

DÉCIMA EDICIÓN

Sistemas digitales

Principios y aplicaciones

Ronald J. Tocci

Monroe Community College

Neal S. Widmer

Purdue University

Gregory L. Moss

Purdue University

Traducción

Alfonso Vidal Romero Elizondo

Ingeniero en Electrónica y Comunicación
Instituto Tecnológico y de Estudios Superiores de
Monterrey - Campus Monterrey

Revisión técnica

Reynaldo Félix Acuña

Profesor investigador
Departamento de Ingeniería Eléctrica
y Electrónica
Instituto Tecnológico y de Estudios
Superiores de Monterrey
Campus Estado de México

Marcos de Alba

Profesor investigador
Departamento de Ingeniería Eléctrica
y Electrónica
Instituto Tecnológico y de Estudios
Superiores de Monterrey
Campus Estado de México



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

FLIP-FLOPS Y DISPOSITIVOS RELACIONADOS

CONTENIDO

- 5-1 Latch de compuerta NAND
- 5-2 Latch de compuerta NOR
- 5-3 Ejemplo práctico de diagnóstico de fallas
- 5-4 Pulsos digitales
- 5-5 Señales de reloj y flip-flops sincronizados por reloj
- 5-6 Flip-flop sincronizado por reloj en S-R
- 5-7 Flip-flop sincronizado por reloj en J-K
- 5-8 Flip-flop sincronizado por reloj en D
- 5-9 Latch *D* (latch transparente)
- 5-10 Entradas asíncronas
- 5-11 Símbolos IEEE/ANSI
- 5-12 Consideraciones de sincronización de los flip-flops
- 5-13 Problema potencial de sincronización en los circuitos con FF
- 5-14 Aplicaciones de los flip-flops
- 5-15 Sincronización de los flip-flops
- 5-16 Detección de una secuencia de entrada
- 5-17 Almacenamiento y transferencia de datos
- 5-18 Transferencia de datos en serie: registros de desplazamiento
- 5-19 División y conteo de frecuencia
- 5-20 Aplicación de microcomputadora
- 5-21 Dispositivos disparadores de Schmitt
- 5-22 One-Shot (Multivibrador monoestable)
- 5-23 Circuitos generadores de reloj
- 5-24 Diagnóstico de fallas en circuitos con flip-flops
- 5-25 Circuitos secuenciales mediante el uso de HDL
- 5-26 Dispositivos disparados por flanco
- 5-27 Circuitos de HDL con varios componentes

■ OBJETIVOS

Al terminar este capítulo, usted podrá:

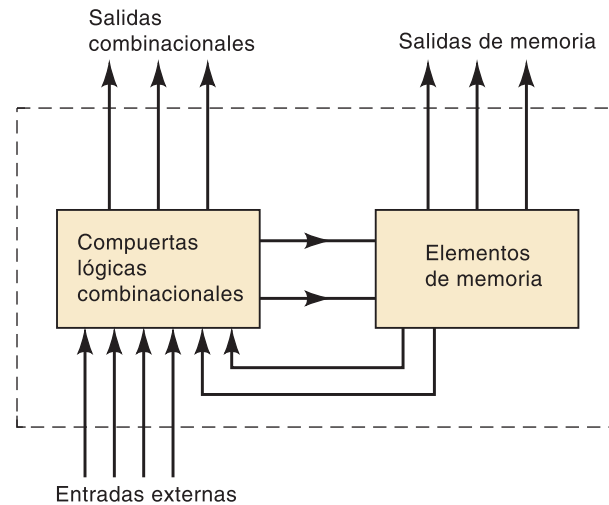
- Construir y analizar la operación de un flip-flop tipo match, compuesto a partir de compuertas NAND o NOR.
- Describir la diferencia entre los sistemas síncronos y asíncronos.
- Comprender la operación de los flip-flops disparados por flanco.
- Analizar y aplicar los diversos parámetros de sincronización de los flip-flops especificados por los fabricantes.
- Comprender las principales diferencias entre las transferencias de datos en paralelo y en serie.
- Dibujar las formas de onda de sincronización de salida de varios tipos de flip-flops en respuesta a un conjunto de señales de entrada.
- Reconocer los diversos símbolos IEEE/ANSI para los flip-flops.
- Utilizar los diagramas de transición de estado para describir la operación de un contador.
- Utilizar flip-flops en circuitos de sincronización.
- Conectar registros de desplazamiento como circuitos de transferencia de datos.
- Emplear los flip-flops como circuitos de división y conteo de frecuencia.
- Comprender las características comunes de los disparadores de Schmitt.
- Aplicar dos tipos distintos de monoestables (one-shots) en el diseño de circuitos.
- Diseñar un oscilador de carrera libre mediante el uso de un temporizador 555.
- Reconocer y predecir los efectos del desfase de reloj en los circuitos síncronos.
- Diagnosticar fallas en diversos tipos de circuitos con flip-flops.
- Escribir código en HDL para los latches.
- Utilizar primitivas, componentes y bibliotecas lógicas en el código de HDL.
- Construir circuitos a nivel estructural a partir de ciertos componentes.

■ INTRODUCCIÓN

Los circuitos lógicos que hemos considerado hasta ahora han sido combinacionales, cuyos niveles de salida en cualquier instante son dependientes de los niveles presentes en las entradas en ese instante. Cualquier condición anterior en relación con los niveles de entrada no tiene efecto alguno sobre las salidas actuales, ya que los circuitos lógicos combinacionales no tienen memoria. La mayoría de los sistemas digitales consisten tanto de circuitos combinacionales como de elementos de memoria.

La figura 5-1 muestra un diagrama de bloques de un sistema digital general que combina las compuertas lógicas combinacionales con los dispositivos de memoria. La porción combinacional acepta las señales lógicas de las entradas externas y de las salidas de los elementos de memoria. El circuito combinacional opera sobre estas entradas para producir varias salidas, algunas de las cuales se utilizan para determinar los valores binarios que se van a almacenar en los elementos de memoria.

FIGURA 5-1 Diagrama de un sistema digital general.



A su vez, las salidas de algunos de los elementos de memoria van a las entradas de las compuertas lógicas en los circuitos combinacionales. Este proceso indica que las salidas externas de un sistema digital son funciones tanto de sus entradas externas como de la información almacenada en sus elementos de memoria.

El elemento de memoria más importante es el **flip-flop**, el cual está formado por un conjunto de compuertas lógicas. Una compuerta lógica por sí sola no tiene capacidad de almacenamiento, sin embargo, varias de ellas pueden interconectarse de cierta forma logrando que permitan almacenar información. Para producir estos flip-flop (que se abrevian como FF) se utilizan varios arreglos distintos de compuertas.

La figura 5-2(a) es el tipo general de símbolo que se utiliza para un flip-flop. Ahí se muestran dos salidas identificadas como Q y $\bar{Q}$, que son el inverso una de otra. $Q/\bar{Q}$ son las designaciones más comunes que se utilizan para las salidas de un FF. De vez en cuando utilizaremos otras designaciones tales como $X/\bar{X}$ y $A/\bar{A}$, por conveniencia, para identificar los distintos FFs en un circuito lógico.

La salida Q es la salida *normal* del FF, y $\bar{Q}$ es la salida *invertida* del FF. Cada vez que nos referimos al estado de un FF, estamos haciendo referencia al estado de su salida normal (Q); se sobreentiende que su salida invertida ($\bar{Q}$) está en el estado opuesto. Por ejemplo, si decimos que un FF se encuentra en el estado ALTO (1), significa que $Q = 1$; si decimos que un FF se encuentra en el estado BAJO (0), significa que $Q = 0$. Desde luego que el estado de $\bar{Q}$ siempre será el inverso de Q .

La figura 5-2(b) sintetiza los dos estados de operación posibles para un FF. Observe que al estado ALTO o 1 ($Q = 1/\bar{Q} = 0$) también se le conoce como **SET** (establecer). Cada vez que las entradas a un FF provocan que cambie al estado $Q = 1$, le llamamos *establecer* el FF; es decir, el FF ha sido establecido. De manera similar, al estado BAJO o 0 ($Q = 0/\bar{Q} = 1$) también se le conoce como **CLEAR** (borrar) o **RESET** (restablecer). Cada vez que las entradas de un FF hacen que cambie al estado $Q = 0$, le llamamos *borrar* o *restablecer* el FF; es decir, el FF ha sido

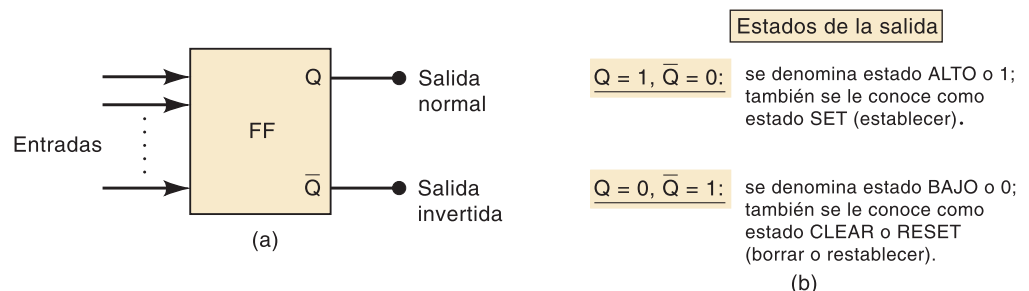


FIGURA 5-2 Símbolo general de un flip-flop y la definición de sus dos posibles estados de salida.

borrado (restablecido). Como veremos, muchos FFs tienen una entrada **SET** y/o una entrada **CLEAR** o **RESET**, la cual se utiliza para excitar el FF y llevarlo a un estado de salida específico.

Como indica el símbolo en la figura 5-2(a), un FF puede tener una o más entradas, las cuales se utilizan para hacer que el FF alterne (“flip-flop”) entre sus posibles estados de salida. Más adelante veremos que la mayoría de las entradas de un FF necesitan activarse sólo en forma momentánea (pulsos) para provocar un cambio en el estado de salida del FF, y que la salida permanecerá en ese nuevo estado incluso hasta después de que se acabe el pulso de entrada. Ésta es la característica de *memoria* del FF.

Al flip-flop se le conoce también con otros nombres, incluyendo *latch* y *multivibrador biestable*. El término *latch* se utiliza para ciertos tipos de flip-flops que describiremos más adelante. El término *multivibrador biestable* es el nombre técnico más adecuado en español para un flip-flop, pero es demasiado largo como para utilizarlo regularmente.

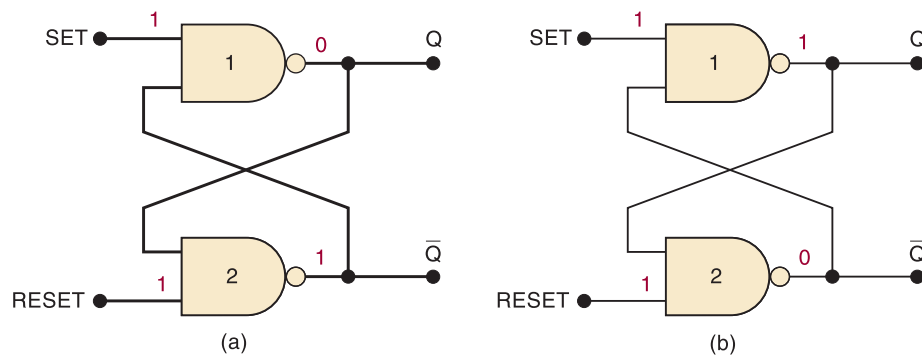
5-1 LATCH DE COMPUERTA NAND

El circuito de FF más básico puede crearse a partir de dos compuertas NAND o de dos compuertas NOR. En la figura 5-3(a) se muestra la versión con compuertas NAND, a la cual se le conoce como **latch de compuerta NAND** o simplemente **latch**. Las dos compuertas NAND están retroalimentadas en forma transversal, de manera que la salida de la compuerta NAND-1 está conectada a una de las entradas de la compuerta NAND-2 y viceversa. Las salidas de las compuertas, identificadas como Q y $\bar{Q}$ respectivamente, son las salidas del latch. Bajo condiciones normales, una salida siempre será el inverso de la otra. Existen dos entradas para el latch: la entrada **SET** es la que *establece* Q en el estado 1; la entrada **RESET** es la que *restablece* Q al estado 0.

Por lo general, las entradas **SET** y **RESET** permanecen en el estado ALTO, y una de ellas cambiará a BAJO mediante un pulso cada vez que se quiera cambiar el estado de las salidas del latch. Comenzaremos nuestro análisis mostrando que hay dos estados de salida con la misma probabilidad cuando $\text{SET} = \text{RESET} = 1$. La figura 5-3(a) muestra una de esas posibilidades, en donde tenemos que $Q = 0$ y $\bar{Q} = 1$. Cuando $Q = 0$, las entradas para la compuerta NAND-2 son 0 y 1, lo cual produce $\bar{Q} = 1$. El 1 de Q hace que la compuerta NAND-1 tenga un 1 en ambas entradas para producir una salida de 0 en $\bar{Q}$. En efecto, lo que tenemos es el nivel BAJO en la salida de la compuerta NAND-1 que produce un nivel ALTO en la salida de la compuerta NAND-2, lo cual a su vez mantiene la salida de la compuerta NAND-1 en BAJO.

La segunda posibilidad se muestra en la figura 5-3(b), en donde $Q = 1$ y $\bar{Q} = 0$. El nivel ALTO de la compuerta NAND-1 produce un nivel BAJO en la salida de la compuerta NAND-2, el cual a su vez mantiene la salida de la compuerta NAND-1 en ALTO. De esta forma, hay dos posibles estados de salida cuando $\text{SET} = \text{RESET} = 1$; como veremos más adelante, la que se dé en un momento dado, dependerá de lo que haya ocurrido antes en las entradas.

FIGURA 5-3 Un latch NAND tiene dos posibles estados en los que se puede mantener cuando $\text{SET} = \text{RESET} = 1$.



Cómo establecer el latch (FF)

Ahora vamos a investigar lo que ocurre cuando se aplica un pulso a la entrada SET para mantenerla unos momentos en nivel BAJO mientras que RESET se mantiene en ALTO. La figura 5-4(a) muestra lo que ocurre cuando $Q = 0$ antes de que se produzca el pulso. A medida que se aplica un pulso a SET para que cambie a nivel BAJO en el tiempo t_0 , Q cambiará a ALTO y este nivel ALTO obligará a $\bar{Q}$ a cambiar a BAJO, de manera que la compuerta NAND-1 ahora tiene dos entradas en BAJO. Por lo tanto, cuando SET regrese al estado 1 en t_1 , la salida de la compuerta NAND-1 *permanecerá* en ALTO, lo cual a su vez mantendrá la salida de la compuerta NAND-2 en BAJO.

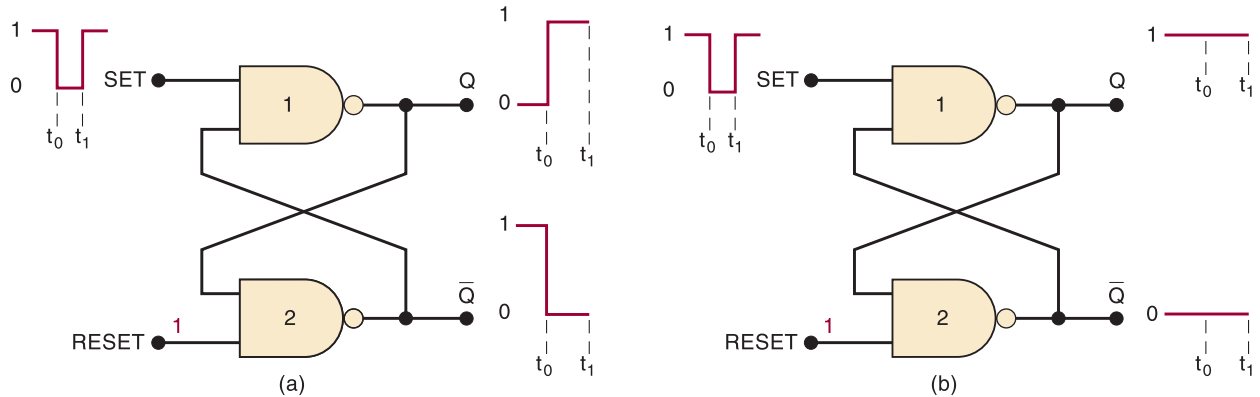


FIGURA 5-4 Aplicación de un pulso a la entrada SET para que cambie al estado 0 cuando (a) $Q = 0$ antes del pulso en SET; (b) $Q = 1$ antes del pulso en SET. Observe que, en ambos casos, Q termina en ALTO.

La figura 5-4(b) muestra lo que ocurre cuando $Q = 1$ y $\bar{Q} = 0$ antes de la aplicación del pulso en SET. Como $\bar{Q} = 0$ ya está manteniendo la salida de la compuerta NAND-1 en ALTO, el pulso BAJO en SET no cambiará nada. Así, cuando SET regrese a ALTO, las salidas del latch seguirán en el estado $Q = 1$, $\bar{Q} = 0$.

Para sintetizar la figura 5-4 diremos que un pulso BAJO en la entrada SET siempre hará que el latch termine en el estado $Q = 1$. A esta operación se le conoce como *establecer* el latch o FF.

Cómo restablecer el latch (FF)

Ahora consideremos lo que ocurre cuando se aplica un pulso a la entrada RESET para que cambie a nivel BAJO mientras que SET se mantiene en ALTO. La figura

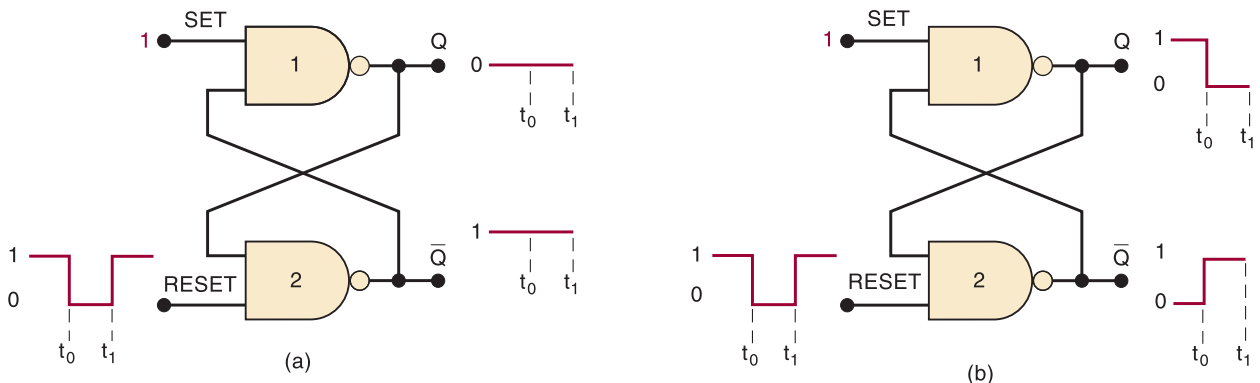


FIGURA 5-5 Aplicación de un pulso a la entrada RESET para que cambie al estado BAJO cuando (a) $Q = 0$ antes del pulso en RESET; (b) $Q = 1$ antes del pulso en RESET. En cada caso, Q termina en BAJO.

5-5(a) muestra lo que ocurre cuando $Q = 0$ y $\bar{Q} = 1$, antes de aplicar el pulso. Como $Q = 0$ ya está manteniendo la salida de la compuerta NAND-2 en ALTO, el pulso BAJO en RESET no tendrá ningún efecto. Cuando RESET regrese a ALTO, las salidas del latch seguirán en el estado $Q = 0$ y $\bar{Q} = 1$.

La figura 5-5(b) muestra la situación en donde $Q = 1$ antes de la ocurrencia del pulso en RESTABLECER. A medida que se aplica un pulso en dicha terminal, para que cambie a BAJO en t_0 , $\bar{Q}$ cambiará a ALTO y este nivel ALTO obligará a Q a cambiar a BAJO, de manera que la compuerta NAND-2 ahora tiene dos entradas en BAJO. Así, cuando RESET regrese a ALTO en t_1 , la salida de la compuerta NAND-2 *permanecerá* en ALTO, lo cual a su vez mantendrá la salida de la compuerta NAND-1 en BAJO.

