especifica el estado actual y las condiciones de entrada que harán que el siguiente estado sea 1. Puesto que todas las variables de las expresiones booleanas son función del estado actual, se omite la designación (t) después de cada variable, por conveniencia, a fin de expresar las ecuaciones de estado en la forma más compacta:

$$A(t + 1) = Ax + Bx$$
$$B(t + 1) = A'x$$

Las expresiones booleanas para las ecuaciones de estado se deducen directamente de las compuertas que forman la parte de circuito combinacional del circuito secuencial, ya que los valores D del circuito combinacional determinan el siguiente estado. Asimismo, el valor del estado actual de la salida se expresa algebraicamente como

$$y(t) = [A(t) + B(t)]x'(t)$$

Al omitir el símbolo (t) para el estado actual, se obtiene la ecuación booleana de salida:

$$y = (A + B)x'$$

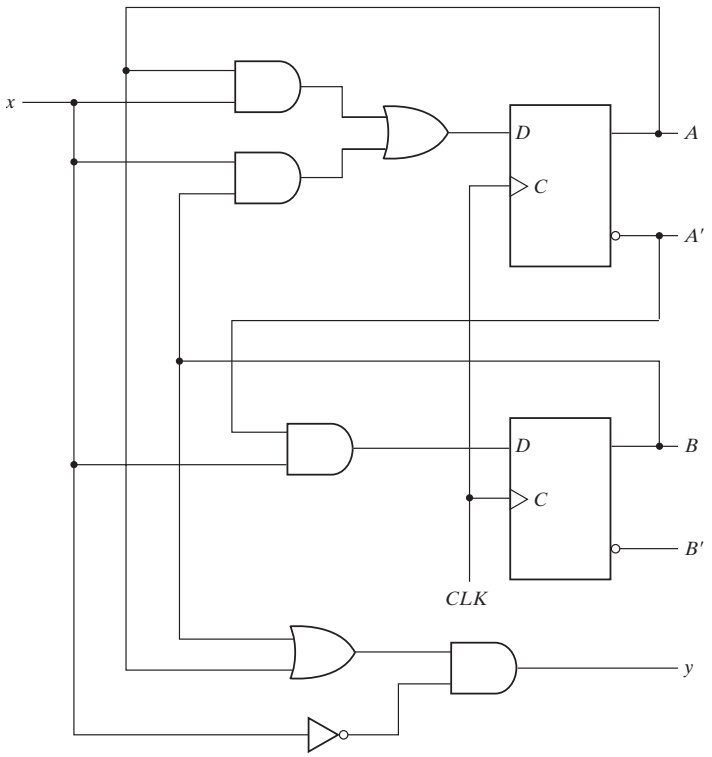


FIGURA 5-15
Ejemplo de circuito secuencial

Tabla de estados

La sucesión temporal de entradas, salidas y estados de flip-flop se puede presentar de forma compacta en una *tabla de estados* (también llamada *tabla de transición*). La tabla de estados para el circuito de la figura 5-15 se presenta en la tabla 5-2. La tabla consta de cuatro secciones rotuladas *estado actual*, *entrada*, *siguiente estado* y *salida*. La sección de estado actual muestra los estados de los flip-flops A y B en cualquier instante dado t. La sección de entrada da un valor de x para cada posible estado actual. La sección de siguiente estado muestra los estados de los flip-flops un ciclo de reloj después, en el tiempo $t + 1$. La sección de salida da el valor de y en el tiempo t para cada estado actual y condición de entrada.

La preparación de una tabla de estados requiere enumerar todas las posibles combinaciones binarias de estado actual y entradas. En este caso, tenemos ocho combinaciones binarias, de 000 a 111. Luego se determinan los valores de siguiente estado a partir del diagrama lógico o de las ecuaciones de estado. El siguiente estado del flip-flop A deberá satisfacer la ecuación de estado

$$A(t + 1) = Ax + Bx$$

La sección de siguiente estado en la columna A de la tabla de estados tiene tres unos donde el estado actual y el valor de entrada satisfacen las condiciones de que el estado actual de A y la

Tabla 5-2
Tabla de estados para el circuito de la figura 5-15

Estado actual		Entrada	Siguiente estado		Salida
A	*B*	*x*	*A*	*B*	*y*
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	1	0	0

entrada x son ambos 1, o el estado actual de B y la entrada x son ambos 1. De forma similar, el siguiente estado del flip-flop B se deduce de la ecuación de estado

$$B(t + 1) = A'x$$

y es igual a 1 cuando el estado actual de A es 0 y la entrada x es 1. La columna de salida se deduce de la ecuación de salida

$$y = Ax' + Bx'$$

La tabla de estados de un circuito secuencial con flip-flops tipo D se obtiene por el mismo procedimiento delineado en el ejemplo anterior. En general, un circuito secuencial con m flip-flops y n entradas necesita 2^{m+n} filas en la tabla de estados. Se hace una lista de los números binarios del 0 hasta $2^{m+n} - 1$ bajo las columnas de estado actual y entrada. La sección de siguiente estado tiene m columnas, una para cada flip-flop. Los valores binarios para el siguiente estado se deducen directamente de las ecuaciones de estado. La sección de salida tiene tantas columnas como variables de salida haya. Su valor binario se deduce del circuito o de la función booleana de la misma manera que se deduce una tabla de verdad.

A veces es conveniente expresar la tabla de estados en una forma un poco distinta. En la otra configuración, la tabla de estados sólo tiene tres secciones: estado actual, siguiente estado y salida. Las condiciones de entrada se enumeran en las secciones de siguiente estado y salida. En la tabla 5-3 se repite la tabla de estados de la tabla 5-2, en el segundo formato. Para cada estado actual, hay dos siguientes estados y salidas posibles, dependiendo del valor de la entrada. Una forma podría ser preferible a la otra, dependiendo de la aplicación.

Tabla 5-3
Segunda forma de la tabla de estados

Estado actual	Siguiente estado		Salida	
	$x = 0$	$x = 1$	$x = 0$	$x = 1$
AB	*AB*	*AB*	*y*	*y*
00	00	01	0	0
01	00	11	1	0
10	00	10	1	0
11	00	10	1	0

Diagrama de estados

La información contenida en una tabla de estados se representa gráficamente en forma de diagrama de estados. En este tipo de diagramas, un estado se representa con un círculo, y las transiciones entre estados se indican con flechas que conectan a los círculos. En la figura 5-16 se aprecia el diagrama de estados del circuito secuencial de la figura 5-15. El diagrama de estados proporciona la misma información que la tabla de estados y se obtiene directamente de la tabla 5-2 o 5-3. El número binario dentro de cada círculo identifica el estado de los flip-flops. Las flechas se rotulan con dos números binarios separados por una diagonal. Primero se da el valor de entrada durante el estado actual, y el número después de la diagonal indica la salida durante el estado actual, con esa entrada. (Es importante recordar que el valor de bit indicado para la salida a lo largo de la flecha se da durante el estado actual y con la entrada indicada, y nada tiene que ver con la transición al siguiente estado.) Por ejemplo, la flecha del estado 00 a 01 lleva el rótulo 1/0, lo que significa que cuando el circuito secuencial está en el estado actual 00 y la entrada es 1, la salida es 0. Después del siguiente ciclo de reloj, el circuito pasa al siguiente estado, 01. Si la entrada cambia a 0, la salida será 1, pero si la entrada sigue siendo 1, la salida se mantendrá en 0. Esta información se obtiene del diagrama de estados siguiendo las dos flechas que salen del círculo correspondiente al estado 01. Una flecha que conecta a un círculo consigo mismo indica que no hay cambio de estado.

No hay diferencia entre una tabla de estados y un diagrama de estados, como no sea en la forma de representación. La tabla de estados se deduce más fácilmente de un diagrama lógico dado y la ecuación de estado. El diagrama de estados se sigue directamente de la tabla de estados. El diagrama de estados muestra una perspectiva gráfica de las transiciones de estado y es la forma más apropiada para interpretar el funcionamiento del circuito, si quien lo interpreta es un ser humano. Por ejemplo, el diagrama de estados de la figura 5-16 indica claramente que, partiendo del estado 00, la salida será 0 en tanto la entrada se mantenga en 1. La primera entrada 0 después de una serie de unos da una salida de 1 y transfiere al circuito de vuelta al estado inicial 00.

Ecuaciones de entrada de flip-flops

El diagrama lógico de un circuito secuencial consiste en flip-flops y compuertas. Las interconexiones de compuertas forman un circuito combinacional y podrían especificarse algebraicamente con expresiones booleanas. El conocimiento del tipo de flip-flops y una lista de las expresiones booleanas del circuito combinacional proporcionan la información necesaria para dibujar el dia-

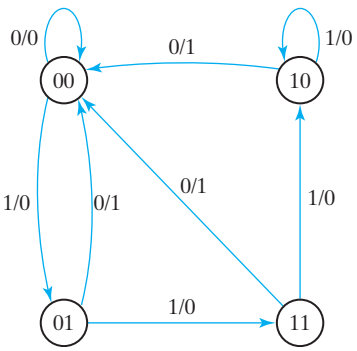


FIGURA 5-16
Diagrama de estados del circuito de la figura 5-15

grama lógico del circuito secuencial. La parte del circuito combinacional que genera salidas externas se describe algebraicamente con un conjunto de funciones booleanas llamadas *ecuaciones de salida*. La parte del circuito que genera las entradas a los flip-flops se describe algebraicamente con un conjunto de funciones booleanas llamadas *ecuaciones de entrada* de flip-flops (o *ecuaciones de excitación*). Adoptaremos la convención de usar el símbolo de entrada de flip-flop para denotar la variable de ecuación de entrada y un subíndice para indicar el nombre de la salida de flip-flop. Por ejemplo, la ecuación de entrada siguiente especifica la compuerta OR con entradas x y y conectada a la entrada D de un flip-flop cuya salida se rotula con el símbolo Q:

$$D_Q = x + y$$

El circuito secuencial de la figura 5-15 consta de dos flip-flops D, A y B, una entrada x y una salida y. El diagrama lógico del circuito se expresa algebraicamente con dos ecuaciones de entrada de flip-flops y una ecuación de salida:

$$\begin{aligned} D_A &= Ax + Bx \\ D_B &= A'x \\ y &= (A + B)x' \end{aligned}$$

Las tres ecuaciones proporcionan la información necesaria para dibujar el diagrama lógico del circuito secuencial. El símbolo D_A especifica un flip-flop D rotulado A. D_B especifica un segundo flip-flop D rotulado B. Las expresiones booleanas asociadas a estas dos variables, y la expresión de la salida y, especifican la parte de circuito combinacional del circuito secuencial.

Las ecuaciones de entrada de flip-flop son una forma algebraica conveniente para especificar el diagrama lógico de un circuito secuencial. Implican el tipo de flip-flop con el símbolo de letra, y especifican cabalmente el circuito combinacional que alimenta a los flip-flops. Cabe señalar que la expresión de la ecuación de entrada de un flip-flop D es idéntica a la expresión de la ecuación de estado correspondiente. Ello se debe a la ecuación característica que iguala el siguiente estado al valor de la entrada D: $Q(t + 1) = D_Q$.

Análisis con flip-flops *D*

Resumiremos el procedimiento para analizar un circuito secuencial con reloj con flip-flops D utilizando un ejemplo sencillo. El circuito que queremos analizar se describe con la ecuación de entrada

$$D_A = A \oplus x \oplus y$$

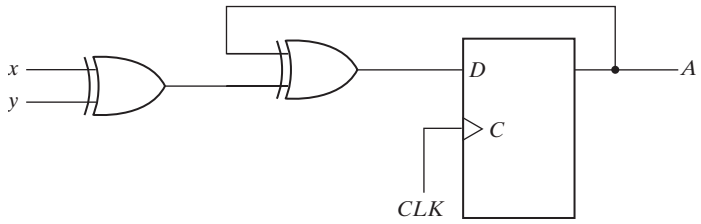
El símbolo D_A implica un flip-flop D con salida A. Las variables x y y son las entradas del circuito. No se dan ecuaciones de salida, así que la salida proviene implícitamente de la salida del flip-flop. Obtenemos el diagrama lógico de la ecuación de entrada [figura 5-17a)].

La tabla de estados tiene una columna para el estado actual del flip-flop A, dos columnas para las dos entradas y una columna para el siguiente estado de A. Los números binarios bajo Axy van de 000 a 111, como se observa en la figura 5-17b). Los valores de siguiente estado se obtienen de la ecuación de estado

$$A(t + 1) = A \oplus x \oplus y$$

La expresión especifica una función impar y es igual a 1 cuando sólo una variable es 1 o cuando las tres variables son 1. Esto se indica en la columna de siguiente estado de A.

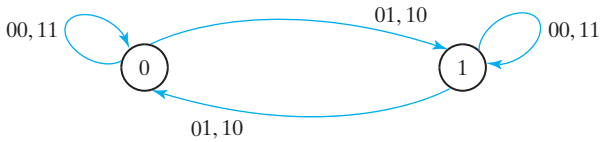
El circuito tiene un flip-flop y dos estados. El diagrama de estados consiste en dos círculos, uno para cada estado [figura 5-17c)]. El estado actual y la salida pueden ser 0 o 1, como indica el número dentro de los círculos. No se necesita una diagonal en las flechas porque no hay salida de circuito combinacional. Las dos entradas pueden tener cuatro posibles combinacio-



a) Diagrama de circuito

Estado actual	Salidas		Siguiente estado
A	*x*	*y*	*A*
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

b) Tabla de estados



c) Diagrama de estados

FIGURA 5-17
Circuito secuencial con flip-flop *D*

nes para cada estado. Las dos combinaciones de entrada durante cada transición de estado se separan con una coma para simplificar la notación.

Análisis con flip-flops *JK*

Una tabla de estados consta de cuatro secciones: estado actual, entradas, siguiente estado y salidas. Las dos primeras se obtienen enumerando todas las combinaciones binarias. La sección de salida se determina con base en las ecuaciones de salida. Los valores de siguiente estado se obtienen de las ecuaciones de estado. En el caso de un flip-flop tipo *D*, la ecuación de estado es igual a la ecuación de entrada. Cuando se usa un flip-flop de otro tipo, como un *JK* o un *T*, es necesario consultar la tabla característica o ecuación característica correspondiente para obtener los valores de siguiente estado. Ilustraremos el procedimiento primero utilizando la tabla característica, y luego lo repetiremos usando la ecuación característica.

Los valores de siguiente estado de un circuito secuencial que usa flip-flops tipo *JK* o *T* se deducen con el procedimiento siguiente:

1. Determine las ecuaciones de entrada del flip-flop en términos del estado actual y las variables de entrada.
2. Enumere los valores binarios de cada ecuación de entrada.
3. Use la tabla característica del flip-flop en cuestión para determinar los valores de siguiente estado de la tabla de estados.

Por ejemplo, consideremos el circuito secuencial con dos flip-flops *JK*, *A* y *B*, y una entrada, *x*, que se ilustra en la figura 5-18. El circuito no tiene salidas, de modo que la tabla de estados no necesita una columna de salida. (Las salidas de los flip-flops se consideran como las salidas en este caso.)

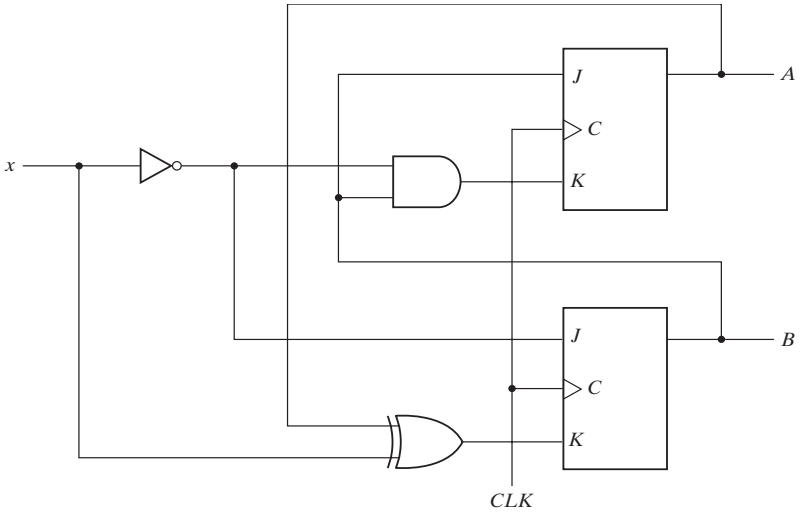


FIGURA 5-18
Circuito secuencial con flip-flop *JK*

El circuito se puede especificar con las ecuaciones de entrada de flip-flop

$$J_A = B \qquad K_A = Bx'$$
$$J_B = x' \qquad K_B = A'x + Ax' = A \oplus x$$

La tabla de estados del circuito secuencial aparece en la tabla 5-4. Las columnas de estado actual y entrada presentan las ocho combinaciones binarias. Los valores binarios de las columnas rotuladas *entradas de flip-flops* no forman parte de la tabla de estados, pero las necesitamos para evaluar el siguiente estado, como especifica el paso 2 del procedimiento. Estos valores binarios se obtienen directamente de las cuatro ecuaciones de entrada de forma similar a como se deduce una tabla de verdad de una expresión booleana. El siguiente estado de cada flip-flop se evalúa a partir de las entradas J y K correspondientes y la tabla característica del flip-flop JK dada en la tabla 5-1. Debemos considerar cuatro casos. Cuando $J = 1$ y $K = 0$, el siguiente estado es 1. Cuando $J = 0$ y $K = 1$, el siguiente estado es 0. Cuando $J = K = 0$, no hay cambio de estado y el valor de siguiente estado es igual al de estado actual. Cuando $J = K = 1$,

Tabla 5-4
Tabla de estados de un circuito secuencial con flip-flops JK

Estado Actual		Entrada	Siguiente estado		Entradas de flip-flop			
A	*B*	*x*	*A*	*B*	J_A	K_A	J_B	K_B
0	0	0	0	1	0	0	1	0
0	0	1	0	0	0	0	0	1
0	1	0	1	1	1	1	1	0
0	1	1	1	0	1	0	0	1
1	0	0	1	1	0	0	1	1
1	0	1	1	0	0	0	0	0
1	1	0	0	0	1	1	1	1
1	1	1	1	1	1	0	0	0

el bit de siguiente estado es el complemento del bit de estado actual. Se dan ejemplos de los últimos dos casos de la tabla cuando el estado actual AB es 10 y la entrada x es 0. JA y KA son ambos 0 y el estado actual de A es 1. Por tanto, el siguiente estado de A sigue siendo el mismo y es igual a 1. En la misma fila de la tabla, JB y KB son ambos 1. Puesto que el estado actual de B es 0, el siguiente estado de B se complementará y cambiará a 1.

También es posible obtener los valores de siguiente estado evaluando las ecuaciones de estado de la ecuación característica, mediante el procedimiento siguiente:

1. Obtenga las ecuaciones de entrada de flip-flop en términos del estado actual y las variables de entrada.
2. Sustituya las ecuaciones de salida en la ecuación característica del flip-flop para obtener las ecuaciones de estado.
3. Use las ecuaciones de estado correspondientes para determinar los valores de siguiente estado de la tabla de estados.

Las ecuaciones de entrada para los dos flip-flops JK de la figura 5-18 se presentaron en la página anterior. Obtenemos las ecuaciones características de los flip-flops sustituyendo A o B por el nombre del flip-flop, en vez de Q:

$$A(t + 1) = JA' + K'A$$
$$B(t + 1) = JB' + K'B$$

Sustituyendo los valores de J_A y K_A de las ecuaciones de entrada, se obtiene la ecuación de estado para A:

$$A(t + 1) = BA' + (Bx')'A = A'B + AB' + Ax$$

La ecuación de estado proporciona los valores de bits para la columna de siguiente estado de A en la tabla de estados. De forma similar, la ecuación de estado del flip-flop B se deduce de la ecuación característica sustituyendo los valores de J_B y K_B:

$$B(t + 1) = x'B' + (A \oplus x)'B = B'x' + ABx + A'Bx'$$

La ecuación de estado proporciona los valores de bit para la columna de siguiente estado de B en la tabla de estados. Observe que las columnas de entradas de flip-flop de la tabla 5-4 no se necesitan cuando se usan ecuaciones de estado.

El diagrama de estados del circuito secuencial se presenta en la figura 5-19. Vemos que, como el circuito no tiene salidas, las flechas que salen de los círculos se marcan con un solo número binario para indicar el valor de la entrada x.

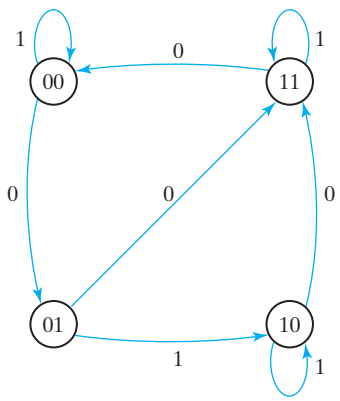


FIGURA 5-19
Diagrama de estados del circuito de la figura 5-18

Análisis con flip-flops *T*

El análisis de un circuito secuencial con flip-flops *T* sigue el mismo procedimiento que delineamos para los flip-flops *JK*. Los valores de siguiente estado de la tabla de estados se obtienen utilizando la tabla característica de la tabla 5-1 o bien la ecuación característica

$$Q(t + 1) = T \oplus Q = T'Q + TQ'$$

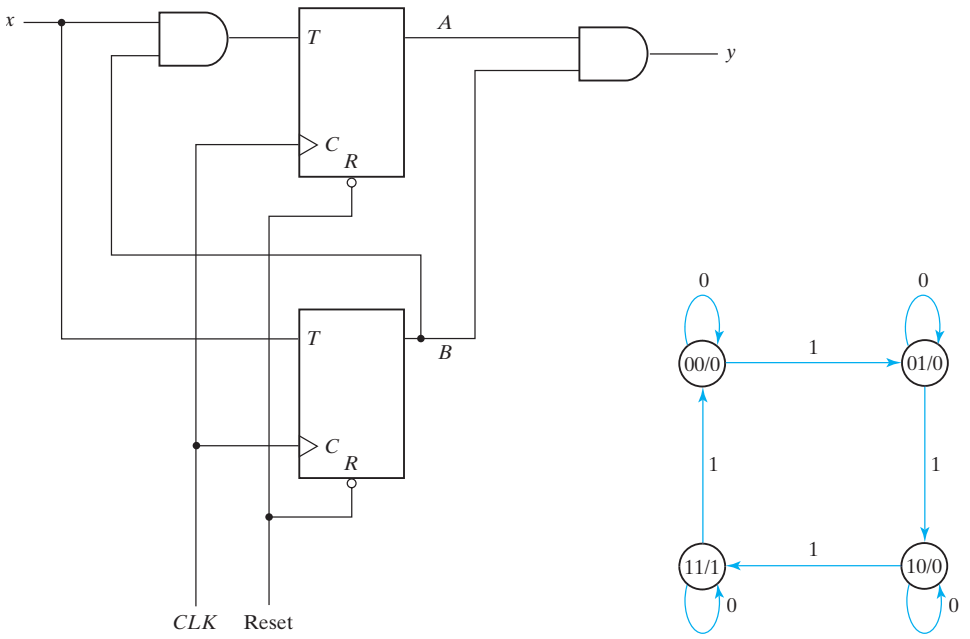
Consideremos el circuito secuencial de la figura 5-20. Tiene dos flip-flops A y B, una entrada x y una salida y. Se describe algebraicamente con dos ecuaciones de entrada y una de salida:

$$T_A = Bx$$
$$T_B = x$$
$$y = AB$$

La tabla de estados del circuito se presenta en la tabla 5-5. Los valores de y se obtienen de la ecuación de salida. Los valores para el siguiente estado se deducen de las ecuaciones de estado sustituyendo T_A y T_B en las ecuaciones características para dar

$$A(t + 1) = (Bx)'A + (Bx)A' = AB' + Ax' + A'Bx$$
$$B(t + 1) = x \oplus B$$

Los valores de siguiente estado para A y B en la tabla de estados se obtienen de las expresiones para las dos ecuaciones de estado.



a) Diagrama de circuito b) Diagrama de estados

FIGURA 5-20
Circuito secuencial con dos flip-flops *T*

Tabla 5-5
Tabla de estados para un circuito secuencial con flip-flops T

Estado actual		Entrada	Siguiente estado		Salida
A	*B*	*x*	*A*	*B*	*y*
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	1	0
0	1	1	1	0	0
1	0	0	1	0	0
1	0	1	1	1	0
1	1	0	1	1	1
1	1	1	0	0	1

El diagrama de estados del circuito se reproduce en la figura 5-20b). En tanto la entrada x sea 1, el circuito se comportará como un contador binario con la sucesión de estados 00, 01, 10, 11 y de vuelta a 00. Cuando $x = 0$, el circuito permanece en el mismo estado. La salida y es 1 cuando el estado actual es 11. Aquí la salida depende únicamente del estado actual y es independiente de la entrada. Los dos valores separados por una diagonal dentro de cada círculo corresponden al estado actual y a la salida.

Modelos Mealy y Moore

El modelo más general de un circuito secuencial tiene entradas, salidas y estados internos. Se acostumbra distinguir entre dos modelos de circuitos secuenciales: el modelo Mealy y el modelo Moore. Difieren en la forma en que se genera la salida. En el modelo Mealy, la salida es función tanto del estado actual como de la entrada. En el modelo Moore, la salida sólo es función del estado actual. Al tratar los dos modelos, algunos libros y otras fuentes técnicas ven el circuito secuencial como una máquina de estados finitos (FSM, *finite state machine*). El modelo Mealy de un circuito secuencial es una FSM Mealy o máquina Mealy. El modelo Moore es una FSM Moore o máquina Moore.

En la figura 5-15 se ilustra un ejemplo de modelo Mealy. La salida y es función tanto de la entrada x como del estado actual de A y B. El diagrama de estados correspondiente de la figura 5-16 muestra los valores de entrada y de salida separados por una diagonal sobre las flechas entre los estados.

En la figura 5-18 aparece un ejemplo de modelo Moore. Aquí la salida es función únicamente del estado actual. El diagrama de estados correspondiente de la figura 5-19 sólo tiene entradas marcadas sobre las flechas. Las salidas son los estados de flip-flop indicados dentro de los círculos. Otro ejemplo de modelo Moore es el circuito secuencial de la figura 5-20. La salida depende únicamente de los valores de los flip-flops, así que sólo es función del estado actual. El valor de la entrada se marca en el diagrama de estados junto a las flechas, mientras que el valor de salida se indica dentro del círculo junto con el estado actual.