Para sintetizar la figura 5-5 diremos que un pulso BAJO en la entrada RESET siempre hará que el latch termine en el estado $Q = 0$. A esta operación se le conoce como *borrar o restablecer* el latch.

Establecer y restablecer en forma simultánea

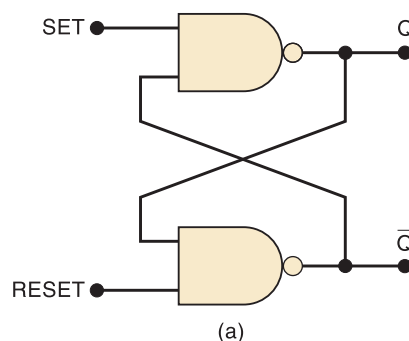
El último caso a considerar es cuando se aplica un pulso simultáneo a las entradas SET y RESET para que cambien a BAJO. Esto producirá niveles en ALTO en ambas salidas de las compuertas NAND, de manera que $Q = \bar{Q} = 1$. Es evidente que ésta es una condición indeseable, ya que se supone que las dos salidas deben ser el inverso una de la otra. Además, cuando las entradas SET y RESET regresen al nivel ALTO, el estado de salida resultante dependerá de cuál sea la entrada que regrese a ALTO primero. Las transiciones simultáneas al estado 1 producirán resultados impredecibles. Por estas razones, no es común utilizar la condición $SET = RESET = 0$ en el latch NAND.

Resumen del latch NAND

La operación antes descrita puede colocarse de manera conveniente en una tabla de funciones (figura 5-6) y se sintetiza de la siguiente manera:

1. $SET = RESET = 1$. Esta condición es el estado normal de reposo, y no tiene efecto sobre el estado de la salida. Las salidas Q y $\bar{Q}$ permanecerán en el estado en el que se hayan encontrado antes de esta condición de entrada.
2. $SET = 0, RESET = 1$. Esta condición siempre provocará que la salida cambie al estado $Q = 1$, en donde permanecerá aún después de que SET regrese a ALTO. A esto se le conoce como *establecer* el latch.
3. $SET = 1, RESET = 0$. Esta condición siempre producirá el estado $Q = 0$, en donde permanecerá la salida aún después de que RESET regrese a ALTO. A esto se le conoce como *borrar o restablecer* el latch.
4. $SET = RESET = 0$. Esta condición trata de establecer y borrar el latch al mismo tiempo, y produce $Q = \bar{Q} = 1$. Si las entradas se regresan a 1 en forma simultánea, el estado resultante será impredecible. Esta condición de entrada no debe utilizarse.

FIGURA 5-6 (a) latch NAND; (b) tabla de funciones.



Set	Reset	Salida
1	1	Sin cambio
0	1	$Q = 1$
1	0	$Q = 0$
0	0	Inválido*

* Produce $Q = \bar{Q} = 1$.

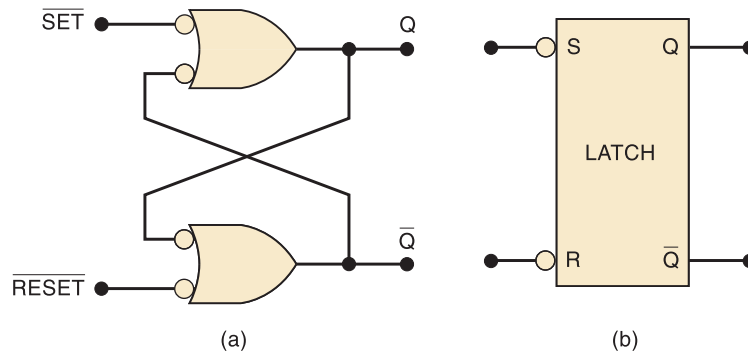
(b)

Representaciones alternativas

De la descripción de la operación del latch NAND debe quedar claro que las entradas SET y RESET son activas en BAJO. La entrada SET establecerá $Q = 1$ cuando SET cambie a BAJO; la entrada RESET borrará $Q = 0$ cuando RESET cambie a BAJO. Por esta razón, a menudo el latch NAND se dibuja utilizando la representación alternativa para cada compuerta NAND, como se muestra en la figura 5-7(a). Las burbujas en las entradas, así como el etiquetado de las señales como $\overline{SET}$ y $\overline{RESET}$, indican el estado activo en BAJO de estas entradas. (Tal vez quiera repasar las secciones 3-13 y 3-14 que hablan sobre este tema.)

La figura 5-7(b) muestra una representación en bloque simplificada, la cual utilizaremos algunas veces. Las etiquetas S y R representan las entradas SET y RESET, y las burbujas indican la naturaleza activa en BAJO de estas entradas. Cada vez que utilicemos este símbolo de bloque, representará a un latch NAND.

FIGURA 5-7
(a) representación equivalente del latch NAND; (b) símbolo de bloque simplificado.



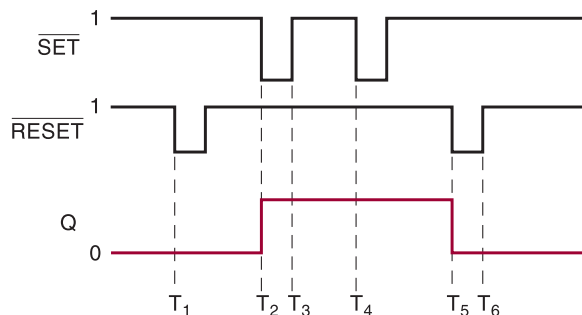
Terminología

A la acción de *restablecer* un FF o un latch también se le conoce como *borrar*; ambos términos se utilizan sin distinción en el campo digital. De hecho, una entrada RESET puede llamarse también entrada CLEAR, y un latch SET-RESET puede llamarse latch SET-CLEAR.

EJEMPLO 5-1

Las formas de onda de la figura 5-8 se aplican a las entradas del latch de la figura 5-7. Suponga que al inicio $Q = 0$; determine la forma de onda de Q .

FIGURA 5-8 Ejemplo 5-1.



Solución

Al principio $\overline{SET} = \overline{RESET} = 1$, por lo que Q permanecerá en el estado 0. El pulso BAJO que ocurre en la entrada RESET en el tiempo T_1 no tendrá efecto, ya que Q se encuentra de antemano en el estado borrado (0).

La única forma en que Q puede pasar al estado 1 es mediante un pulso BAJO en la entrada $\overline{\text{SET}}$. Esto ocurre en el tiempo T_2 , cuando $\overline{\text{SET}}$ primero cambia a BAJO. Cuando $\overline{\text{SET}}$ regresa al nivel ALTO en T_3 , Q permanecerá en su nuevo estado ALTO.

En el tiempo T_4 , cuando $\overline{\text{SET}}$ cambia a BAJO de nuevo, no habrá efecto sobre Q ya que se encuentra de antemano en el estado 1.

La única forma de regresar a Q de nuevo al estado 0 es mediante un pulso BAJO en la entrada $\overline{\text{RESET}}$. Esto ocurre en el tiempo T_5 . Cuando $\overline{\text{RESET}}$ regresa a 1 en el tiempo T_6 , Q permanece en el estado BAJO.

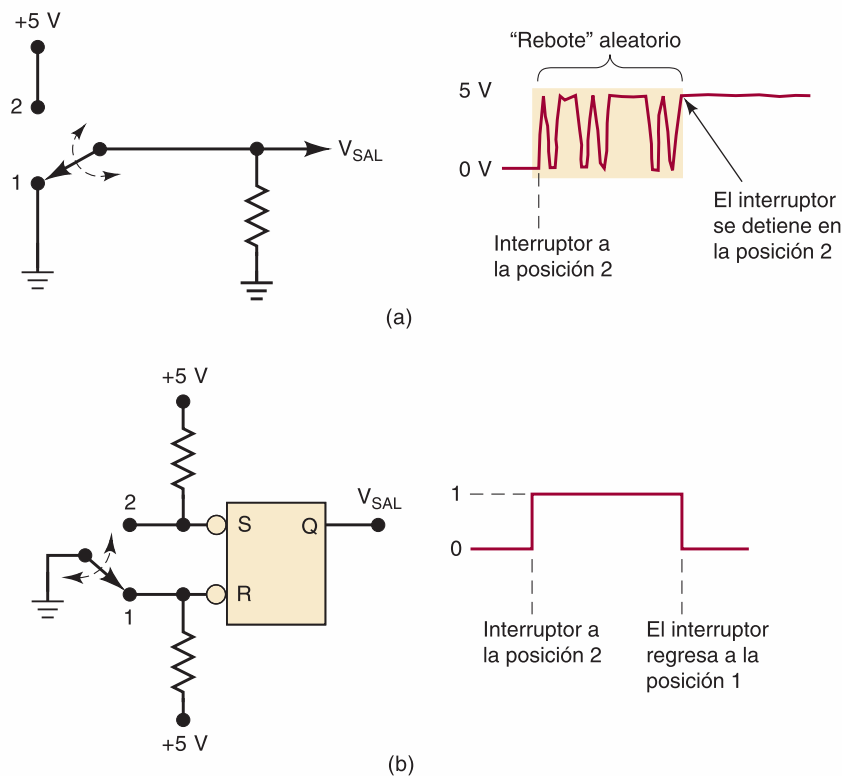
El ejemplo 5-1 muestra que la salida del latch “recuerda” la última entrada que se activó, y no cambiará de estado sino hasta que se active la entrada opuesta.

EJEMPLO 5-2

Es casi imposible obtener una transición de voltaje “limpia” de un interruptor mecánico, debido al fenómeno conocido como **rebotes de contacto**. Esto se ilustra en la figura 5-9(a), en donde la acción de mover el interruptor de la posición de contacto 1 a la 2 produce varias transiciones en el voltaje de salida, a medida que el interruptor rebota (hace contacto y lo interrumpe con el contacto 2 varias veces) antes de detenerse en el contacto 2.

Por lo general, las transiciones múltiples en la señal de salida no duran más de unos cuantos milisegundos, pero podrían ser inaceptables en muchas aplicaciones. Un latch NAND puede utilizarse para evitar que la presencia del rebote de contactos afecte la salida. Describa la operación del circuito para “eliminar rebotes en el interruptor” de la figura 5-9(b).

FIGURA 5-9 (a) El rebote mecánico de los contactos producirá transiciones múltiples; (b) latch NAND utilizado para eliminar los rebotes en un interruptor mecánico.



Solución

Suponga que el interruptor está detenido en la posición 1, de manera que la entrada $\overline{\text{RESET}}$ está en BAJO y $Q = 0$. Cuando el interruptor se mueve a la posición 2, $\overline{\text{RESET}}$ cambiará al nivel ALTO y aparecerá un nivel BAJO en la entrada $\overline{\text{SET}}$ cuando el interruptor haga el primer contacto. Esto hará que $Q = 1$ en cuestión de unos cuantos nanosegundos (el tiempo de respuesta de la compuerta NAND). Ahora, si el interruptor rebota del contacto 2, tanto $\overline{\text{SET}}$ como $\overline{\text{RESET}}$ estarán en ALTO y Q no se verá afectada; permanecerá en ALTO. Por lo tanto, nada ocurrirá en Q a medida que el interruptor rebote del contacto 2 antes de que por fin se detenga en la posición 2.

De igual forma, cuando el interruptor se mueva de la posición 2 a la posición 1, se colocará un BAJO en la entrada $\overline{\text{RESET}}$ cuando haga el primer contacto. Esto pondrá a Q en el estado BAJO, en donde permanecerá aún y cuando el interruptor rebote en el contacto 1 varias veces antes de detenerse.

De esta forma, la salida en Q consistirá de una sola transición cada vez que el interruptor se mueva de una posición a la otra.

PREGUNTAS DE REPASO

1. ¿Cuál es el estado normal de reposo para las entradas $\overline{\text{SET}}$ y $\overline{\text{RESET}}$? ¿Cuál es el estado activo de cada una de estas entradas?
2. ¿Cuáles serán los estados de Q y $\overline{Q}$ después de que se haya restablecido (borrado) un FF?
3. *Verdadero o falso:* la entrada $\overline{\text{SET}}$ nunca puede usarse para hacer que $Q = 0$.
4. Cuando se aplica energía por primera vez a cualquier circuito FF, es imposible predecir los estados iniciales de Q y $\overline{Q}$. ¿Qué puede hacerse para asegurar que un latch NAND siempre empiece en el estado $Q = 1$?

5-2 LATCH DE COMPUERTA NOR

Pueden usarse dos compuertas NOR acopladas en forma transversal para formar lo que se conoce como **latch de compuerta NOR**. El arreglo, que se muestra en la figura 5-10(a), es similar al latch NAND a excepción de que las salidas Q y $\overline{Q}$ aparecen en posiciones invertidas.

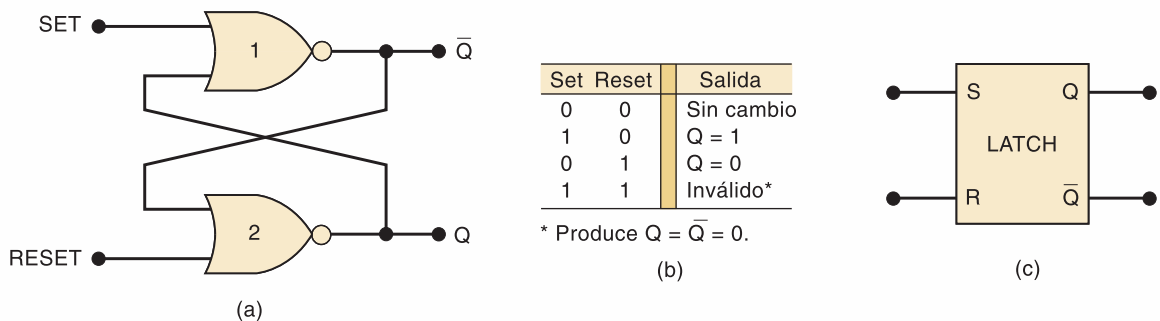


FIGURA 5-10 (a) Latch de compuerta NOR; (b) tabla de funciones; (c) símbolo de bloque simplificado.

Podemos realizar el análisis de la operación del latch NOR exactamente de la misma forma que lo hicimos para el latch NAND. Los resultados se proporcionan en la tabla de funciones de la figura 5-10(b) y se sintetizan de la siguiente manera:

1. $\text{SET} = \text{RESET} = 0$. Éste es el estado normal de reposo para el latch NOR y no tiene efecto sobre el estado de la salida. Q y $\overline{Q}$ permanecerán en el estado que tenían antes de que se produjera esta condición de entrada.

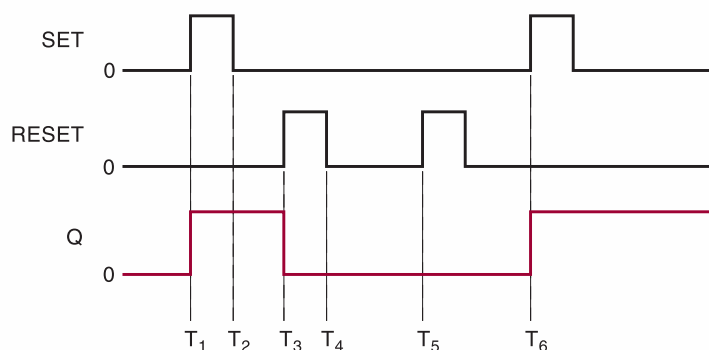
2. $SET = 1$, $RESET = 0$. Esta condición siempre establecerá $Q = 1$, en donde permanecerá aún después de que SET regrese a 0.
3. $SET = 0$, $RESET = 1$. Esta condición siempre borrará $Q = 0$, en donde permanecerá aún después de que $RESET$ regrese a 0.
4. $SET = 1$, $RESET = 1$. Esta condición trata de establecer y restablecer el latch al mismo tiempo, y produce $Q = \bar{Q} = 0$. Si las entradas se regresan a 0 en forma simultánea, el estado de salida resultante será impredecible. Esta condición de entrada no debe utilizarse.

El latch de compuerta NOR opera exactamente de la misma forma que el latch NAND, sólo que las entradas SET y $RESET$ son activas en ALTO, en vez de activas en BAJO, y el estado detenido normal es $SET = RESET = 0$. Q cambiará a nivel ALTO mediante un pulso ALTO en la entrada SET , y cambiará a BAJO mediante un pulso ALTO en la entrada $RESET$. El símbolo de bloque simplificado para el latch NOR en la figura 5-10(c) se muestra sin burbujas en las entradas S y R ; esto indica que las entradas son activas en ALTO.

EJEMPLO 5-3

Suponga que al principio $Q = 0$ y determine la forma de onda de Q para las entradas del latch NOR de la figura 5-11.

FIGURA 5-11 Ejemplo 5-3.



Solución

En un principio, $SET = RESET = 0$, lo cual no tiene efecto sobre Q , que permanece en BAJO. Cuando SET cambie a nivel ALTO en el tiempo T_1 , Q se establecerá en 1 y permanecerá ahí aún después de que SET regrese a 0 en T_2 .

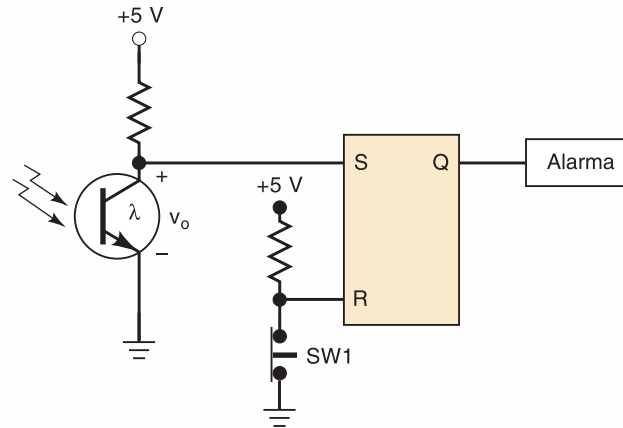
En T_3 la entrada $RESET$ cambia a nivel ALTO y borra a Q para que quede en el estado 0, en donde permanecerá aún después de que $RESET$ regrese a BAJO en T_4 .

El pulso $RESET$ en T_5 no tiene efecto sobre Q , ya que de antemano se encuentra en BAJO. El pulso en SET en el tiempo T_6 regresa a Q de nuevo a 1, en donde permanecerá por tiempo indefinido.

El ejemplo 5-3 muestra que el latch “recuerda” la última entrada que se activó, y no cambiará de estado sino hasta que se active la entrada opuesta.

EJEMPLO 5-4

La figura 5-12 muestra un circuito simple que puede utilizarse para detectar la interrupción de un haz de luz. La luz se enfoca en un fototransistor que está conectado en la configuración de emisor común para operar como un interruptor. Suponga que el latch se ha borrado previamente para quedar en el estado 0, para lo cual se abrió el interruptor $SW1$ durante unos instantes, y describa lo que ocurre si se interrumpe por un momento el haz de luz.

FIGURA 5-12 Ejemplo 5-4.**Solución**

Cuando hay luz en el fototransistor, podemos suponer que está conduciendo al 100%, de manera que la resistencia entre el colector y el emisor es muy pequeña. Por ende, v_o estará cerca de 0 V. Esto coloca un BAJO en la entrada SET del latch, de manera que SET = RESET = 0.

Cuando se interrumpe el haz de luz, el fototransistor se apaga y su resistencia de colector-emisor se hace muy alta (es decir, en esencia es un circuito abierto). Esto hace que v_o se eleve hasta un valor aproximado de 5 V; con esto se activa la entrada SET, la cual establece a Q en ALTO y desactiva la alarma.

Por otro lado, Q permanecerá en ALTO y la alarma permanecerá encendida aún cuando v_o regrese a 0 V (es decir, el haz de luz se interrumpió sólo por un momento), ya que SET y RESET estarán ambas en BAJO, lo cual no producirá cambio alguno en Q .

En esta aplicación se utiliza la característica de memoria del latch para convertir una ocurrencia momentánea (interrupción del haz) en una salida constante.

Estado de un flip-flop al momento de aplicarle energía

Cuando se aplica energía a un circuito, no es posible predecir el estado inicial de la salida de un flip-flop si sus entradas SET y RESET se encuentran en su estado inactivo (por ejemplo, $S = R = 1$ para un latch NAND, $S = R = 0$ para un latch NOR). Se tiene la misma probabilidad de que el estado inicial sea $Q = 0$ que $Q = 1$. Esto dependerá de ciertos factores tales como los tiempos de propagación internos, la capacitancia parásita y la carga externa. Si un latch o FF debe empezar en cierto estado específico para asegurar la operación apropiada de un circuito, entonces debe colocarse en ese estado mediante la activación momentánea de la entrada SET o RESET al principio de la operación del circuito. A menudo, esto se logra aplicando un pulso en la entrada apropiada.

PREGUNTAS DE REPASO

1. ¿Cuál es el estado normal en reposo de las entradas del latch NOR? ¿Cuál es el estado activo?
2. Cuando un latch se establece, ¿cuáles son los estados de Q y $\bar{Q}$?
3. ¿Cuál es la única manera de hacer que la salida Q de un latch NOR cambie de 1 a 0?
4. Si el latch NOR de la figura 5-12 se sustituyera por un latch NAND, ¿por qué el circuito no funcionaría en forma correcta?

5-3 EJEMPLO PRÁCTICO DE DIAGNÓSTICO DE FALLAS

Los siguientes dos ejemplos presentarán una ilustración de los tipos de razonamiento utilizados al diagnosticar fallas en un circuito que contenga un latch.

EJEMPLO 5-5

Analice y describa la operación del circuito de la figura 5-13.

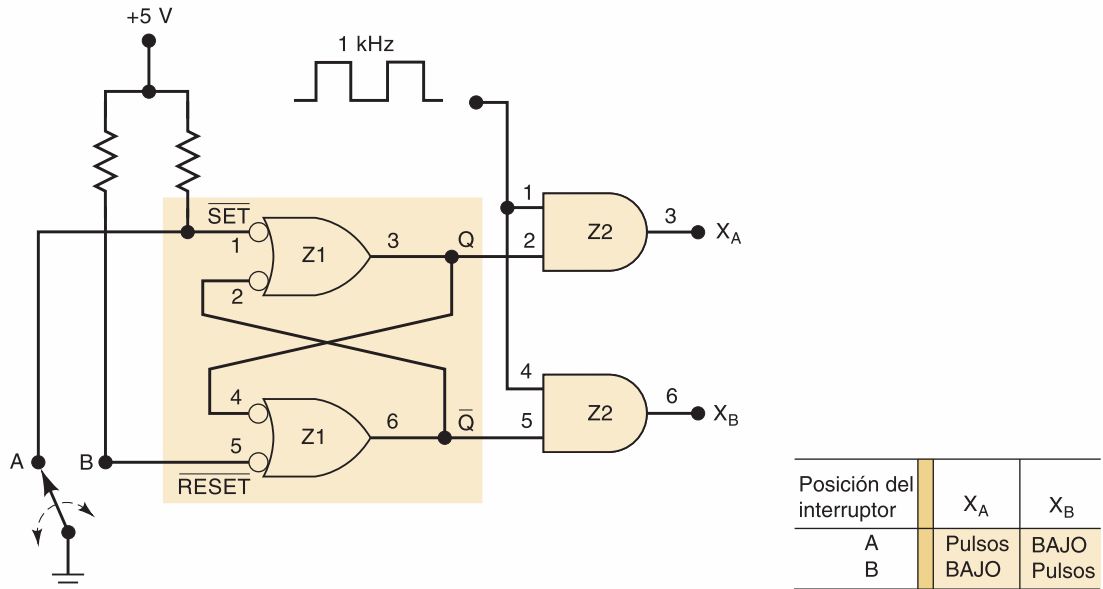


FIGURA 5-13 Ejemplos 5-5 y 5-6.

Solución

El interruptor se utiliza para establecer o borrar el latch NAND y producir señales limpias, libres de rebotes en Q y $\bar{Q}$. Estas salidas del latch controlan el paso de la señal de pulso de 1-kHz hacia las salidas X_A y X_B de la compuerta AND.

Cuando el interruptor se mueve hacia la posición A, el latch se establece para que $Q = 1$. Esto permite que los pulsos de 1-kHz pasen hacia X_A , mientras que el nivel BAJO en $\bar{Q}$ mantiene a $X_B = 0$. Cuando el interruptor se mueve hacia la posición B, el latch se borra para que $Q = 0$, lo cual mantiene a $X_A = 0$, mientras que el nivel ALTO en $\bar{Q}$ permite que los pulsos pasen hacia X_B .

EJEMPLO 5-6

Un técnico evalúa el circuito de la figura 5-13 y registra las observaciones, que se muestran en la tabla 5-1. Él observa que cuando el interruptor se encuentra en la posición B el circuito funciona de manera correcta, pero en la posición A el latch no se establece para quedar en el estado $Q = 1$. ¿Cuáles son las posibles fallas que podrían ocasionar este funcionamiento incorrecto?

Solución

Existen varias posibilidades:

1. Una conexión abierta interna en Z1-1, lo cual evitaría que Q respondiera a la entrada $\overline{\text{SET}}$.

TABLA 5-1

Posición del interruptor	$\overline{\text{SET}}$ (Z1-1)	$\overline{\text{RESET}}$ (Z1-5)	Q (Z1-3)	$\overline{Q}$ (Z1-6)	X_A (Z2-3)	X_B (Z2-6)
<i>A</i>	BAJO	ALTO	BAJO	ALTO	BAJO	Pulsos
<i>B</i>	ALTO	BAJO	BAJO	ALTO	BAJO	Pulsos

2. Una falla interna en un componente de la compuerta NAND Z1 que evite que responda en forma apropiada.
3. La salida Q se mantiene en BAJO, lo cual podría ser ocasionado por:
 - (a) Z1-3 en corto interno a tierra
 - (b) Z1-4 en corto interno a tierra
 - (c) Z2-2 en corto interno a tierra
 - (d) El nodo Q en corto externo a tierra

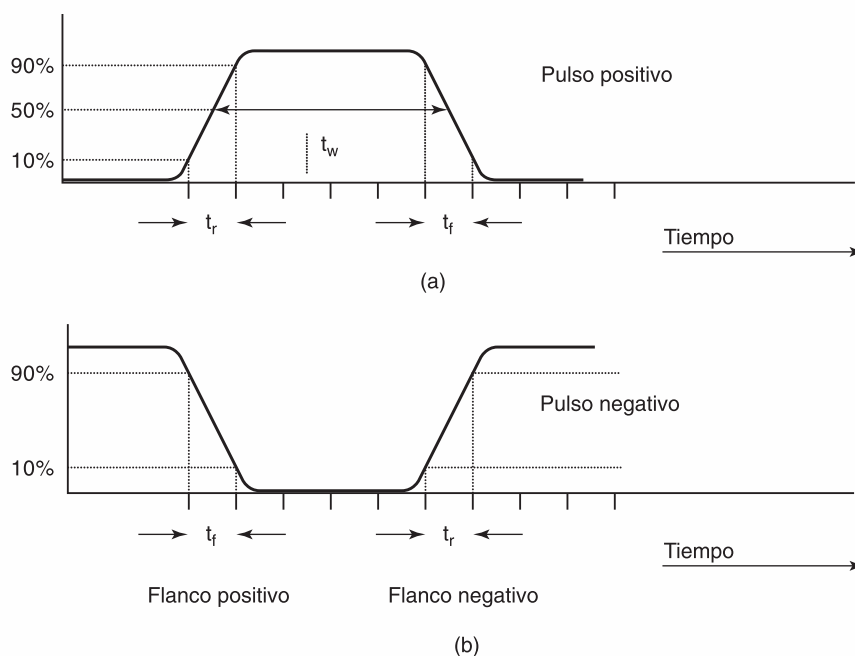
La comprobación con un óhmetro de Q a tierra determinará si cualquiera de estas condiciones está presente. Una revisión visual deberá revelar cualquier corto externo.

¿Podría $\overline{Q}$ estar en corto interno o externo con V_{CC} ? Al realizar el análisis nos llevará a la conclusión de que ésta no podría ser la falla. Si $\overline{Q}$ estuviera en corto con V_{CC} , esto no evitaría que la salida Q cambiara al nivel ALTO cuando SET cambiar a BAJO. Como Q no cambia a ALTO, ésta no puede ser la falla. La razón por la cual $\overline{Q}$ parece permanecer en ALTO es que Q permanece en BAJO, y eso mantiene a Q en ALTO mediante la compuerta NAND inferior.

5-4 PULSOS DIGITALES

Como se puede ver de nuestra disertación sobre los latches SR, hay situaciones en los sistemas digitales en las que una señal cambia de un estado normal inactivo al estado opuesto (activo), provocando que ocurra algo en el circuito. Después, la señal regresa a su estado inactivo mientras que el efecto de la señal que se activó recientemente permanece en el sistema. A estas señales se les conoce como **pulsos** y es muy importante comprender la terminología asociada con los pulsos y las formas de onda de los mismos. A un pulso que realiza su adecuada función cuando cambia a ALTO se le conoce como pulso *positivo*, y a un pulso que realiza su debida función cuando cambia a BAJO se le conoce como pulso *negativo*. En los circuitos reales, a una forma de onda de pulso le toma tiempo cambiar de un nivel al otro. A estos tiempos de transición se les conoce como tiempo de subida (t_r) y tiempo de caída (t_f), y se les define como el tiempo que tarda el voltaje en cambiar entre el 10 y el 90% del voltaje de nivel ALTO, como se muestra en el pulso positivo de la figura 5-14(a). A la transición al principio del pulso se le conoce como flanco positivo (de subida) y a la transición al final del pulso se le conoce como flanco negativo (de caída). La duración (ancho) del pulso (t_w) se le define como el tiempo entre los puntos cuando los flancos de subida y de caída se encuentran al 50% del nivel de voltaje ALTO. La figura 5-14(b) muestra un pulso activo en BAJO o negativo.

FIGURA 5-14 (a) Un pulso positivo y (b) un pulso negativo.



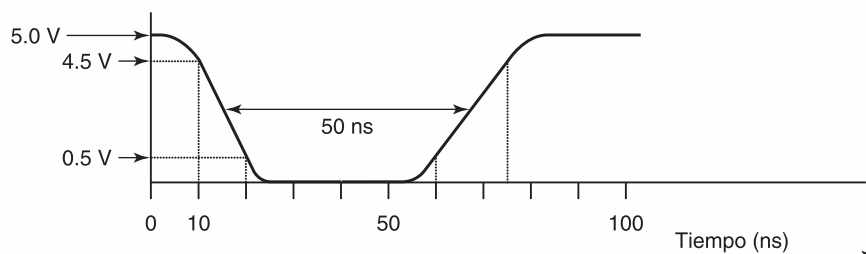
EJEMPLO 5-7

Cuando un microcontrolador desea acceder a los datos en su memoria externa, activa una terminal de salida activa en BAJO conocida como $\overline{RD}$ (lectura). La hoja de datos dice que, por lo general, el pulso $\overline{RD}$ tiene un ancho (t_w) de 50 ns, un tiempo de subida t_r de 15 ns y un tiempo de caída t_f de 10 ns. Haga un dibujo en escala del pulso $\overline{RD}$.

Solución

La figura 5-15 muestra el dibujo del pulso. El pulso $\overline{RD}$ es activo en BAJO, por lo que el flanco anterior es un flanco de caída medido en base a t_f y el flanco posterior es el flanco de subida medido en base t_r .

FIGURA 5-15 Ejemplo 5-7.



5-5 SEÑALES DE RELOJ Y FLIP-FLOPS SINCRONIZADOS POR RELOJ

Los sistemas digitales pueden operar en forma *asíncrona* o *síncrona*. En los sistemas asíncronos, las salidas de los circuitos lógicos pueden cambiar de estado en cualquier momento en el que una o más de las entradas cambien. Un sistema asíncrono es, por lo general, más difícil de diseñar que un sistema síncrono; de igual forma el proceso de diagnóstico de fallas es más sencillo para sistemas síncronos.

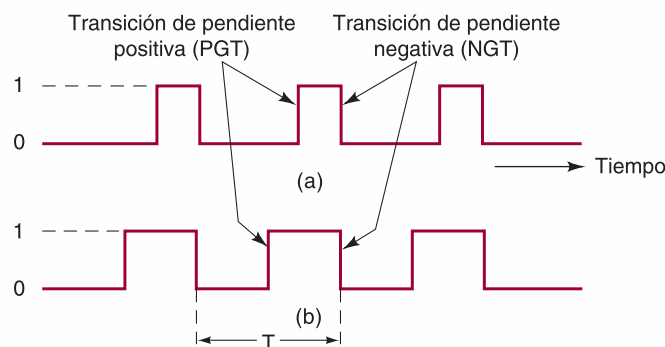
En los sistemas síncronos, los tiempos exactos en los que cualquier entrada puede cambiar de estados se determinan con base en una señal que se conoce comúnmente como el **reloj**. Por lo común, esta señal de reloj es un tren de pulsos rectangulares o una onda cuadrada, como se muestra en la figura 5-16. La señal del

reloj se distribuye en todas las partes del sistema, y la mayoría (si no es que todas) de las salidas del sistema pueden cambiar de estado sólo cuando el reloj hace una transición. La figura 5-16 muestra las transiciones (también llamadas *flancos*). Cuando el reloj cambia de un 0 a un 1 se le llama **transición de pendiente positiva (PGT)**; cuando el reloj cambia de 1 a 0 se le llama **transición de pendiente negativa (NGT)**. Utilizaremos las abreviaciones PGT y NGT ya que estos términos aparecen con mucha frecuencia a lo largo del texto.

La mayoría de los sistemas digitales son principalmente síncronos (aunque siempre hay algunas partes asíncronas), ya que es más fácil diseñar y diagnosticar fallas en los circuitos síncronos. El diagnóstico de fallas se facilita debido a que las salidas del circuito sólo pueden cambiar en instantes específicos. En otras palabras, casi todo se sincroniza con las transiciones de la señal de reloj.

La acción de sincronización de las señales del reloj se logra a través del uso de **flip-flops sincronizados por reloj**, los cuales están diseñados para cambiar de estado en una de las dos transiciones del reloj.

FIGURA 5-16 Señales de reloj.



La velocidad a la que opera un sistema digital síncrono depende de la frecuencia con la que ocurren los ciclos del reloj. Un ciclo de reloj se mide desde una PGT hasta la siguiente PGT, o desde una NGT hasta la siguiente NGT. Al tiempo que se requiere para completar un ciclo (segundos/ciclo) se le conoce como **periodo (T)**, como se muestra en la figura 5-16(b). Por lo general, se hace referencia a la velocidad de un sistema digital con base en el número de ciclos de reloj que ocurren en 1 s (ciclos/segundo), a lo cual se le conoce como **frecuencia (F)** del reloj. La unidad estándar para la frecuencia son los hertz. Un hertz (1Hz) = 1 ciclo/segundo.

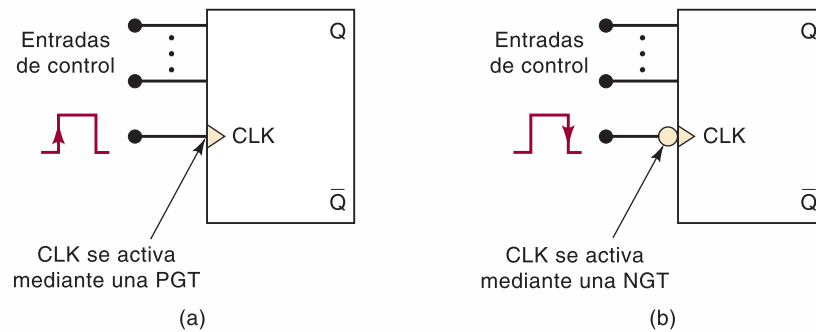
Flip-flops sincronizados por reloj

Varios tipos de FFs sincronizados por reloj se utilizan en una amplia variedad de aplicaciones, pero antes de comenzar nuestro estudio al respecto, describiremos las ideas principales que son comunes para todos ellos.

1. Los FFs sincronizados por reloj tienen una entrada de reloj que, por lo general, se identifica como *CLK*, *CK* o *CP*. Aquí utilizaremos *CLK*, como muestra la figura 5-17. En la mayoría de los FFs sincronizados por reloj la entrada *CLK* es **disparada por flanco**, lo cual significa que se activa mediante una transición de señal; esto se indica por la presencia de un pequeño triángulo en la entrada *CLK*, lo cual contrasta con los latches, que se disparan por nivel.

La figura 5-17(a) es un FF con un pequeño triángulo en su entrada *CLK* para indicar que esta entrada se activa sólo cuando ocurre una transición de pendiente positiva (PGT); ninguna otra parte del pulso de entrada tendrá efecto sobre la entrada *CLK*. En la figura 5-17(b) el símbolo FF tiene una burbuja, así como un triángulo en su entrada *CLK*. Esto significa que la entrada *CLK* se activa sólo cuando ocurre una transición de pendiente negativa; ninguna otra parte del pulso de entrada tendrá efecto sobre la entrada *CLK*.

FIGURA 5-17 Los FFs sincronizados por reloj tienen una entrada de reloj (CLK) que se activa ya sea en (a) la PGT o (b) la NGT. Las entradas de control determinan el efecto de la transición activa del reloj.



- Los FFs sincronizados por reloj tienen una o más **entradas de control** que pueden tener varios nombres, dependiendo de su operación. Las entradas de control no tendrán efecto sobre Q sino hasta que ocurra la transición activa del reloj. En otras palabras, su efecto está sincronizado con la señal que se aplica a CLK . Por esta razón se les conoce como **entradas de control síncronas**.

Por ejemplo, las entradas de control del FF de la figura 5-17(a) no tendrán efecto sobre Q sino hasta que ocurra la PGT de la señal del reloj. De igual forma, las entradas de control en la figura 5-17(b) no tendrán efecto sino hasta que ocurra la NGT de la señal del reloj.

- En resumen podemos decir que las entradas de control preparan a las salidas de FF para cambiar, mientras que la transición activa en la entrada CLK *dispara* el cambio. Las entradas de control controlan el **QUE** (es decir, a qué estado cambiará la salida); la entrada CLK determina el **CUANDO**.

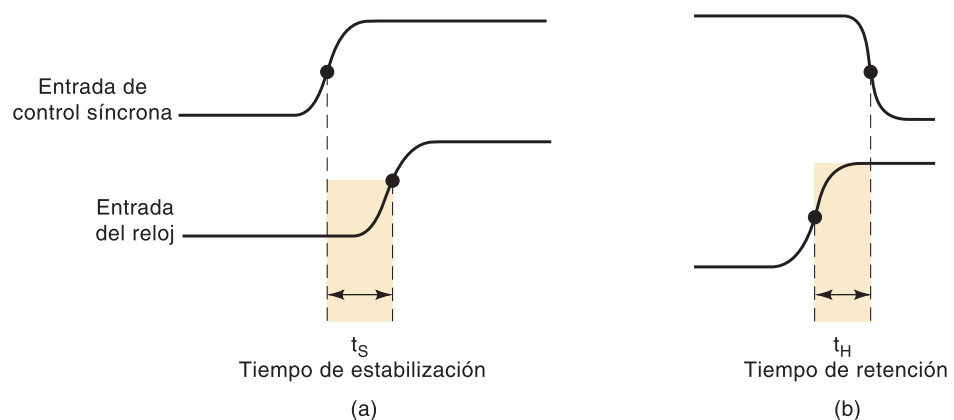
Tiempos de estabilización y de retención

Deben cumplirse dos requerimientos de sincronización si un FF sincronizado por reloj va a responder en forma confiable a sus entradas de control cuando ocurra la transición activa en CLK . Estos requerimientos se ilustran en la figura 5-18 para un FF que se dispara en una PGT.