En un modelo Moore, las salidas del circuito secuencial se sincronizan con el reloj porque sólo dependen de salidas de flip-flop que están sincronizadas con el reloj. En un modelo Mealy, las salidas podrían cambiar si las entradas cambian durante el ciclo de reloj. Además, las sali-

das podrían tener valores falsos momentáneos debidos al retardo entre el momento en que las entradas cambian y el momento en que cambian las salidas de flip-flop. Para sincronizar un circuito tipo Mealy, las entradas del circuito secuencial se deben sincronizar con el reloj y las salidas se deben muestrear únicamente durante el borde del reloj.

5-5 HDL PARA CIRCUITOS SECUENCIALES

Presentamos el lenguage de descripción de hardware (HDL) Verilog en la sección 3-9. En la sección 4-11 se hizo una descripción de los circuitos combinacionales y una introducción al modelado de comportamiento. En esta sección seguiremos estudiando el modelado de comportamiento y presentaremos ejemplos de descripciones de flip-flops y circuitos secuenciales.

Modelado de comportamiento

Hay dos tipos de enunciados de comportamiento en Verilog HDL: *inicial* y *siempre*. El comportamiento inicial se ejecuta una vez en el tiempo = 0. El comportamiento "siempre" se ejecuta una y otra vez hasta que la simulación termina. Los comportamientos se declaran dentro de los módulos con las palabras clave **initial** y **always** seguidas de un enunciado o bloque de enunciados delimitado por las palabras clave **begin** y **end.** Un módulo puede contener un número arbitrario de enunciados **initial** o **always.** Estos enunciados se ejecutan de forma concurrente a partir del tiempo 0.

Un enunciado **initial** se ejecuta una sola vez. Inicia su ejecución al principio de la simulación y termina una vez que han terminado de ejecutarse todos los enunciados. Como se mencionó al final de la sección 4-11, el enunciado **initial** es útil para generar señales de entrada a fin de simular un diseño. Al simular un circuito secuencial, es necesario generar una fuente de reloj para disparar los flip-flops. He aquí dos posibles formas de incluir un reloj de operación libre:

```
initial
   begin
      clock = 1'b0 ;
      repeat (30)
      #10 clock = ~ clock;
   end
```

```
initial
    begin
      clock = 1'b0;
      #300  $finish;
    end
always
      #10  clock = ~clock;
```

En la primera versión, el bloque **initial** está encerrado entre las palabras clave **begin** y **end.** El reloj se pone en 0 en el tiempo = 0; se complementa cada 10 unidades de tiempo y se repite 30 veces. Esto produce 15 ciclos de reloj, cada uno con una duración de 20 unidades de tiempo. En la segunda versión, el bloque **initial** pone el reloj en 0 en el tiempo = 0. Después de 10 unidades de tiempo, el enunciado **always** complementa repetidamente el reloj cada 10 unidades de tiempo, lo que proporciona un reloj con una duración de 20 unidades de tiempo. La simulación termina en respuesta a la tarea del sistema **$finish** en el tiempo = 300.

El enunciado **always** se controla con retardos que esperan cierto tiempo o esperan hasta que ciertas condiciones se cumplen o se presentan ciertos sucesos. Aquí sólo se explicará la condición de control por suceso. Este tipo de enunciado tiene la forma

```
always @ (expresión de control de sucesos)
Enunciados procedimentales de asignación.
```

La expresión de control de sucesos especifica la condición que debe presentarse para activar la ejecución de los enunciados procedimentales de asignación. Las variables del miembro iz-

quierdo de los enunciados procedimentales deben ser del tipo de datos **reg** y declararse como tales. El miembro derecho puede ser cualquier expresión que produzca un valor empleando operadores definidos en Verilog.

La expresión de control de sucesos (también llamada lista de sensibilidad) especifica los sucesos que deben darse para iniciar la ejecución de los enunciados procedimentales del bloque **always.** Los enunciados de ese bloque se ejecutan sucesivamente y la ejecución se suspende después del último enunciado. Luego, el enunciado **always** espera otra vez que se presente un suceso. Aquí consideraremos dos tipos de sucesos: sucesos sensibles al nivel y sucesos disparados por flanco. Los primeros se dan en los circuitos combinacionales y en latches. Por ejemplo, el enunciado

```
always @ (A or B or Reset)
```

hace que se ejecuten los enunciados procedimentales del bloque **always** si hay un cambio en *A* o en *B* o en *Reset*. En los circuitos secuenciales sincrónicos, los flip-flops sólo deben cambiar como respuesta a una transición de pulso de reloj. La transición podría ser un disparador de borde positivo o de borde negativo. Verilog HDL maneja estas condiciones con dos palabras clave: **posedge** y **negedge.** Por ejemplo,

```
always @(posedge clock or negedge reset)
```

hace que se ejecuten los enunciados procedimentales sólo si el reloj sufre una transición positiva o si *Reset* pasa por una transición negativa.

Los enunciados procedimentales son enunciados contenidos en un enunciado **initial** o **always.** Esto contrasta con las asignaciones continuas que vimos en la sección 4-11 al hablar del modelado de flujo de datos, donde el enunciado se evalúa continuamente. Hay dos tipos de enunciados procedimentales, *bloqueadores* y *no bloqueadores*, y se distinguen por los símbolos que usan. Los enunciados bloqueadores emplean el símbolo (=) como operador de asignación, mientras que los no bloqueadores usan el operador (<=). Los enunciados de asignación bloqueadores se ejecutan sucesivamente en el orden en que aparecen en un bloque secuencial. Los enunciados no bloqueadores evalúan las expresiones del miembro derecho pero no efectúan la asignación al miembro izquierdo sino hasta que se han evaluado todas las expresiones. Se entenderán mejor los dos tipos de asignaciones con un ejemplo. Consideremos las dos asignaciones procedimentales bloqueadoras:

```
B = A
C = B + 1
```

El primer enunciado transfiere *A* a *B*. El segundo enunciado incrementa el nuevo valor de *B* y lo transfiere a *C*. Al final, *C* contiene el valor de $A + 1$.

Consideremos ahora los dos enunciados en forma de asignaciones no bloqueadoras:

```
B <= A
C <= B + 1
```

Cuando se ejecutan los enunciados, las expresiones de la derecha se evalúan y se almacenan en un lugar temporal. El valor de *A* se guarda en un lugar y el valor de $B + 1$ se guarda en otro. Una vez que se han evaluado y almacenado todas las expresiones del bloque secuencial, se efectúa la asignación a los destinos de la izquierda. En este caso, *C* contendrá el valor original de *B* más uno. Casi todos los ejemplos de este capítulo y el siguiente pueden usar enunciados bloqueadores. Los enunciados no bloqueadores son indispensables cuando se efectúa diseño en el nivel de transferencia de registros, como se verá en el capítulo 8.

Flip-flops y latches

Los ejemplos HDL 5-1 a 5-4 muestran descripciones de diversos flip-flops y un latch *D*. El latch *D* es transparente y responde a un cambio en la entrada de datos con un cambio en la salida en tanto la entrada de control esté habilitada. El módulo de descripción del latch *D* se presenta en el ejemplo HDL 5-1. Tiene dos entradas, *D* y *control*, y una salida, *Q*. Puesto que *Q* se evalúa en un enunciado procedimental, se le debe declarar como de tipo **reg.** Los latches responden a niveles de señal de entrada, así que las dos entradas se dan sin calificadores de borde (**posedge** o **negedge**) en la expresión de control de sucesos que sigue al símbolo @ en el enunciado **always.** Sólo hay un enunciado de asignación procedimental bloqueador, y especifica la transferencia de la entrada *D* a la salida *Q* si el control es 1 lógico. Advierta que este enunciado se ejecuta cada vez que hay un cambio en *D* si el control es 1.

El ejemplo HDL 5-2 describe dos flip-flops *D* de borde positivo en dos módulos. El primero responde únicamente al reloj; el segundo incluye una entrada de restablecimiento asincrónico. La salida *Q* debe declararse como de tipo de datos **reg** además de darse como salida. El motivo es que es una salida de destino en un enunciado procedimental de asignación. La palabra clave **posedge** garantiza que la transferencia de la entrada *D* a *Q* sólo se dará durante la transición de borde positivo de CLK. Un cambio en *D* en cualquier otro momento no cambia *Q*.

Ejemplo HDL 5-1

```
//Descripción de un latch D (Véase la figura 5-6)
module D_latch (Q,D,control);
   output Q;
   input D,control;
   reg Q;
   always @ (control or D)
   if (control) Q = D;      //Igual que: if (control == 1)
endmodule
```

Ejemplo HDL 5-2

```
//Flip-flop D
module D_FF (Q,D,CLK);
   output Q;
   input D,CLK;
   reg Q;
   always @ (posedge CLK)
     Q = D;
endmodule

//Flip-flop D con restablecimiento asincrónico.
module DFF (Q,D,CLK,RST);
   output Q;
   input D,CLK,RST;
   reg Q;
   always @(posedge CLK or negedge RST)
     if (~RST) Q = 1'b0;     // Igual a: if (RST == 0)
     else Q = D;
endmodule
```

El segundo módulo incluye una entrada de restablecimiento asincrónico además del reloj sincrónico. Se usa una forma especial del enunciado **if** para generar este tipo de flip-flop. La expresión de sucesos después del símbolo @ en el enunciado **always** puede tener cualquier número de sucesos de borde, sean **posedge** o **negedge.** Uno de ellos debe ser un suceso de reloj. Los demás especifican condiciones en las que debe ejecutarse lógica asincrónica. Cada enunciado **if** o **else if** de los enunciados procedimentales de asignación corresponde a un suceso asincrónico. El último enunciado **else** corresponde al suceso de reloj. Hay dos sucesos de borde en el segundo módulo del ejemplo 5-2. El suceso **negedge** RST (restablecimiento) es asincrónico porque equivale al suceso **if** (~RST). En tanto RST sea 0, Q se pondrá en 0. Si CLK tiene una transición positiva, su efecto se bloqueará. Sólo si RST = 1 podrá el suceso de reloj **posedge** transferir sincrónicamente D a Q.

Por lo regular es necesario que los flip-flops incluyan una señal de entrada de restablecimiento (o preestablecimiento, *preset*); de lo contrario, no se podrá determinar el estado inicial del circuito secuencial. Los circuitos secuenciales no se pueden probar con simulación HDL si no es posible asignar un estado inicial con una señal de entrada.

El ejemplo HDL 5-3 describe la construcción de un flip-flop T o JK a partir de un flip-flop D y compuertas. El circuito se describe utilizando las ecuaciones características de los flip-flops:

$$Q(t+1) = Q \oplus T \qquad \text{para un flip-flop } T$$
$$Q(t+1) = JQ' + K'Q \qquad \text{para un flip-flop } JK$$

Ejemplo HDL 5-3

```
//Flip-flop T hecho con flip-flop D y compuertas
module TFF (Q,T,CLK,RST);
   output Q;
   input T,CLK,RST;
   wire DT;
   assign DT = Q ^ T ;
//Crear ejemplar del flip-flop D
   DFF TF1 (Q,DT,CLK,RST);
endmodule

//Flip-flop JK hecho con flip-flop D y compuertas
module JKFF (Q,J,K,CLK,RST);
   output Q;
   input J,K,CLK,RST;
   wire JK;
   assign JK = (J & ~Q) | (~K & Q);
//Crear ejemplar de flip-flop D
   DFF JK1 (Q,JK,CLK,RST);
endmodule

//Flip-flop D
module DFF (Q,D,CLK,RST);
   output Q;
   input D,CLK,RST;
   reg Q;
   always @ (posedge CLK or negedge RST)
     if (~RST) Q = 1'b0;
     else Q = D;
endmodule
```

Ejemplo HDL 5-4

```
//Descripción funcional de flip-flop JK
module  JK_FF (J,K,CLK,Q,Qnot);
   output Q,Qnot;
   input  J,K,CLK;
   reg  Q;
   assign Qnot = ~ Q ;
   always @ (posedge CLK)
            case ({J,K})
              2'b00: Q = Q;
              2'b01: Q = 1'b0;
              2'b10: Q = 1'b1;
              2'b11: Q = ~ Q;
            endcase
endmodule
```

El primer módulo TFF describe un flip-flop *T* creando un ejemplar de DFF (la creación de ejemplares se explica en la sección 4-11). A la declaración **wire** DT se asigna el OR exclusivo de *Q* y *T*, lo cual es necesario para convertir un flip-flop *D* en un flip-flop *T*. La creación de un ejemplar en la que el valor de DT sustituya a *D* en el módulo DFF produce el flip-flop *T* requerido. El flip-flop *JK* se especifica de forma similar utilizando su ecuación característica para definir lo que sustituirá a *D* en el ejemplar de DFF.

El ejemplo HDL 5-4 muestra otra forma de describir un flip-flop *JK*. Aquí optamos por describirlo empleando la tabla característica en lugar de la ecuación característica. La condición de ramificación multivía **case** examina el número de dos bits que se obtiene concatenando los bits de *J* y *K*. El valor **case** ({J,K}) se evalúa y compara con los valores de la lista de enunciados que sigue. Se ejecuta el primer valor que coincide con la condición verdadera. Puesto que la concatenación de *J* y *K* produce un número de dos bits, puede ser igual a 00, 01, 10 o 11. El primer bit da el valor de *J*, y el segundo, el de *K*. Las cuatro posibles condiciones especifican el valor del siguiente estado de *Q* después de la aplicación de un reloj de borde positivo.

Diagrama de estados

El funcionamiento de los circuitos secuenciales se describe en HDL en el mismo formato que los diagramas de estados. En el ejemplo HDL 5-5 se presenta un diagrama de estados de modelo Mealy. La entrada, salida, reloj y restablecimiento se declaran de la forma acostumbrada. El estado de los flip-flops se declara con los identificadores `Prstate` (estado actual) y `Nxtstate` (siguiente estado). Estas variables contienen el valor de estado del circuito secuencial. La asignación binaria de estado se efectúa con un enunciado de parámetro. (Verilog permite definir constantes en un módulo con la palabra clave **parameter**). Se asignan los números binarios 00 a 11 a los cuatro estados *S*0 a *S*3. La notación *S*2 = 2 ' b10 es preferible a la alternativa, *S*2 = 2. La primera usa dos bits para almacenar la constante. La segunda notación produce un número binario de 32 (o 64) bits.

La descripción HDL utiliza tres bloques **always** que se ejecutan de forma concurrente e interactúan a través de variables en común. El primer enunciado **always** restablece el circuito al estado inicial *S*0 = 00 y especifica la operación sincrónica con reloj. El enunciado

Ejemplo HDL 5-5

```
//Diagrama de estados Mealy (figura 5-16)
module Mealy_mdl (x,y,CLK,RST);
  input x,CLK,RST;
  output y;
  reg y;
  reg [1:0] Prstate, Nxtstate;
  parameter S0 = 2'b00, S1 = 2'b01, S2 = 2'b10, S3 = 2'b11;
   always @ (posedge CLK or negedge RST)
      if (~RST) Prstate = S0;  //Iniciar en estado S0
      else Prstate = Nxtstate; //Operaciones de reloj
   always @ (Prstate or x)     //Determinar siguiente estado
         case (Prstate)
            S0: if (x) Nxtstate = S1;
                  else Nxtstate = S0;
            S1: if (x) Nxtstate = S3;
                  else Nxtstate = S0;
            S2: if (~x)Nxtstate = S0;
                  else Nxtstate = S2;
            S3: if (x) Nxtstate = S2;
                  else Nxtstate = S0;
         endcase
   always @ (Prstate or x)     //Evaluar salida
         case (Prstate)
            S0: y = 0;
            S1: if (x) y = 1'b0; else y = 1'b1;
            S2: if (x) y = 1'b0; else y = 1'b1;
            S3: if (x) y = 1'b0; else y = 1'b1;
         endcase
  endmodule
```

`Prstate = Nxtstate` se ejecuta únicamente en respuesta a una transición de borde positivo del reloj. Esto implica que cualquier cambio que sufra el valor de `Nxtstate` en el segundo bloque **always** se transferirá a `Prstate` como resultado de un suceso **posedge.** El segundo bloque **always** determina la transición al siguiente estado en función del estado actual y de la entrada. La condición de ramificación multivía sigue la sucesión especificada en el diagrama de estados de la figura 5-16. El tercer bloque **always** evalúa la salida en función del estado actual y de la entrada. Aunque se presenta aparte por claridad, podría combinarse con el segundo bloque. Observe que el valor de la salida *y* podría cambiar si cambia el valor de la entrada *x* mientras el circuito está en cualquier estado dado.

En el ejemplo HDL 5-6 se describe un ejemplo de diagrama de estados de modelo Moore. Este ejemplo demuestra que es posible especificar las transiciones de estado con un solo bloque **always.** El estado actual del circuito se identifica con la variable `state`. Las transiciones de estado se presentan con el CLK **posedge** de acuerdo con las condiciones dadas en los enunciados **case.** La salida del circuito es independiente de la entrada y se toma directamente de las salidas de los flip-flops. La salida de dos bits *AB* se especifica con un enunciado **assign** y es igual al valor del estado actual.

Ejemplo HDL 5-6

```
//Diagrama de estados de Moore (figura 5-19)
module Moore_mdl (x,AB,CLK,RST);
   input x,CLK,RST;
   output [1:0]AB;
   reg [1:0] state;
   parameter S0 = 2'b00, S1 = 2'b01, S2 = 2'b10, S3 = 2'b11;
      always @ (posedge CLK or negedge RST)
         if (~RST) state = S0;  //Iniciar en estado S0
         else
         case (state)
           S0: if (~x) state = S1; else state = S0;
           S1: if (x)  state = S2; else state = S3;
           S2: if (~x) state = S3; else state = S2;
           S3: if (~x) state = S0; else state = S3;
         endcase
   assign AB = state;        //Salida de flip-flops
endmodule
```

Descripción estructural

Los circuitos combinacionales se describen en HDL empleando enunciados de flujo de datos o en el nivel de compuertas. En el caso de los circuitos secuenciales, el funcionamiento de los flip-flops se describe con enunciados de comportamiento. Puesto que un circuito secuencial consta de flip-flops y compuertas, su estructura se puede describir con una combinación de enunciados de flujo de datos y de comportamiento. Los flip-flops se describen con un enunciado **always.** La parte combinacional se describe con enunciados **assign** y ecuaciones booleanas. Los módulos individuales se pueden combinar creando ejemplares.

La descripción estructural de un circuito secuencial se ilustra en el ejemplo HDL 5-7. El ejemplo tiene dos módulos. El primero describe el circuito de la figura 5-20a). El segundo describe un flip-flop *T*. Otro módulo genera un estímulo para probar el funcionamiento del circuito. El circuito secuencial es un contador binario de dos bits controlado por la entrada *x*. La salida *y* es 1 cuando la cuenta llega al 11 binario. Se incluyen los flip-flops *A* y *B* como salidas para verificar su funcionamiento. Las ecuaciones de entrada de los flip-flops y la ecuación de salida se evalúan con enunciados **assign** que tienen las expresiones booleanas correspondientes. Luego se crean ejemplares del flip-flop *T* empleando TA y TB definidos por las ecuaciones de entrada.

El segundo módulo describe el flip-flop *T*. La entrada RST restablece el flip-flop a 0 con una señal negativa. El funcionamiento del flip-flop se especifica con su ecuación característica $Q(t + 1) = Q \oplus T$.

El módulo de estímulo alimenta entradas al circuito para verificar la respuesta de salida. El primer bloque **initial** produce ocho ciclos de reloj con un periodo de 10 ns. El segundo bloque **initial** especifica un cambio alterno de la entrada *x* que se presenta en la transición de borde negativo del reloj. El resultado de la simulación se presenta en la figura 5-21. Las salidas *A* y *B* pasan por la sucesión binaria 00, 01, 10, 11 y de vuelta a 00. El cambio en el conteo se da durante un borde positivo del reloj siempre que $x = 1$. Si $x = 0$, la cuenta no cambia. La salida *y* es 1 cuando tanto *A* como *B* son 1. Esto verifica el funcionamiento del circuito.

Ejemplo HDL 5-7

```
//Descripción estructural de circuito secuencial
//Véase la figura 5-20a)
module Tcircuit (x,y,A,B,CLK,RST);
   input x,CLK,RST;
   output y,A,B;
   wire TA,TB;
//Ecuaciones de entrada de flip-flop
   assign TB = x,
          TA = x & B;
//Ecuación de salida
   assign y = A & B;
//Se crean ejemplares de flip-flops T
   T_FF BF (B,TB,CLK,RST);
   T_FF AF (A,TA,CLK,RST);
endmodule

//Flip-flop T
module T_FF (Q,T,CLK,RST);
   output Q;
   input T,CLK,RST;
   reg Q;
     always @ (posedge CLK or negedge RST)
       if (~RST) Q = 1'b0;
       else Q = Q ^ T;
endmodule

//Estímulo para probar el circuito secuencial
module testTcircuit;
  reg x,CLK,RST;  //entradas del circuito
  wire y,A,B;    //salida del circuito
  Tcircuit TC (x,y,A,B,CLK,RST);  // se crea un ejemplar
                                  // del circuito
  initial
     begin
         RST = 0;
         CLK = 0;
      #5 RST = 1;
         repeat (16)
      #5 CLK = ~CLK;
     end
  initial
     begin
          x = 0;
      #15 x = 1;
          repeat (8)
      #10 x = ~ x;
     end
endmodule
```

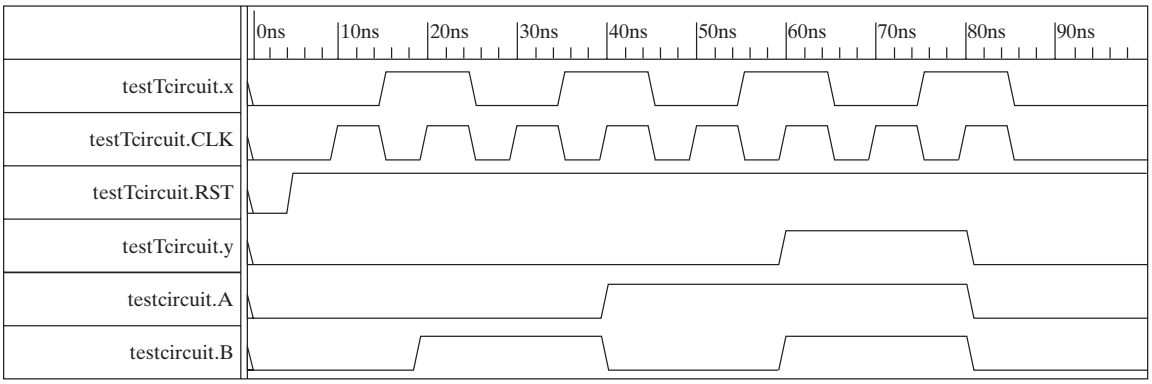


FIGURA 5-21
Salida de la simulación del ejemplo HDL 5-7

5-6 REDUCCIÓN Y ASIGNACIÓN DE ESTADOS

El análisis de circuitos secuenciales parte de un diagrama de circuitos y culmina en una tabla o diagrama de estados. El diseño de un circuito secuencial parte de un conjunto de especificaciones y culmina en un diagrama lógico. Presentaremos los procedimientos de diseño en la sección 5-7. En esta sección veremos ciertas propiedades de los circuitos secuenciales que podrían servir para reducir el número de compuertas y flip-flops durante el diseño.

Reducción de estados

La reducción en el número de flip-flops de un circuito secuencial se conoce como problema de *reducción de estados*. Los algoritmos de reducción de estados dan pie a procedimientos para reducir el número de estados de una tabla de estados, pero sin alterar los requisitos externos de entrada-salida. Puesto que m flip-flops producen 2^m estados, una reducción en el número de estados podría (o no) reducir el número de flip-flops. Un efecto impredecible al reducir el número de flip-flops es que a veces el circuito equivalente (con menos flip-flops) podría requerir más compuertas combinacionales.

Ilustraremos el procedimiento de reducción de estados con un ejemplo. Partiremos de un circuito secuencial cuya especificación se da en el diagrama de estados de la figura 5-22. En este ejemplo, sólo son importantes las sucesiones de entrada-salida; los estados internos sólo sirven para producir las sucesiones requeridas. Por ello, los estados marcados dentro de los círculos se denotan con letras en vez de sus valores binarios. Esto contrasta con un contador binario, en el que la sucesión de valores binarios de los estados mismos se toma como las salidas.

Hay un número infinito de sucesiones de entrada que podrían aplicarse al circuito; cada una produce una sucesión de salida única. Por ejemplo, consideremos la sucesión de entrada 01010110100 partiendo del estado inicial a. Cada entrada de 0 o 1 produce una salida de 0 o 1 y hace que el circuito pase al siguiente estado. Del diagrama de estados, obtenemos las su-

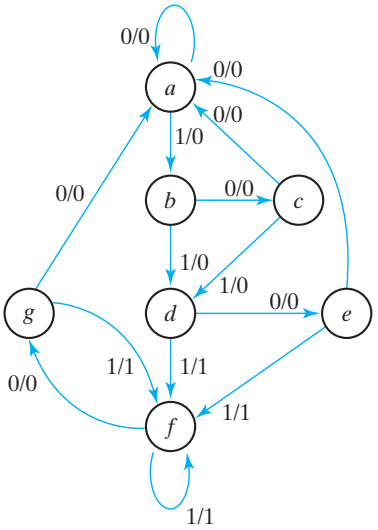


FIGURA 5-22
Diagrama de estados

cesiones de salida y de estados para la sucesión dada de entrada como sigue: con el circuito en el estado inicial *a*, una entrada de 0 produce una salida de 0 y el circuito permanece en el estado *a*. Con estado actual *a* y entrada de 1, la salida es 0 y el siguiente estado es *b*. Con estado actual *b* y entrada de 0, la salida es 0 y el siguiente estado es *c*. Continuando este proceso, se obtiene la sucesión total siguiente:

Estado	*a*	*a*	*b*	*c*	*d*	*e*	*f*	*f*	*g*	*f*	*g*	*a*
Entrada	0	1	0	1	0	1	1	0	1	0	0	
Salida	0	0	0	0	0	1	1	0	1	0	0	

En cada columna, tenemos el estado actual, el valor de entrada y el valor de salida. El siguiente estado aparece hasta arriba en la siguiente columna. Es importante darse cuenta de que, en este circuito, los estados mismos tienen importancia secundaria porque únicamente nos interesan las sucesiones de salida causadas por sucesiones de entrada.

Suponga ahora que hemos hallado un circuito secuencial cuyo diagrama de estados tiene menos de siete estados y queremos compararlo con el circuito cuyo diagrama de estados está dado por la figura 5-22. Si se aplican sucesiones de entrada idénticas a los dos circuitos y se producen salidas idénticas para todas las sucesiones de entrada, decimos que los dos circuitos son equivalentes (en lo que se refiere a entrada-salida), y podemos sustituir uno por el otro. El problema de la reducción de estados consiste en hallar formas de reducir el número de estados de un circuito secuencial sin alterar las relaciones de entrada-salida.

Ahora procedemos a reducir el número de estados de este ejemplo. Primero, necesitamos la tabla de estados; es más fácil aplicar los procedimientos de reducción de estados a una tabla que a un diagrama. La tabla de estados del circuito aparece en la tabla 5-6 y se obtiene directamente del diagrama de estados.

Tabla 5-6
Tabla de estados

	Siguiente estado		Salida	
Estado actual	$x = 0$	$x = 1$	$x = 0$	$x = 1$
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	f	0	1
e	a	f	0	1
f	g	f	0	1
g	a	f	0	1

Presentaremos aquí, sin demostrarlo, un algoritmo para reducir los estados de una tabla de estados plenamente especificada: "Decimos que dos estados son equivalentes si, para cada miembro del conjunto de entradas, dan exactamente la misma salida y pasan el circuito al mismo estado o a un estado equivalente". Si dos estados son equivalentes, uno de ellos puede eliminarse sin alterar las relaciones de entrada-salida.

Apliquemos ahora este algoritmo a la tabla 5-6. Examinamos la tabla en busca de estados actuales que pasen al mismo siguiente estado y tengan la misma salida con ambas combinaciones de entrada. Los estados g y e cumplen con esos requisitos: ambos pasan a los estados a y f y tienen salidas de 0 y 1 con $x = 0$ y $x = 1$, respectivamente. Por tanto, los estados g y e son equivalentes y podemos eliminar uno de ellos. En la tabla 5-7 se indica el procedimiento para eliminar un estado y sustituirlo por su equivalente. Se elimina la fila del estado actual g y el estado g se sustituye por e en todos los lugares en que aparece en las columnas de siguiente estado.

El estado actual f ahora tiene como siguientes estados a e y f, y como salidas, 0 y 1 cuando $x = 0$ y $x = 1$, respectivamente. En la fila del estado actual d aparecen los mismos siguientes estados y salidas. Por tanto, los estados f y d son equivalentes y podemos eliminar el estado f, sustituyéndolo por d. La tabla reducida final se reproduce en la tabla 5-8. El diagrama de estados de la tabla reducida consta únicamente de cinco estados y se aprecia en la

Tabla 5-7
Reducción de la tabla de estados

	Siguiente estado		Salida	
Estado actual	$x = 0$	$x = 1$	$x = 0$	$x = 1$
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	f	0	1
e	a	f	0	1
f	e	f	0	1

Tabla 5-8
Tabla de estados reducida

	Siguiente estado		Salida	
Estado actual	$x = 0$	$x = 1$	$x = 0$	$x = 1$
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	d	0	1
e	a	d	0	1

figura 5-23. Este diagrama de estados satisface las especificaciones de entrada-salida originales y produce la sucesión de salida requerida con cualquier sucesión de entrada dada. La lista que sigue se dedujo del diagrama de estados de la figura 5-23 y corresponde a la sucesión de entrada que se utilizó antes (advierta que se obtiene la misma sucesión de salida, aunque la sucesión de estados es distinta):

Estado	a	a	b	c	d	e	d	d	e	d	e	a
Entrada	0	1	0	1	0	1	1	0	1	0	0	
Salida	0	0	0	0	0	1	1	0	1	0	0	

De hecho, esta sucesión es exactamente la misma que se obtuvo con la figura 5-21 si se sustituye g por e y f por d.

Podemos verificar sistemáticamente la posible equivalencia de cada par de estados con la ayuda de un procedimiento que utiliza una tabla de implicación. Dicha tabla tiene un cuadrado para cada par de estados que se sospecha podrían ser equivalentes. Si usamos la tabla acertadamente, podremos encontrar todos los pares de estados equivalentes de una tabla de estados. Ilustraremos el uso de la tabla de implicación para reducir el número de estados de una tabla de estados en la sección 9.5.

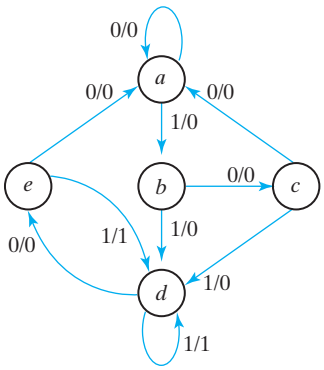


FIGURA 5-23
Diagrama de estados reducido

El circuito secuencial de este ejemplo se redujo de siete a cinco estados. En general, la reducción del número de estados de una tabla de estados da pie a un circuito con menos componentes. No obstante, el hecho de que una tabla de estados se haya reducido a menos estados no garantiza un ahorro en el número de flip-flops o de compuertas.

Asignación de estados

Para diseñar un circuito secuencial con componentes físicos, es necesario asignar valores binarios codificados a los estados. En el caso de un circuito con m estados, los códigos deben contener n bits, donde $2^n = \geq m$. Por ejemplo, con tres bits es posible asignar códigos a ocho estados denotados por los números binarios de 000 a 111. Si usamos la tabla de estados de la tabla 5-6, deberemos asignar valores binarios a siete estados; el estado restante no se usa. Si utilizamos la tabla de estados de la tabla 5-8, sólo cinco estados requerirán asignación binaria, y nos quedarán tres estados no utilizados. Los estados no utilizados se tratan como condiciones de indiferencia durante el diseño. Dado que las condiciones de indiferencia por lo regular ayudan a obtener un circuito más sencillo, es más probable que el circuito con cinco estados requiera menos compuertas combinacionales que el circuito con siete estados.

La forma más sencilla de codificar cinco estados es usar los primeros cinco enteros en el orden del conteo binario, como se muestra en la primera asignación de la tabla 5-9. Otra asignación similar es el código Gray que se muestra como asignación 2. En este caso, sólo un bit del grupo de código cambia al pasar de un número al siguiente. Este código facilita la colocación de las funciones booleanas en el mapa para simplificarlas. Otra posible asignación que se usa a menudo en el diseño de control es la asignación de un solo uno (*one-hot*). Esta configuración utiliza tantos bits como estados hay en el circuito. En cualquier momento, sólo un bit es 1; todos los demás son 0. Este tipo de asignación utiliza un flip-flop por estado.

La tabla 5-10 es la tabla de estados reducida, después de sustituir los símbolos de letra de los estados por la asignación binaria 1. Una asignación distinta producirá una tabla de estados con valores binarios distintos para los estados. Usamos la forma binaria de la tabla de estados para deducir la parte combinacional del circuito secuencial. La complejidad del circuito combinacional dependerá de la asignación binaria de estados que se escoja.

A veces se usa el término *tabla de transiciones* para referirse a una tabla de estados con asignación binaria. Esto la distingue de las tablas de estados que usan nombres simbólicos para los estados. En este libro usaremos el mismo término para referirnos a ambos tipos de tablas de estados.

Tabla 5-9
Tres posibles asignaciones binarias de estados

Estado	Asignación 1 Binaria	Asignación 2 Código Gray	Asignación 3 Un solo uno
a	000	000	00001
b	001	001	00010
c	010	011	00100
d	011	010	01000
e	100	110	10000

Tabla 5-10
Tabla de estados reducida, con la asignación binaria 1

Estado actual	Siguiente estado $x = 0$	Siguiente estado $x = 1$	Salida $x = 0$	Salida $x = 1$
000	000	001	0	0
001	010	011	0	0
010	000	011	0	0
011	100	011	0	1
100	000	011	0	1

5-7 PROCEDIMIENTO DE DISEÑO

El diseño de un circuito secuencial con reloj parte de un conjunto de especificaciones y culmina en un diagrama lógico o una lista de funciones booleanas de la cual puede obtenerse el diagrama lógico. En contraste con los circuitos combinacionales, que se especifican cabalmente con una tabla de verdad, los circuitos secuenciales requieren una tabla de estados para su especificación. El primer paso en el diseño de circuitos secuenciales es la obtención de una tabla de estados o una representación equivalente, como un diagrama de estados.

Un circuito secuencial sincrónico consta de flip-flops y compuertas combinacionales. El diseño del circuito consiste en escoger los flip-flops y luego encontrar una estructura de compuertas combinacionales que, junto con los flip-flops, produzca un circuito que satisfaga las especificaciones planteadas. El número de flip-flops se deduce del número de estados que se requieren en el circuito. El circuito combinacional se deduce de la tabla de estados evaluando las ecuaciones de entrada y de salida de los flip-flops. De hecho, una vez determinados el tipo y el número de los flip-flops, el proceso de diseño implica una transformación de un problema de circuito secuencial a un problema de circuito combinacional. De este modo, pueden aplicarse las técnicas del diseño de circuitos combinacionales.

El procedimiento para diseñar circuitos secuenciales sincrónicos se resume en una lista de pasos recomendados:

1. Deduzca, de la descripción textual y las especificaciones del funcionamiento deseado, un diagrama de estados para el circuito.
2. Reduzca el número de estados si es necesario.
3. Asigne valores binarios a los estados.
4. Obtenga la tabla de estados codificada en binario.
5. Escoja el tipo de flip-flops que se usarán.
6. Deduzca las ecuaciones simplificadas de entrada y de salida de los flip-flops.
7. Dibuje el diagrama lógico.

La especificación textual del comportamiento del circuito por lo regular supone que el lector conoce la terminología de lógica digital. Es necesario que el diseñador utilice intuición y experiencia para interpretar correctamente las especificaciones del circuito, porque las descripciones textuales podrían ser incompletas e inexactas. Una vez establecida tal especificación, y habiéndose obtenido el diagrama de estados, será posible aplicar procedimientos conocidos de síntesis para completar el diseño. Aunque existen procedimientos formales para la reducción y asignación de estados, los diseñadores experimentados casi nunca los usan. Los pasos

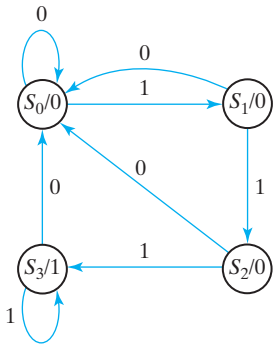


FIGURA 5-24
Diagrama de estados para el detector de sucesiones

4 a 7 del diseño se implementan con algoritmos exactos y por tanto pueden automatizarse. La parte del diseño que sigue un procedimiento bien definido se denomina *síntesis*.

El primer paso es la parte más difícil del diseño. Aquí mostraremos un ejemplo sencillo para ilustrar la forma de obtener un diagrama de estados a partir de la especificación textual.

Queremos diseñar un circuito que detecte tres o más unos consecutivos en una cadena de bits que llegan por una línea de entrada. El diagrama de estados del circuito se presenta en la figura 5-24. Se obtiene partiendo del estado S_0. Si la entrda es 0, el circuito permanece en el mismo estado, pero si es 1, pasa al estado S_1 para indicar que se detectó un 1. Si la siguiente entrada es 1, el cambio es al estado S_2, para indicar que han llegado dos unos consecutivos, pero si la entrada es 0 volvemos al estado S_0. El tercer uno consecutivo envía al circuito al estado S_3. Si se detectan más unos, el circuito permanecerá en S_3. Cualquier entrada 0 devolverá el circuito a S_0. Así, el circuito permanecerá en S_3 en tanto se hayan recibido tres o más unos consecutivos. Se trata de un circuito secuencial de modelo Moore porque la salida es 1 cuando el circuito está en el estado S_3, y 0 en los demás casos.

Síntesis con flip-flops

Una vez deducido el diagrama de estados, el resto del diseño sigue un procedimiento de síntesis directo. De hecho, es posible diseñar el circuito con una descripción HDL del diagrama de estados y las herramientas de síntesis HDL apropiadas para obtener una lista de red sintetizada. (La descripción HDL del diagrama de estados será similar al ejemplo HDL 5-6 de la sección 5-5.) Para diseñar el circuito a mano, necesitamos asignar códigos binarios a los estados y preparar la tabla de estados. Esto se hizo en la tabla 5-11, que se dedujo del diagrama de estados de la figura 5-24 con una asignación binaria directa. Escogimos dos flip-flops *D* para representar los cuatro estados y rotulamos sus salidas *A* y *B*. Hay una entrada *x* y una salida *y*. La ecuación característica del flip-flop *D* es $Q(t + 1) = D_Q$, lo que significa que los valores de siguiente estado de la tabla de estados especifican la condición de entrada *D* para el flip-flop. Las ecuaciones de entrada del flip-flop se obtienen directamente de las columnas de siguiente estado de *A* y *B*, y se expresan en forma de suma de minitérminos así:

$$A(t + 1) = D_A(A, B, x) = \sum(3, 5, 7)$$
$$B(t + 1) = D_B(A, B, x) = \sum(1, 5, 7)$$
$$y(A, B, x) = \sum(6, 7)$$

Tabla 5-11
Tabla de estados para el detector de sucesiones

Estado actual		Entrada	Siguiente Estado		Salida
A	*B*	*x*	*A*	*B*	*y*
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	0	0	0
1	0	1	1	1	0
1	1	0	0	0	1
1	1	1	1	1	1

donde A y B son los valores de estado actual de los flip-flops A y B, x es la entrada, y D_A y D_B son las ecuaciones de entrada. Los minitérminos para la salida y se obtienen de la columna de salida de la tabla de estados.

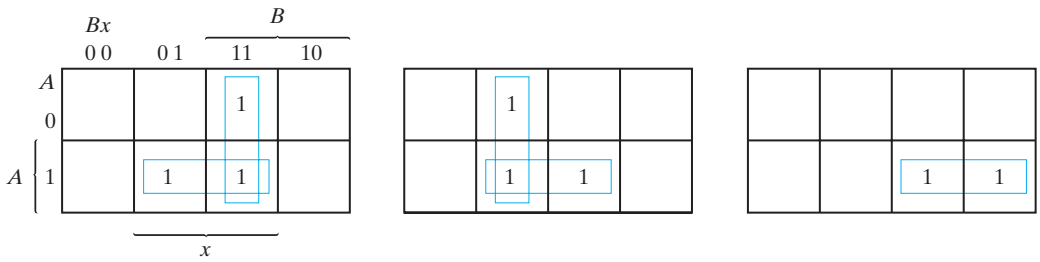
Las ecuaciones booleanas se simplifican con ayuda de los mapas de la figura 5-25. Las ecuaciones simplificadas son

$$D_A = Ax + Bx$$
$$D_B = Ax + B'x$$
$$y = AB$$

El diagrama lógico del circuito secuencial se presenta en la figura 5-26.

Tablas de excitación

El diseño de un circuito secuencial con flip-flops de otro tipo que no sea D se complica por el hecho de que las ecuaciones de entrada del circuito se deben deducir de manera indirecta de la tabla de estados. Cuando se usan flip-flops D, las ecuaciones de entrada se obtienen direc-



$D_A = Ax + Bx$ $D_B = Ax + B'x$ $y = AB$

FIGURA 5-25
Mapas para el detector de sucesiones

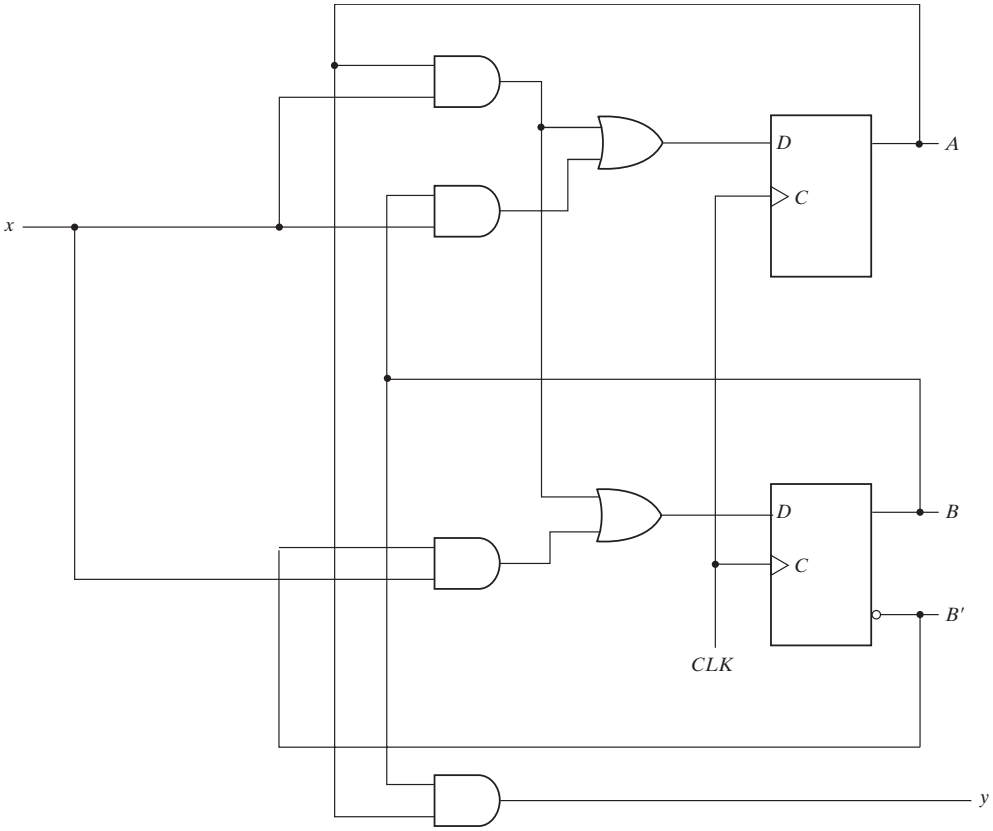


FIGURA 5-26
Diagrama lógico del detector de sucesiones

tamente del siguiente estado. No sucede así con los flip-flops *JK* y *T*. Para determinar las ecuaciones de entrada de estos flip-flops, es necesario deducir una relación funcional entre la tabla de estados y las ecuaciones de entrada.

Las tabla características de flip-flops que presentamos en la tabla 5-1 dan el valor del siguiente estado cuando se conocen las entradas y el estado actual. Estas tablas son útiles para analizar circuitos secuenciales y para definir el funcionamiento de los flip-flops. Durante el proceso de diseño, normalmente se conoce la transición de estado actual a siguiente estado y se desea conocer las condiciones de entrada del flip-flop que dan pie a la transición requerida. Por ello, se necesita una tabla que dé las entradas requeridas para un cambio de estado dado. Ese tipo de tablas se llaman *tablas de excitación*.

La tabla 5-12 presenta las tablas de excitación de los dos flip-flops. Cada tabla tiene una columna para el estado actual, $Q(t)$ y el siguiente estado, $Q(t + 1)$, y una columna para cada entrada, a fin de mostrar cómo se logra la transición requerida. Hay cuatro posibles transiciones de estado actual a siguiente estado. Las condiciones de entrada necesarias para cada una se deducen de la información proporcionada por la tabla característica. El símbolo X en las tablas representa una condición de indiferencia, es decir, que no importa si la entrada es 1 o 0.

La tabla de excitación del flip-flop *JK* corresponde a la parte a). Cuando tanto el estado actual como el siguiente son 0, la entrada *J* debe permanecer en 0 y la entrada *K* puede ser 0 o 1.

Tabla 5-12
Tablas de excitación de flip-flops

$Q(t)$	$Q(t+1)$	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

a) JK

$Q(t)$	$Q(t+1)$	T
0	0	0
0	1	1
1	0	1
1	1	0

b) T

Asimismo, cuando tanto el estado actual como el siguiente son 1, la entrada K debe permanecer en 0, mientras que la J puede ser 0 o 1. Si es preciso que el flip-flop tenga una transición del estado 0 al 1, J deberá ser 1, porque la entrada J establece el flip-flop; en cambio, la entrada K puede ser 0 o 1. Si $K = 0$, la condición $J = 1$ establecerá el flip-flop como se requiere; si $K = 1$ y $J = 1$, el flip-flop se complementará y pasará del estado 0 al 1, como se requiere. Por tanto, marcamos la entrada K con una condición de indiferencia para la transición de 0 a 1. En el caso de la transición del estado 1 al 0, necesitamos $K = 1$, porque la entrada K despeja el flip-flop. En cambio, la entrada J puede ser 0 o 1, porque $J = 0$ no tiene ningún efecto, mientras que $J = 1$ con $K = 1$ complementa el flip-flop y produce la transición del estado 1 al estado 0.

La parte b) corresponde a la tabla de excitación para el flip-flop T. Por la tabla de características, sabemos que, cuando la entrada $T = 1$, el estado del flip-flop se complementa; cuando $T = 0$, el estado del flip-flop no cambia. Por tanto, si el estado del flip-flop debe permanecer sin cambio, el requisito es que $T = 0$. Si es preciso complementar el estado del flip-flop, T debe ser 1.

Síntesis con flip-flops *JK*

El procedimiento de síntesis de circuitos secuenciales con flip-flops JK es el mismo que con flip-flops D, excepto que las ecuaciones de entrada se deben evaluar a partir de la transición de estado actual a siguiente estado deducida de la tabla de excitación. Para ilustrar el procedimiento, sintetizaremos el circuito secuencial especificado por la tabla 5-13. Además de tener columnas para el estado actual, la entrada y el siguiente estado, como en una tabla convencional de estados, la tabla también muestra las condiciones de entrada del flip-flop de las que se deducen las ecuaciones de entrada. Estas entradas del flip-flop se deducen de la tabla de estados y de la tabla de excitación del flip-flop JK. Por ejemplo, en la primera fila de la tabla 5-13 tenemos una transición para el flip-flop A, de 0 en el estado actual a 0 en el siguiente estado. En la tabla 5-12 para el flip-flop JK, vemos que una transición de estado actual 0 a siguiente estado 0 requiere que la entrada J sea 0; la entrada K no importa. Por tanto, se anota un 0 y una X en la primera fila, bajo J_A y K_A. Puesto que la primera fila también muestra una transición para el flip-flop B, de 0 en el estado actual a 0 en el siguiente estado, se anota un 0 y una X en esa fila, bajo J_B y K_B. La segunda fila de la tabla indica una transición del flip-flop B, de 0 en el estado actual a 1 en el siguiente estado. La tabla de excitación nos dice que una transición de 0 a 1 requiere $J = 1$, mientras que K no importa, así que anotamos 1 y X en la segunda fila bajo J_B y K_B. Continuamos este proceso con todas las filas de la tabla y con cada flip-flop, copiando las condiciones de entrada de la tabla de excitación en la fila apropiada del flip-flop que se está considerando.

Las entradas del flip-flop de la tabla 5-13 especifican la tabla de verdad de las ecuaciones de entrada en función de los estados actuales A y B y de la entrada x. Las ecuaciones de entrada se simplifican en los mapas de la figura 5-27. Los valores de siguiente estado no se usan duran-

Tabla 5-13
Tabla de estados y entradas de flip-flops JK

Estado Actual		Entrada	Siguiente Estado		Entradas del flip-flop			
A	*B*	*x*	*A*	*B*	J_A	K_A	J_B	K_B
0	0	0	0	0	0	X	0	X
0	0	1	0	1	0	X	1	X
0	1	0	1	0	1	X	*X*	1
0	1	1	0	1	0	X	*X*	0
1	0	0	1	0	X	0	0	X
1	0	1	1	1	X	0	1	X
1	1	0	1	1	X	0	*X*	0
1	1	1	0	0	X	1	*X*	1

te la simplificación porque las ecuaciones de entrada son función únicamente del estado actual y de la entrada. Considere la ventaja de usar flip-flops *JK* al diseñar circuitos secuenciales. El hecho de que haya tantas condiciones de indiferencia indica que el circuito combinacional para las ecuaciones de entrada seguramente será más sencillo, porque los minitérminos de indiferencia normalmente ayudan a obtener expresiones más simples. Si la tabla de estados tiene estados no utilizados, habrá condiciones de indiferencia adicionales en el mapa.

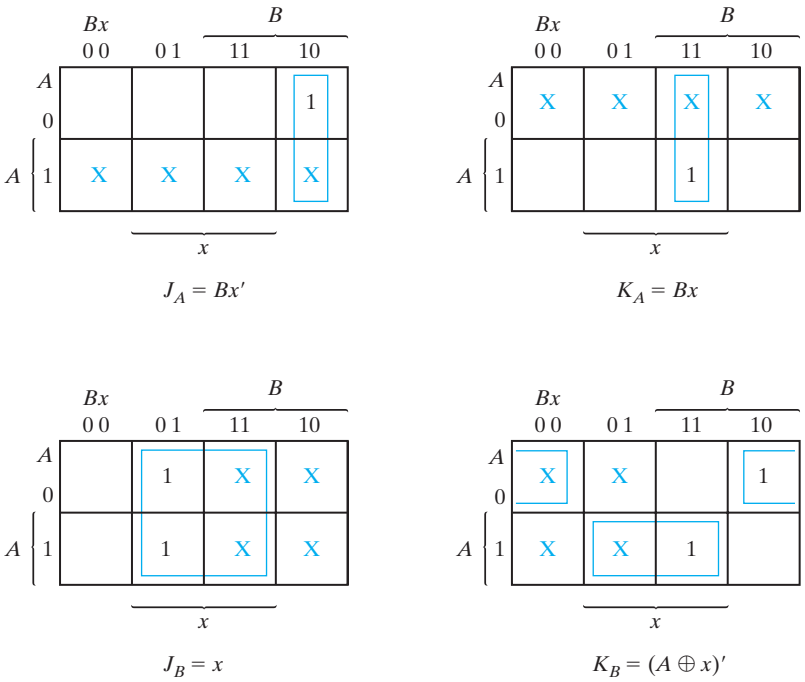


FIGURA 5-27
Mapas para las ecuaciones de entrada *J* y *K*

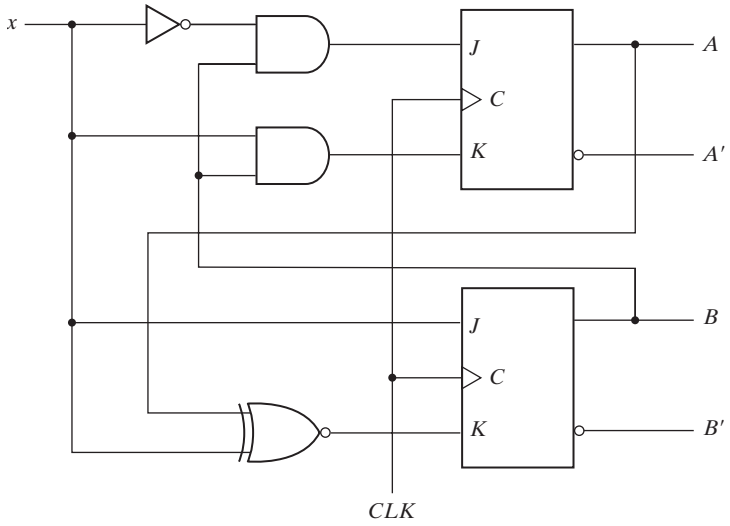


FIGURA 5-28
Diagrama lógico para el circuito secuencial con flip-flops *JK*

Las cuatro ecuaciones de entrada de los dos flip-flops *JK* se dan bajo los mapas de la figura 5-27. El diagrama lógico del circuito secuencial se presenta en la figura 5-28.