El **tiempo de estabilización** (t_s) es el tiempo que va justo antes de la transición activa de la señal CLK , durante la cual la entrada de control debe mantenerse en el nivel apropiado. Por lo general, los fabricantes de circuitos integrados especifican el tiempo de estabilización mínimo permisible t_s (min). Si no se cumple este requerimiento de tiempo, el FF tal vez no responda en forma confiable cuando ocurra el flanco del reloj.

El **tiempo de retención** (t_H) es el tiempo que sigue justo después de la transición activa de la señal CLK , durante la cual la entrada de control síncrona debe mantenerse en el nivel apropiado. Por lo general, los fabricantes de circuitos inte-

FIGURA 5-18 Las entradas de control deben mantenerse estables durante (a) un tiempo t_s antes de la transición activa del reloj y (b) durante un tiempo t_H después de la transición activa del reloj.



grados especifican el valor mínimo aceptable para el tiempo de retención t_H (min). Si no se cumple este requerimiento, el FF no se disparará en forma confiable.

Por lo anterior, para asegurar que un FF sincronizado por reloj responderá en forma apropiada cuando ocurra la transición activa del reloj, las entradas de control deben estar estables (sin cambios) durante cuando menos un tiempo igual a t_S (min) *antes* de la transición del reloj, y durante cuando menos un tiempo igual a t_H (min) *después* de la transición del reloj.

Los flip-flops de CI tendrán valores mínimos permisibles para t_S y t_H en el orden de los nanosegundos. Por lo general, los tiempos de estabilización se encuentran en el intervalo de 5 a 50 ns, mientras que los de retención son de 0 a 10 ns. Observe que estos tiempos se miden entre los puntos del 50 por ciento en las transiciones.

Estos requerimientos de sincronización son muy importantes en los sistemas síncronos ya que, como veremos, habrá muchas situaciones en las que las entradas de control síncronas para un FF estén cambiando a un tiempo casi igual que la entrada CLK .

PREGUNTAS DE REPASO

1. ¿Cuáles son los dos tipos de entradas que tiene un FF sincronizado por reloj?
2. ¿Qué significa el término *disparado por flanco*?
3. *Verdadero o falso*: la entrada CLK afectará a la salida del FF sólo cuando ocurra la transición activa de la entrada de control.
4. Defina los requerimientos para el tiempo de estabilización y el tiempo de retención de un FF sincronizado por reloj.

5-6 FLIP-FLOP SINCRONIZADO POR RELOJ EN S-R

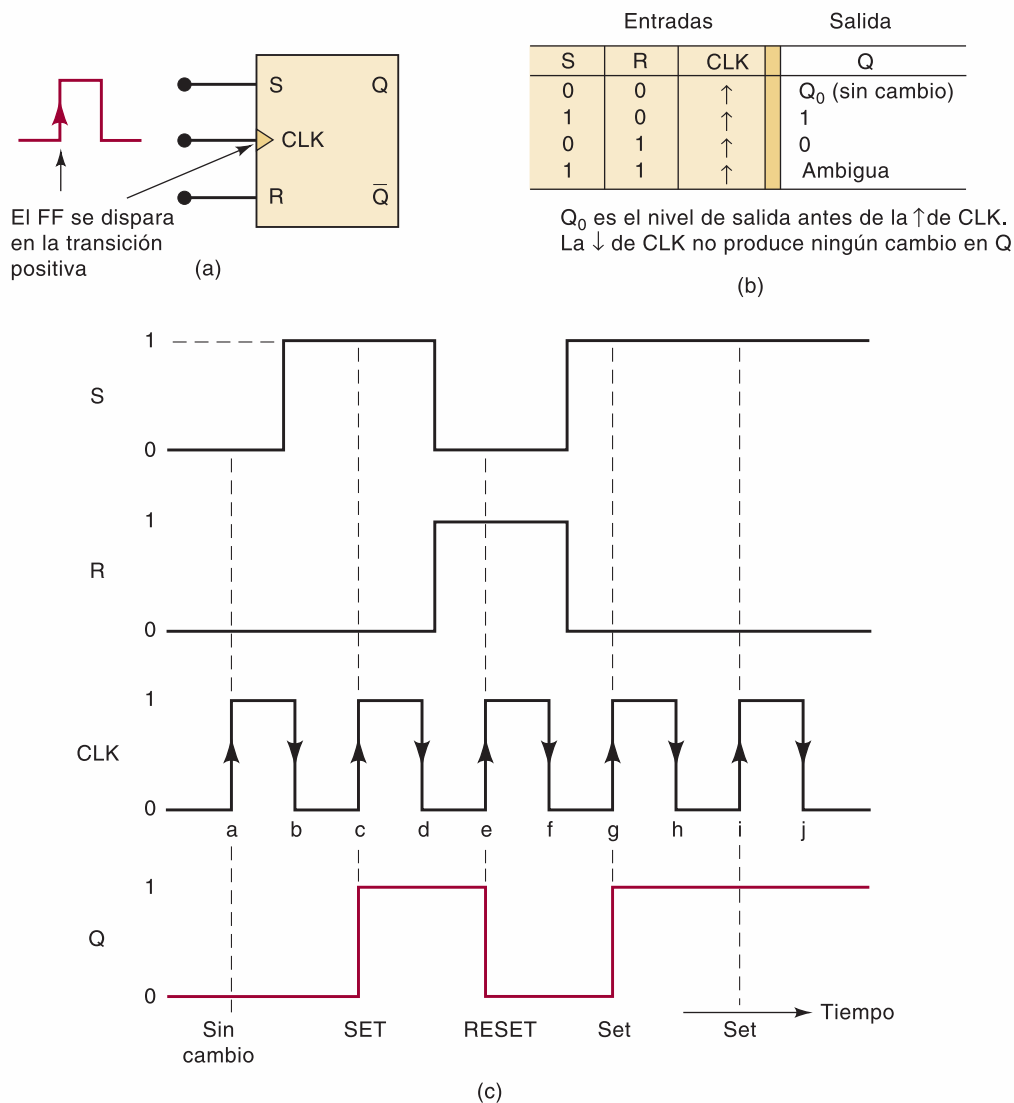
La figura 5-19(a) muestra el símbolo lógico para un **flip-flop sincronizado por reloj en S-R** que se dispara por el flanco de pendiente positiva de la señal del reloj. Esto significa que el FF puede cambiar de estado *sólo* cuando una señal que se aplica a su entrada de reloj realiza la transición de 0 a 1. Las entradas S y R controlan el estado del FF de la misma forma como se describió antes para el latch de compuerta NOR, pero el FF no responde a estas entradas sino hasta que ocurre la PGT en la señal del reloj.

La tabla de funciones de la figura 5-19(b) muestra cómo responderá la salida del FF a la PGT en la entrada CLK para las diversas combinaciones de las entradas S y R . Esta tabla de funciones utiliza cierta nomenclatura nueva. La flecha hacia arriba (a) indica que se requiere una PGT en CLK ; la etiqueta Q_0 indica el nivel en Q antes de la PGT. Esta nomenclatura se utiliza a menudo por los fabricantes de circuitos integrados en las hojas de datos de los CIs.

Las formas de onda de la figura 5-19(c) ilustran la operación del flip-flop S-R sincronizado por reloj. Si suponemos que se están cumpliendo los requerimientos para el tiempo de estabilización y el tiempo de retención en todos los casos, podremos analizar estas formas de onda de la siguiente manera:

1. Al principio todas las entradas son 0 y se asume que la salida Q es 0; esto es, $Q_0 = 0$.
2. Cuando ocurre la PGT del primer pulso de reloj (punto *a*) las entradas S y R son ambas 0, por lo que el FF no se ve afectado y permanece en el estado $Q = 0$ (es decir, $Q = Q_0$).
3. Al ocurrir la PGT del segundo pulso de reloj (punto *c*) la entrada S ahora está en nivel ALTO, mientras que R sigue en BAJO. Por ende, el FF se establece para quedar en el estado 1 durante el flanco de subida de este pulso de reloj.
4. Cuando el tercer pulso de reloj produce su transición positiva (punto *e*), encuentra que $S = 0$ y $R = 1$, lo cual hace que el FF se borre para quedar en el estado 0.

FIGURA 5-19 (a) Flip-flop S-R sincronizado por reloj que responde sólo al flanco positivo de un pulso de reloj; (b) tabla de funciones; (c) formas de onda comunes.

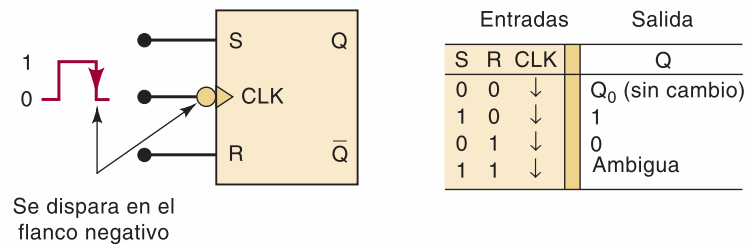


- El cuarto pulso establece el FF una vez más para quedar en el estado $Q = 1$ (punto g), ya que $S = 1$ y $R = 0$ cuando ocurre el flanco positivo.
- El quinto pulso también encuentra que $S = 1$ y $R = 0$ cuando realiza su transición de pendiente positiva. No obstante, Q ya se encuentra en nivel ALTO por lo que permanece en ese estado.
- La condición $S = R = 1$ no debe utilizarse, ya que produce una condición ambigua.

Hay que recalcar de estas formas de onda que el FF no se ve afectado por las transiciones de pendiente negativa de los pulsos del reloj. Además, observe que los niveles de S y de R no tienen efecto sobre el FF, excepto al momento en que ocurre una transición de pendiente positiva en la señal del reloj. Las entradas S y R son entradas de *control* síncronas; controlan el estado al que cambiará el FF cuando ocurra el pulso de reloj. La entrada CLK es la entrada de **disparo**, la cual hace que el FF cambie de estado de acuerdo con el nivel de las entradas S y R cuando ocurre la transición activa del reloj.

La figura 5-20 muestra el símbolo y la tabla de funciones para un flip-flop sincronizado por reloj en S-R que se dispara con la transición de pendiente *negativa* en su entrada CLK . El pequeño círculo y el pequeño triángulo en la entrada CLK

FIGURA 5-20 Flip-flop sincronizado por reloj en S-R que se dispara sólo en las transiciones de pendiente negativa.



indican que este FF se disparará sólo cuando la entrada *CLK* cambie de 1 a 0. Este FF opera de la misma forma que el FF de flanco positivo, con la diferencia de que la salida puede cambiar de estado sólo en el flanco negativo de los pulsos de reloj (puntos *b*, *d*, *f*, *h* y *j* en la figura 5-19). Tanto el FF de disparo en flanco positivo como el de disparo en flanco negativo se utilizan en los sistemas digitales.

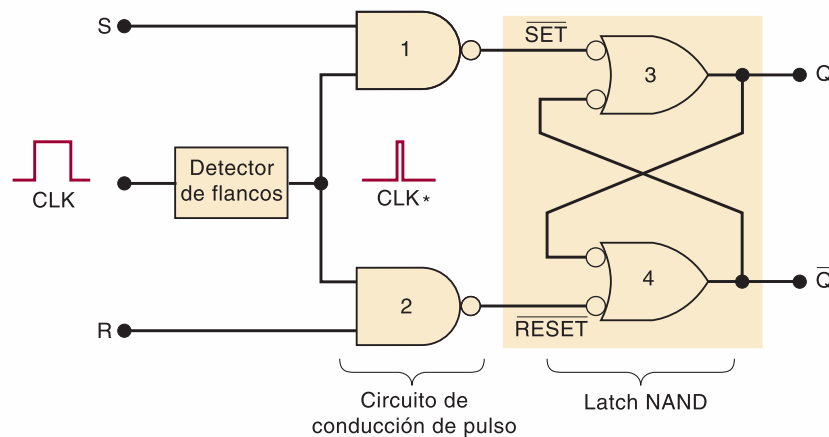
Circuitos internos del flip-flop sincronizado por reloj en S-R

No es necesario un análisis detallado de los circuitos internos de un FF sincronizado por reloj, ya que todos los tipos se encuentran disponibles como CIs. Aunque nuestro interés principal es en la operación externa del FF, nuestra comprensión de esta operación externa puede mejorar si analizamos una versión simplificada de los circuitos internos del FF. La figura 5-21 muestra lo anterior para un flip-flop S-R disparado por flanco.

El circuito contiene tres secciones:

1. Un latch de compuerta NAND básico, formado por las compuertas NAND-3 y NAND-4.
2. Un **circuito de conducción de pulso**, formado por las compuertas NAND-1 y NAND-2.
3. Un **circuito detector de flancos**.

FIGURA 5-21 Versión simplificada de los circuitos internos para un flip-flop sincronizado por reloj en S-R.



Como muestra la figura 5-21, el detector de flancos produce un pico estrecho de pendiente positiva (CLK^*), el cual ocurre en forma coincidente con la transición activa del pulso de la entrada *CLK*. El circuito de conducción de pulso “guía” el pico a través de la entrada SET o de la entrada RESET del latch, de acuerdo con los niveles presentes en *S* y *R*. Por ejemplo, con $S = 1$ y $R = 0$ la señal CLK^* se invierte y se pasa a través de la compuerta NAND-1 para producir un pulso BAJO en la entrada SET del latch, la cual establece $Q = 1$. Con $S = 0$, $R = 1$ la señal CLK^* se invierte y se pasa a través de la compuerta NAND-2 para producir un pulso BAJO en la entrada RESET del latch, la cual restablece a $Q = 0$.

La figura 5-22(a) muestra cómo se genera la señal CLK^* para los FFs disparados por flanco que se disparan con una PGT. El INVERSOR produce un retraso de unos cuantos nanosegundos, de manera que las transiciones de $\bar{CLK}$ ocurran un poco

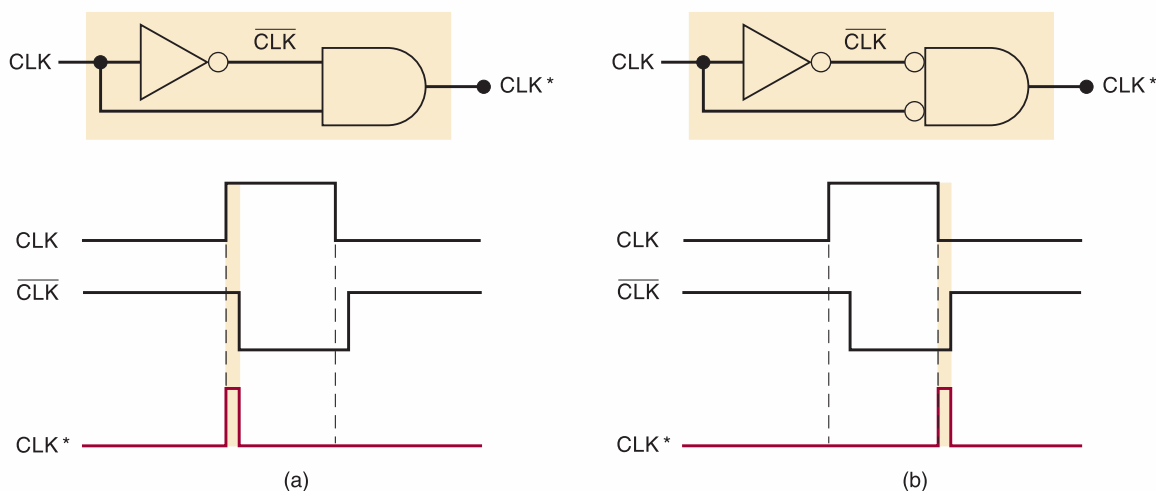


FIGURA 5-22 Implementación de circuitos detectores de flancos que se utilizan en flip-flops disparados por flanco: (a) PGT; (b) NGT. La duración de los pulsos en CLK^* es por lo general de 2 a 5 ns.

después de las de $\overline{CLK}$. La compuerta AND produce un pulso de salida que está en nivel ALTO sólo durante los pocos nanosegundos cuando CLK y $\overline{CLK}$ se encuentran ambas en ALTO. El resultado es un pulso estrecho en CLK^* , que ocurre en la PGT de CLK . El arreglo de la figura 5-22(b) produce de igual forma a CLK^* en la NGT de CLK para los FFs que se disparan con una NGT.

Como la señal CLK^* está en ALTO durante unos cuantos nanosegundos, Q se ve afectada por los niveles en S y R sólo durante un corto intervalo, durante y después de la ocurrencia del flanco activo de CLK . Esto es lo que proporciona al FF su propiedad de dispararse por flancos.

PREGUNTAS DE REPASO

1. Suponga que las formas de onda de la figura 5-19(c) se aplican a las entradas del FF de la figura 5-20. ¿Qué ocurrirá con Q en el punto b ? ¿En el punto f ? ¿En el punto h ?
2. Explique por qué las entradas S y R afectan a Q sólo durante la transición activa de CLK .

5-7 FLIP-FLOP SINCRONIZADO POR RELOJ EN J-K

La figura 5-23(a) muestra un **flip-flop sincronizado por reloj en J-K**, el cual se dispara por el flanco de pendiente positiva de la señal de reloj. Las entradas J y K controlan el estado del FF de la misma forma que las entradas S y R controlan el flip-flop S-R, sólo por una gran diferencia: *la condición $J = K = 1$ no produce una salida ambigua*. Para esta condición de 1, 1, el FF siempre cambiará a su estado opuesto al momento en que ocurra la transición positiva de la señal de reloj. A este modo de operación se le conoce como el **modo de conmutación**. En este modo, si tanto J como K se dejan en ALTO, el FF cambiará de estado (conmutará) para cada PGT del reloj.

La tabla de funciones de la figura 5-23(a) sintetiza la forma en que el flip-flop J-K responde a la PGT para cada combinación de J y de K . Observe que la tabla de funciones es la misma que para el flip-flop sincronizado por reloj en S-R (figura 5-19), excepto por la condición $J = K = 1$. Esta condición produce $Q = \overline{Q_0}$, lo cual significa que el nuevo valor de Q será el inverso del valor que tenía antes de la PGT; ésta es la operación de conmutación.

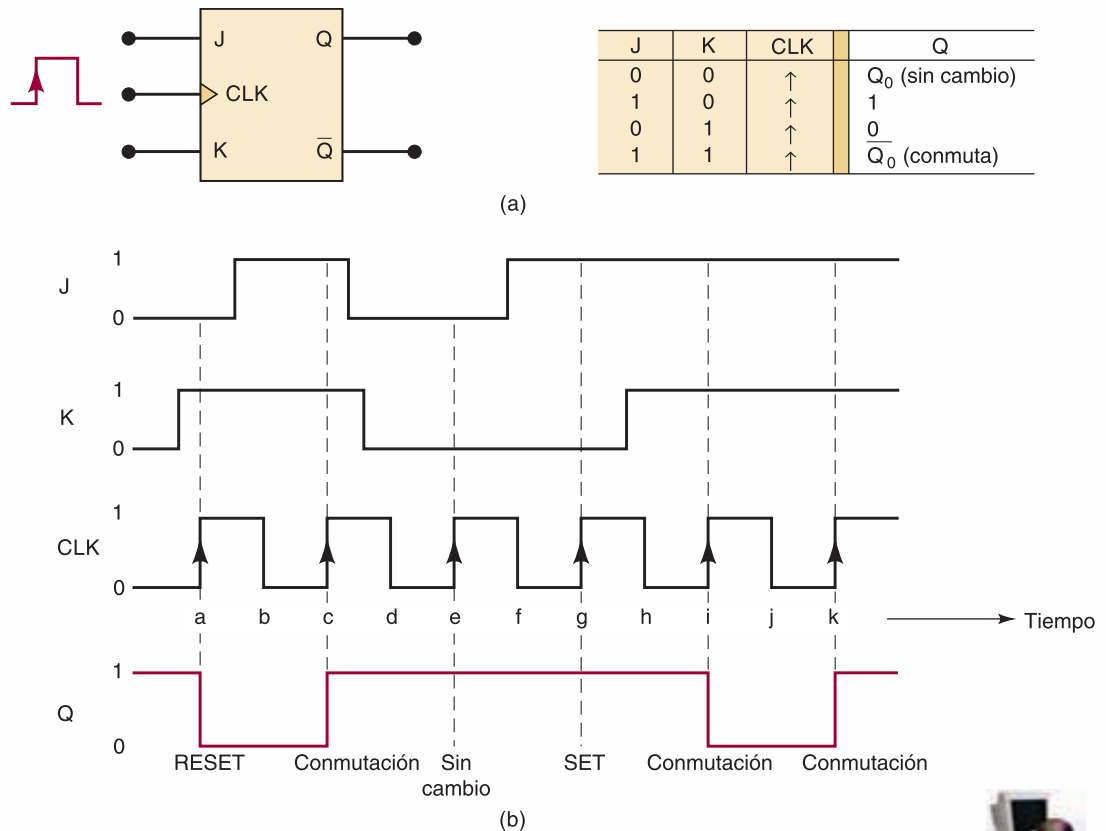


FIGURA 5-23 (a) Flip-flop sincronizado por reloj en J-K que responde sólo al flanco positivo del reloj; (b) formas de onda.