Síntesis con flip-flops *T*

Ilustraremos la síntesis con flip-flops *T* diseñando un contador binario. Un contador binario de *n* bits consiste en *n* flip-flops capaces de contar en binario de 0 hasta $2^n - 1$. El diagrama de estados de un contador de tres bits se reproduce en la figura 5-29. Como se ve por los estados binarios indicados dentro de los círculos, las salidas de los flip-flops repiten la sucesión de conteo binario, volviendo a 000 después de 111. Las flechas entre los círculos no se han marcado con valores de entrada y salida como en otros diagramas de estados. Recuerde que las transiciones de estado en los circuitos secuenciales con reloj se dan durante un borde de reloj; los flip-flops permanecen en su estado actual si no se aplica reloj. Por ello, el reloj no aparece explícitamente como variable de entrada en el diagrama de estados ni en la tabla de estados. Desde este

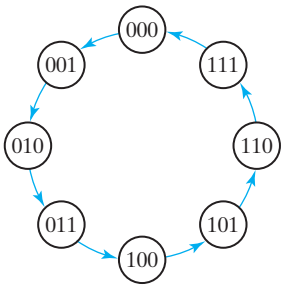


FIGURA 5-29
Diagrama de estados de un contador binario de tres bits

Tabla 5-14
Tabla de estados para el contador de tres bits

Estado actual			Siguiente estado			Entradas de los flip-flops		
A_2	A_1	A_0	A_2	A	A_0	T_{A2}	T_{A1}	T_{A0}
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	1
0	1	0	0	1	1	0	0	1
0	1	1	1	0	0	1	1	1
1	0	0	1	0	1	0	0	1
1	0	1	1	1	0	0	1	1
1	1	0	1	1	1	0	1	1
1	1	1	0	0	0	1	1	1

punto de vista, el diagrama de estados de un contador no tiene que indicar valores de entrada y salida a lo largo de las flechas. La única entrada del circuito es el reloj, y las salidas están especificadas por el estado actual de los flip-flops. El siguiente estado de un contador depende exclusivamente de su estado actual, y la transición de estado se efectúa cada vez que el reloj tiene una transición.

La tabla 5-14 es la tabla de estados para el contador binario de tres bits. Los tres flip-flops se designan con A_2, A_1 y A_0. La forma más eficiente de construir contadores binarios es con flip-flops T gracias a su propiedad de complemento. La excitación de los flip-flops para las entradas T se deduce de la tabla de excitación del flip-flop T y de una inspección de la transición del estado actual al siguiente estado. Por ejemplo, consideremos las entradas de flip-flop que van en la fila 001. El estado actual aquí es 001, y el siguiente, 010, que es el siguiente conteo sucesivo. Si comparamos estos dos conteos, veremos que A_2 pasa de 0 a 0; por tanto, marcamos T_{A2} con 0 porque el flip-flop A_2 no debe cambiar cuando hay una transición de reloj. A_1 pasa de 0 a 1, así que marcamos T_{A1} con 1 porque este flip-flop se deberá complementar en el siguiente borde de reloj. Por su parte, A_0 pasa de 1 a 0, lo que indica que se debe complementar; por tanto, marcamos T_{A0} con 1. La última fila, con estado actual 111, se compara con el primer conteo, 000, que es su siguiente estado. El cambio de ceros a unos en todos los bits requiere complementar los tres flip-flops.

Las ecuaciones de entrada de los flip-flops se simplifican con los mapas de la figura 5-30. Vemos que T_{A0} tiene unos en los ocho minitérminos porque el bit menos significativo del con-

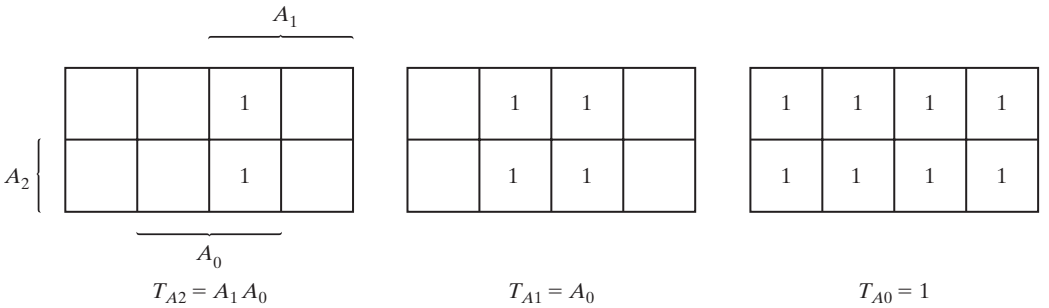


FIGURA 5-30
Mapas para el contador binario de tres bits

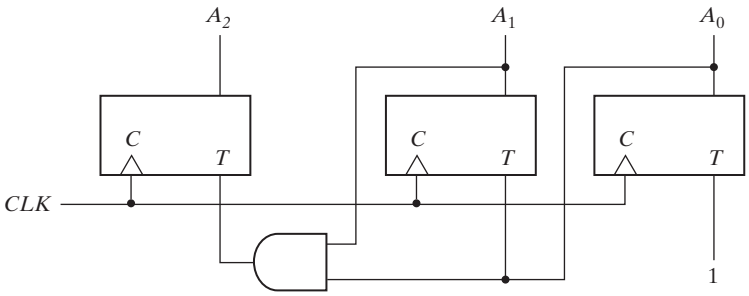


FIGURA 5-31
Diagrama lógico del contador de tres bits

tador se complementa en cada conteo. Una función booleana que incluye a todos los minitérminos define un valor constante de 1. Las ecuaciones de entrada que se dan abajo de cada mapa especifican la parte combinacional del contador. Al incluir estas funciones con los tres flip-flops, obtenemos el diagrama lógico del contador de la figura 5-31.

PROBLEMAS

5-1 El latch *D* de la figura 5-6 se construyó con cuatro compuertas NAND y un inversor. Considere estas otras tres formas de obtener un latch *D*. En cada caso, dibuje el diagrama lógico y verifique el funcionamiento del circuito.

a) Use compuertas NOR para la parte de latch *SR* y compuertas AND para las otras dos. Se podría necesitar un inversor.

b) Use compuertas NOR para las cuatro compuertas. Se podrían requerir inversores.

c) Use únicamente cuatro compuertas NAND (sin inversor). Esto se logra conectando la salida de la compuerta superior de la figura 5-6 (que va al latch *SR*) con la entrada de la compuerta inferior (en vez de la salida del inversor).

5-2 Construya un flip-flop *JK* con un flip-flop *D*, un multiplexor de 2 líneas a 1 y un inversor.

5-3 Demuestre que la ecuación característica para la salida de complemento de un flip-flop *JK* es

$$Q'(t + 1) = J'Q' + KQ$$

5-4 Un flip-flop *PN* tiene cuatro operaciones: despeje a 0, ningún cambio, complemento y establecimiento a 1, cuando las entradas *P* y *N* son 00, 01, 10 y 11, respectivamente.

a) Tabule la tabla de características.

b) Deduzca la ecuación característica.

c) Tabule la tabla de excitación.

d) Muestre cómo el flip-flop *PN* se puede convertir en un flip-flop *D*.

5-5 Explique la diferencia entre tabla de verdad, tabla de estados, tabla característica y tabla de excitación. Explique también la diferencia entre una ecuación booleana, una ecuación de estado, una ecuación característica y una ecuación de entrada de flip-flop.

5-6 Un circuito secuencial con dos flip-flops D, A y B; dos entradas, x y y; y una salida, z, se especifica con las ecuaciones de estado y salida siguientes

$$A(t + 1) = x'y + xA$$
$$B(t + 1) = x'B + xA$$
$$z = B$$

a) Dibuje el diagrama lógico del circuito.

b) Prepare la tabla de estados del circuito secuencial.

c) Dibuje el diagrama de estados correspondiente.

5-7 Un circuito secuencial tiene un flip-flop Q, dos entradas x y y, y una salida S. Consta de un circuito sumador completo conectado a un flip-flop D, como se indica en la figura P5-7. Deduzca la tabla de estados y el diagrama de estados del circuito secuencial.

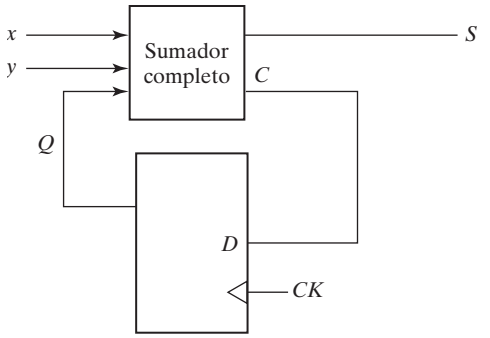


FIGURA P5-7

5-8 Deduzca la tabla de estados y el diagrama de estados del circuito secuencial que se muestra en la figura P5-8. Explique la función del circuito.

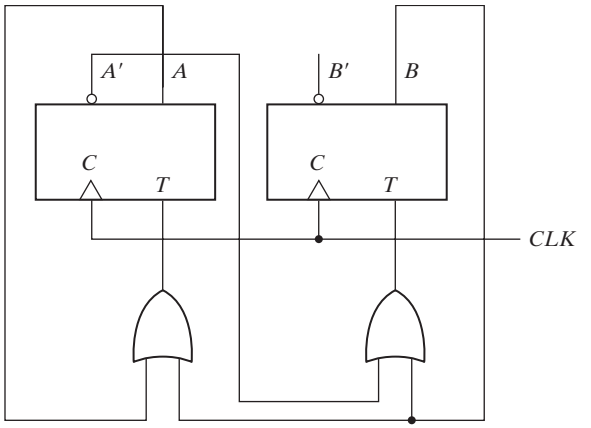


FIGURA P5-8

5-9 Un circuito secuencial tiene dos flip-flops *JK*, *A* y *B*, y una entrada, *x*. El circuito se describe con estas ecuaciones de entrada de flip-flop:

$$J_A = x \qquad K_A = B'$$
$$J_B = x \qquad K_B = A$$

a) Deduzca las ecuaciones de estado $A(t + 1)$ y $B(t + 1)$ sustituyendo las ecuaciones de entrada por las variables *J* y *K*.

b) Dibuje el diagrama de estados del circuito.

5-10 Un circuito secuencial tiene dos flip-flops *JK*, *A* y *B*, dos entradas, *x* y *y*, y una salida, *z*. Las ecuaciones de entrada de los flip-flops y la ecuación de salida del circuito son

$$J_A = Bx + B'y' \qquad K_A = B'xy'$$
$$J_B = A'x \qquad K_B = A + xy'$$
$$z = Ax'y' + Bx'y'$$

a) Dibuje el diagrama lógico del circuito. b) Prepare la tabla de estados.

c) Deduzca las ecuaciones de estado para *A* y *B*.

5-11 Partiendo del estado 00 en el diagrama de estados de la figura 5-16, determine las transiciones de estados y sucesión de salida que se generarán cuando se aplique la sucesión de entrada 010110111011110.

5-12 Reduzca el número de estados de la siguiente tabla de estados y tabule la tabla de estados reducida.

Estado actual	Siguiente estado		Salida	
	$x = 0$	$x = 1$	$x = 0$	$x = 1$
a	*f*	*b*	0	0
b	*d*	*c*	0	0
c	*f*	*e*	0	0
d	*g*	*a*	1	0
e	*d*	*c*	0	0
f	*f*	*b*	1	1
g	*g*	*h*	0	1
h	*g*	*a*	1	0

5-13 Partiendo del estado *a* y la sucesión de entrada 01110010011, determine la sucesión de salida de

a) la tabla de estados del problema anterior y

b) la tabla de estados reducida del problema anterior. Demuestre que se obtiene la misma sucesión de salida con ambas.

5-14 Sustituya la asignación binaria 2 de la tabla 5-9 en los estados de la tabla 5-8 y obtenga la tabla de estados binaria.

5-15 Prepare una tabla de estados para el flip-flop *JK* utilizando *Q* como estado actual y siguiente, y *J* y *K* como entradas. Diseñe el circuito secuencial especificado por la tabla de estados y demuestre que es equivalente a la figura 5-12a).

5-16 Diseñe un circuito secuencial con dos flip-flops *D*, *A* y *B*, y una entrada, *x*. Cuando $x = 0$, el estado del circuito no cambia. Cuando $x = 1$, el circuito pasa por las transiciones de estado de 00 a 01 a 11 a 10 y de vuelta a 00, y repite.

5-17 Diseñe un complementador a dos en serie con una entrada y una salida. El circuito acepta una cadena de bits de la entrada y genera el complemento a dos en la salida. El circuito se puede restablecer asincrónicamente para iniciar y terminar la operación.

5-18 Diseñe un circuito secuencial con dos flip-flops *JK*, *A* y *B*, y dos entradas, *E* y *x*. Si $E = 0$, el circuito permanece en el mismo estado sea cual sea el valor de *x*. Si $E = 1$ y $x = 1$, el circuito pasa por las transiciones de estado de 00 a 01 a 10 a 11 y de vuelta a 00, y repite. Cuando $E = 1$ y $x = 0$, el circuito pasa por las transiciones de estado de 00 a 11 a 10 a 01 y de vuelta a 00, y repite.

5-19 Un circuito secuencial tiene tres flip-flops, *A*, *B*, *C*; una entrada, *x*; y una salida, *y*. El diagrama de estados aparece en la figura P5-19. El circuito se diseñará tratando los estados no utilizados como condiciones de indiferencia. Analice el circuito obtenido del diseño para determinar el efecto de los estados no utilizados.

a) Use flip-flops *D* en el diseño.

b) Use flip-flops *JK* en el diseño.

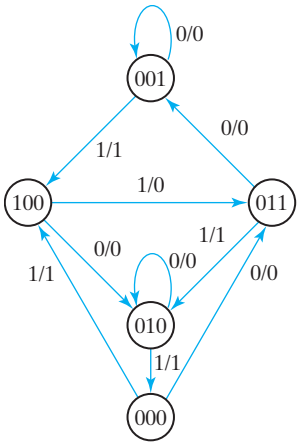


FIGURA P5-19

5-20 Diseñe el circuito secuencial especificado por el diagrama de estados de la figura 5.19 empleando flip-flops *T*.

5-21 Explique la principal diferencia entre un enunciado **initial** y un enunciado **always** en Verilog HDL.

5-22 Dibuje la forma de onda generada por el enunciado **initial**

```
initial
  begin
  w = 0;  #20 w = 1;  # 50 w = 0;  # 30 w = 1;  #10 w = 0;
  end
```

5-23 Considere estos enunciados suponiendo que RegA contiene inicialmente el valor 30.

a)
```
RegA = 125
RegB = RegA
```

b)
```
RegA <= 125
RegB <= RegA
```

¿Qué valores tienen RegA y RegB después de la ejecución?

5-24 Escriba una descripción HDL del comportamiento de un flip-flop *D* con preestablecimiento y restablecimiento asincrónicos. (Este tipo de flip-flop se reproduce en la figura 11-13.)

5-25 Un flip-flop especial disparado por borde positivo tiene dos entradas, *D*1 y *D*2, y una entrada de control que escoge una de las dos. Escriba una descripción HDL del comportamiento de este flip-flop.

5-26 Escriba una descripción HDL del comportamiento de un flip-flop *JK* utilizando un enunciado **if-else** basado en el valor del estado actual. (*Sugerencia:* Considere la ecuación característica cuando $Q = 0$ o $Q = 1$.)

5-27 Reescriba la descripción del ejemplo HDL 5-5 combinando las transiciones de estado y la salida en un bloque **always.**

5-28 Simule el circuito secuencial de la figura 5-17.

a) Escriba la descripción HDL del diagrama de estados.

b) Escriba la descripción HDL del diagrama de circuito.

c) Escriba un estímulo HDL con una sucesión de entradas: 00, 01, 11, 10. Verifique que la respuesta sea la misma con ambas descripciones.

5-29 Escriba la descripción HDL del contador binario de dos bits que se ilustra en la figura 5-20. Utilice el módulo de estímulo del ejemplo HDL 5-7 y verifique que su respuesta de salida sea la misma que las formas de onda de la figura 5-21.

5-30 Dibuje el diagrama lógico del circuito secuencial descrito por el módulo HDL siguiente:

```
module Seqcrt (A,B,C,Q,CLK);
   input A,B,C,CLK;
   output Q;
   reg Q,E;
   always @ (posedge CLK)
      begin
     E <= A & B;
     Q <= E | C;
      end
endmodule
```

¿Qué cambios, si acaso, deben hacerse al circuito si los dos últimos enunciados usan asignación bloqueadora en vez de no bloqueadora?

REFERENCIAS

1. HAYES, J. P. 1993. *Introduction to Digital Logic Design*. Reading, MA: Addison-Wesley.
2. WAKERLY, J. F. 2000. *Digital Design: Principles and Practices*, 3a. ed. Upper Saddle River, NJ: Prentice-Hall.
3. KATZ, R. H. 1994. *Contemporary Logic Design*. Upper Saddle River, NJ: Prentice-Hall.
4. MANO, M. M. y C. R. KIME. 2000. *Logic and Computer Design Fundamentals*, 2a. ed. Upper Saddle River, NJ: Prentice-Hall.
5. NELSON V. P., H. T. NAGLE, J. D. IRWIN y B. D. CARROLL. 1995. *Digital Logic Circuit Analysis and Design*. Upper Saddle River, NJ: Prentice-Hall.
6. DIETMEYER, D. L. 1988. *Logic Design of Digital Systems*, 3a. ed. Boston: Allyn Bacon.
7. GAJSKI, D. D. 1997. *Principles of Digital Design*. Upper Saddle River, NJ: Prentice-Hall.
8. ROTH, C. H. 1992. *Fundamentals of Logic Design*, 4a. ed. St. Paul: West.
9. BHASKER, J. 1997. *A Verilog HDL Primer*. Allentown, PA: Star Galaxy Press.
10. BHASKER, J. 1998. *Verilog HDL Synthesis*. Allentown, PA: Star Galaxy Press.
11. CILETTI, M. D. 1999. *Modeling, Synthesis and Rapid Prototyping with Verilog HDL*. Upper Saddle River, NJ: Prentice-Hall.
12. PALNITKAR, S. 1996. *Verilog HDL: A Guide to Digital Design and Synthesis*. SunSoft Press (un título Prentice-Hall).
13. THOMAS, D. E. y P. R. MOORBY. 1998. *The Verilog Hardware Description Language,* 4a. ed. Boston: Kluwer Academic Publishers.

6 Registros y contadores

6-1 REGISTROS

Un circuito secuencial con reloj consiste en un grupo de flip-flops y compuertas combinacionales conectados para formar un camino de retroalimentación. Los flip-flops son indispensables porque, sin ellos, el circuito se reduce a un circuito puramente combinacional (suponiendo que no haya retroalimentación entre las compuertas). Un circuito con flip-flops se considera secuencial aunque no tenga compuertas combinacionales. Los circuitos que incluyen flip-flops por lo regular se clasifican según la función que desempeñan, más que por el nombre del circuito secuencial. Dos de esos circuitos son los registros y los contadores.

Un registro es un grupo de flip-flops. Cada flip-flop puede almacenar un bit de información. Un registro de *n* bits consiste en un grupo de *n* flip-flops capaces de almacenar *n* bits de información binaria. Además de los flip-flops, un registro puede tener compuertas combinacionales que realizan ciertas tareas de procesamiento de datos. En su definición más amplia, un registro consiste en un grupo de flip-flops y compuertas que efectúan su transición. Los flip-flops contienen la información binaria y las compuertas determinan cómo esa información se transfiere al registro.

Un contador es básicamente un registro que pasa por una sucesión predeterminada de estados. Las compuertas del contador están conectadas de tal manera que producen la sucesión prescrita de estados binarios. Aunque los contadores son un tipo especial de registros, es común distinguirlos dándoles otro nombre.

Hay diversos tipos de registros en el comercio. El más sencillo consiste únicamente en flip-flops, sin compuertas. La figura 6-1 muestra uno de esos registros construido con cuatro flip-flops tipo *D*. La entrada de reloj, común a todos los flip-flops, los dispara en el flanco positivo de cada pulso, y los datos binarios disponibles en las cuatro entradas se transfieren en el registro de cuatro bits. Es posible muestrear las cuatro salidas en cualquier momento para obtener la información binaria almacenada en el registro. La entrada de despeje (*clear*) se conecta a la entrada *R* (restablecimiento, *reset*) de los cuatro flip-flops. Cuando esta entrada cambia

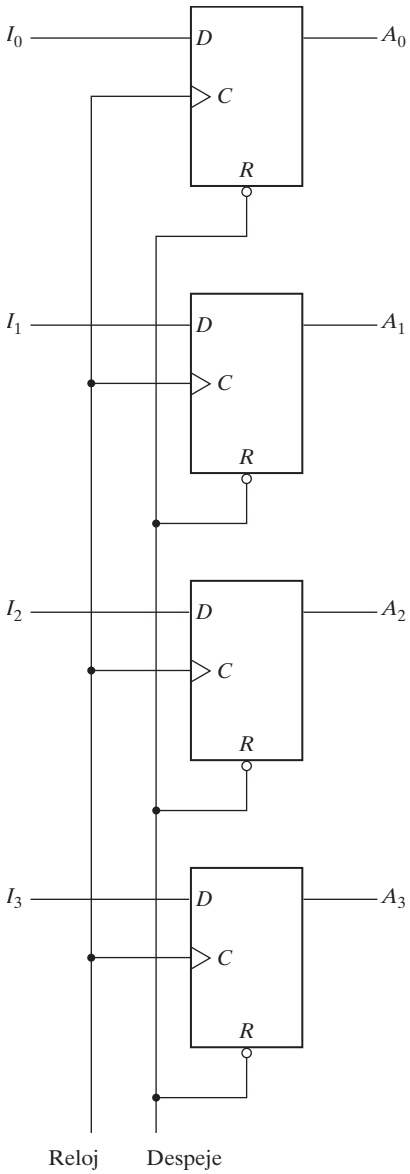


FIGURA 6-1
Registro de cuatro bits

a 0, todos los flip-flops se restablecen asincrónicamente. La entrada *clear* sirve para poner en ceros el registro antes de que comience a funcionar con reloj. Las entradas *R* se deben mantener en 1 lógico durante el funcionamiento normal con reloj. Se utiliza tanto *clear* como *reset* para indicar la transferencia del registro al estado de ceros.

Registro con carga paralela

Los sistemas digitales sincrónicos tienen un generador maestro de reloj que suministra un tren continuo de pulsos de reloj. Estos pulsos se aplican a todos los flip-flops y registros del sistema. El reloj maestro actúa como una bomba que alimenta un latido constante a todas las partes del sistema. Se requiere una señal de control aparte para decidir qué pulso de reloj específico tendrá efecto sobre un registro dado. La transferencia de información nueva a un registro se describe como *carga* del registro. Si todos los bits del registro se cargan simultáneamente, con un pulso de reloj común, se dice que la carga se efectúa en paralelo. Un borde de reloj aplicado a las entradas *C* del registro de la figura 6-1 carga las cuatro entradas en paralelo. En esta configuración, el reloj deberá inhibirse del circuito cuando se desee que el registro conserve intacto su contenido. Esto se hace controlando la señal de entrada del reloj con una compuerta habilitadora. Sin embargo, la inserción de compuertas en la trayectoria del reloj implica que la lógica se efectúa con pulsos de reloj. La inserción de compuertas lógicas produce retardos de propagación desiguales entre el reloj maestro y las entradas de los flip-flops. Para sincronizar plenamente el sistema, hay que cerciorarse de que todos los pulsos de reloj lleguen al mismo tiempo a todos los puntos del sistema, para que todos los flip-flops se disparen en forma simultánea. La lógica efectuada con pulsos de reloj inserta retardos variables y podría hacer que el sistema se desincronice. Por ello, es aconsejable controlar el funcionamiento del registro con las entradas *D*, en vez de controlar el reloj en las entradas *C* de los flip-flops.

En la figura 6-2 se observa un registro de cuatro bits con una entrada de control de carga que se hace pasar por compuertas y llega a las entradas *D* de los flip-flops. La entrada de carga del registro determina la acción que se realizará en cada pulso de reloj. Si la entrada de carga es 1, los datos de las cuatro entradas se transfieren al registro en el siguiente borde positivo del reloj. Si la entrada de carga es 0, las salidas de los flip-flops se conectan a sus respectivas entradas. La conexión de retroalimentación de salida a entrada es necesaria porque el flip-flop *D* no tiene una condición de "sin cambio". En cada flanco de reloj, la entrada *D* determina el siguiente estado del registro. Para que la salida no cambie, es necesario hacer que la entrada *D* sea igual al valor actual de la salida.

Los pulsos de reloj se aplican a las entradas *C* en todo momento. La entrada de carga determina si el siguiente pulso va a aceptar nueva información o va a dejar intacta la información que está en el registro. La transferencia de información de las entradas de datos o las salidas del registro se efectúa simultáneamente con los cuatro bits, en respuesta a un borde de reloj.

6-2 REGISTROS DE DESPLAZAMIENTO

Un registro capaz de desplazar su información binaria en una dirección o en la otra se llama *registro de desplazamiento*. La configuración lógica de un registro de desplazamiento consiste en una cadena de flip-flops en cascada, con la salida de un flip-flop conectada a la entrada del siguiente flip-flop. Todos los flip-flops reciben pulsos de reloj comunes, que activan el desplazamiento de una etapa a la siguiente.