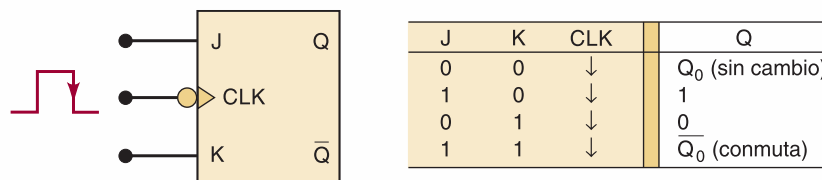
La operación de este FF se ilustra en la figura 5-23(b) mediante las formas de onda. Una vez más, suponemos que se han cumplido los requerimientos para el tiempo de estabilización y el de retención.

1. Al principio todas las entradas son 0 y se asume que la salida Q es 1; esto es, $Q_0 = 1$.
2. Cuando ocurre el flanco de pendiente positiva del primer pulso de reloj (punto a), existe la condición $J = 0, K = 1$. Por ende, el FF se restablecerá para quedar en el estado $Q = 0$.
3. El segundo pulso de reloj encuentra que $J = K = 1$ cuando realiza su transición positiva (punto c). Esto hace que el FF *conmute* a su estado opuesto, $Q = 1$.
4. En el punto e en la forma de onda del reloj, J y K son ambas 0 por lo que el FF no cambia de estado en esta transición.
5. En el punto g , $J = 1$ y $K = 0$. Ésta es la condición que establece Q para quedar en el estado 1. No obstante, ya se encuentra en 1 por lo que permanecerá ahí.
6. En el punto i , $J = K = 1$ por lo que el FF conmuta a su estado opuesto. Lo mismo ocurre en el punto k .

De estas formas de onda podemos observar que el FF no se ve afectado por el flanco de pendiente negativa de los pulsos de reloj. Además, los niveles de las entradas J y K no tienen efecto, excepto cuando ocurre la PGT de la señal del reloj. Las entradas J y K por sí solas no pueden hacer que el FF cambie de estado.

La figura 5-24 muestra el símbolo para un flip-flop sincronizado por reloj en J-K que se dispara con las transiciones de pendiente negativa de la señal del reloj. El pequeño círculo en la entrada CLK indica que este FF se disparará cuando la

FIGURA 5-24 Flip-flop J-K que se dispara sólo con las transiciones de pendiente negativa.



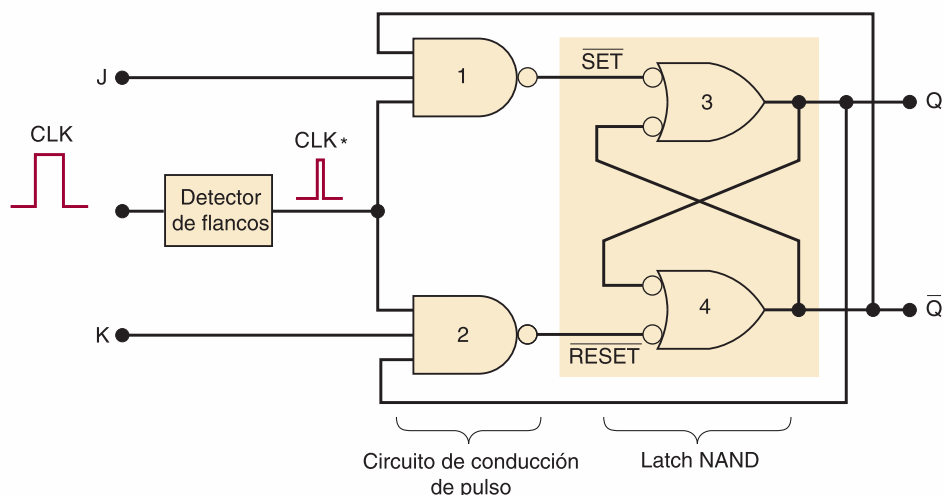
entrada CLK cambie de 1 a 0. Este FF opera de la misma forma que el FF de flanco positivo de la figura 5-23, excepto que la salida puede cambiar de estado sólo en las transiciones de pendiente negativa de la señal del reloj (puntos *b*, *d*, *f*, *h* y *j*). Ambas polaridades de los flip-flops J-K disparados por flanco son de uso común.

El flip-flop J-K es mucho más versátil que el flip-flop S-R, ya que no tiene estados ambiguos. La condición $J = K = 1$, que produce la operación de conmutación, tiene un uso extenso en todos los tipos de contadores binarios. En esencia, el flip-flop J-K puede hacer cualquier cosa que hace el flip-flop S-R, además de operar en el modo de conmutación.

Circuitos internos del flip-flop J-K disparado por flanco

En la figura 5-25 se muestra una versión simplificada de los circuitos internos de un flip-flop J-K disparado por flanco. Contiene las mismas tres secciones que el flip-flop S-R disparado por flanco (figura 5-21). De hecho, la única diferencia entre los dos circuitos es que las salidas Q y $\bar{Q}$ se alimentan de vuelta a las compuertas NAND de conducción de pulso. Esta conexión de retroalimentación es lo que proporciona al flip-flop J-K su operación de conmutación para la condición $J = K = 1$.

FIGURA 5-25 Circuito interno del flip-flop J-K disparado por flanco.



Examinemos ahora esta condición de conmutación más de cerca, asumiendo que $J = K = 1$ y que Q permanece en el estado BAJO cuando ocurre un pulso en CLK . Con $Q = 0$ y $\bar{Q} = 1$, la compuerta NAND 1 dirigirá a CLK^* (invertida) hacia la entrada $\overline{SET}$ del latch NAND para producir $Q = 1$. Si asumimos que Q está en ALTO cuando ocurre un pulso en CLK , la compuerta NAND 2 dirigirá a CLK^* (invertida) hacia la entrada $\overline{RESET}$ del latch para producir $Q = 0$. Por ende, Q siempre terminará en el estado opuesto.

Para que la operación de conmutación funcione como se describió antes, el pulso en CLK^* debe ser muy estrecho. Debe regresar a 0 antes de que las salidas Q y $\bar{Q}$ conmuten a sus nuevos valores; de no ser así, los nuevos valores de Q y $\bar{Q}$ harán que el pulso CLK^* conmute de nuevo las salidas del latch.

PREGUNTAS DE REPASO

1. Verdadero o falso: un flip-flop J-K puede usarse como un flip-flop S-R, pero un flip-flop S-R no puede usarse como un flip-flop J-K.
2. ¿Tiene un flip-flop J-K condiciones de entrada ambiguas?
3. ¿Qué condición de entrada en J-K siempre establecerá $Q=1$ cuando ocurra la transición activa en CLK ?

5-8 FLIP-FLOP SINCRONIZADO POR RELOJ EN D

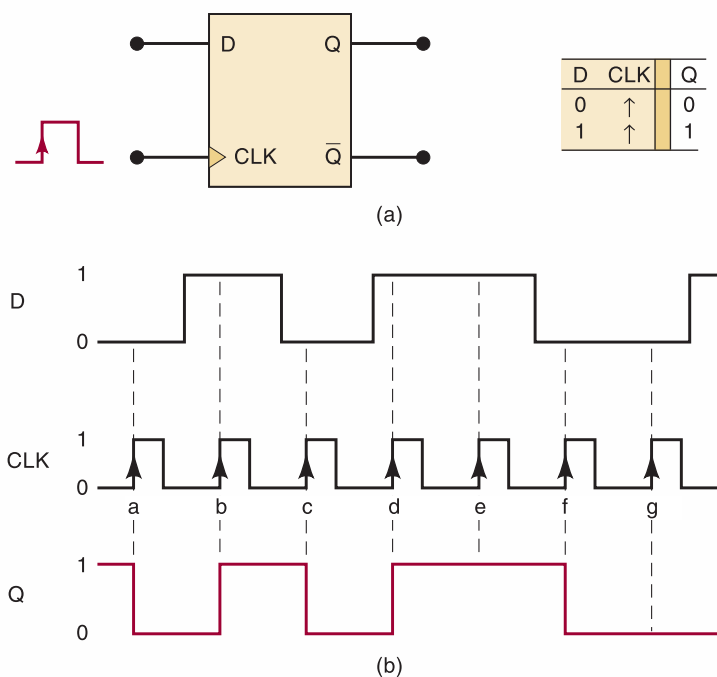
La figura 5-26(a) muestra el símbolo y la tabla de funciones para un **flip-flop sincronizado por reloj en D** que se dispara con una PGT. A diferencia de los flip-flops S-R y J-K, este flip-flop sólo tiene una entrada de control síncrona D (la cual significa *datos*). La operación del flip-flop D es muy simple: Q cambiará al mismo estado que esté presente en la entrada D cuando ocurra una PGT en CLK . En otras palabras, el nivel presente en D se *almacenará* en el flip-flop en el instante en que ocurra la PGT. Las formas de onda de la figura 5-26(b) ilustran esta operación.

Suponga que al principio Q está en nivel ALTO. Cuando ocurre la primera PGT en el punto a , la entrada D está en BAJO; por ende, Q cambiará al estado 0. Aún y cuando el nivel de la entrada D cambie entre los puntos a y b , no tiene efecto sobre Q ; Q está almacenando el nivel BAJO que se encontraba en D en el punto a . Cuando ocurre la PGT en b , Q cambia a ALTO porque D está en ALTO en ese momento. Q almacena este nivel ALTO hasta que la PGT en el punto c hace que Q cambie a BAJO, ya que D se encuentra en BAJO en ese momento. De una manera similar, la salida Q toma los niveles presentes en D cuando ocurren las PGTs en los puntos d , e , f y g . Observe que Q permanece en ALTO en el punto e , porque D aún está en ALTO.

De nuevo, es importante recordar que Q puede cambiar sólo cuando ocurre una PGT. La entrada D no tiene efecto entre las PGTs.

Un flip-flop D disparado por flanco negativo opera de la misma forma antes descrita, sólo que Q tomará el valor de D cuando ocurra una NGT en CLK . El símbolo para el flip-flop D que se dispara con las NGTs tendrá una burbuja en la entrada CLK .

FIGURA 5-26 (a) Flip-flop D que se dispara sólo en las transiciones de pendiente positiva; (b) formas de onda.

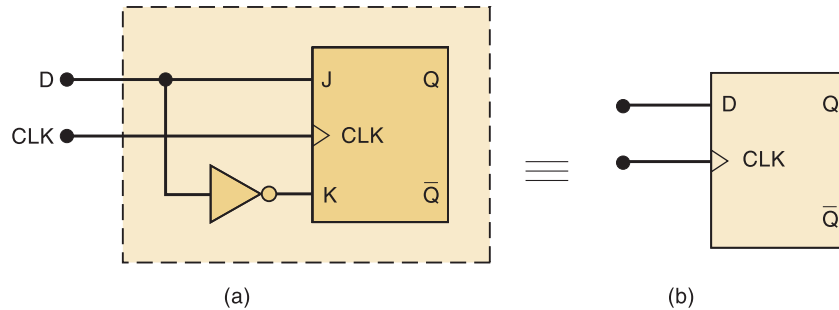


Implementación del flip-flop D

Un flip-flop D disparado por flanco se implementa con facilidad con sólo agregar un INVERSOR al flip-flop J-K disparado por flanco, como se muestra en la figura 5-27. Si usted prueba ambos valores de D , podrá ver que Q toma el nivel presente en D cuando ocurre una PGT. Lo mismo puede hacerse para convertir un flip-flop S-R en un flip-flop D.

FIGURA 5-27

Implementación de un flip-flop D disparado por flanco, a partir de un flip-flop J-K.



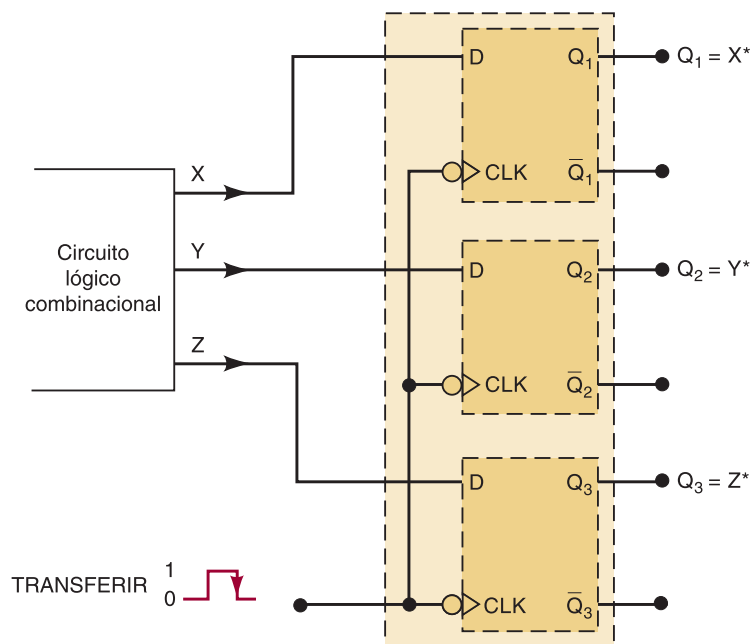
Transferencia de datos en paralelo

En este punto tal vez se esté preguntando acerca de la utilidad del flip-flop D, ya que parece que la salida Q es la misma que la entrada D . Pero no es exactamente así; recuerde que Q toma el valor de D sólo en ciertos intervalos de tiempo, por lo cual no es idéntica a D (por ejemplo, vea las formas de onda de la figura 5-26).

En la mayoría de las aplicaciones del flip-flop D, la salida Q debe tomar el valor de su entrada D sólo en tiempos definidos con precisión. Un ejemplo de esto se ilustra en la figura 5-28. Las salidas X , Y , Z de un circuito lógico van a transferirse a los FFs Q_1 , Q_2 y Q_3 para su almacenamiento. Si utilizamos los flip-flops D, los niveles presentes en X , Y y Z se transferirán a Q_1 , Q_2 y Q_3 en forma respectiva, al momento de aplicar un pulso en TRANSFERIR a las entradas CLK comunes. Los FFs pueden almacenar estos valores para su posterior procesamiento. Éste es un ejemplo de la **transferencia de datos en paralelo** para datos binarios; los tres bits X , Y y Z se transfieren en forma *simultánea*.

FIGURA 5-28

Transferencia en paralelo de datos binarios mediante el uso de flip-flops D.



*Después de que ocurre la NGT

PREGUNTAS DE REPASO

1. ¿Qué ocurrirá con la forma de onda de Q en la figura 5-26(b) si la entrada D se mantiene en BAJO de manera permanente?
2. Verdadero o falso: la salida Q será igual al nivel en la entrada D en todo momento.
3. ¿Pueden usarse los FFs J-K para la transferencia de datos en paralelo?

5-9 LATCH D (LATCH TRANSPARENTE)

El flip-flop D disparado por flanco utiliza un circuito detector de flancos para asegurar que la salida responda a la entrada D sólo cuando ocurre la transición activa del reloj. Si no se utiliza este detector de flancos, el circuito resultante opera en forma un poco distinta. A éste se le conoce como **latch D** y tiene el arreglo que se muestra en la figura 5-29(a).

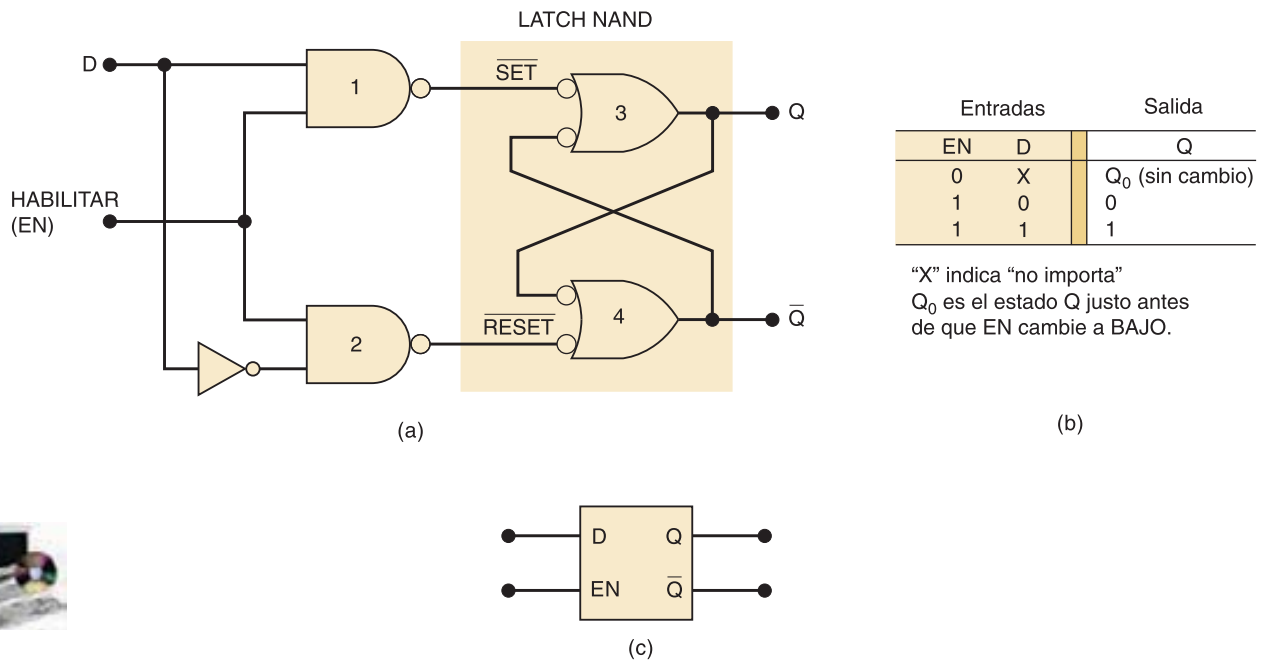


FIGURA 5-29 Latch D : (a) estructura; (b) tabla de funciones; (c) símbolo lógico.

El circuito contiene el latch NAND y las compuertas NAND de dirección 1 y 2 sin el circuito detector de flancos. La entrada común para las compuertas de dirección se llama entrada de *habilitación* (se abrevia como EN) en vez de una entrada de reloj, ya que su efecto sobre las salidas Q y $\bar{Q}$ no se restringe a ocurrir sólo en sus transiciones. La operación del latch D se describe a continuación:

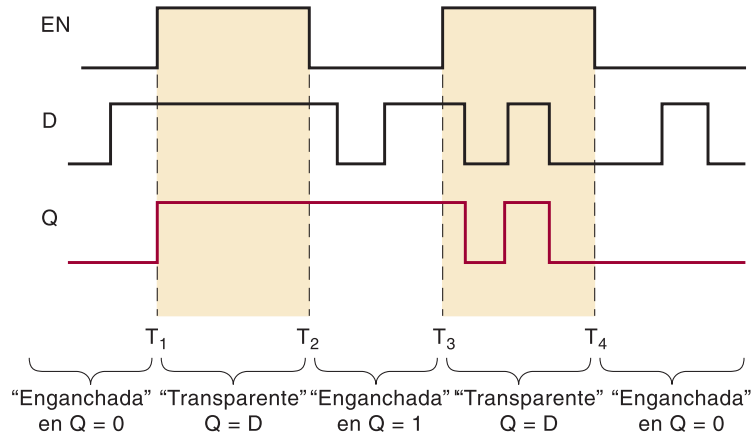
1. Cuando EN está en ALTO, la entrada D producirá un nivel BAJO ya sea en la entrada $\overline{SET}$ o en la entrada $\overline{RESET}$ del latch NAND, para hacer que Q tenga el mismo nivel que D . Si D cambia mientras que EN está en ALTO, Q seguirá los mismos cambios. En otras palabras, mientras que $EN = 1$ la salida Q se verá justo igual que D ; en este modo, se dice que el latch D es "transparente".
2. Cuando EN cambia a BAJO, la entrada D queda inhibida y no puede afectar al latch NAND debido a que las salidas de ambas compuertas de dirección se mantendrán en ALTO. Por ende, las salidas Q y $\bar{Q}$ permanecerán en el nivel que tenían justo antes de que EN cambiara a BAJO. En otras palabras, las salidas se "enganchan" a su nivel actual y no pueden cambiar mientras que EN se encuentre en BAJO, aún si D cambia.