El registro de desplazamiento más sencillo posible usa sólo flip-flops, como se indica en la figura 6-3. La salida de un flip-flop dado se conecta a la entrada *D* del flip-flop que está a su derecha. Cada pulso de reloj desplaza el contenido del registro una posición de bit a la derecha. La *entrada en serie* determina qué sucede en el flip-flop de la extrema izquierda durante el desplazamiento. La *salida en serie* se toma de la salida del flip-flop de la extrema derecha. A veces se hace necesario controlar el desplazamiento de modo que sólo se efectúe con ciertos pulsos, pe-

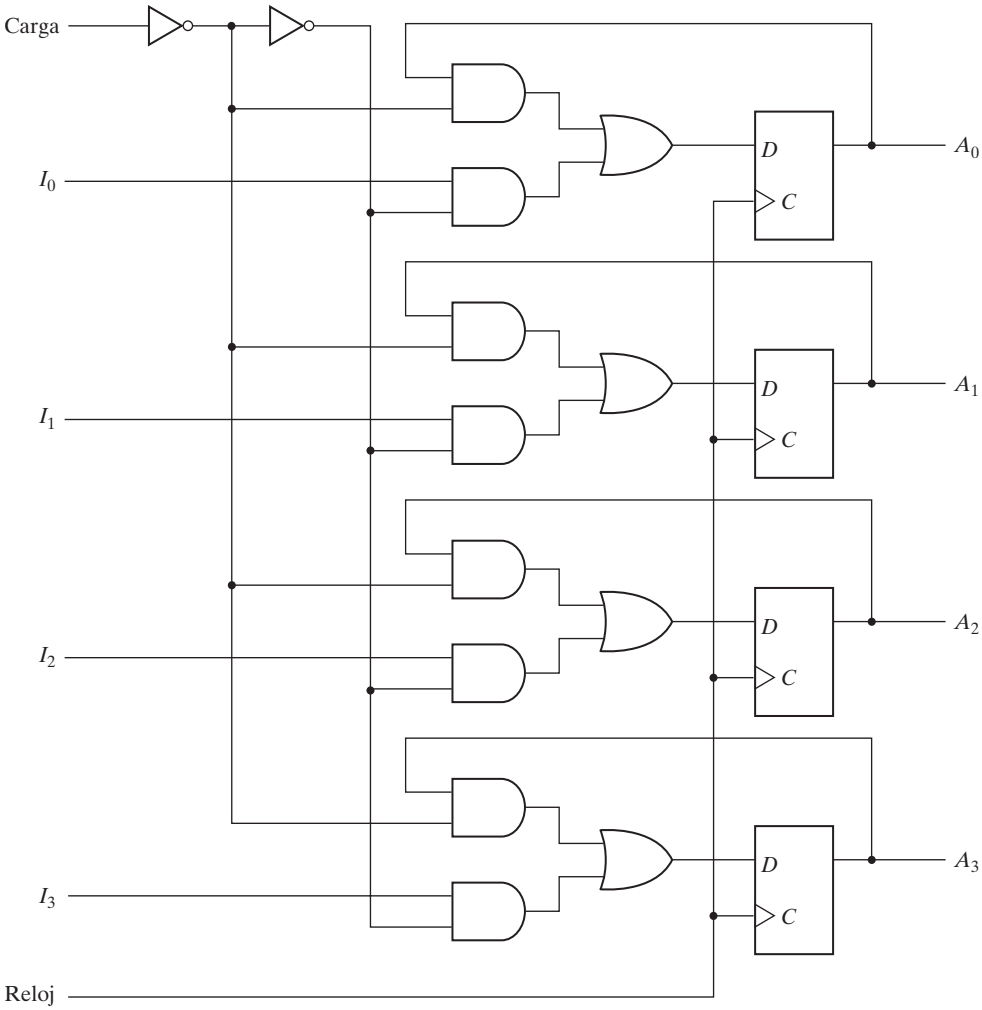


FIGURA 6-2
Registro de cuatro bits con carga paralela

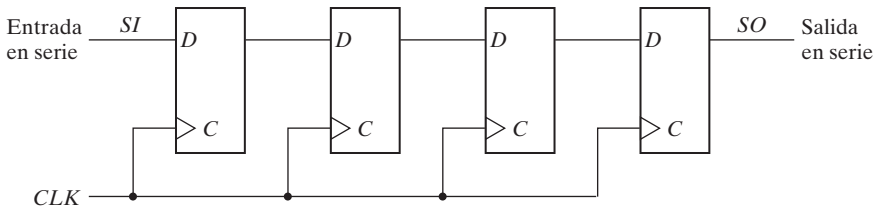


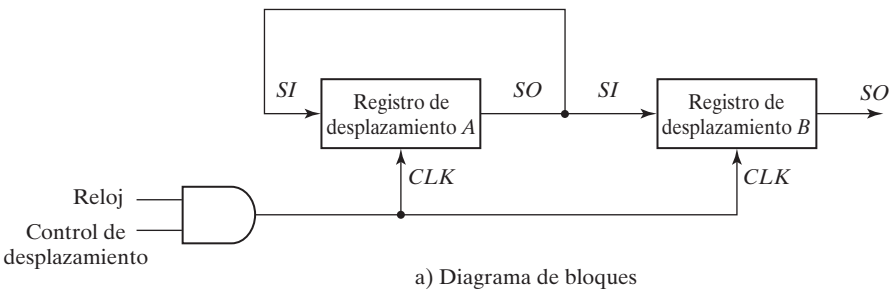
FIGURA 6-3
Registro de desplazamiento de cuatro bits

ro no con otros. Esto se logra inhibiendo el reloj de la entrada del registro para impedir que se desplace. Más adelante se verá que la operación de desplazamiento se controla a través de las entradas *D* de los flip-flops, en vez de hacerse a través de la entrada de reloj. Pero si se se usa el registro de desplazamiento de la figura 6-3, el desplazamiento podrá controlarse conectando el reloj a través de una compuerta AND con una entrada que controle el desplazamiento.

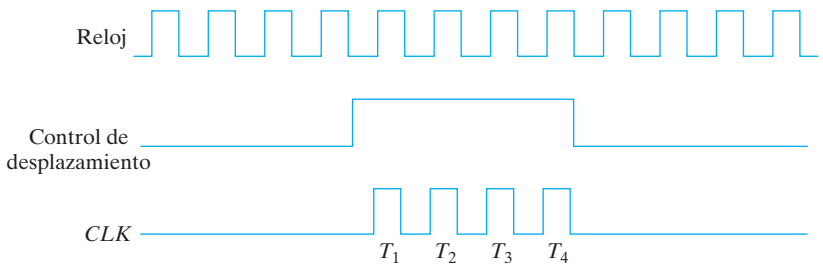
Transferencia en serie

Decimos que un sistema digital opera en modo en serie cuando la información se transfiere y manipula bit por bit. La información se transfiere bit por bit desplazando los bits del registro de origen hacia el registro de destino. Esto contrasta con la transferencia paralela, en la que todos los bits del registro se transfieren al mismo tiempo.

La transferencia en serie de información del registro *A* al registro *B* se efectúa con registros de desplazamiento, como se observa en el diagrama de bloques de la figura 6-4a). La salida en serie (*SO*) del registro *A* se conecta a la entrada en serie (*SI*) del registro *B*. Para evitar la pérdida de información almacenada en el registro de origen, se hace que la información del registro *A* circule conectando la salida en serie con la entrada en serie. El contenido inicial del registro *B* se desplaza hacia su salida en serie y se pierde a menos que se transfiera a un tercer registro de desplazamiento. La entrada de control de desplazamiento determina cuándo y cúantas veces se desplazan los registros. Esto se hace con una compuerta AND que permite el paso de pulsos de reloj a las terminales *CLK* únicamente cuando el control de desplazamiento está activo.



a) Diagrama de bloques



b) Diagrama de temporización

FIGURA 6-4
Transferencia en serie del registro *A* al registro *B*

Tabla 6-1
Ejemplo de transferencia en serie

Pulso de temporización	Registro de desplazamiento *A*	Registro de desplazamiento *B*
Valor inicial	1 0 1 1	0 0 1 0
Después de T_1	1 1 0 1	1 0 0 1
Después de T_2	1 1 1 0	1 1 0 0
Después de T_3	0 1 1 1	0 1 1 0
Después de T_4	1 0 1 1	1 0 1 1

Supongamos que cada uno de los registros de desplazamiento tiene cuatro bits. La unidad de control que supervisa la transferencia debe diseñarse de tal manera que habilite los registros de desplazamiento, a través de la señal de control de desplazamiento, durante un tiempo fijo de cuatro pulsos de reloj. Esto se muestra en el diagrama de temporización de la figura 6-4b). La señal de control de desplazamiento se sincroniza con el reloj y cambia de valor inmediatamente después del flanco negativo del reloj. Los cuatro pulsos de reloj siguientes encuentran la señal de control de desplazamiento en el estado activo, así que la salida de la compuerta AND conectada a las entradas *CLK* produce cuatro pulsos, T_1, T_2, T_3 y T_4. Cada borde ascendente del pulso causa un desplazamiento en ambos registros. El cuarto pulso cambia el control de desplazamiento a 0 y los registros de desplazamiento quedan inhabilitados.

Supongamos que el contenido binario de *A* antes del desplazamiento es 1011 y que el de *B* es 0010. La transferencia en serie de *A* a *B* se efectúa en cuatro pasos, como se indica en la tabla 6-1. Con el primer pulso T_1, el bit de la extrema derecha de *A* se desplaza al bit de la extrema izquierda de *B* y también se circula a la posición de extrema izquierda de *A*. Al mismo tiempo, todos los bits de *A* y *B* se desplazan una posición a la derecha. La anterior salida en serie de *B*, en la posición de extrema derecha, se pierde, y su valor cambia de 0 a 1. Los tres pulsos siguientes efectúan operaciones idénticas, desplazando los bits de *A* a *B*, uno por uno. Después del cuarto desplazamiento, el control de desplazamiento cambia a 0 y ambos registros, *A* y *B*, tienen el valor 1011. Así, el contenido de *A* se transfiere a *B*, pero permanece inalterado.

La diferencia entre los modos de operación en serie y en paralelo deberá ser obvia por este ejemplo. En el modo paralelo, se cuenta con información de todos los bits de un registro, y todos los bits se pueden transferir simultáneamente durante un pulso de reloj. En el modo en serie, los registros tienen una sola entrada en serie y una sola salida en serie. La información se transfiere bit por bit mientras los registros se desplazan en la misma dirección.

Suma en serie

Las operaciones de las computadoras digitales por lo regular se efectúan en paralelo porque este modo de operación es más rápido. Las operaciones en serie son más lentas, pero tienen la ventaja de requerir menos equipo. Para ilustrar el modo de operación en serie, presentaremos aquí el diseño de un sumador en serie. Su contraparte paralela se presentó en la sección 4-4.

Los dos números binarios que se sumarán en serie se almacenan en dos registros de desplazamiento. Los bits se suman par por par utilizando un solo circuito de sumador completo (SC), como se observa en la figura 6-5. El acarreo de salida del sumador completo se transfiere a un flip-flop *D*. La salida de este flip-flop se utiliza entonces como acarreo de entrada para el siguiente par de bits significativos. El bit de suma de la salida *S* del sumador completo podría transferirse a un tercer registro de desplazamiento. Al desplazar la suma a *A* mientras se des-

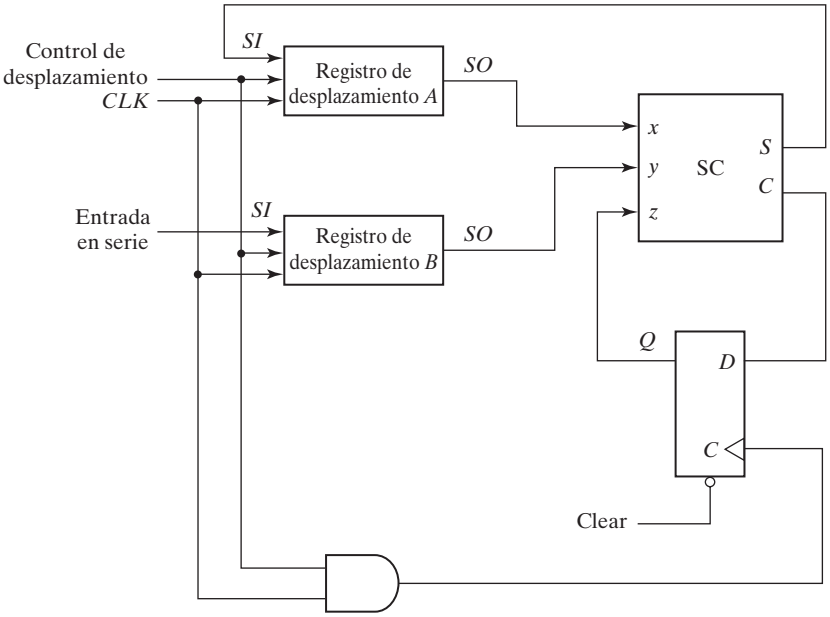


FIGURA 6-5
Sumador en serie

plazan hacia afuera los bits de *A*, es posible utilizar un solo registro para almacenar los bits tanto de un sumando como de la suma. La entrada en serie del registro *B* sirve para transferir a *B* un nuevo número binario mientras los bits del sumando se desplazan hacia afuera durante la suma.

El sumador en serie funciona como sigue. Inicialmente, el registro *A* contiene el primer sumando, el registro *B* contiene el segundo sumando y el flip-flop de acarreo está en 0. Las salidas (*SO*) de *A* y *B* alimentan un par de bits significativos al sumador completo en *x* y *y*. La salida *Q* del flip-flop alimenta el acarreo de entrada en *z*. El control de desplazamiento habilita ambos registros y el flip-flop de acarreo, de modo que, en el siguiente pulso de reloj, ambos registros se desplazarán una vez a la derecha; el bit de suma de *S* ingresará en el flip-flop de extrema izquierda de *A*, y el acarreo de salida se transferirá al flip-flop *Q*. El control de desplazamiento habilita los registros durante un número de pulsos de reloj igual al número de bits que hay en los registros. Con cada pulso de reloj sucesivo, un nuevo bit de suma se transfiere a *A*, un nuevo acarreo se transfiere a *Q* y ambos registros se desplazan una vez a la derecha. Este proceso continúa hasta que el control de desplazamiento se inhabilita. Así pues, la suma se efectúa pasando cada par de bits, junto con el acarreo anterior, por un solo circuito de sumador completo, y transfiriendo la suma, bit por bit, al registro *A*.

Inicialmente, *A* y el flip-flop de acarreo se ponen en 0 (con *clear*), y luego se suma el primer número de *B*. Mientras *B* se desplaza a través del sumador completo, se transfiere a él un segundo número a través de su entrada en serie. Luego el segundo número se suma al contenido del registro *A* mientras un tercer número se transfiere en serie al registro *B*. Esto puede repetirse para efectuar la suma de dos, tres o más números y acumular su suma en el registro *A*.

Si se compara el sumador en serie con el sumador paralelo descrito en la sección 4-4, notaremos varias diferencias. El sumador paralelo utiliza registros de carga paralela, mientras que el sumador en serie usa registros de desplazamiento. El número de circuitos de sumador

completo en el sumador paralelo es igual al número de bits de los números binarios, mientras que el sumador en serie sólo requiere un circuito de sumador completo y un flip-flop de acarreo. Sin contar los registros, el sumador paralelo es un circuito combinacional, mientras que el sumador en serie es un circuito secuencial. El circuito secuencial del sumador en serie consiste en un sumador completo y un flip-flop que almacena el acarreo de salida. Esto es típico en las operaciones en serie porque el resultado de una operación en un tiempo de un bit podría depender no sólo de las entradas actuales, sino también de las entradas anteriores que se deben almacenar en flip-flops.

Para demostrar que es posible diseñar operaciones en serie empleando el procedimiento de circuitos secuenciales, volveremos a diseñar el sumador en serie utilizando una tabla de estados. Primero, suponga que contamos con dos registros de desplazamiento para almacenar los números binarios que se sumarán en serie. Las salidas en serie de los registros se designarán x y y. El circuito secuencial a diseñar no incluirá los registros de desplazamiento; éstos se insertarán después para mostrar el circuito completo. El circuito secuencial propiamente dicho tiene las dos entradas, x y y, que alimentan un par de bits significativos, una salida S que genera el bit de suma y un flip-flop Q para almacenar el acarreo. La tabla de estados que especifica el circuito secuencial se presenta en la tabla 6-2. El estado actual de Q es el valor actual del acarreo. El acarreo que está en Q se suma a las entradas x y y para producir el bit de suma en la salida S. El siguiente estado de Q es igual al acarreo de salida. Vemos que las filas de la tabla de estados son idénticas a las de la tabla de verdad de un sumador completo, excepto que el acarreo de entrada ahora es el estado actual de Q y el acarreo de salida ahora es el siguiente estado de Q.

Si usamos un flip-flop D para Q, el circuito se reducirá al que se muestra en la figura 6-5. Si usamos un flip-flop JK para Q, será necesario determinar los valores de las entradas J y K consultando la tabla de excitación (tabla 5-12). Esto se hace en las últimas dos columnas de la tabla 6-2. Las dos ecuaciones de entrada de flip-flop y la ecuación de salida se simplifican por medio de mapas para obtener

$$J_Q = xy$$
$$K_Q = x'y' = (x + y)'$$
$$S = x \oplus y \oplus Q$$

Tabla 6-2
Tabla de estados del sumador en serie

Estado actual	Entradas		Siguiente estado	Salida	Entradas del flip-flop	
Q	X	y	Q	S	J_Q	K_Q
0	0	0	0	0	0	X
0	0	1	0	1	0	X
0	1	0	0	1	0	X
0	1	1	1	0	1	X
1	0	0	0	1	X	1
1	0	1	1	0	X	0
1	1	0	1	0	X	0
1	1	1	1	1	X	0

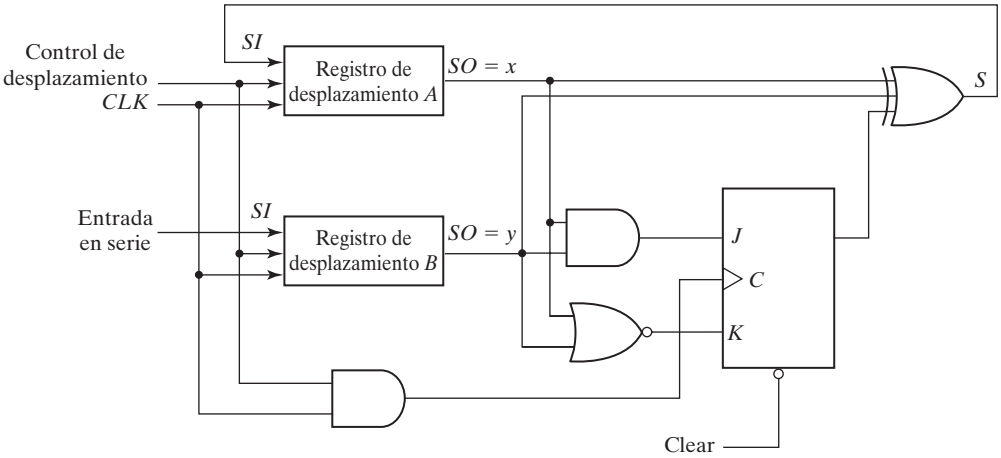


FIGURA 6-6
Segunda forma del sumador en serie

El diagrama de circuito se reproduce en la figura 6-6. El circuito consta de tres compuertas y un flip-flop *JK*. Se han incluido los dos registros de desplazamiento en el diagrama para mostrar el sumador en serie completo. Cabe señalar que la salida *S* es función no sólo de *x* y *y*, sino también del estado actual de *Q*. El siguiente estado de *Q* es función del estado actual de *Q* y de los valores *x* y *y* que salen de las salidas en serie de los registros de desplazamiento.

Registro de desplazamiento universal

Si las salidas de flip-flop de un registro de desplazamiento están accesibles, la información que se introduce en serie por desplazamiento se puede sacar en paralelo de las salidas de los flip-flops. Si se añade la capacidad de carga en paralelo a un registro de desplazamiento, los datos introducidos en paralelo se podrán sacar en serie desplazando los datos almacenados en el registro.

Algunos registros de desplazamiento proporcionan las terminales de entrada y salida necesarias para la transferencia en paralelo. También podrían tener capacidad de desplazamiento tanto a la derecha como a la izquierda. El registro de desplazamiento más general tiene estas capacidades:

1. Un control de despeje (*clear*) para poner en ceros el registro.
2. Una entrada de reloj (*clock*) para sincronizar las operaciones.
3. Un control de desplazamiento a derecha (*shift-right*) para habilitar la operación de desplazamiento a la derecha, y las líneas de *entrada* y *salida en serie* asociadas al desplazamiento a la derecha.
4. Un control de desplazamiento a izquierda (*shift-left*) para habilitar la operación de desplazamiento a la izquierda, y las líneas de *entrada* y *salida en serie* asociadas al desplazamiento a la izquierda.
5. Un control de carga en paralelo (*parallel-load*) para habilitar la transferencia en paralelo y las *n* líneas de entrada asociadas a la transferencia en paralelo.
6. *n* líneas de salida en paralelo.
7. Un control de estado que deja la información del registro como está, en presencia del reloj.

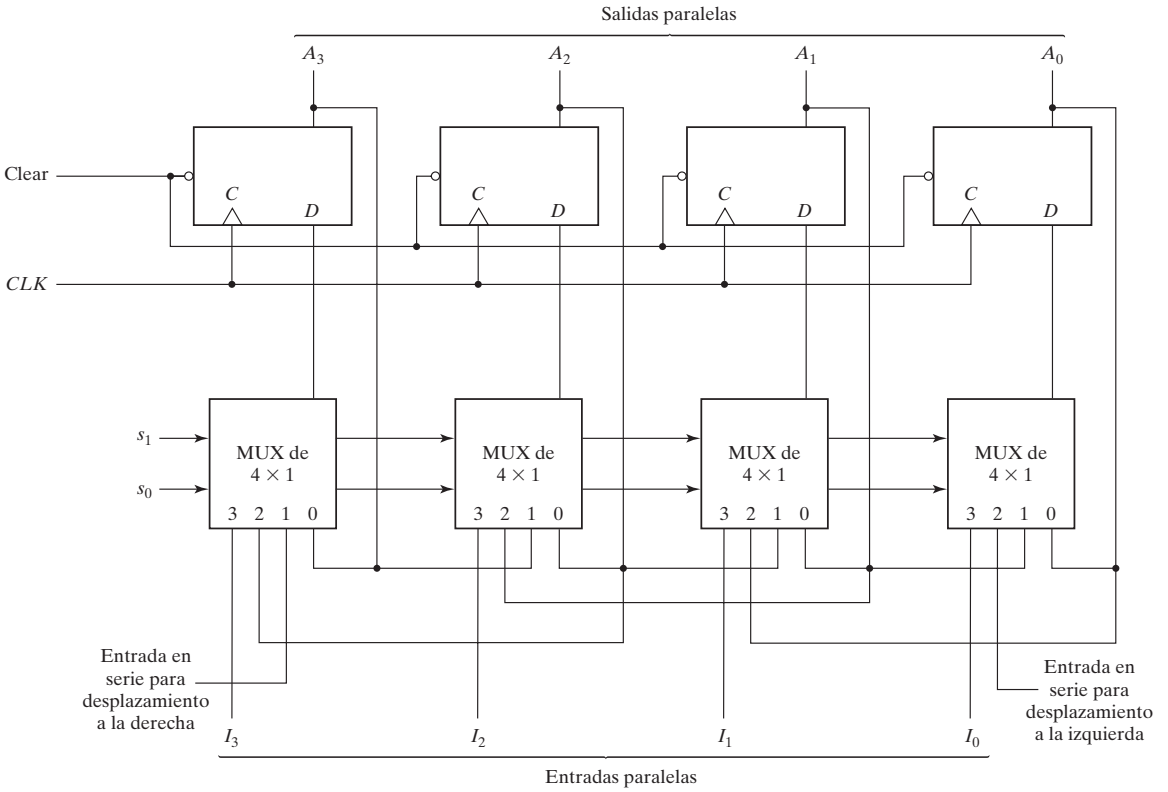


FIGURA 6-7
Registro de desplazamiento universal de 4 bits

Otros registros de desplazamiento podrían tener sólo algunas de las funciones anteriores, con una operación de desplazamiento por lo menos.

Un registro que sólo puede desplazar en una dirección es un registro de desplazamiento unidireccional. Uno que puede hacerlo en ambas direcciones es un registro de desplazamiento bidireccional. Si el registro tiene ambos desplazamientos y capacidad de carga paralela, se denomina *registro de desplazamiento universal*.

En la figura 6-7 se presenta el diagrama de un registro de desplazamiento universal de cuatro bits que posee todas las capacidades de la lista anterior. Consta de cuatro flip-flops *D* y cuatro multiplexores. Los cuatro multiplexores tienen dos entradas de selección en común, s_1 y s_0. La entrada 0 de cada multiplexor se selecciona cuando $s_1s_0 = 00$, la entrada 1 se selecciona cuando $s_1s_0 = 01$, y de manera análoga para las otras dos entradas. Las entradas de selección controlan el modo de operación del registro según las funciones enumeradas en la tabla 6-3. Cuando $s_1s_0 = 00$, el valor actual del registro se aplica a las entradas *D* de los flip-flops. Esta condición establece una trayectoria desde la salida de cada flip-flop hasta la entrada del mismo flip-flop. El siguiente borde de reloj transfiere a cada flip-flop el valor binario que con-

Tabla 6-3
Tabla de función para el registro de la figura 6-7

Control de modo		
s_1	s_0	**Operación del *registro***
0	0	Sin cambio
0	1	Desplazamiento a la derecha
1	0	Desplazamiento a la izquierda
1	1	Carga en paralelo

tenía antes, así que no hay cambio de estado. Cuando $s_1s_0 = 01$, la terminal 1 de las entradas de multiplexor tiene una trayectoria a las entradas D de los flip-flops. Esto causa una operación de desplazamiento a la derecha, transfiriéndose la entrada en serie al flip-flop A_3. Cuando $s_1s_0 = 10$, el resultado es una operación de desplazamiento a la izquierda, y la otra entrada en serie pasa al flip-flop A_0. Por último, cuando $s_1s_0 = 11$, la información binaria que está en las líneas de entrada paralelas se transfiere al registro simultáneamente durante el siguiente borde de reloj.

Los registros de desplazamiento se usan mucho como interfaz de sistemas digitales situados lejos uno del otro. Por ejemplo, supongamos que es necesario transmitir una cantidad de n bits entre dos puntos. Si la distancia es grande, sería costoso usar n líneas para transmitir los n bits en paralelo. Resulta más económico usar una sola línea y transmitir la información en serie, bit por bit. Los n bits de datos se colocan en paralelo en un registro del transmisor y luego se transmiten en serie por la línea común. El receptor acepta los datos en serie en un registro de desplazamiento. Una vez que ha recibido los n bits, éstos se pueden tomar de las salidas del registro en paralelo. Así, el transmisor efectúa una conversión de los datos, de paralelo a serie, y el receptor efectúa una conversión de serie a paralelo.

6-3 CONTADORES DE RIZO

Un registro que pasa por una sucesión preescrita de estados cuando se aplican pulsos de entrada se denomina contador. Los pulsos de entrada podrían ser pulsos de reloj u originarse en alguna fuente externa, y podrían presentarse a intervalos fijos de tiempo o al azar. La sucesión de estados podría seguir la sucesión numérica binaria o cualquier otro orden. Un contador que sigue la sucesión numérica binaria es un contador binario. Un contador binario de n bits consiste en n flip-flops y puede contar en binario desde 0 hasta $2^n - 1$.

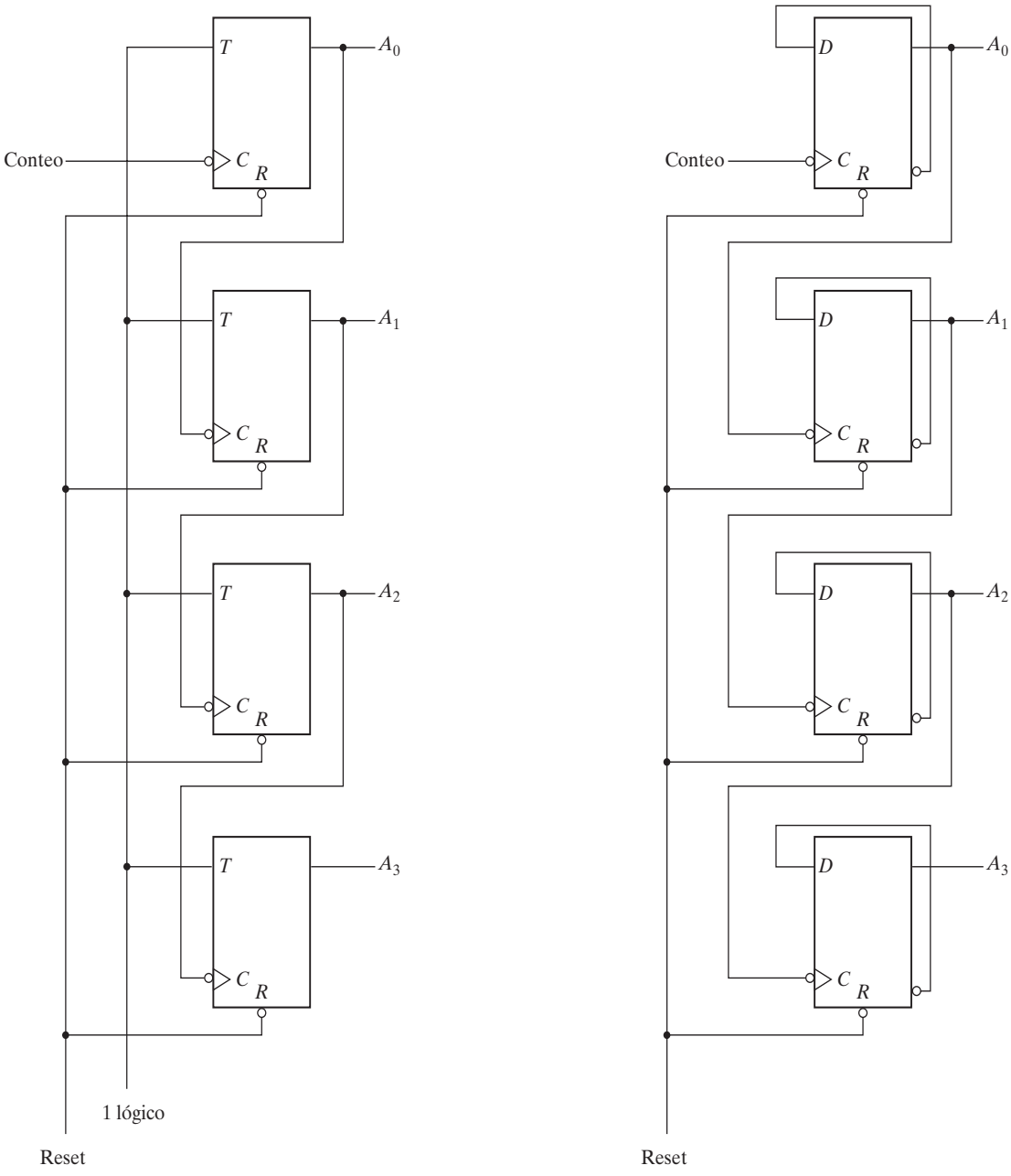
Los contadores se dividen en dos categorías: contadores de rizo y contadores sincrónicos. En un contador de rizo, la transición de salida del flip-flop sirve como disparador de otros flip-flops. Dicho de otro modo, la entrada C de algunos flip-flops, o de todos, se dispara, no con los pulsos del reloj común, sino con la transición que se da en otras salidas de flip-flop. En un contador sincrónico, las salidas C de todos los flip-flops reciben el reloj común. Se hablará de los contadores sincrónicos en las dos secciones siguientes. Aquí presentaremos los contadores de rizo binario y BCD y explicaremos su funcionamiento.

Contador binario de rizo

Un contador binario de rizo consiste en una conexión en serie de flip-flops complementadores; la salida de cada flip-flop se conecta a la entrada *C* del siguiente flip-flop de orden superior. El flip-flop que contiene el bit menos significativo recibe los pulsos de conteo que llegan. Es posible construir un flip-flop complementador con un flip-flop *JK* cuyas entradas *J* y *K* se han conectado entre sí, o con un flip-flop *T*. Una tercera posibilidad es usar un flip-flop *D* con la salida de complemento conectada a la entrada *D*. Así, la entrada *D* siempre es el complemento del estado actual, y el siguiente pulso de reloj hace que el flip-flop se complemente. En la figura 6-8 se presenta el diagrama lógico de dos contadores binarios de rizo de cuatro bits. El contador se construye con flip-flops complementadores del tipo *T* en la parte a) y de tipo *D* en la parte b). La salida de cada flip-flop se conecta a la entrada *C* del siguiente flip-flop sucesivo. El flip-flop que contiene el bit menos significativo recibe los pulsos de conteo que llegan. Las entradas *T* de todos los flip-flops de a) se conectan a un 1 lógico permanente. Esto hace que cada flip-flop se complemente si la señal de su entrada *C* sufre una transición negativa. La burbuja junto al símbolo de indicador dinámico de *C* denota que los flip-flops responden a la transición de borde negativo de la entrada. La transición negativa se presenta cuando la salida del flip-flop anterior al que *C* está conectada cambia de 1 a 0.

Para entender el funcionamiento del contador binario de rizo de cuatro bits, resulta útil remitirse a la lista de los primeros nueve números binarios de la tabla 6-4. El conteo inicia con el 0 binario y se incrementa en uno con cada pulso de conteo introducido. Después de la cuenta de 15, el contador vuelve a 0 para repetir la cuenta. El bit menos significativo, A_0, se complementa con cada pulso de conteo introducido. Cada vez que A_0 pasa de 1 a 0, complementa a A_1. Cada vez que A_1 pasa de 1 a 0, complementa a A_2. Cada vez que A_2 pasa de 1 a 0, complementa a A_3, y así sucesivamente con los demás bits de orden más alto que tenga el contador. Por ejemplo, considere la transición de la cuenta 0011 a 0100. A_0 se complementa con el pulso de conteo. Puesto que A_0 pasa de 1 a 0, dispara a A_1 y lo complementa. El resultado de esto es que A_1 cambia de 1 a 0, lo que a su vez hace que A_2 se complemente, cambiando de 0 a 1. A_2 no dispara a A_3 porque A_2 produce una transición positiva y el flip-flop sólo responde a transiciones negativas. Así, el conteo de 0011 a 0100 se logra cambiando los bits uno por uno, de modo que el conteo pasa primero de 0011 a 0010, luego a 0000 y por último a 0100. Los flip-flops cambian uno por uno sucesivamente, y la señal se propaga por el contador como rizo, de una etapa a la siguiente.

Un contador binario que cuenta al revés se llama contador binario de cuenta regresiva. En él, la cuenta binaria se decrementa en uno con cada pulso de conteo que llega. La cuenta de un contador de cuenta regresiva de cuatro bits inicia en el 15 binario y continúa con las cuentas binarias 14, 13, 12, . . . , 0 y luego de vuelta a 15. Una lista de la sucesión de conteo de un contador binario de cuenta regresiva muestra que el bit menos significativo se complementa con cada pulso de conteo. Cualquier otro bit de la sucesión se complementa si el bit menos significativo precedente pasa de 0 a 1. Por tanto, el diagrama de un contador binario de cuenta regresiva es igual al de la figura 6-8, a condición de que todos los flip-flops se disparen con el borde positivo del reloj. (No debe haber burbuja en las entradas *C*.) Si se usan flip-flops disparados por borde negativo, la entrada *C* de cada flip-flop deberá conectarse a la salida de complemento del flip-flop anterior. Así, cuando la salida verdadera cambie de 0 a 1, el complemento cambiará de 1 a 0 y complementará el siguiente flip-flop, como debe ser.



a) Con flip-flops *T*

b) Con flip-flops *D*

FIGURA 6-8
Contador binario de rizo de cuatro bits

Tabla 6-4
Sucesión binaria de conteo

A_3	A_2	A_1	A_0
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0

Contador BCD de rizo

Un contador decimal sigue una sucesión de diez estados y vuelve a 0 después de la cuenta de 9. Un contador así necesita por lo menos cuatro flip-flops para representar cada dígito decimal, ya que un dígito decimal se representa con un código binario de por lo menos cuatro bits. La sucesión de estados de un contador decimal depende del código binario empleado para representar un dígito decimal. Si se usa BCD, la sucesión de estados es la que se aprecia en el diagrama de estados de la figura 6-9. Este contador es similar al binario, excepto que el estado que sigue a 1001 (código del dígito decimal 9) es 0000 (código para el dígito decimal 0).

En la figura 6-10 se presenta el diagrama lógico de un contador BCD de rizo que utiliza flip-flops *JK*. Las cuatro salidas se designan con la letra *Q* seguida de un subíndice numérico igual al peso binario del bit correspondiente en el código BCD. Vemos que la salida de Q_1 se aplica a las entradas *C* tanto de Q_2 como de Q_8, y que la salida de Q_2 se aplica a la entrada *C* de Q_4. Las entradas *J* y *K* se conectan a una señal de 1 lógico permanente o bien a salidas de otros flip-flops.

Un contador de rizo es un circuito secuencial asincrónico. Las señales que afectan la transición del flip-flop dependen de la forma en que cambian de 1 a 0. Podemos explicar el funcionamiento del contador con una lista de condiciones para transiciones de flip-flop. Estas condiciones se deducen del diagrama lógico y del conocimiento de la forma en que opera un flip-flop *JK*. Recuerde que, cuando la entrada *C* cambia de 1 a 0, el flip-flop se

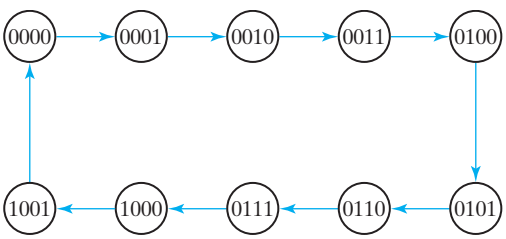


FIGURA 6-9
Diagrama de estados de un contador BCD decimal

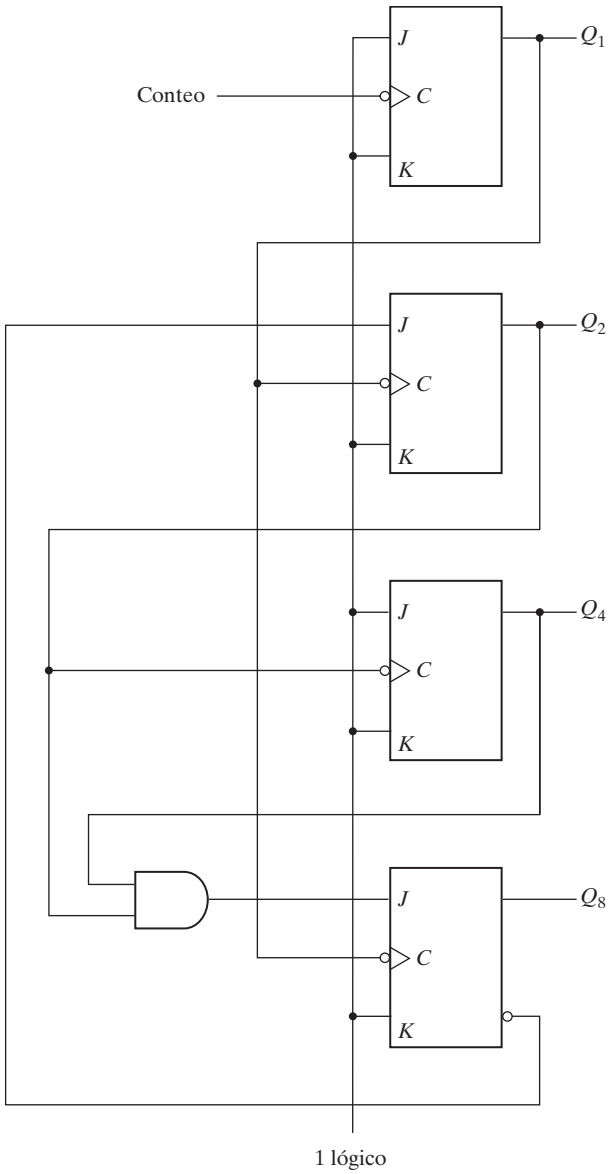


FIGURA 6-10
Contador BCD de rizo

establece si $J = 1$, se restablece si $K = 1$, se complementa si $J = K = 1$ y no cambia si $J = K = 0$.

Para verificar que estas condiciones producen la sucesión requerida por un contador BCD de rizo, es necesario comprobar que las transiciones de flip-flop pasen sucesivamente a los estados especificados por el diagrama de estados de la figura 6-9. Q_1 cambia de estado después de cada pulso de reloj. Q_2 se complementa cada vez que Q_1 pasa de 1 a 0, en tanto $Q_8 = 0$.

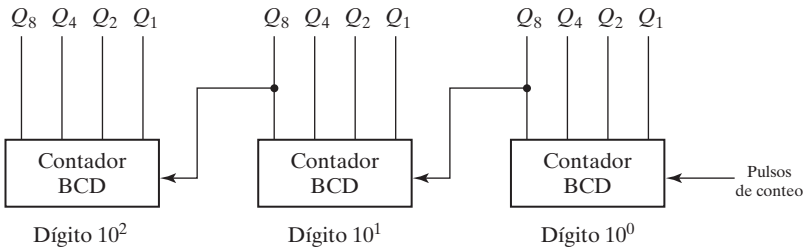


FIGURA 6-11
Diagrama de bloques de un contador BCD decimal de tres décadas

Cuando Q_8 cambia a 1, Q_2 permanece en 0. Q_4 se complementa cada vez que Q_2 cambia de 1 a 0. Q_8 permanece en 0 en tanto Q_2 o Q_4 sean 0. Si tanto Q_2 como Q_4 cambian a 1, Q_8 se complementa cuando Q_1 cambia de 1 a 0. Q_8 se despeja en la siguiente transición de Q_1.

El contador BCD de la figura 6-10 es un contador de *década*, pues cuenta de 0 a 9. Para contar en decimal de 0 a 99, se necesita un contador de dos décadas. Para contar de 0 a 999, es necesario uno de tres décadas. Los contadores de varias décadas se construyen conectando contadores BCD en cascada, uno para cada década. En la figura 6-11 se ilustra un contador de tres décadas. Las entradas de la segunda y la tercera décadas provienen de Q_8 de la década anterior. Cuando Q_8 de una década cambia de 1 a 0, dispara el conteo de la siguiente década de orden superior mientras su propia década cambia de 9 a 0.

6-4 CONTADORES SINCRÓNICOS

Los contadores sincrónicos difieren de los de rizo en que se aplican pulsos de reloj a las entradas de todos los flip-flops. Un mismo reloj dispara todos los flip-flops simultáneamente en vez de hacerlo uno por uno sucesivamente como en los contadores de rizo. La decisión de si un flip-flop debe complementarse o no depende de los valores de las entradas de datos, como T o J y K, en el momento en que llega el borde de reloj. Si $T = 0$ o $J = K = 0$, el flip-flop no cambia de estado. Si $T = 1$ o $J = K = 1$, el flip-flop se complementa.

Ya presentamos el procedimiento de diseño de los contadores sincrónicos en la sección 5-7, y el diseño de un contador binario de tres bits se efectuó con la ayuda de la figura 5-31. En esta sección se presentarán algunos contadores sincrónicos representativos y se explicará su funcionamiento.

Contador binario

El diseño de un contador binario sincrónico es tan sencillo que no es preciso realizar un proceso secuencial de diseño lógico. En un contador binario sincrónico, el flip-flop de la posición menos significativa se complementa con cada pulso. Un flip-flop en cualquier otra posición se complementa cuando todos los bits de las posiciones significativas inferiores son 1. Por ejemplo, si el estado actual de un contador de cuatro bits es $A_3A_2A_1A_0 = 0011$, el siguiente conteo será 0100. A_0 siempre se complementa. A_1 se complementa porque el estado actual de $A_0 = 1$. A_2 se complementa porque el estado actual de $A_1A_0 = 11$. En cambio, A_3 no se complementa porque el estado actual de $A_2A_1A_0 = 011$, y no cumple la condición de "todos unos".

Los contadores binarios sincrónicos tienen un patrón regular y se pueden construir con flip-flops complementadores y compuertas. El patrón regular se distingue en el contador de cuatro bits que se representa en la figura 6-12. Las entradas *C* de todos los flip-flops se conectan a un reloj común. El contador se habilita con la entrada de habilitar contador. Si esa entrada es 0, todas las entradas *J* y *K* son 0 y el reloj no cambia el estado del contador. La primera etapa A_0

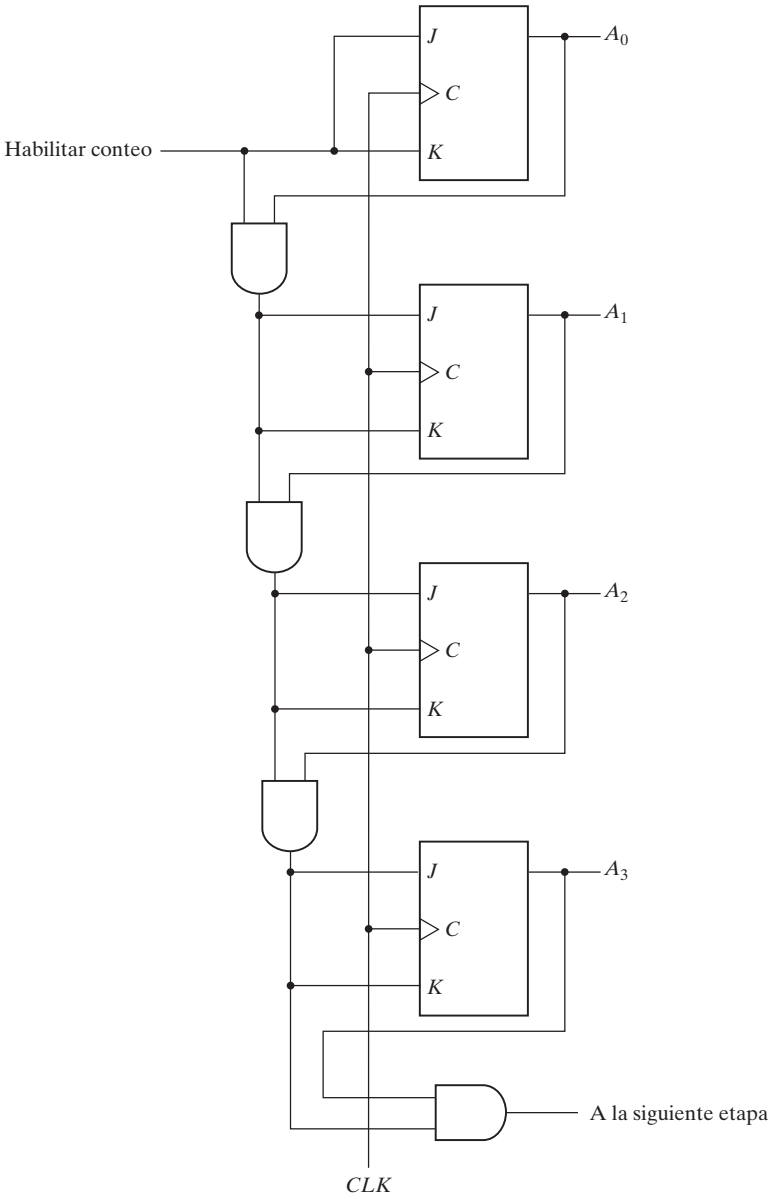


FIGURA 6-12
Contador binario sincrónico de cuatro bits

tiene 1 en su *J* y en su *K* si el contador está habilitado. Las otras entradas *J* y *K* tienen 1 si todas las etapas anteriores, menos significativas, producen 1 y el conteo está habilitado. La cadena de compuertas AND genera la lógica requerida para las entradas *J* y *K* de cada etapa. El contador puede extenderse a cualquier cantidad de etapas, cada una de las cuales tiene un flip-flop adicional y una compuerta AND que produce una salida de 1 si las salidas de todos los flip-flops anteriores son 1.

Cabe señalar que los flip-flops se disparan con el borde positivo del reloj. La polaridad del reloj no es fundamental en este caso como lo era en el contador de rizo. El contador sincrónico se dispara con el borde positivo del reloj, o con el negativo. Los flip-flops complementadores del contador binario pueden ser del tipo *JK* o del tipo *T*, o del tipo *D* con compuertas XOR. La equivalencia de los tres tipos se señala en la figura 5-13.

Contador binario ascendente-descendente

Un contador binario sincrónico de cuenta regresiva pasa por los estados binarios en el orden inverso, de 1111 hasta 0000, pasando después a 1111 para repetir el conteo. Es posible diseñar un contador de cuenta regresiva de la forma acostumbrada, pero el resultado es predecible por inspección del conteo binario descendente. El bit de la posición menos significativa se complementa con cada pulso. Un bit en cualquier otra posición se complementa si todos los bits menos significativos son 0. Por ejemplo, el estado que sigue al estado actual 0100 es 0011. El bit menos significativo siempre se complementa. El segundo bit hacia la izquierda se complementa porque el primero es 0. El tercero se complementa porque los dos primeros son 0. El cuarto bit, en contraste, no cambia porque no todos los bits menos significativos son 0.

Podemos construir un contador binario de cuenta regresiva como el de la figura 6-12, excepto que las entradas de las compuertas AND deben provenir de las salidas complementadas de los flip-flops anteriores, no de las salidas normales. Es posible combinar las dos operaciones en un solo circuito para formar un contador capaz de contar hacia arriba o hacia abajo. En la figura 6-13 se representa el circuito de un contador binario **ascendente-descendente** que utiliza flip-flops *T*. Tiene una entrada de control para conteo ascendente (arriba) y una entrada de control para conteo descendente (abajo). Cuando la entrada arriba es 1, el circuito cuenta hacia arriba, porque las entradas *T* reciben sus señales de las salidas normales de los flip-flops anteriores. Cuando la entrada abajo es 1 y la entrada arriba es 0, el circuito cuenta hacia abajo, porque se aplican a las entradas *T* las salidas complementadas de los flip-flops anteriores. Si ambas entradas, arriba y abajo, son 0, el circuito no cambia de estado y permanece en la misma cuenta. Si ambas entradas son 1, el circuito cuenta hacia arriba. Esto garantiza que sólo una operación se efectúe en todo momento.

Contador BCD

Los contadores BCD cuentan en decimal codificado en binario, de 0000 hasta 1001 y luego regresan a 0000. Debido al regreso a cero después de contar hasta 9, el contador BCD no sigue un patrón regular como en el conteo binario directo. Para deducir el circuito de un contador BCD sincrónico, es preciso efectuar un procedimiento secuencial de diseño de circuitos.

La tabla de estados de un contador BCD se presenta en la tabla 6-5. Las condiciones de entrada de los siete flip-flops se obtienen de las condiciones de estado actual y siguiente estado. También se da una salida *y* en la tabla. Esta salida es 1 cuando el estado actual es 1001. Así, *y*

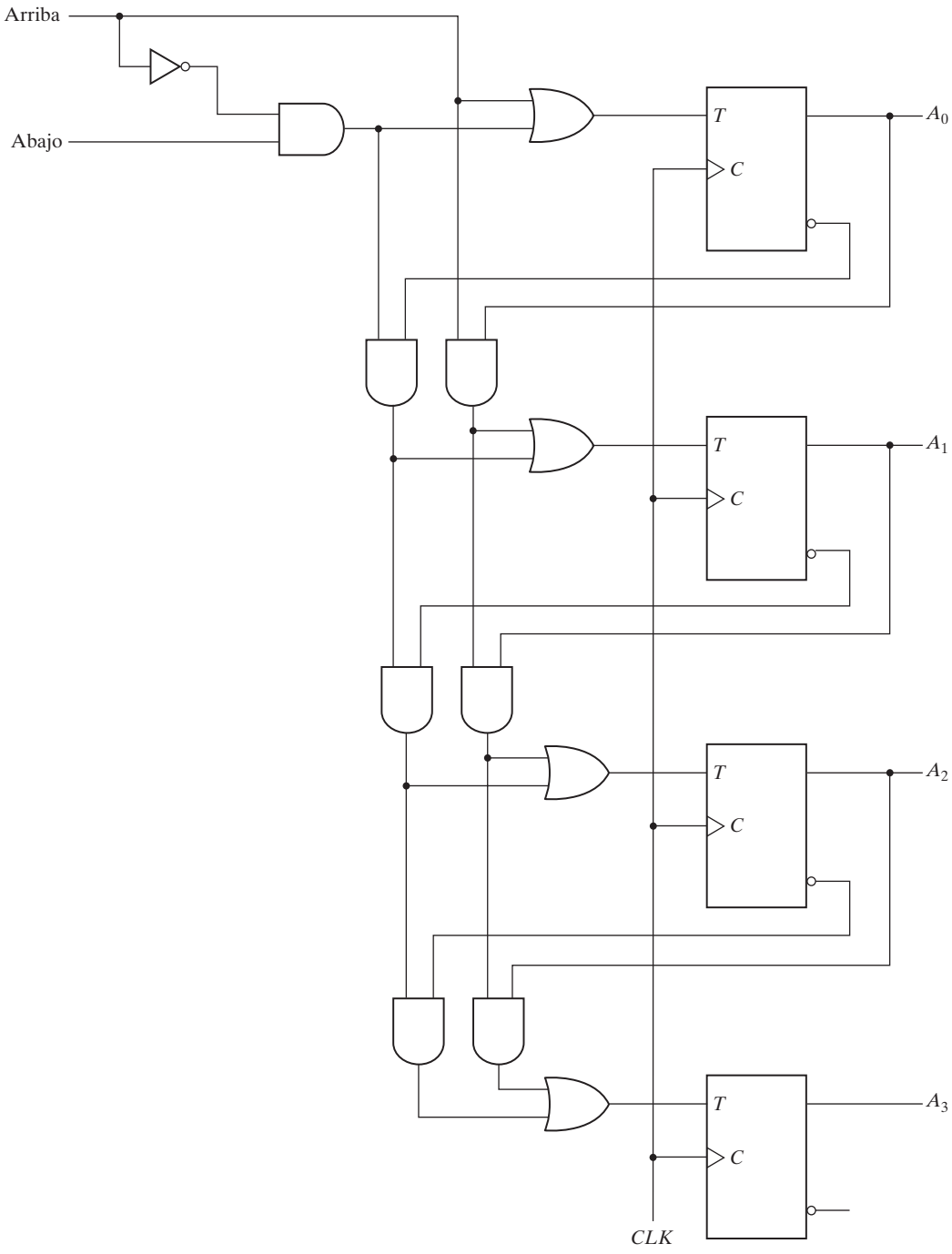


FIGURA 6-13
Contador binario ascendente-descendente de cuatro bits

Tabla 6-5
Tabla de estados para el contador BCD

Estado actual				Siguiente estado				Salida	Entradas de flip-flop			
Q_8	Q_4	Q_2	Q_1	Q_8	Q_4	Q_2	Q_1	y	TQ_8	TQ_4	TQ_2	TQ_1
0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	1	0	0	1	0	0	0	0	1	1
0	0	1	0	0	0	1	1	0	0	0	0	1
0	0	1	1	0	1	0	0	0	0	1	1	1
0	1	0	0	0	1	0	1	0	0	0	0	1
0	1	0	1	0	1	1	0	0	0	0	1	1
0	1	1	0	0	1	1	1	0	0	0	0	1
0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	0	0	1	0	0	0	0	1
1	0	0	1	0	0	0	0	1	1	0	0	1

puede habilitar el conteo de la siguiente década más significativa al tiempo que cambia la década actual de 1001 a 0000.