Esta operación se sintetiza en la tabla de funciones de la figura 5-29(b). El símbolo lógico para el latch D se muestra en la figura 5-29(c). Observe que aún cuando la entrada EN opera en forma muy parecida a la entrada CLK de un FF disparado por flanco, no hay un pequeño triángulo en la entrada EN . Esto se debe a que el símbolo del pequeño triángulo se utiliza estrictamente para entradas que pueden provocar un cambio en la salida sólo cuando ocurre una transición. El latch D no se dispara por flanco.

EJEMPLO 5-8

FIGURA 5-30 Formas de onda para el ejemplo 5-8, las cuales muestran los dos modos de operación del latch D transparente.

Determine la forma de onda de Q para un latch D con las entradas EN y D de la figura 5-30. Asuma que al principio $Q = 0$.



Solución

Antes del tiempo T_1 EN está en BAJO, por lo que Q está “enganchada” en su nivel 0 actual y no puede cambiar aunque cambie D . Durante el intervalo de T_1 a T_2 , EN está en ALTO, de manera que Q seguirá la señal presente en D . Por ende, Q cambia a ALTO en T_1 y permanece ahí porque D no está cambiando. Cuando EN regrese a BAJO en T_2 , Q se enganchará en el nivel ALTO que tiene en T_2 y permanecerá ahí hasta que EN cambie a BAJO.

En T_3 , cuando EN cambia de nuevo a ALTO, Q seguirá los cambios en la entrada D hasta T_4 , cuando EN regrese a BAJO. Durante el intervalo de T_3 a T_4 , el latch D es “transparente” ya que las variaciones en D pasan hasta la salida Q . En T_4 , cuando EN cambia a BAJO, Q se enganchará en el nivel 0 debido a que es su nivel en T_4 . Después de T_4 las variaciones en D no tendrán efecto sobre Q , ya que está enganchada (es decir, $EN = 0$).

PREGUNTAS DE REPASO

1. Describa cómo un latch D opera en forma distinta a un flip-flop D disparado por flanco.
2. Verdadero o falso: un latch D se encuentra en su modo transparente cuando $EN = 0$.
3. Verdadero o falso: en un latch D , la entrada D sólo puede afectar a Q cuando $EN = 1$.

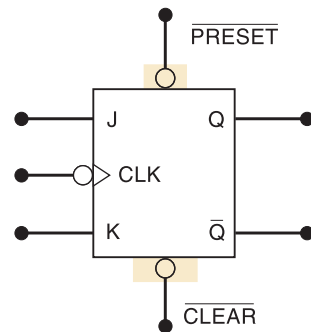
5-10 ENTRADAS ASÍNCRONAS

Para los flip-flops sincronizados por reloj que hemos estudiado, las entradas S , R , J , K y D se han designado como entradas de *control*. A estas entradas también se les conoce como entradas síncronas, ya que su efecto sobre la salida del FF está sincronizado con la entrada CLK . Como hemos visto, las entradas de control síncronas deben usarse en conjunto con una señal de reloj para disparar el FF.

La mayoría de los FFs sincronizados por reloj tienen también una o más **entradas asíncronas** que operan de manera independiente a las entradas síncronas y a la entrada del reloj. Estas entradas asíncronas pueden usarse para establecer el FF al estado 1 o borrar (RESET) el FF al estado 0 *en cualquier momento, sin importar las condiciones en las otras entradas*. Dicho de otra forma, las entradas asíncronas son **entradas predominantes**, las cuales pueden usarse para ignorar todas las demás entradas y colocar el FF en un estado u otro.

La figura 5-31 muestra un flip-flop J-K con dos entradas asíncronas designadas como $\overline{\text{PRESET}}$ y $\overline{\text{CLEAR}}$, que son entradas activas en BAJO, como lo indican las burbujas en el símbolo del FF. La tabla de funciones que se muestra sintetiza la manera en que afectan a la salida del FF. Vamos a examinar los diversos casos.

FIGURA 5-31 Flip-flop sincronizado por reloj en J-K con entradas asíncronas.



J	K	Clk	$\overline{\text{PRE}}$	$\overline{\text{CLR}}$	Q
0	0	↓	1	1	Q (sin cambio)
0	1	↓	1	1	0 (RESET síncrona)
1	0	↓	1	1	1 (SET síncrona)
1	1	↓	1	1	$\bar{Q}$ (Conmutación síncrona)
x	x	x	1	1	Q (sin cambio)
x	x	x	1	0	0 (CLEAR asíncrona)
x	x	x	0	1	1 (PRESET síncrona)
x	x	x	0	0	(Inválida)

- $\overline{\text{PRESET}} = \overline{\text{CLEAR}} = 1$. Las entradas asíncronas están inactivas y el FF es libre de responder a las entradas J , K y CLK ; en otras palabras, puede llevarse a cabo la operación con sincronización por reloj.
- $\overline{\text{PRESET}} = 0$; $\overline{\text{CLEAR}} = 1$. La señal $\overline{\text{PRESET}}$ está activa y Q se establece en forma *inmediata* en 1, sin importar qué condiciones estén presentes en las entradas J , K y CLK . La entrada CLK no puede afectar al FF mientras que $\overline{\text{PRESET}} = 0$.
- $\overline{\text{PRESET}} = 1$; $\overline{\text{CLEAR}} = 0$. La señal $\overline{\text{CLEAR}}$ se activa y Q se borra *de inmediato* para quedar en 0, sin importar las condiciones en las entradas J , K o CLK . La entrada CLK no tiene efecto mientras que $\overline{\text{CLEAR}} = 0$.
- $\overline{\text{PRESET}} = \overline{\text{CLEAR}} = 0$. Esta condición no debe utilizarse, ya que puede producir una respuesta ambigua.

Es importante tener en cuenta que estas entradas asíncronas responden a los niveles de corriente directa. Esto significa que si se mantiene un 0 constante en la entrada $\overline{\text{PRESET}}$, el FF permanecerá en el estado $Q = 1$ sin importar lo que ocurra en las otras entradas. De manera similar, un BAJO constante en la entrada $\overline{\text{CLEAR}}$ mantendrá al FF en el estado $Q = 0$. Por ende, las entradas asíncronas pueden utilizarse para mantener el FF en un estado específico durante cualquier intervalo de tiempo deseado. No obstante, es más común utilizar las entradas asíncronas para establecer o borrar el FF para que quede en el estado deseado, mediante la aplicación de un pulso momentáneo.

Muchos FFs sincronizados por reloj que están disponibles como CIs tendrán ambas entradas asíncronas; otros sólo tendrán la entrada $\overline{\text{CLEAR}}$. Algunos FFs tendrán entradas asíncronas que sean activas en ALTO, en lugar de ser activas en BAJO. Para estos FFs, el símbolo del FF no tendrá una burbuja en las entradas asíncronas.

Designaciones para las entradas asíncronas

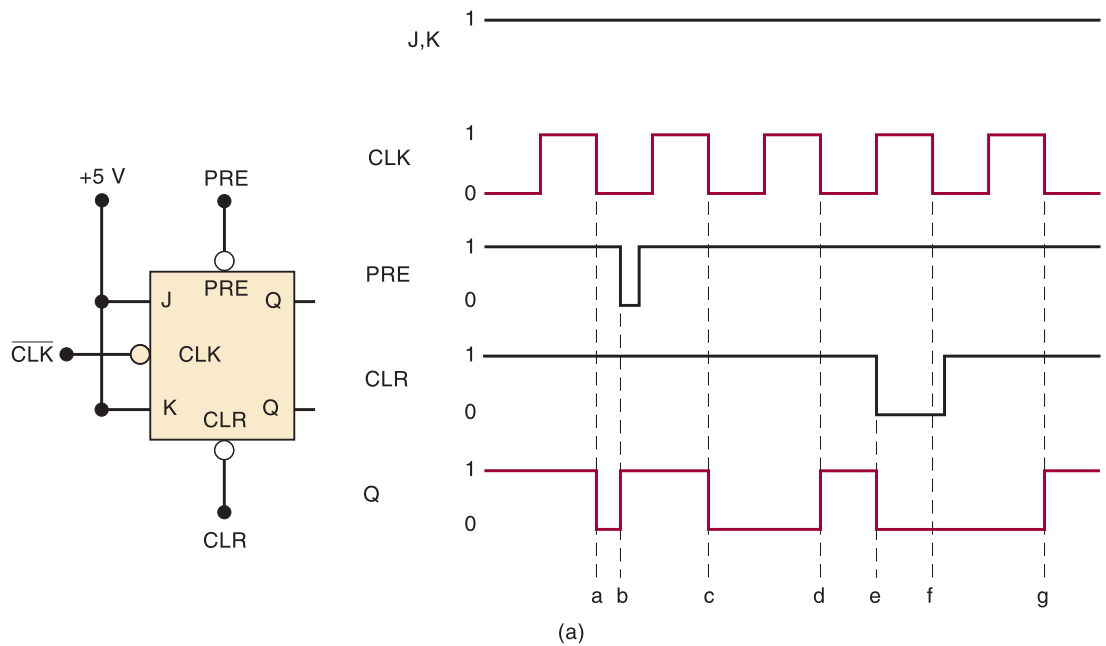
No todos los fabricantes de circuitos integrados están de acuerdo en la nomenclatura a usar para estas entradas asíncronas. Las designaciones más comunes son $\overline{\text{PRE}}$ (abreviación de $\overline{\text{PRESET}}$) y $\overline{\text{CLR}}$ (abreviación de $\overline{\text{CLEAR}}$). Estas etiquetas las

distinguen con claridad de las entradas SET y RESET síncronas. También se utilizan otras etiquetas tales como S_D (SET directo) y R_D (RESET directo). De ahora en adelante, utilizaremos las etiquetas *PRE* y *CLR* para representar las entradas asíncronas, ya que parecen ser las que se utilizan con más frecuencia. Cuando estas entradas asíncronas sean activas en BAJO, como casi siempre, utilizaremos la barra superior para indicar su estado de activas en BAJO; esto es, $\overline{PRE}$ y $\overline{CLR}$.

Aunque la mayoría de los flip-flops de CI tienen cuando menos una o más entradas asíncronas, hay ciertas aplicaciones de circuitos en las que no se utilizan. En tales casos se mantienen de forma permanente en su nivel inactivo. A menudo, en nuestro uso de los FFs a lo largo del resto de este libro, no mostraremos las entradas asíncronas de un FF que no se utilicen; asumiremos que se mantienen conectadas en forma permanente a su nivel lógico inactivo.

EJEMPLO 5-9

La figura 5-32(a) muestra el símbolo para un FF J-K que responde a una NGT en su entrada de reloj y que tiene entradas asíncronas activas en BAJO. Antes de seguir con el ejemplo, tome nota de la manera en que están etiquetadas las entradas. Primero observe que la señal de reloj que se aplica al FF está etiquetada como *CLK* (la barra superior indica que esta señal se activa en la NGT), mientras que en el otro lado de la burbuja (dentro del bloque) se etiqueta como $\overline{CLK}$. De igual forma,



Punto	Operación
a	Conmutación síncrona en NGT de $\overline{CLK}$
b	SET asíncrona en $\overline{PRE} = 0$
c	Conmutación síncrona
d	Conmutación síncrona
e	CLEAR asíncrona en $\overline{CLR} = 0$
f	$\overline{CLR}$ predomina sobre la NGT de $\overline{CLK}$
g	Conmutación síncrona

(b)



FIGURA 5-32 Formas de onda para el ejemplo 5-9 que muestran cómo responde un flip-flop sincronizado por reloj a sus entradas asíncronas.

las entradas asíncronas externas activas en BAJO se etiquetan como $\overline{PRE}$ y $\overline{CLR}$, mientras que dentro del bloque al otro lado de la burbuja se etiquetan como PRE y CLR . El punto importante a recordar es que la presencia de la burbuja en una entrada indica que esa entrada responde a una señal de nivel BAJO.

Las entradas J y K se muestran enlazadas al nivel ALTO en este ejemplo. Determine la salida Q en respuesta a las formas de onda de entrada que se muestran en la figura 5-32(a), Suponga que al principio Q está en ALTO.

Solución

Al principio, $\overline{PRE}$ y $\overline{CLR}$ se encuentran en su estado ALTO inactivo, por lo que no tendrán efecto sobre Q . Por ende, cuando ocurra la primera NGT de la señal $\overline{CLK}$ en el punto a , Q conmutará a su estado opuesto; recuerde que $J = K = 1$ produce la operación de conmutación.

En el punto b , se aplica un pulso a la entrada $\overline{PRE}$ para que cambie a su estado de activa en BAJO. Esto hará que $Q = 1$ de inmediato. Observe que $\overline{PRE}$ produce $Q = 1$ sin esperar una NGT en $\overline{CLK}$. Las entradas asíncronas operan en forma independiente de $\overline{CLK}$.

En el punto c , la NGT de $\overline{CLK}$ hará de nuevo que Q conmute a su estado opuesto. Observe que $\overline{PRE}$ ha regresado a su estado inactivo antes del punto c . De igual forma, la NGT de $\overline{CLK}$ en el punto d conmutará a Q de vuelta al nivel ALTO.

En el punto e , se aplica un pulso a la entrada $\overline{CLR}$ para que cambie a su estado de activa en BAJO y $Q = 0$ de inmediato. De nuevo, esto se hace en forma independiente de $\overline{CLK}$.

La NGT de $\overline{CLK}$ en el punto f no conmutará a Q , ya que la entrada $\overline{CLR}$ aún está activa. El nivel BAJO en $\overline{CLR}$ predomina sobre la entrada $\overline{CLK}$ y mantiene a $Q = 0$.

Cuando ocurra la NGT de $\overline{CLK}$ en el punto g , conmutará a Q al estado ALTO, ya que ninguna de las entradas asíncronas está activa en ese punto.

Estos pasos se sintetizan en la figura 5-32(b).

PREGUNTAS DE REPASO

1. ¿Cómo difiere la operación de una entrada asíncrona en comparación con una entrada síncrona?
2. ¿Puede un flip-flop D responder a sus entradas D y CLK mientras $\overline{PRE} = 1$?
3. Liste las condiciones necesarias para un flip-flop J-K disparado por flanco positivo, con entradas asíncronas activas en BAJO para conmutarlo a su estado opuesto.

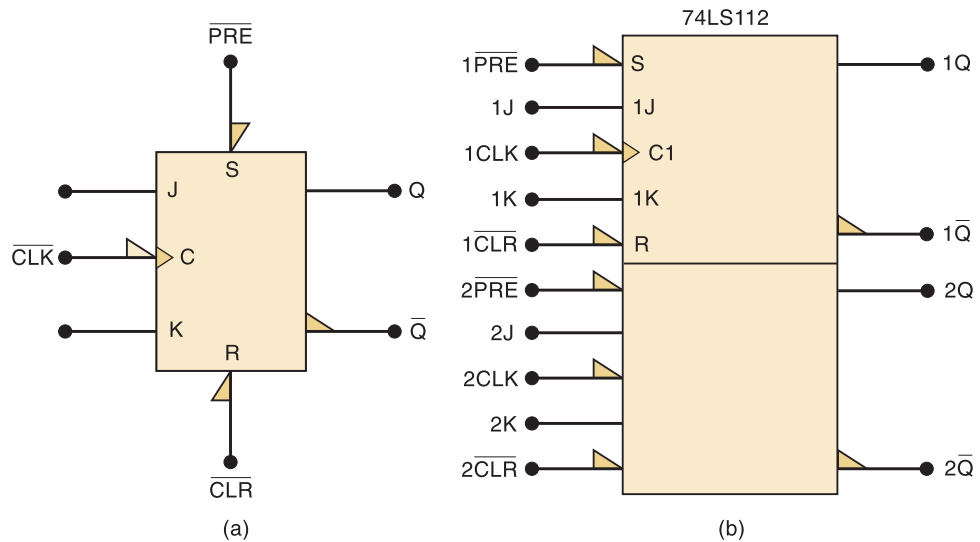
5-11 SÍMBOLOS IEEE/ANSI

La figura 5-33(a) muestra el símbolo IEEE/ANSI para un flip-flop J-K disparado por flanco con entradas asíncronas. Observe el triángulo recto en la entrada CLK para indicar que se activa mediante una NGT. Recuerde que en los símbolos IEEE/ANSI un triángulo recto tiene el mismo significado que la pequeña burbuja en los símbolos tradicionales. Observe además que la entrada de reloj se etiqueta como “C” para denotar cualquier entrada que *controla* cuando otras entradas van a afectar a la salida. Las entradas $\overline{PRE}$ y $\overline{CLR}$ son activas en BAJO, según lo indican los triángulos rectos en estas entradas. Los símbolos IEEE/ANSI también utilizan las etiquetas “S” y “R” dentro del rectángulo para denotar las operaciones SET y RESET asíncronas, que son las mismas que PRESET y CLEAR respectivamente.

La figura 5-33(b) muestra el símbolo lógico IEEE/ANSI para un CI que forma parte de la serie 74LS de dispositivos TTL. El 74LS112 es un CI con dos flip-flop J-K disparados por flanco negativo, con capacidades de preestablecer (PRESET) y borrar (CLEAR). Contiene dos flip-flops J-K, igual al que se simboliza en la figura 5-33(a).

FIGURA 5-33

Símbolos IEEE/ANSI para (a) un solo flip-flop J-K disparado por flanco y (b) un CI real (dos flip-flop J-K disparados por flanco negativo 74LS112).

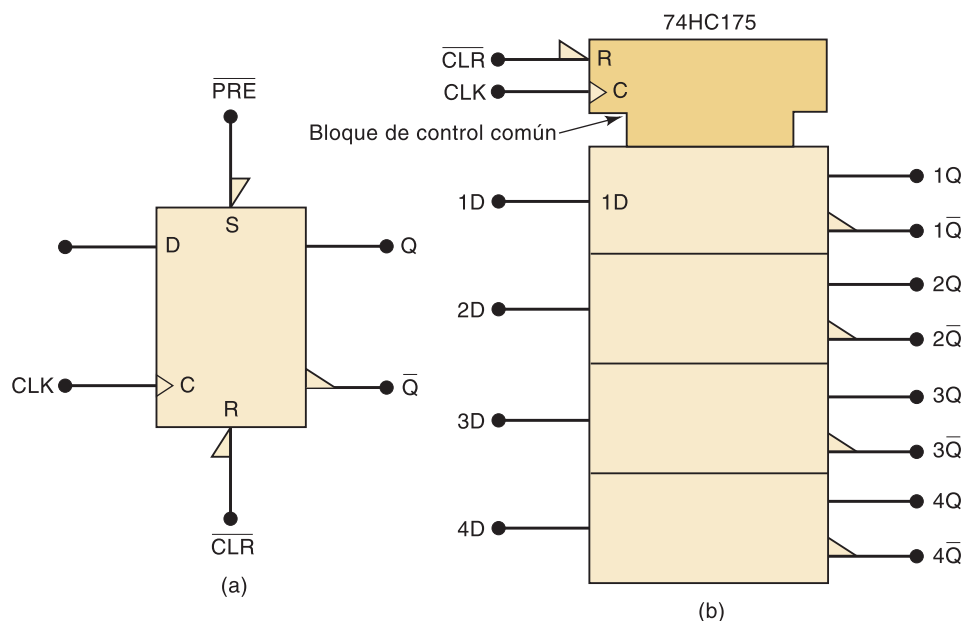


Observe cómo las entradas y salidas están numeradas. Observe además que las etiquetas de las entradas dentro de los rectángulos se muestran sólo para el FF superior. Se sobreentiende que las entradas para el FF inferior tienen el mismo arreglo que el FF superior. Este mismo símbolo de CI se aplica para el CMOS 74HC112.