Las ecuaciones de entrada de los flip-flops se simplifican con la ayuda de mapas. Los estados no utilizados de los minitérminos 10 a 15 se toman como términos de indiferencia. Las funciones simplificadas son

$$T_{Q1} = 1$$
$$T_{Q2} = Q_8' Q_1$$
$$T_{Q4} = Q_2 Q_1$$
$$T_{Q8} = Q_8 Q_1 + Q_4 Q_2 Q_1$$
$$y = Q_8 Q_1$$

Es fácil dibujar el circuito con cuatro flip-flops *T*, cinco compuertas AND y una compuerta OR. Los contadores BCD sincrónicos se pueden conectar en cascada para formar un contador de números decimales de cualquier longitud. La conexión en cascada se hace como en la figura 6-11, excepto que la salida *y* se debe conectar a la entrada de conteo de la siguiente década más significativa.

Contador binario con carga paralela

Es muy común que los contadores empleados en sistemas digitales requieran una capacidad de carga paralela para transferir un número binario inicial al contador antes de la operación de conteo. La figura 6-14 representa el diagrama lógico de un registro de cuatro bits que tiene capacidad de carga paralela y puede operar como contador. Si la entrada de control de carga es 1, la operación de conteo se inhabilita y se efectúa una transferencia de datos de las cuatro entradas de datos a los cuatro flip-flops. Si ambas entradas de control son 0, los pulsos de reloj no alteran el estado del registro.

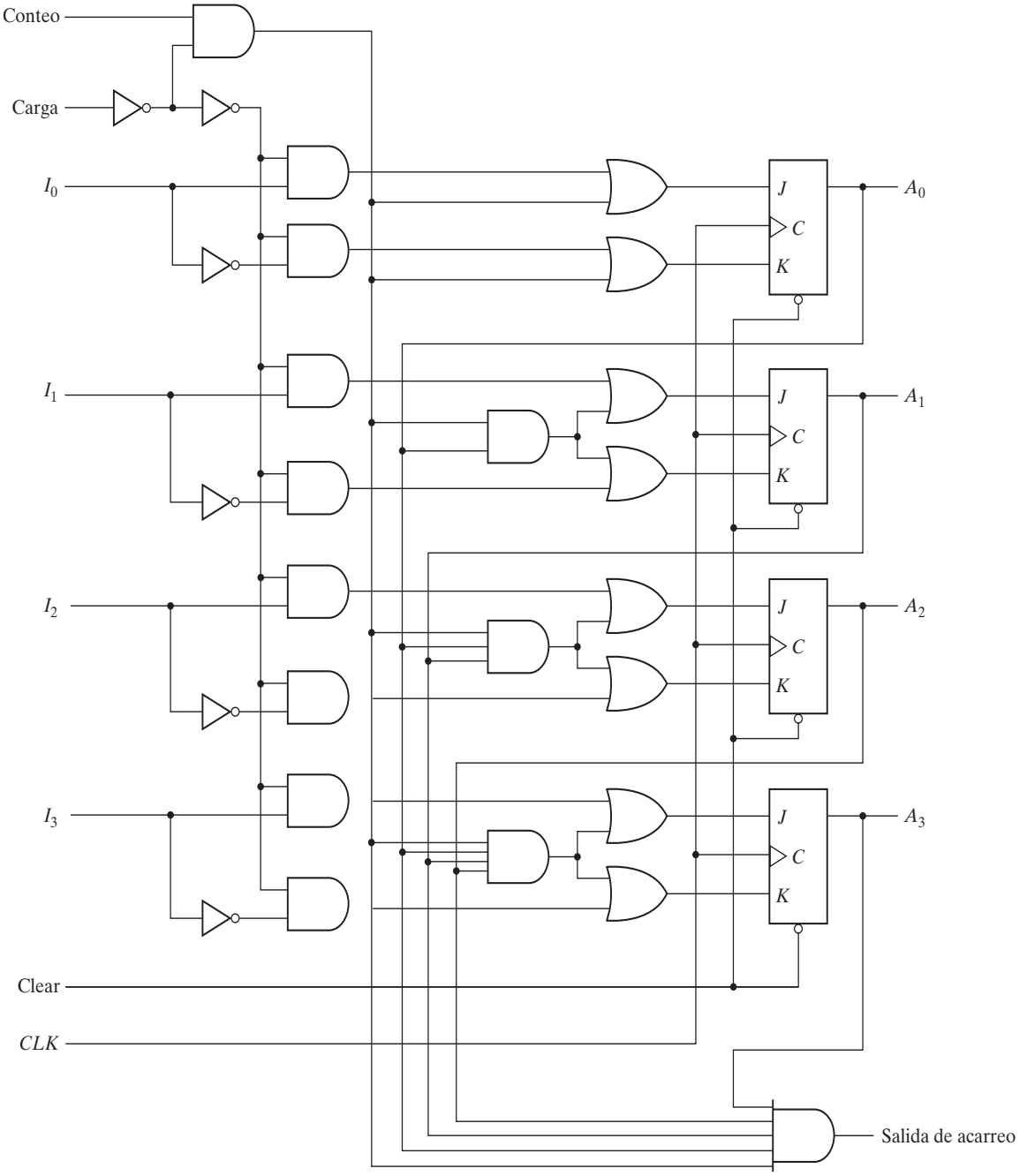


FIGURA 6-14
Contador binario de cuatro bits con carga paralela

Tabla 6-6
Tabla de función para el contador de la figura 6-14

Clear	CLK	Carga	Conteo	Función
0	X	X	X	Poner en ceros
1	↑	1	*X*	Cargar entradas
1	↑	0	1	Contar al siguiente estado binario
1	↑	0	0	Sin cambio

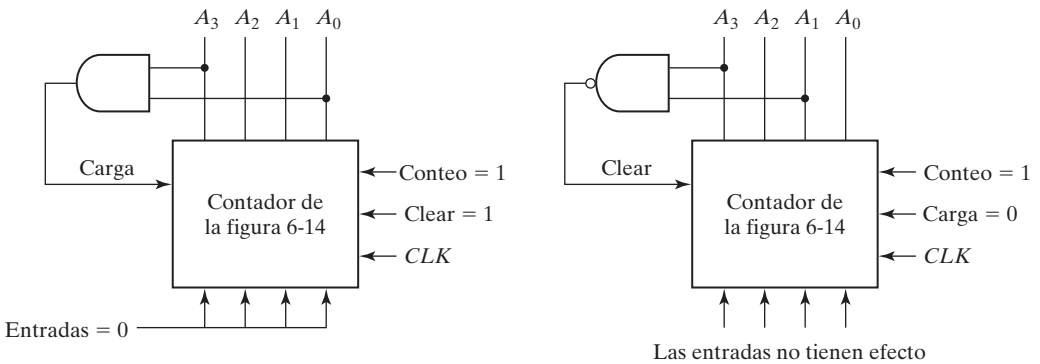
La salida de acarreo es 1 si todos los flip-flops son 1 y la entrada de conteo está habilitada. Ésta es la condición para complementar el flip-flop que contiene el siguiente bit significativo. La salida de acarreo es útil para expandir el contador a más de cuatro bits. La rapidez del contador aumenta si el acarreo se genera directamente a partir de las salidas de los cuatro flip-flops, pues ello reduce el retardo de generación del acarreo. Al pasar del estado 1111 a 0000, sólo hay un retardo de compuerta; en cambio, en la cadena de compuertas AND de la figura 6-12 hay cuatro retardos de compuerta. Asimismo, cada flip-flop se asocia a una compuerta AND que recibe directamente las salidas de todos los flip-flops anteriores en vez de conectar las compuertas AND en cadena.

El funcionamiento del contador se resume en la tabla 6-6. Las cuatro entradas de control: clear, CLK, carga y conteo, determinan el siguiente estado. La entrada clear es asincrónica y, si es 0, hace que el contador se despeje (se ponga en ceros) sin importar si hay pulsos de reloj u otras entradas. Esto se indica en la tabla con las entradas X, que representan condiciones de indiferencia para las demás entradas. La entrada clear debe estar en el estado 1 para que se efectúen todas las demás operaciones. Si la entrada de carga y la de conteo son 0, las salidas no cambian, aunque se apliquen pulsos de reloj. Una entrada de carga 1 causa una transferencia de las entradas I_0-I_3 al registro durante un borde positivo del reloj. Los datos de entrada se cargan en el registro sin importar qué valor tenga la entrada de conteo, porque ésta se inhibe cuando la entrada de carga está habilitada. La entrada de carga debe ser 0 para que la entrada de conteo controle el funcionamiento del contador.

Podemos usar un contador con carga paralela para generar cualquier sucesión de conteo deseada. La figura 6-15 muestra dos formas de usar un contador con carga paralela para generar un conteo BCD. En ambos casos, el control de conteo se pone en 1 para habilitar el conteo a través de la entrada CLK. Recuerde también que el control de entrada inhibe el conteo y que la operación de despeje es independiente de las demás entradas de control.

La compuerta AND de la figura 6-15a) detecta la ocurrencia del estado 1001. Inicialmente, el contador se pone en ceros y luego las entradas clear y conteo se ponen en 1 para que el contador esté siempre activo. En tanto la salida de la compuerta AND sea 0, cada borde positivo del reloj incrementará el contador en uno. Cuando la salida llegue a la cuenta de 1001, tanto A_0 como A_3 serán 1, y la salida de la compuerta AND será 1. Esta condición activa la entrada de carga; entonces, cuando llegue el siguiente borde de reloj, el registro no contará, sino que se cargará de sus cuatro entradas de datos. Puesto que esas cuatro entradas están conectadas a 0 lógico, se cargará un valor de ceros en el registro después de la cuenta de 1001. Así pues, el circuito efectúa el conteo de 0000 hasta 1001 y luego vuelve a 0000, como debe hacer un contador BCD.

En la figura 6-15b), la compuerta NAND detecta la cuenta de 1010, y en ese mismo instante el registro se despeja. La cuenta de 1010 no tiene oportunidad de durar un tiempo aprecia-



a) Utilizando la entrada de carga

b) Utilizando la entrada clear

FIGURA 6-15
Dos formas de construir un contador BCD empleando un contador con carga paralela

ble, porque el registro pasa inmediatamente a 0. Hay un pico momentáneo en la salida A_0 cuando la cuenta pasa de 1010 a 1011 e inmediatamente a 0000. Este pico momentáneo podría ser indeseable, y es por ello que no se recomienda esta configuración. Si el contador tiene una entrada clear sincrónica, sería posible despejar el contador con el reloj después de que se presenta la cuenta 1001.

6-5 OTROS CONTADORES

Es posible diseñar contadores que generen cualquier sucesión de estados deseada. Un contador de división entre N (también llamado contador módulo-N) pasa por una sucesión repetida de N estados. Dicha sucesión podría seguir el conteo binario o podría ser cualquier otra sucesión arbitraria. Se emplean contadores para generar señales de temporización que controlan la sucesión de operaciones de un sistema digital. También es posible construir contadores con registros de desplazamiento. En esta sección, se presentarán unos cuantos ejemplos de contadores no binarios.

Contador con estados no utilizados

Un circuito con n flip-flops tiene 2^n estados binarios. Hay ocasiones en que un circuito secuencial utiliza menos de este máximo número posible de estados. Los estados que no se usan para especificar el circuito secuencial no se incluyen en la tabla de estados. Al simplificar las ecuaciones de entrada, los estados no utilizados podrían tratarse como condiciones de indiferencia, o asignárseles siguientes estados específicos. Una vez diseñado y construido el circuito, una interferencia externa podría hacer que el circuito quede en uno de los estados no utilizados. En tal caso, será necesario asegurarse de que el circuito pase en algún momento a uno de los estados válidos para poder reanudar su operación normal. De lo contrario, si el circuito secuencial circula entre estados no utilizados, no habrá forma de que regrese a la suce-

Tabla 6-7
Tabla de estados de un contador

Estado actual			Siguiente estado			Entradas de flip-flops					
A	B	C	A	B	C	J_A	K_A	J_B	K_B	J_C	K_C
0	0	0	0	0	1	0	X	0	X	1	X
0	0	1	0	1	0	0	X	1	X	X	1
0	1	0	1	0	0	1	X	X	1	0	X
1	0	0	1	0	1	X	0	0	X	1	X
1	0	1	1	1	0	X	0	1	X	X	1
1	1	0	0	0	0	X	1	X	1	0	X

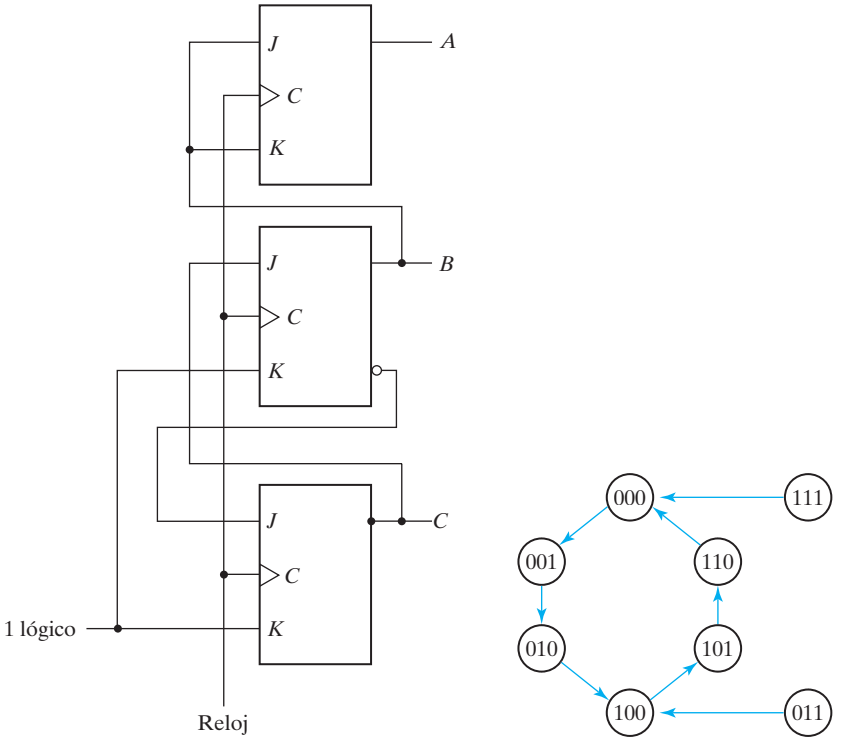
sión diseñada de transiciones de estado. Si los estados no utilizados se tratan como condiciones de indiferencia, entonces, una vez que el circuito se haya diseñado, se le deberá investigar para determinar el efecto de los estados no utilizados. El estado que sigue a un estado no utilizado se determina efectuando un análisis del circuito ya diseñado.

Como ilustración, consideremos el contador especificado en la tabla 6-7. El conteo tiene una sucesión repetida de seis estados, en la que los flip-flops B y C repiten el conteo binario 00, 01, 10, y el flip-flop A alterna entre 0 y 1 cada tres conteos. La sucesión de conteo no es binaria directa, y dos estados, 011 y 111, no están incluidos en el conteo. La decisión de usar flip-flops JK da pie a las condiciones de entrada de flip-flop que se especifican en la tabla. Las entradas K_B y K_C sólo tienen unos y cruces en sus columnas, de modo que estas entradas siempre son 1. Las demás ecuaciones de entrada de flip-flop se simplifican utilizando los minitérminos 3 y 7 como condiciones de indiferencia. Las ecuaciones simplificadas son

$$J_A = B \qquad K_A = B$$
$$J_B = C \qquad K_B = 1$$
$$J_C = B' \qquad K_C = 1$$

El diagrama lógico del contador se reproduce en la figura 6-16a). Puesto que hay dos estados no utilizados, analizaremos el circuito para determinar su efecto. Si el circuito llega a estar en el estado 011 debido a un error de señal, pasará al estado 100 después de la aplicación de un pulso de reloj. Esto se averigua por inspección del diagrama lógico, observando que, cuando $B = 1$, el siguiente borde de reloj complementa a A y pone a C en 0, y cuando $C = 1$, el siguiente borde de reloj complementa a B. De forma similar, podemos ver que el estado que sigue a 111 es 000.

En la figura 6-16b) se observa el diagrama de estados que incluye el efecto de los estados no utilizados. Si el circuito llega a quedar en uno de los estados no utilizados debido a una interferencia externa, el siguiente pulso de conteo lo transferirá a uno de los estados válidos y el circuito seguirá contando correctamente. Así pues, este circuito tiene autocorrección. Un contador tiene autocorrección si, en caso de quedar en uno de los estados no utilizados, llega a la sucesión normal de conteo después de uno o más pulsos de reloj.



a) Diagrama lógico

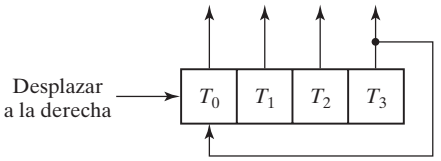
b) Diagrama de estados

FIGURA 6-16
Contador con estados no utilizados

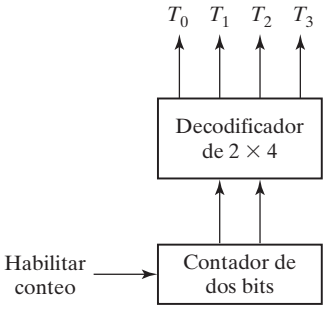
Contador anular

Las señales de temporización que controlan la sucesión de operaciones de un sistema digital se pueden generar con un registro de desplazamiento o con un contador provisto de decodificador. Un *contador anular* (o de anillo) es un registro de desplazamiento circular en el que sólo un flip-flop está establecido en cualquier instante dado; los demás están despejados. El bit solitario se desplaza de un flip-flop al siguiente para producir la sucesión de señales de temporización. En la figura 6-17a) se aprecia un registro de desplazamiento de cuatro bits conectado como contador anular. El valor inicial del registro es 1000. El bit 1 se desplaza a la derecha con cada pulso de reloj y al llegar a T_3 circula de vuelta a T_0. Cada flip-flop está en el estado 1 una vez cada cuatro ciclos de reloj y produce una de las cuatro señales de temporización que se indican en la figura 6-17c). Cada salida se convierte en 1 después de la transición de borde negativo de un pulso de reloj y sigue siendo 1 durante el siguiente ciclo de reloj.

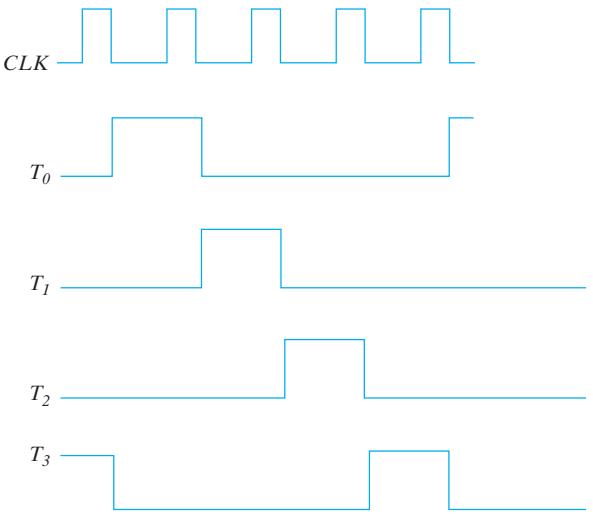
Las señales de temporización también pueden generarse con un contador de dos bits que pasa por cuatro estados distintos. El decodificador que se ilustra en la figura 6-17b) decodifica los cuatro estados del contador y genera la sucesión requerida de señales de temporización.



a) Contador anular (valor inicial = 1000)



b) Contador y decodificador



c) Sucesión de cuatro señales de temporización

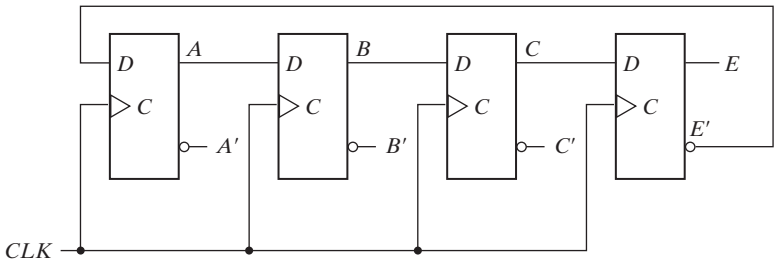
FIGURA 6-17
Generación de señales de temporización

Para generar 2^n señales de temporización, se requiere un registro de desplazamiento con 2^n flip-flops o bien un contador binario de n bits junto con un decodificador de n a 2^n líneas. Por ejemplo, podemos generar 16 señales de temporización con un registro de desplazamiento de 16 bits conectado como contador anular, o con un contador binario de cuatro bits y un decodificador de 4 a 16 líneas. En el primer caso, se necesitan 16 flip-flops. En el segundo, necesitaremos cuatro flip-flops y 16 compuertas AND de cuatro entradas para el decodificador. También

es posible generar las señales de temporización con una combinación de un registro de desplazamiento y un decodificador. En este caso, el número de flip-flops es menor que con un contador anular, y el decodificador sólo requiere compuertas de dos entradas. La combinación se denomina *contador Johnson*.

Contador Johnson

Un contador anular de k bits circula un solo bit entre los flip-flops para producir k estados distinguibles. El número de estados puede duplicarse si el registro de desplazamiento se conecta como contador anular *con extremo conmutado*. Un contador anular con extremo conmutado es un registro de desplazamiento circular en el que la salida de complemento del último flip-flop está conectada a la entrada del primer flip-flop. La figura 6-18a) muestra un registro de desplazamiento de este tipo. La conexión circular se efectúa entre la salida de complemento del flip-flop de la extrema derecha y la entrada del flip-flop de la extrema izquierda. El registro desplaza su contenido una vez a la derecha con cada pulso de reloj y, al mismo tiempo, el valor complementado del flip-flop E se transfiere al flip-flop A. Empezando en el estado despejado, el contador anular con extremo conmutado pasa por una sucesión de ocho estados, la cual se representa en la figura 6-18b). En general, un contador anular con extremo conmutado de k bits pasa por una sucesión de $2k$ estados. Partiendo de ceros, cada operación de desplazamiento inserta unos por la izquierda hasta que el registro queda lleno de unos. A continuación, se insertan ceros por la izquierda hasta que el registro vuelve a estar lleno de ceros.



a) Contador anular con extremo conmutado de cuatro etapas

Número sucesivo	Salidas de los flip-flops				Compuerta AND requerida para la salida
	A	B	C	E	
1	0	0	0	0	$A'E'$
2	1	0	0	0	AB'
3	1	1	0	0	BC'
4	1	1	1	0	CE'
5	1	1	1	1	AE
6	0	1	1	1	$A'B$
7	0	0	1	1	$B'C$
8	0	0	0	1	$C'E$

b) Sucesión de conteo y decodificación requerida

FIGURA 6-18
Construcción de un contador Johnson

Un contador Johnson es un contador anular con extremo conmutado de k bits provisto de $2k$ compuertas decodificadoras para generar salidas correspondientes a $2k$ señales de temporización. Las compuertas decodificadoras no se indican en la figura 6-18, pero se especifican en la última columna de la tabla. Las ocho compuertas AND que se registran en la tabla, conectadas al circuito, completan la construcción del contador Johnson. Puesto que cada compuerta se habilita durante una sucesión dada de estados, las salidas de las compuertas generan ocho señales sucesivas de temporización.

La decodificación de un contador anular con extremo conmutado de k bits para obtener $2k$ señales de temporización sigue un patrón regular. El estado de puros ceros se decodifica tomando el complemento de las salidas de los dos flip-flops de los extremos. El estado de puros unos se decodifica tomando las salidas normales de los dos flip-flops de los extremos. Todos los demás estados se decodifican a partir de un patrón 1, 0 o 0, 1 adyacente en la sucesión. Por ejemplo, la sucesión 7 tiene un patrón 0, 1 adyacente en los flip-flops B y C. La salida decodificada se obtiene tomando el complemento de B y la salida normal de C, es decir, $B'C$.

Una desventaja del circuito de la figura 6-18a) es que si llega a estar en un estado no utilizado, persistirá en pasar de un estado no válido a otro y nunca llegará a un estado válido. Esto se corrige modificando el circuito a modo de evitar esta condición indeseable. Un procedimiento de corrección consiste en desconectar la salida del flip-flop B que va a la entrada D del flip-flop C, y habilitar la entrada del flip-flop C con la función

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

donde D_C es la ecuación para la entrada D del flip-flop C.

Podemos construir contadores Johnson para cualquier número de sucesiones de temporización. El número de flip-flops requeridos es la mitad del número de señales de temporización. El número de compuertas decodificadoras es igual al número de señales de temporización, y sólo se necesitan compuertas de dos entradas.