La figura 5-34(a) es el símbolo IEEE/ANSI para un flip-flop D disparado por flanco positivo con entradas asíncronas. No hay un triángulo recto en la entrada del reloj debido a que este FF se sincroniza mediante las PGTs del reloj.

FIGURA 5-34

Símbolos IEEE/ANSI para (a) un solo flip-flop D disparado por flanco y (b) un CI actual (flip-flop cuádruple 74HC175 con señales de reloj y CLEAR comunes).



La figura 5-34(b) es el símbolo IEEE/ANSI para un CI 74HC175, el cual contiene cuatro flip-flops D que comparten una entrada CLK común y una entrada CLR común. Los FFs no tienen entrada $\overline{PRE}$. Este símbolo contiene un rectángulo separado para representar cada FF y un **bloque de control común** especial, identificado por el rectángulo con muescas en la parte superior. El bloque de control común se utiliza siempre que un CI tiene una o más entradas que son comunes para más de uno de los circuitos dentro del chip. Para el 74HC175, las entradas CLK y CLR son

comunes para los cuatro flip-flops D en el CI. Esto significa que una PGT en CLK hará que cada salida Q tome el nivel presente en su entrada D ; también significa que un nivel BAJO en CLR borrará todas las salidas Q para que queden en el estado BAJO.

PREGUNTAS DE REPASO

1. Explique el significado de los dos triángulos distintos que pueden formar parte de la simbología IEEE/ANSI en una entrada de reloj.
2. Describa el significado del bloque de control común.

5-12 CONSIDERACIONES DE SINCRONIZACIÓN DE LOS FLIP-FLOPS

Los fabricantes de flip-flops en circuitos integrados especificarán varios parámetros de sincronización y características importantes que deben considerarse antes de utilizar un FF en cualquier aplicación de circuitos. Describiremos las más importantes y después proporcionaremos algunos ejemplos reales de flip-flops en CIs específicos de las familias lógicas TTL y CMOS.

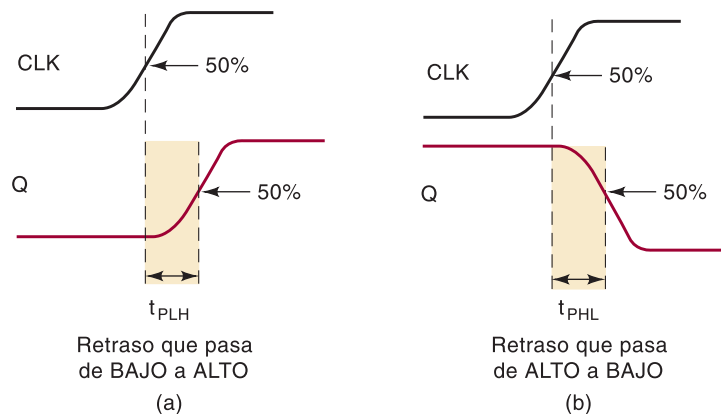
Tiempos de estabilización y de retención

Ya hemos hablado sobre estos tiempos, y tal vez recuerde de la sección 5-5 que representan requerimientos que deben cumplirse para que un FF pueda dispararse en forma confiable. La hoja técnica del fabricante del CI siempre especificará los valores *mínimos* para t_S y t_H .

Tiempos de propagación

Cada vez que una señal va a cambiar el estado de la salida de un FF, hay un retraso a partir del tiempo en el que se aplica la señal hasta el tiempo en el que la salida realiza su cambio. La figura 5-35 ilustra los **retrasos por tiempos de propagación** que se producen en respuesta a una transición positiva en la entrada CLK . Observe que estos retrasos se miden entre los puntos del 50 por ciento en las formas de onda de entrada y de salida. Los mismos tipos de retrasos ocurren en respuesta a las señales en las entradas asíncronas de un FF (PRESET y CLEAR). Las hojas técnicas de los fabricantes, por lo general, especifican los tiempos de propagación en respuesta a todas las entradas, y es común que especifiquen los valores *máximos* para t_{PLH} y t_{PHL} .

FIGURA 5-35 Tiempos de propagación de un FF.



Los flip-flops de CI modernos tienen tiempos de propagación que varían desde unos cuantos nanosegundos hasta un valor aproximado de 100 ns. Por lo general, los valores de t_{PLH} y t_{PHL} no son iguales, y se incrementan en proporción directa al número de cargas que controla la salida Q . Los tiempos de propagación de los FFs juegan un papel importante en ciertas situaciones que veremos más adelante.

Frecuencia máxima de reloj, f_{MAX}

Ésta es la frecuencia más alta que puede aplicarse a la entrada CLK de un FF y que éste pueda dispararse en forma confiable. El límite de f_{MAX} variará de un FF a otro, incluso entre FFs que tengan el mismo número de dispositivo. Por ejemplo, el fabricante del CI 7470 con flip-flops J-K prueba muchos de estos FFs y puede llegar a encontrar que los valores de f_{MAX} caen en el intervalo de 20 a 35 MHz. Entonces especificará el valor *mínimo* de f_{MAX} como de 20 MHz. Esto puede parecer confuso, pero un poco de análisis nos dejará claro que lo que el fabricante está diciendo es que no puede garantizar que el FF 7470 que usted ponga en su circuito vaya a trabajar por encima de 20 MHz; la mayoría de ellos lo hará pero algunos no. No obstante, si usted los opera por debajo de 20 MHz, el fabricante garantiza que todos funcionarán.

Tiempos de pulso de reloj en ALTO y en BAJO

El fabricante también especificará la *mínima* duración de tiempo que la señal CLK debe permanecer en BAJO antes de cambiar a ALTO; a esto algunas veces se le conoce como $t_{W(L)}$. Al tiempo *mínimo* que CLK debe permanecer en ALTO antes de regresar a BAJO se le conoce algunas veces como $t_{W(H)}$. Estos tiempos se definen en la figura 5-36(a). No cumplir con estos requerimientos de tiempo mínimo puede provocar un disparo no confiable. Observe que estos valores de tiempo se miden entre los puntos intermedios en las transiciones de la señal.

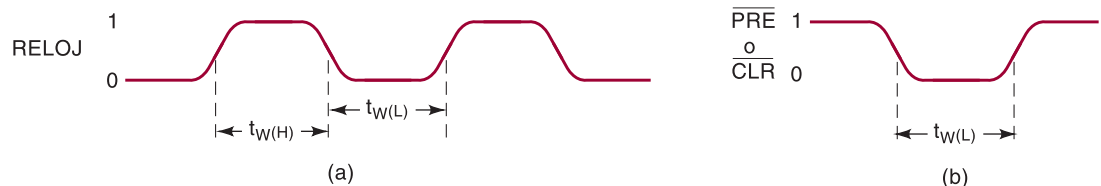


FIGURA 5-36 (a) Tiempos para los niveles BAJO y ALTO del reloj; (b) anchura de pulso asíncrono.

Anchura de pulso activo asíncrono

El fabricante también especificará la *mínima* duración de tiempo que debe mantenerse una entrada PRESET o CLEAR en su estado activo, para poder establecer o borrar el FF en forma confiable. La figura 5-36(b) muestra el valor de $t_{W(L)}$ para las entradas asíncronas activas en BAJO.

Tiempos de transición del reloj

Para un disparo confiable, los tiempos de transición de la forma de onda del reloj (tiempos de subida y de caída) deben mantenerse muy cortos. Si la señal del reloj tarda demasiado en hacer las transiciones de un nivel al otro, el FF podría dispararse en forma errática o tal vez no se dispararía. Por lo general, los fabricantes no listan un requerimiento de tiempo de transición mínimo para cada circuito integrado de FF. En vez de ello se proporciona como un requerimiento general para todos los CIs dentro de una familia lógica dada. Por ejemplo, los tiempos de transición deben ser, por lo general, ≤ 50 ns para los dispositivos TTL y ≤ 200 ns para los dispositivos CMOS. Estos requerimientos variarán entre los distintos fabricantes y entre las diversas subfamilias dentro de las extensas familias lógicas TTL y CMOS.

CIs reales

Como ejemplos prácticos de estos parámetros de sincronización, veamos un análisis de varios FFs de circuitos integrados reales. En especial veremos los siguientes CIs:

7474	Dos Flip-flop D, disparados por flanco (TTL estándar).
74LS112	Dos Flip-flop J-K, disparados por flanco (Schottky de baja potencia TTL).
74C74	Dos Flip-flop D, disparados por flanco (CMOS de compuerta metálica).
74HC112	Dos Flip-flop J-K, disparados por flanco (CMOS de alta velocidad).

La tabla 5-2 lista los diversos valores de sincronización para cada uno de estos FFs, a medida que aparecen en los libros de datos del fabricante. Todos los valores listados son *mínimos*, excepto para los tiempos de propagación, que son valores *máximos*. El análisis de la tabla 5-2 nos revela varios puntos interesantes.

TABLA 5-2
Valores de sincronización de los flip-flops (en nanosegundos)

	TTL		CMOS	
	7474	74LS112	74C74	74HC112
t_S	20	20	60	25
t_H	5	0	0	0
t_{PHL} de CLK a Q	40	24	200	31
t_{PLH} de CLK a Q	25	16	200	31
t_{PHL} de $\overline{CLR}$ a Q	40	24	225	41
t_{PLH} de $\overline{PRE}$ a Q	25	16	225	41
$t_{W(L)}$ tiempo en BAJO para CLK	37	15	100	25
$t_{W(H)}$ Tiempo en ALTO para CLK	30	20	100	25
$t_{W(L)}$ en $\overline{PRE}$ o $\overline{CLR}$	30	15	60	25
$f_{MÁX}$ en MHz	15	30	5	20

1. Todos los FFs tienen requerimientos muy bajos para t_H ; esto es común en la mayoría de los FFs modernos disparados por flanco.
2. La serie 74HC de dispositivos CMOS tiene valores de sincronización que se comparan con los de los dispositivos TTL. La serie 74C es mucho más lenta que la serie 74HC.

EJEMPLO 5-10

De la tabla 5-2 determine lo siguiente.

- (a) Suponga que $Q = 0$. ¿Cuánto tiempo puede tardar Q en cambiar a ALTO cuando ocurre una PGT en la entrada CLK de un 7474?
- (b) Suponga que $Q = 1$. ¿Cuánto tiempo puede tardar Q en cambiar a BAJO, en respuesta a la entrada $\overline{CLR}$ de un 74HC112?
- (c) ¿Cuál es el pulso más estrecho que debe aplicarse a la entrada $\overline{CLR}$ del FF 74LS112 para borrar a Q de manera confiable?
- (d) ¿Cuál FF en la tabla 5-2 requiere que las entradas de control permanezcan estables *después* de que ocurra la transición activa del reloj?
- (e) ¿Para cuáles FFs deben mantenerse estables las entradas de control durante un tiempo mínimo, antes de la transición activa del reloj?

Solución

- La PGT hará que Q cambie de BAJO a ALTO. El retraso de CLK a Q se lista como $t_{PLH} = 25$ ns para el 7474.
- Para el 74HC112, el tiempo requerido para que Q cambie de ALTO a BAJO en respuesta a la entrada $\overline{CLR}$ se lista como $t_{PHL} = 41$ ns.
- Para el 74LS112, el pulso más estrecho en la entrada $\overline{CLR}$ se lista como $t_W(L) = 15$ ns.
- El 7474 es el único FF en la tabla 5-2 que tiene un requerimiento de tiempo de retención distinto de cero.
- Todos los FFs tienen un requerimiento de tiempo de estabilización distinto de cero.

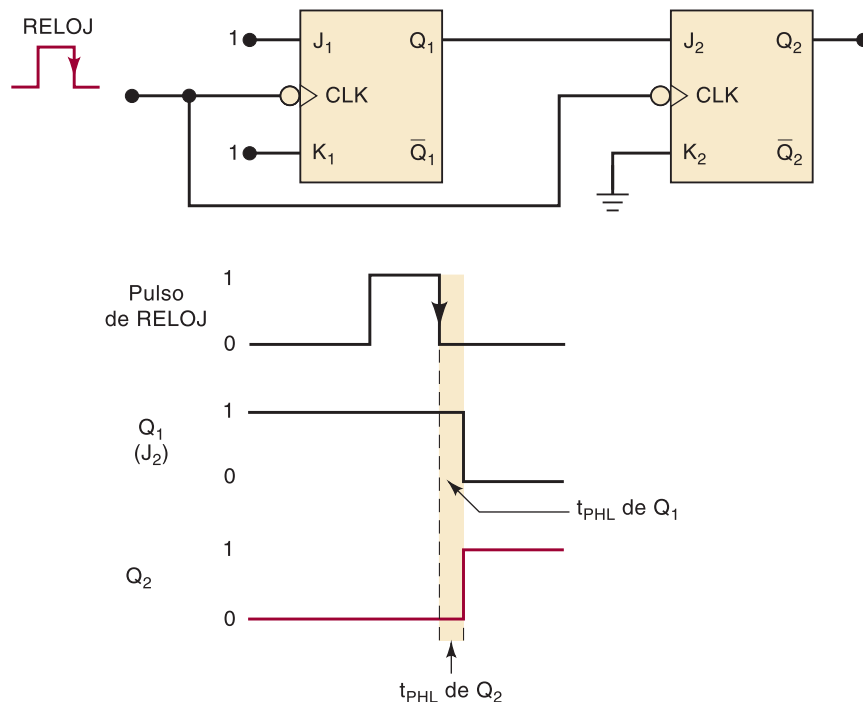
PREGUNTAS DE REPASO

- ¿Cuáles parámetros de sincronización de los FFs indican el tiempo que tarda la salida Q en responder a una entrada?
- Verdadero o falso:* un FF que tiene una clasificación de f_{MAX} de 25 MHz puede dispararse en forma confiable mediante cualquier forma de onda de pulso en CLK con una frecuencia menor a 25 MHz.

5-13 PROBLEMA POTENCIAL DE SINCRONIZACIÓN EN CIRCUITOS CON FF

En muchos circuitos digitales, la salida de un FF se conecta ya sea en forma directa o a través de compuertas lógicas a la entrada de otro FF, y ambos FFs se disparan mediante la misma señal de reloj. Esto presenta un problema potencial de sincronización. Una situación común se ilustra en la figura 5-37, en donde la salida de Q_1 se conecta a la entrada J de Q_2 y ambos FFs se sincronizan por la misma señal en sus entradas CLK .

FIGURA 5-37 Q_2 responderá en forma apropiada al nivel presente en Q_1 antes de la NGT de CLK , siempre y cuando el requerimiento de tiempo de retención de Q_2 (t_H) sea menor que el tiempo de propagación de Q_1 .



El problema potencial de sincronización es el siguiente: como Q_1 cambiará en la NGT del pulso de reloj, la entrada J_2 de Q_2 cambiará cuando reciba la misma NGT. Esto podría producir una respuesta impredecible en Q_2 .

Vamos a suponer que al principio $Q_1 = 1$ y $Q_2 = 0$. Por ende, el FF de Q_1 tiene $J_1 = K_1 = 1$ y Q_2 tiene $J_2 = Q_1 = 1$, $K_2 = 0$ antes de la NGT del pulso de reloj. Cuando se produce la NGT, Q_1 conmutará al estado BAJO pero no cambiará a BAJO sino hasta después de su retraso de propagación t_{PHL} . La misma NGT cambiará de forma confiable a Q_2 a su estado ALTO siempre y cuando t_{PHL} sea mayor que el requerimiento de tiempo de retención de Q_2 , t_H . Si no se cumple esta condición, la respuesta de Q_2 será impredecible.

Por fortuna, todos los FFs modernos disparados por flanco tienen requerimientos de tiempo de retención de 5 ns o menos; la mayoría tiene $t_H = 0$, lo cual significa que no tienen requerimiento de tiempo de retención. Para estos FFs, las situaciones como la de la figura 5-37 no serán problema.

A menos que se indique lo contrario, en todos los circuitos de FF que nos encontremos a lo largo del libro supondremos que el requerimiento de tiempo de retención del FF es lo bastante corto como para responder en forma confiable, de acuerdo con la siguiente regla:

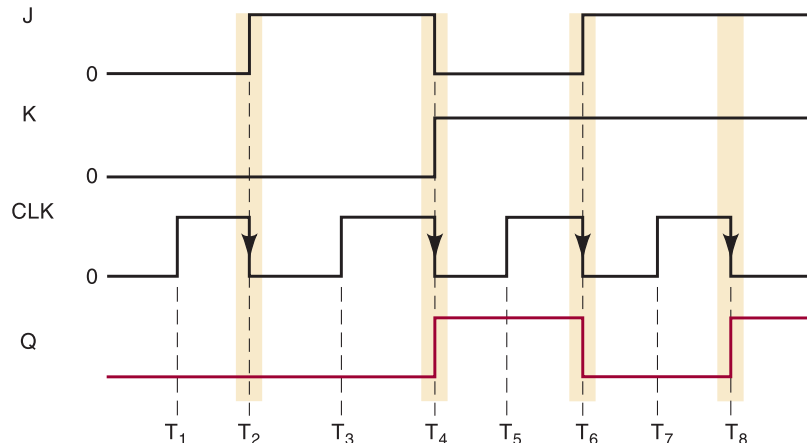
La salida del FF cambiará a un estado determinado por los niveles lógicos presentes en sus entradas de control síncronas justo antes de la transición activa del reloj.

Si aplicamos esta regla a la figura 5-37, nos dice que Q_2 cambiará a un estado determinado por la condición $J_2 = 1$, $K_2 = 0$, que esté presente justo antes de la NGT del pulso de reloj. El hecho de que J_2 esté cambiando en respuesta a la misma NGT no tiene efecto.

EJEMPLO 5-11

Determine la salida Q para un flip-flop J-K disparado por flanco negativo, para las formas de onda de entrada que se muestran en la figura 5-38. Suponga que al principio $t_H = 0$ y que $Q = 0$.

FIGURA 5-38 Ejemplo 5-11.



Solución

El FF responderá sólo en los tiempos T_2 , T_4 , T_6 y T_8 . En T_2 , Q responderá a la condición $J = K = 0$ presente justo antes de T_2 . En T_4 , Q responderá a la condición $J = 1$, $K = 0$ presente justo antes de T_4 . En T_6 , Q responderá a la condición $J = 0$, $K = 1$ presente justo antes de T_6 . Q responde a $J = K = 1$.

5-14 APLICACIONES DE LOS FLIP-FLOPS

Los flip-flops disparados por flanco (sincronizados por reloj) son dispositivos versátiles que pueden usarse en una amplia variedad de aplicaciones, incluyendo el conteo, el almacenamiento de datos binarios, la transferencia de datos binarios desde un lugar hacia otro y muchas más. Casi todas estas aplicaciones utilizan la operación sincronizada del FF. Muchas de ellas entran en la categoría de los **circuitos secuenciales**. En un circuito secuencial las salidas siguen una secuencia predeterminada de estados, en donde un nuevo estado ocurre cada vez que se produce un pulso de reloj. En las siguientes secciones presentaremos algunas de las aplicaciones básicas y las veremos con más detalle en los siguientes capítulos.

5-15 SINCRONIZACIÓN DE LOS FLIP-FLOPS

La mayoría de los sistemas digitales son principalmente síncronos en su operación, ya que la mayoría de las señales cambiará de estado en sincronía con las transiciones del reloj. No obstante, en muchos casos habrá una señal externa que no esté sincronizada con el reloj; en otras palabras, será asíncrona. Las señales asíncronas se presentan con frecuencia, como resultado de la participación de un operador humano que activa un interruptor de entrada durante cierto tiempo aleatorio, relativo a la señal del reloj. Esta característica aleatoria puede producir resultados impredecibles e indeseables. El siguiente ejemplo ilustra cómo puede usarse un FF para sincronizar el efecto de una entrada asíncrona.