6-6 HDL PARA REGISTROS Y CONTADORES

Los registros y contadores se describen en HDL en el nivel de comportamiento o en el estructural. En el primero, el registro se especifica describiendo las diversas operaciones que realiza, de forma similar a una tabla de función. Una descripción en el nivel estructural muestra el circuito en términos de una colección de componentes como compuertas, flip-flops y multiplexores. Se crean ejemplares de los diversos componentes para formar una descripción jérárquica del diseño, similar a una representación de un diagrama lógico. Utilizaremos tres circuitos de este capítulo para ilustrar los dos tipos de descripciones.

Registro de desplazamiento

El registro de desplazamiento universal que se presentó en la sección 6-2 es un registro de desplazamiento bidireccional con carga paralela. En la tabla 6-6 se especifican las cuatro operaciones con reloj que se efectúan con el registro. El registro también se puede despejar asincrónicamente. La descripción del comportamiento de un registro de desplazamiento universal de cuatro bits se ilustra en el ejemplo HDL 6-1. Hay dos entradas de selección, dos entradas en serie, una entrada en paralelo de cuatro bits y una salida en paralelo de cuatro bits. El bloque **always** describe las cinco operaciones que es posible efectuar con el registro. La en-

Ejemplo HDL 6-1

```
//Descripción del comportamiento de un
//registro de desplazamiento universal
// figura 6-7 y tabla 6-3
module shftreg (s1,s0,Pin,lfin,rtin,A,CLK,Clr);
   input s1,s0;                      //Seleccionar entradas
   input lfin, rtin;                 //Entradas en serie
   input CLK,Clr;                    //Reloj y Clear
   input [3:0] Pin;                  //Entrada paralela
   output [3:0] A;                   //Salida del registro
   reg [3:0] A;
   always @ (posedge CLK or negedge Clr)
     if (~Clr) A = 4'b0000;
      else
        case ({s1,s0})
         2'b00: A = A;               //Sin cambio
         2'b01: A = {rtin,A[3:1]};   //Desplazamiento a la derecha
         2'b10: A = {A[2:0],lfin};   //Desplazamiento a la izquierda
         2'b11: A = Pin;             //Entrada de carga paralela
        endcase
endmodule
```

trada Clr despeja (pone en ceros) el registro asincrónicamente con una señal negativa. Clr debe estar alta para que el registro responda al borde positivo del reloj. Las cuatro operaciones con reloj del registro se determinan a partir de los valores de las dos entradas de selección en el enunciado **case** (s1 y s0 se concatenan en un vector de dos bits después de la palabra clave **case**). El desplazamiento se especifica con la concatenación de la entrada en serie y tres flip-flops. Por ejemplo, el enunciado

```
A = {rtin, A[3:1]}
```

especifica una concatenación de la entrada en serie para desplazamiento a la derecha (rtin) y los flip-flops *A*3, *A*2 y *A*1 para formar un número de cuatro bits, que se transfiere a *A*[3:0]. Esto produce una operación de desplazamiento a la derecha. Considere que sólo se ha descrito la función del circuito, independientemente del hardware específico.

Podemos describir la estructura del registro remitiéndonos al diagrama lógico de la figura 6-7. Ese diagrama indica que el registro se construye con cuatro multiplexores y cuatro flip-flops *D*. La descripción estructural del registro se muestra en el ejemplo HDL 6-2. El ejemplo tiene dos módulos. El primero declara las entradas y las salidas, y luego crea ejemplares para las etapas del registro. Los cuatro ejemplares creados especifican las interconexiones entre las cuatro etapas y proporcionan los pormenores de construcción del registro especificados en el diagrama lógico. El segundo módulo tiene dos bloques **always.** El primero describe al multiplexor, y el segundo, al flip-flop. Juntos definen una etapa del registro.

Ejemplo HDL 6-2

```
//Descripción estructural de registro
//universal de desplazamiento (figura 6-7)
module SHFTREG (I,select,lfin,rtin,A,CLK,Clr);
   input [3:0] I;            //Entrada paralela
   input [1:0] select;       //Seleccionar modo
   input lfin,rtin,CLK,Clr;  //Entradas en serie, reloj, clear
   output [3:0] A;           //Salida paralela
 //Crear ejemplares para las cuatro etapas
   stage ST0 (A[0],A[1],lfin,I[0],A[0],select,CLK,Clr);
   stage ST1 (A[1],A[2],A[0],I[1],A[1],select,CLK,Clr);
   stage ST2 (A[2],A[3],A[1],I[2],A[2],select,CLK,Clr);
   stage ST3 (A[3],rtin,A[2],I[3],A[3],select,CLK,Clr);
endmodule

//Una etapa del registro de desplazamiento
module stage(i0,i1,i2,i3,Q,select,CLK,Clr);
   input i0,i1,i2,i3,CLK,Clr;
   input [1:0] select;
   output Q;
   reg Q;
   reg D;
//Multiplexor 4x1
   always @ (i0 or i1 or i2 or i3 or select)
         case (select)
            2'b00: D = i0;
            2'b01: D = i1;
            2'b10: D = i2;
            2'b11: D = i3;
         endcase
//Flip-flop D
   always @ (posedge CLK or negedge Clr)
         if (~Clr) Q = 1'b0;
         else Q = D;
endmodule
```

Contador sincrónico

El ejemplo HDL 6-3 describe el contador sincrónico con carga paralela de la figura 6-14. *Count*, *Load*, *CLK* y *Clr* (conteo, carga, reloj y despeje) son entradas que determinan el funcionamiento del registro según la función especificada en la tabla 6-6. El contador tiene cuatro entradas de datos, cuatro salidas de datos y una salida de acarreo. Esta última, *CO* se genera con un circuito combinacional y se especifica con un enunciado **assign.** $CO = 1$ cuando la cuenta llega a 15 y el contador está en el estado de conteo. Así pues, $CO = 1$ si

Ejemplo HDL 6-3

```
//Contador binario con carga paralela
//Véase la figura 6-14 y la tabla 6-6
module counter (Count,Load,IN,CLK,Clr,A,CO);
  input Count,Load,CLK,Clr;
  input [3:0] IN;                 //Entrada de datos
  output CO;                      //Acarreo de salida
  output [3:0] A;                 //Salida de datos
  reg [3:0] A;
  assign CO = Count & ~Load & (A == 4'b1111);
  always @ (posedge CLK or negedge Clr)
    if (~Clr) A = 4'b0000;
    else if (Load)  A = IN;
    else if (Count) A = A + 1'b1;
    else A = A;
endmodule
```

Count = 1, *Load* = 0 y *A* = 1111; en los demás casos, *CO* = 0. El bloque **always** especifica la operación a efectuar en el registro, dependiendo de los valores de *Clr, Load* y *Count*. Una señal negativa en *Clr* pone *A* en 0. Si *Clr* = 1, una de tres operaciones se ejecutan durante un borde positivo del reloj. Los enunciados **if, else if** y **else** toman las decisiones como sigue:

```
        if Clr = 0                                  Poner A en ceros
  else if (Clr = 1 and) Load = 1                    Cargar entradas en A
  else if (Clr = 1 and Load = 0 and) Count = 1      Incrementar A
     else (Clr = 1 and Load = 0 and Count = 0)      Sin cambio en A
```

La jerarquía implícita en los enunciados **if-else** se ajusta a la precedencia especificada en la tabla 6-6.

Contador de rizo

En el ejemplo HDL 6-4 se muestra la descripción estructural de un contador de rizo. El primer módulo crea ejemplares de cuatro flip-flops complementadores que se definen en el segundo módulo como `CF(Q,CLK,Reset)`. El reloj (entrada C) del primer flip-flop se conecta a la entrada externa *Count* (*Count* sustituye a *CLK* en *F*0). La entrada de reloj del segundo flip-flop está conectada a la salida del primero (*A*0 sustituye a *CLK* en *F*1). De forma similar, el reloj de cada uno de los otros flip-flops está conectado a la salida del flip-flop anterior. Así, los flip-flops se encadenan para formar un contador de rizo, como se advierte en la figura 6-8b).

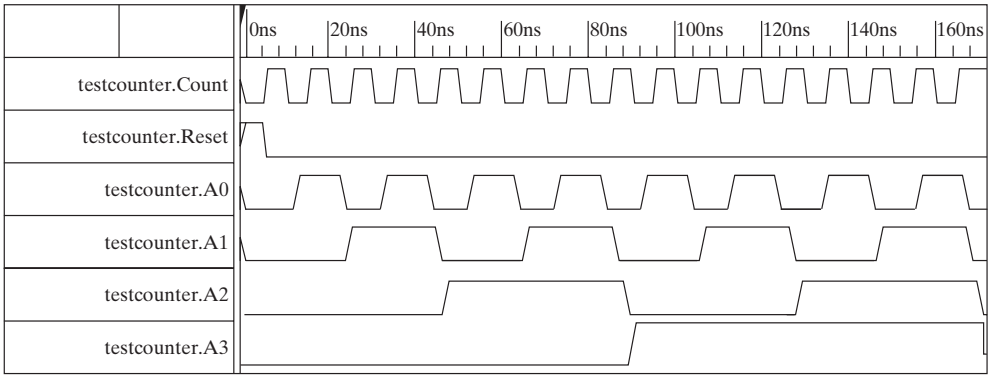
El segundo módulo describe un flip-flop complementador con retardo. El circuito se construye conectando la salida de complemento a la entrada *D*. Se incluye una entrada de restable-

Ejemplo HDL 6-4

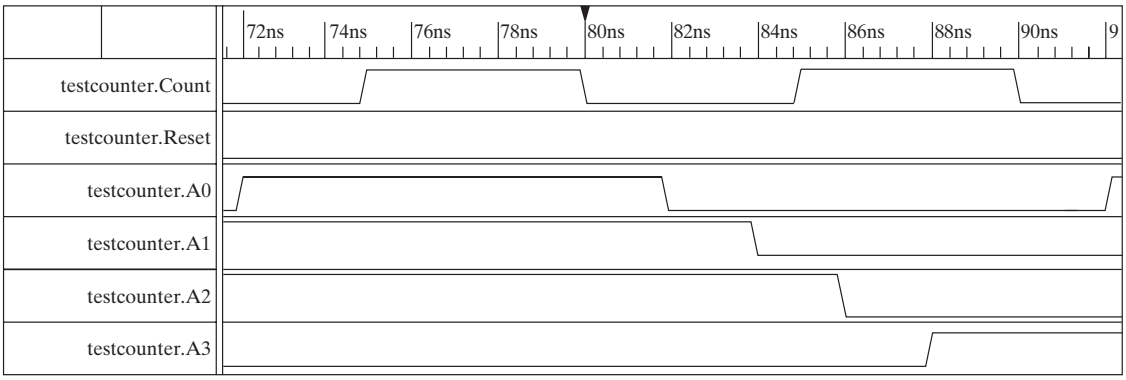
```
//Contador de rizo (Véase la figura 6-8b))
module ripplecounter (A0,A1,A2,A3,Count,Reset);
   output A0,A1,A2,A3;
   input Count,Reset;
//Crear ejemplar de flip-flop complementador
   CF F0 (A0,Count,Reset);
   CF F1 (A1,A0,Reset);
   CF F2 (A2,A1,Reset);
   CF F3 (A3,A2,Reset);
endmodule

//Flip-flop complementador con retardo
//Entrada al flip-flop D = Q'
module CF (Q,CLK,Reset);
   output Q;
   input CLK,Reset;
   reg Q;
   always @ (negedge CLK or posedge Reset)
      if (Reset) Q = 1'b0;
      else  Q = #2 (~Q);     // Retardo de 2 unidades de tiempo
endmodule

//Estímulo para probar el contador de rizo
module testcounter;
   reg Count;
   reg Reset;
   wire A0,A1,A2,A3;
//Crear ejemplar de contador de rizo
   ripplecounter RC (A0,A1,A2,A3,Count,Reset);
always
    #5 Count = ~Count;
initial
 begin
      Count = 1'b0;
      Reset = 1'b1;
   #4 Reset = 1'b0;
   #165 $finish;
 end
endmodule
```

cimiento en el flip-flop para poder iniciar el contador. Los simuladores HDL no proporcionan valores de salida si no se les asigna un valor inicial. Se asigna al flip-flop un retardo de dos unidades de tiempo, desde el momento en que se aplica el reloj hasta el momento en que el flip-flop se complementa. Esto se especifica con el enunciado $Q = \#2\ (\sim Q)$.

El tercer módulo del ejemplo 6-4 genera un estímulo para simular y probar el contador de rizo. El enunciado **always** genera un reloj con un ciclo de 10 unidades de tiempo. Los flip-flops se disparan con el borde negativo del reloj, que se da en $t = 10, 20, 30$, etcétera. Las formas de onda que se obtienen de esta simulación se distinguen en la figura 6-19. *Count* se vuelve negativa cada 10 ns. *A*0 se complementa con cada borde negativo de *Count* pero con un retardo de 2 ns. Cada flip-flop se complementa cuando el flip-flop precedente cambia de 1 a 0. Después de $t = 80$ ns, los cuatro flip-flops se complementan porque el contador cambia de 0111 a 1000. Cada salida se retarda 2 ns y, a causa de ello, *A*3 cambia de 0 a 1 en $t = 88$ ns, y de 1 a 0 a los 168 ns.



a) De 0 a 170 ns



b) De 70 a 92 ns

FIGURA 6-19
Salida de simulación del ejemplo HDL 6-4

PROBLEMAS

6-1 Incluya una compuerta NAND de dos entradas con el registro de la figura 6-1 y conecte la salida de la compuerta a las entradas *C* de todos los flip-flops. Una entrada de la compuerta NAND recibe los pulsos de reloj del generador de reloj, y la otra entrada de la compuerta se encarga de controlar la carga en paralelo. Explique el funcionamiento del registro modificado.

6-2 Incluya una entrada de despeje sincrónica para el registro de la figura 6-2. El registro modificado tendrá una capacidad de carga en paralelo y una capacidad de despeje sincrónico. El registro se despeja (pone en ceros) sincrónicamente cuando el reloj tiene una transición positiva y la entrada de despeje es 1.

6-3 ¿Qué diferencia hay entre transferencia en serie y en paralelo? Explique cómo convertir datos en serie a paralelo y datos en paralelo a datos en serie. ¿Qué tipo de registro se necesita?

6-4 El contenido de un registro de cuatro bits es inicialmente 1101. El registro se desplaza seis veces a la derecha, siendo la entrada en serie 101101. ¿Qué contiene el registro después de cada desplazamiento?

6-5 El registro universal de desplazamiento de cuatro bits mostrado en la figura 6-7 se encierra en un paquete de CI.

a) Dibuje un diagrama de bloques del circuito integrado que señale todas las entradas y salidas. Incluya dos entradas para la alimentación eléctrica.

b) Dibuje un diagrama de bloques empleando dos CI para producir un registro de desplazamiento universal de ocho bits.

6-6 Diseñe un registro de desplazamiento de cuatro bits con carga paralela empleando flip-flops *D*. Hay dos entradas de control: desplazar y cargar. Cuando desplazar $= 1$, el contenido del registro se desplaza una posición. Se transfieren nuevos datos al registro cuando cargar $= 1$ y desplazar $= 0$. Si ambas entradas de control son 0, el contenido del registro no cambia.

6-7 Dibuje el diagrama lógico de un registro de cuatro bits con cuatro flip-flops *D* y cuatro multiplexores 4×1, con entradas de selección de modo s_1 y s_0. El registro opera según la siguiente tabla de función:

s_1	s_0	**Operación del registro**
0	0	Sin cambio
0	1	Complementar las cuatro salidas
1	0	Poner el registro en ceros (sincrónico con el reloj)
1	1	Cargar datos en paralelo

6-8 El sumador en serie de la figura 6-6 usa dos registros de cuatro bits. El registro *A* contiene el número binario 0101, y el registro *B*, 0111. El flip-flop de acarreo se restablece inicialmente en 0. Numere los valores binarios que están en el registro *A* y en el flip-flop de acarreo después de cada desplazamiento.

6-9 En la sección 6-2 se describieron dos formas de implementar un sumador en serie ($A + B$). Es necesario modificar los circuitos para convertirlos en restadores en serie ($A - B$).

a) Utilizando el circuito de la figura 6-5, indique los cambios necesarios para obtener $A +$ complemento a dos de *B*.

b) Utilizando el circuito de la figura 6-6, indique los cambios requeridos modificando la tabla 6-2, de un circuito sumador a uno restador. (Véase el problema 4-12.)

6-10 Diseñe un complementador a dos en serie con un registro de desplazamiento y un flip-flop. El número binario se desplaza hacia afuera por un lado y su complemento a dos se desplaza hacia adentro por el otro lado del registro de desplazamiento.

6-11 Un contador binario de rizo usa flip-flops que se disparan con el borde positivo del reloj. ¿Cuál será el conteo si a) las salidas normales de los flip-flops se conectan al reloj y b) las salidas de complemento de los flip-flops se conectan al reloj?

6-12 Dibuje el diagrama lógico de un sumador binario de rizo de cuatro bits de cuenta regresiva utilizando a) flip-flops que se disparan con el borde positivo del reloj y b) flip-flops que se disparan con el borde negativo del reloj.

6-13 Demuestre que es posible construir un contador BCD de rizo empleando un contador binario de rizo de cuatro bits con despeje asincrónico y una compuerta NAND que detecta la ocurrencia de la cuenta 1010.

6-14 ¿Cuántos flip-flops se complementarán en un contador binario de rizo de 10 bits para llegar a la siguiente cuenta después de la cuenta?: a) 1001100111; b) 0011111111; c) 1111111111.

6-15 Un flip-flop tiene un retardo de 5 ns desde el momento en que se da el borde de reloj hasta el momento en que la salida se complementa. ¿Qué retardo máximo tendría un contador binario de rizo de 10 bits que usara esos flip-flops? ¿Con qué frecuencia máxima puede operar el contador de manera confiable?

6-16 El contador BCD de rizo que se representa en la figura 6-10 tiene cuatro flip-flops y 16 estados, de los cuales sólo se usan 10. Analice el circuito y determine el siguiente estado para cada uno de los otros seis estados no utilizados. ¿Qué sucederá si una señal de ruido hace que el circuito pase a uno de los estados no utilizados?

6-17 Diseñe un contador binario sincrónico de cuatro bits con flip-flops *D*.

6-18 ¿Qué operación se efectúa en el contador ascendente-descendente de la figura 6-13 cuando ambas entradas, arriba y abajo, están habilitadas? Modifique el circuito de modo que cuando ambas entradas sean 1, el contador no cambie de estado, sino que permanezca en la misma cuenta.

6-19 Las ecuaciones de entrada de flip-flops para un contador BCD construido con flip-flops *T* se incluyen en la sección 6-4. Obtenga las ecuaciones de entrada para un contador BCD construido con a) flip-flops *JK* y b) flip-flops *D*. Compare los tres diseños para determinar cuál es el más eficiente.

6-20 Encierre el contador binario con carga paralela de la figura 6-14 en un diagrama de bloques que muestre todas las entradas y salidas.

a) Muestre las conexiones de cuatro de esos bloques para formar un contador de 16 bits con carga paralela.

b) Construya un contador binario que cuente desde 0 hasta 64 binario.

6-21 El contador de la figura 6-14 tiene dos entradas de control —Cargar (*L*) y Conteo c)— y una entrada de datos (I_i).

a) Deduzca las ecuaciones de entrada de flip-flops para *J* y *K* de la primera etapa, en términos de *L*, *C* e *I*.

b) En la figura P6-21 se observa el diagrama lógico de la primera etapa de un circuito integrado equivalente (74161). Compruebe que este circuito sea equivalente al de a).

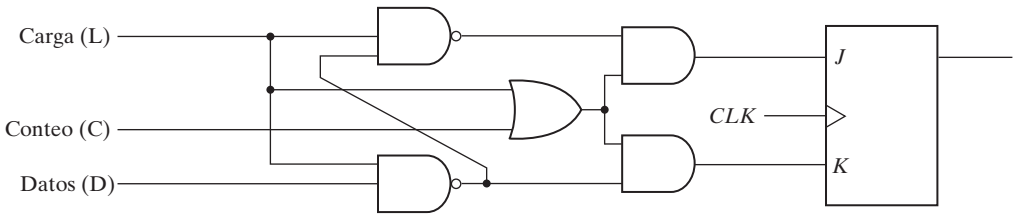


FIGURA P6-21

6-22 Utilizando el circuito de la figura 6-14, dé tres alternativas para un contador mod-12:

a) Utilizando una compuerta AND y la entrada de carga.

b) Utilizando el acarreo de salida.

c) Utilizando una compuerta NAND y la entrada de despeje asincrónico.

6-23 Diseñe un circuito de temporización que genere una señal de salida que se mantenga encendida durante exactamente ocho ciclos de reloj. Una señal de inicio hace que la salida pase al estado 1; después de ocho ciclos de reloj, la señal vuelve al estado 0.

6-24 Diseñe con flip-flops *T* un contador que pase por la siguiente sucesión binaria repetida: 0, 1, 3, 7, 6, 4. Demuestre que si los estados binarios 010 y 101 se consideran condiciones de indiferencia, el contador podría no funcionar correctamente. Encuentre una forma de corregir el diseño.

6-25 Es necesario generar seis señales repetidas de temporización T_0 a T_5 similares a las que se indican en la figura 6-17c). Diseñe el circuito utilizando:

a) Únicamente flip-flops.

b) Un contador y un decodificador.

6-26 Un sistema digital tiene un generador de reloj que produce pulsos con una frecuencia de 80 MHz. Diseñe un circuito que genere un reloj con un tiempo de ciclo de 50 ns.

6-27 Diseñe un contador que siga esta sucesión binaria repetida: 0, 1, 2, 3, 4, 5, 6. Use flip-flops *JK*.

6-28 Diseñe un contador que siga esta sucesión binaria repetida: 0, 1, 2, 4, 6. Use flip-flops *D*.

6-29 Numere los ocho estados no utilizados del contador de anillo con extremo conmutado de la figura 6-18a).

Determine el siguiente estado para cada uno de estos estados y demuestre que, si el contador llega a estar en un estado no válido, no volverá a un estado válido. Modifique el circuito como se recomienda en el texto y demuestre que el contador produce la misma sucesión de estados y que el circuito llega a un estado válido desde cualquiera de los estados no utilizados.

6-30 Demuestre que un contador Johnson con n flip-flops produce una sucesión de $2n$ estados. Numere los 10 estados producidos con cinco flip-flops y los términos booleanos de cada una de las diez salidas de compuerta AND.

6-31 Escriba las descripciones HDL de comportamiento y estructural del registro de cuatro bits de la figura 6-1.

6-32 a) Escriba la descripción HDL del comportamiento de un registro de cuatro bits con carga paralela y despeje asincrónico.

b) Escriba la descripción HDL estructural del registro de cuatro bits con carga paralela de la figura 6-2. Utilice un multiplexor 2×1 para las entradas de flip-flops. Incluya una entrada de despeje asincrónico.

c) Verifique ambas descripciones con un conjunto de pruebas.

6-33 Se usa el programa de estímulo siguiente para simular el contador binario con carga paralela descrito en el ejemplo HDL 6-3. Examine el programa y prediga qué salida tendrá el contador y el acarreo entre $t = 0$ y $t = 155$ ns.

```
//Estímulo para probar el contador
//del ejemplo 6-3
     module testcounter;
       reg Count, Load, CLK, Clr;
       reg [3:0] IN;
       wire C0;
       wire [3:0] A;
       counter cnt (Count, Load, IN, CLK, Clr, A, CO);
       always
           #5 CLK = ~CLK;
         initial
          begin
           Clr = 0;
           CLK = 1;
           Load = 0; Count = 1;
             #5  Clr = 1;
             #50 Load = 1; IN = 4'b1100;
             #10 Load = 0;
             #70 Count = 0;
             #20 $finish;
          end
     endmodule
```

6-34 Escriba la descripción HDL del comportamiento de un registro de desplazamiento de cuatro bits (figura 6-3).

6-35 Escriba las descripciones HDL de comportamiento y estructural del contador arriba-abajo de cuatro bits cuyo diagrama lógico aparece en la figura 6-13.

6-36 Escriba la descripción HDL del comportamiento de un contador arriba-abajo de cuatro bits con carga paralela utilizando las siguientes entradas de control:

a) El contador tiene tres entradas de control para las tres operaciones: Arriba, Abajo y Cargar. El orden de precedencia es: Cargar, Arriba y Abajo.

b) El contador tiene dos entradas de selección para especificar cuatro operaciones: Arriba, Abajo, Cargar y sin cambio.

6-37 Escriba la descripción HDL de un contador anular de ocho bits similar al de la figura 6-17a).

6-38 Escriba la descripción HDL de un contador anular con extremo conmutado de cuatro bits (figura 6-18a).

6-39 Escriba las descripciones HDL de comportamiento y estructural del contador de la figura 6-16.

REFERENCIAS

1. MANO, M. M. y C. R. KIME. 2000. *Logic and Computer Design Fundamentals*. 2a. ed. Upper Saddle River, NJ: Prentice-Hall.
2. NELSON V. P., H. T. NAGLE, J. D. IRWIN y B. D. CARROLL. 1995. *Digital Logic Circuit Analysis and Design*. Upper Saddle River, NJ: Prentice-Hall.
3. HAYES, J. P. 1993. *Introduction to Digital Logic Design*. Reading, MA: Addison-Wesley.
4. WAKERLY, J. F. 2000. *Digital Design: Principles and Practices*. 3a. ed. Upper Saddle River, NJ: Prentice-Hall.
5. DIETMEYER, D. L. 1988. *Logic Design of Digital Systems*. 3a. ed. Boston: Allyn Bacon.
6. GAJSKI, D. D. 1997. *Principles of Digital Design*. Upper Saddle River, NJ: Prentice-Hall.
7. ROTH, C. H. 1992. *Fundamentals of Logic Design*, 4a. ed. St. Paul: West.
8. KATZ, R. H. 1994. *Contemporary Logic Design*. Upper Saddle River, NJ: Prentice-Hall.
9. CILETTI, M. D. 1999. *Modeling, Synthesis and Rapid Prototyping with Verilog HDL*. Upper Saddle River, NJ: Prentice-Hall.
10. BHASKER, J. 1997. *A Verilog HDL Primer*. Allentown, PA: Star Galaxy Press.
11. THOMAS, D. E. y P. R. MOORBY. 1998. *The Verilog Hardware Description Language.* 4a. ed. Boston: Kluwer Academic Publishers.
12. BHASKER, J. 1998. *Verilog HDL Synthesis*. Allentown, PA: Star Galaxy Press.
13. PALNITKAR, S. 1996. *Verilog HDL: A Guide to Digital Design and Synthesis*. SunSoft Press (Un título Prentice-Hall).