EJEMPLO 5-12

La figura 5-39(a) muestra una situación en donde la señal de entrada *A* se genera a partir de un interruptor sin rebotes controlado por un operador (en el ejemplo 5-2 se introdujo por primera vez un interruptor sin rebotes). *A* cambia a ALTO cuando el operador activa el interruptor y cambia a BAJO cuando el operador libera el interruptor. Esta entrada *A* se utiliza para controlar el paso de la señal de reloj hacia la compuerta AND, de manera que aparezcan pulsos de reloj en la salida *X* sólo durante el tiempo que *A* esté en ALTO.

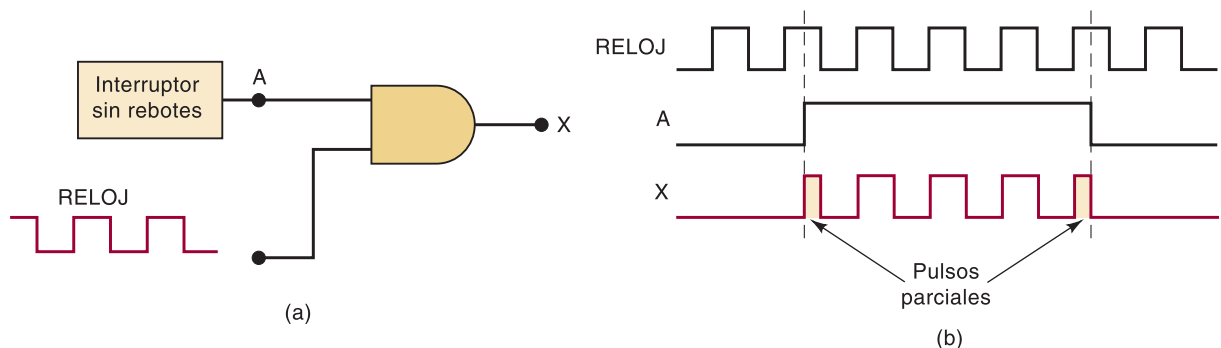
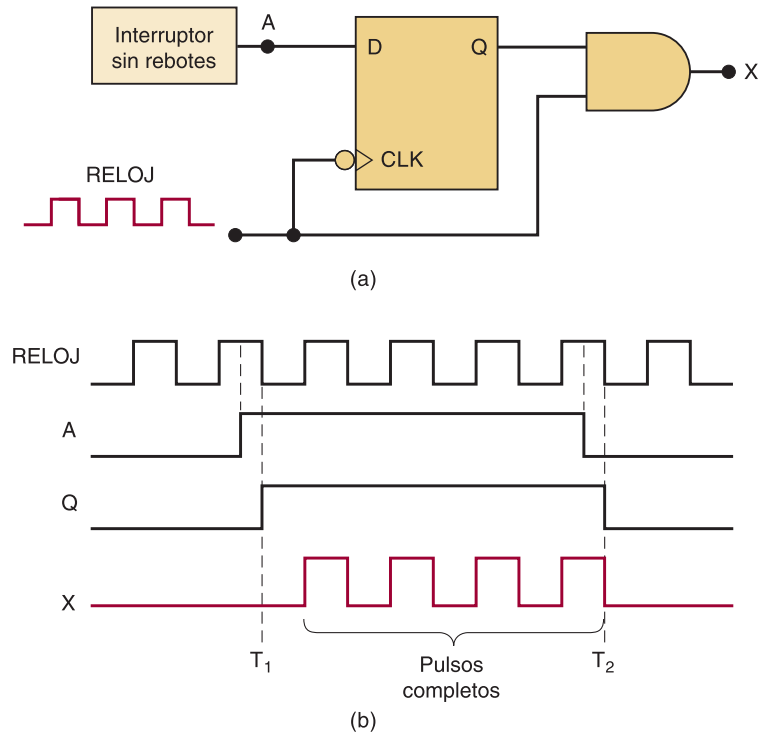


FIGURA 5-39 La señal asíncrona *A* puede producir pulsos parciales en *X*.

El problema con este circuito es que *A* es asíncrona; puede cambiar de estado en cualquier tiempo relativo a la señal del reloj, ya que los tiempos exactos cuando el operador actúa o libera el interruptor son en esencia aleatorios. Esto puede producir pulsos de reloj *parciales* en la salida *X* si cualquiera de las transiciones de *A* ocurre mientras la señal de reloj esté en ALTO, como se muestra en las formas de onda de la figura 5-39(b).

A menudo este tipo de salida no es aceptable, por lo que debemos desarrollar un método para evitar que aparezcan pulsos parciales en *X*. Una solución se muestra en la figura 5-40(a). Describa la manera en que este circuito resuelve el problema y dibuje la forma de onda de *X* para la misma situación que se presenta en la figura 5-39(b).

FIGURA 5-40 Un flip-flop D disparado por flanco se utiliza para sincronizar la habilitación de la compuerta AND con las NGTs del reloj.



Solución

La señal A se conecta a la entrada D del FF Q, el cual está sincronizado por la NGT de la señal del reloj. Por ende, cuando A cambia a ALTO Q cambiará a ALTO hasta la siguiente NGT del reloj en el tiempo T_1 . Este nivel ALTO en Q habilitará a la compuerta AND para que pase los siguientes pulsos de reloj *completos* a X, como muestra la figura 5-40(b).

Cuando A regrese a BAJO, Q no cambiará a BAJO hasta la siguiente NGT del pulso de reloj en T_2 . Por ende, la compuerta AND no inhibirá los pulsos de reloj sino hasta que haya pasado a X el pulso de reloj que termina en T_2 . En consecuencia, la salida X sólo contiene pulsos completos.

Hay un problema potencial con este circuito. Como A podría cambiar a ALTO en cualquier momento, habría una probabilidad aleatoria de que violara los requerimientos de tiempo de estabilización del reloj. En otras palabras, la transición de A podría ocurrir tan cerca del flanco del reloj como para producir una respuesta inestable (deformación) de la salida Q. Para evitar esto se requeriría un circuito de sincronización más complejo.

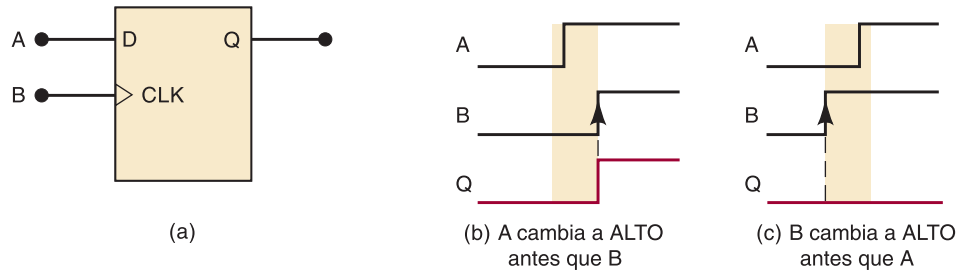
5-16 DETECCIÓN DE UNA SECUENCIA DE ENTRADA

En muchas situaciones una salida debe activarse sólo cuando las entradas se activan en cierta secuencia. Esto no puede lograrse si se utiliza sólo la lógica combinacional, ya que requiere de la característica de almacenamiento de los FFs.

Por ejemplo, una compuerta AND puede utilizarse para determinar cuando dos entradas A y B se encuentran (ambas) en ALTO, pero su salida responderá igual sin importar cuál entrada cambie primero a este estado. Pero suponga que deseamos generar una salida en ALTO sólo si A cambia a ALTO y B lo hace también cierto tiempo después. La figura 5-41(a) muestra una forma de lograr esto.

Las formas de onda de las figuras 5-41(a) y (c) muestran que Q cambiará a ALTO sólo si A cambia a ALTO antes de que B lo haga. Esto se debe a que A primero debe estar en ALTO para que Q pueda cambiar a ALTO con la PGT de B.

FIGURA 5-41 Flip-flop sincronizado por reloj en D, utilizado para responder a una secuencia específica de entradas.



Para que este circuito pueda funcionar en forma apropiada, *A* debe cambiar a ALTO antes que *B* durante cuando menos un lapso igual al requerimiento de tiempo de estabilización del FF.

5-17 ALMACENAMIENTO Y TRANSFERENCIA DE DATOS

Hasta ahora, el uso más común de los flip-flops es para el almacenamiento de datos o información. Los datos pueden representar valores numéricos (por ejemplo, números binarios, números decimales codificados en BCD) o cualquiera de una amplia variedad de tipos de datos que hayan sido codificados en binario. Por lo general, estos datos se almacenan en grupos de FFs, llamados **registros**.

La operación que se realiza con más frecuencia sobre los datos que se almacenan en un FF o un registro es la **transferencia de datos**; lo que implica la transferencia de datos de un FF o registro hacia otro. La figura 5-42 muestra cómo puede lograrse la transferencia de datos entre dos FFs mediante el uso de flip-flops sincronizados por reloj en S-R, J-K y D. En cada caso, el valor lógico que está almacenado en *A* del FF se transfiere a *B* del FF cuando ocurre la NGT del pulso TRANSFERIR. Por ende, después de esta NGT la salida *B* será la misma que la salida *A*.

Las operaciones de transferencia en la figura 5-42 son ejemplos de **transferencia síncrona**, ya que se utilizan las entradas de control y CLK síncronas para realizar la transferencia. También puede obtenerse una operación de transferencia si se utilizan las entradas asíncronas de un FF. La figura 5-43 muestra cómo puede lograrse una **transferencia asíncrona** mediante las entradas PRESET y CLEAR de cualquier tipo de FF.

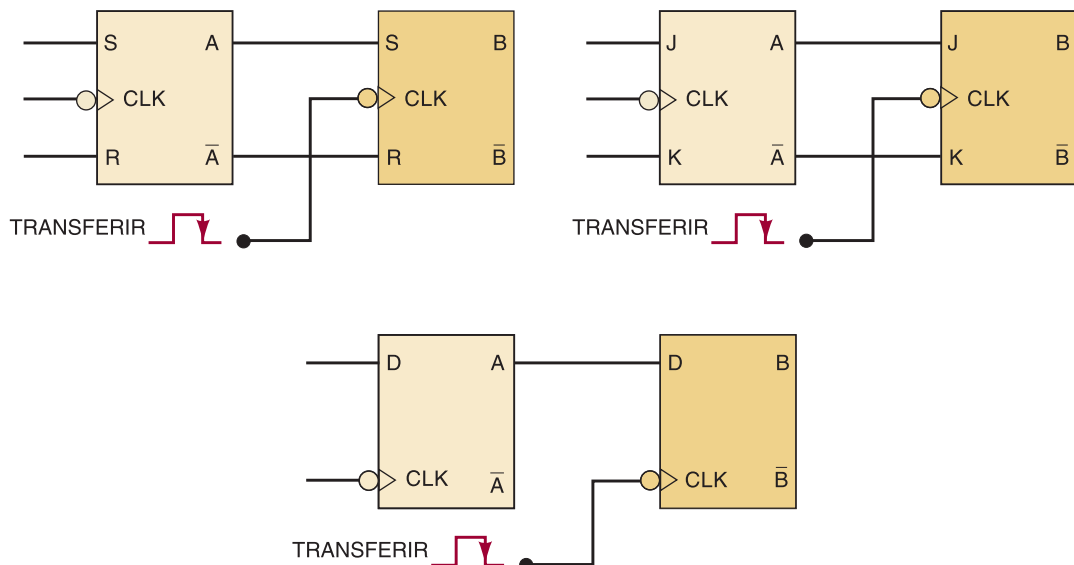
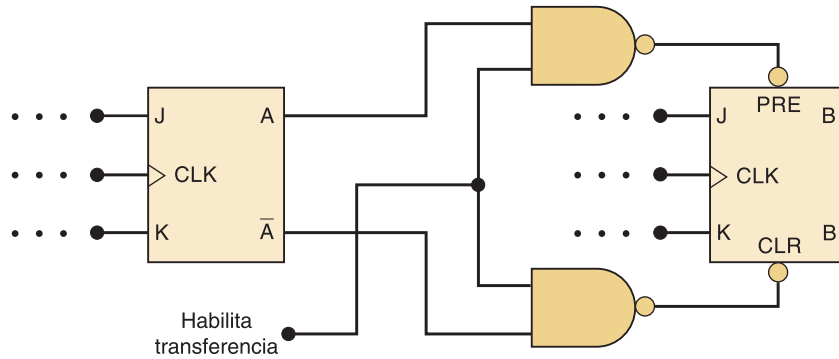


FIGURA 5-42 Operación de transferencia de datos síncrona, realizada por varios tipos de FFs sincronizados por reloj.

FIGURA 5-43

Operación de transferencia de datos asíncrona.



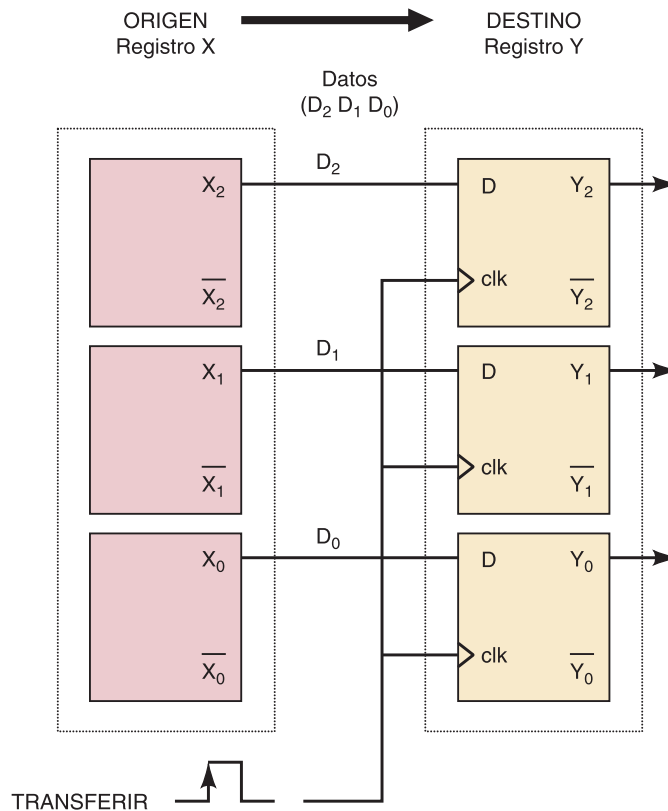
Aquí, las entradas asíncronas responden a los niveles BAJO. Cuando la línea HABILITA TRANSFERENCIA se mantiene en BAJO, las dos salidas de las compuertas NAND se mantienen en ALTO, sin efecto alguno sobre las salidas del FF. Cuando la línea HABILITA TRANSFERENCIA se cambia a ALTO, una de las salidas de las compuertas NAND cambiará a BAJO, dependiendo del estado de las salidas A y $\bar{A}$. Este nivel BAJO establecerá o borrará el FF B para que quede en el mismo estado que el FF A. Esta transferencia asíncrona se realiza de manera independiente de las entradas síncronas y CLK del FF. A la transferencia asíncrona también se le conoce como **transferencia atascada** debido a que los datos pueden “atascarse” en el FF B aún cuando sus entradas síncronas estén activas.

Transferencia de datos en paralelo

La figura 5-44 ilustra la transferencia de datos de un registro a otro mediante el uso de varios FF tipo D. El registro X consiste de los FFs X_2 , X_1 y X_0 ; el registro Y consiste de los FFs Y_2 , Y_1 y Y_0 . Al momento de aplicar la PGT del pulso TRANSFERIR, el nivel almacenado en X_2 se transfiere a Y_2 , X_1 a Y_1 y X_0 a Y_0 . La transferencia del contenido del registro X hacia el registro Y es una transferencia síncrona. También

FIGURA 5-44

Transferencia en paralelo del contenido del registro X hacia el registro Y.



se le conoce como transferencia en paralelo, ya que el contenido de X_2 , X_1 y X_0 se transfiere *de manera simultánea* hacia Y_2 , Y_1 y Y_0 , respectivamente. Si se fuera a realizar una **transferencia de datos en serie**, el contenido del registro X pasaría hacia el registro Y bit por bit. En la siguiente sección examinaremos este proceso.

Es importante comprender que la transferencia en paralelo no cambia el contenido del registro que actúa como el origen de los datos. Por ejemplo, en la figura 5-44 si $X_2X_1X_0 = 101$ y $Y_2Y_1Y_0 = 011$ antes de que ocurra el pulso en TRANSFERIR, entonces ambos registros almacenarán el valor de 101 después del pulso TRANSFERIR.

PREGUNTAS DE REPASO

1. *Verdadero o falso:* la transferencia asíncrona de datos utiliza la entrada CLK .
2. ¿Qué tipo de FF se adapta mejor para la transferencia síncrona debido a que requiere la menor cantidad de interconexiones de un FF al otro?
3. Si se utilizaran flip-flops J-K en los registros de la figura 5-44, en total ¿cuántas interconexiones se requerirían del registro X al registro Y ?
4. *Verdadero o falso:* la transferencia síncrona de datos requiere menos circuitos que la transferencia asíncrona.

5-18 TRANSFERENCIA DE DATOS EN SERIE: REGISTROS DE DESPLAZAMIENTO

Antes de describir la operación de transferencia de datos en serie debemos examinar el arreglo básico de un **registro de desplazamiento**; es decir, un grupo de FFs ordenados de manera que los números binarios almacenados en ellos se desplacen de un FF al siguiente durante cada uno de los pulsos de reloj. Sin duda habrá usted visto en acción los registros de desplazamiento en dispositivos tales como una calculadora electrónica, en la cual los dígitos que se muestran en la pantalla se desplazan un lugar cada vez que usted introduce un nuevo dígito. Es la misma acción que se lleva a cabo en un registro de desplazamiento.

La figura 5-45(a) muestra una manera de ordenar los flip-flops J-K para que operen como un registro de desplazamiento de cuatro bits. Observe que los FFs están conectados de manera que la salida de X_3 se transfiera hacia X_2 hacia X_1 , y X_1 hacia X_0 . Lo que significa que al momento en que ocurre la NGT de un pulso de desplazamiento, cada FF toma el valor almacenado a su izquierda. El flip-flop X_3 toma un valor determinado por las condiciones presentes en sus entradas J y K cuando ocurre la NGT. Por ahora supondremos que las entradas J y K de X_3 se alimentan mediante la forma de onda de DATOS ENT que se muestra en la figura 5-45(b). También supondremos que todos los FFs se encuentran en el estado 0 antes de aplicar los pulsos de desplazamiento.

Las formas de onda en la figura 5-45(b) muestran cómo se desplazan los datos de entrada de izquierda a derecha, de FF en FF a medida que se aplican los pulsos de desplazamiento. Cuando se produce la primera NGT en T_1 , cada uno de los FFs X_2 , X_1 y X_0 tendrán la condición $J = 0$, $K = 1$ presente en sus entradas, debido al estado del FF a su izquierda. El flip-flop X_3 tendrá la condición $J = 1$, $K = 0$ debido a DATOS ENT. Por ende, en T_1 sólo X_3 cambiará a ALTO mientras todos los demás FFs permanecerán en BAJO. Cuando ocurra la segunda NGT en T_2 , el flip-flop X_3 tendrá la condición $J = 0$, $K = 1$ debido a DATOS ENT. El flip-flop X_2 tendrá $J = 1$, $K = 0$ debido al nivel ALTO actual en X_3 . Los flip-flops X_1 y X_0 tendrán todavía la condición $J = 0$, $K = 1$. Por ende, en T_2 sólo el FF X_2 cambiará a ALTO, el FF X_3 cambiará a BAJO y los FFs X_1 y X_0 permanecerán en BAJO.

Podemos usar un razonamiento similar para determinar cómo cambian las formas de onda en T_3 y en T_4 . Observe que en cada NGT de los pulsos de desplazamiento la salida de cada FF toma el nivel que estaba presente en la salida del FF a su izquierda, justo *antes* de la NGT. Desde luego que X_3 toma el nivel que estaba presente en DATOS ENT justo antes de la NGT.

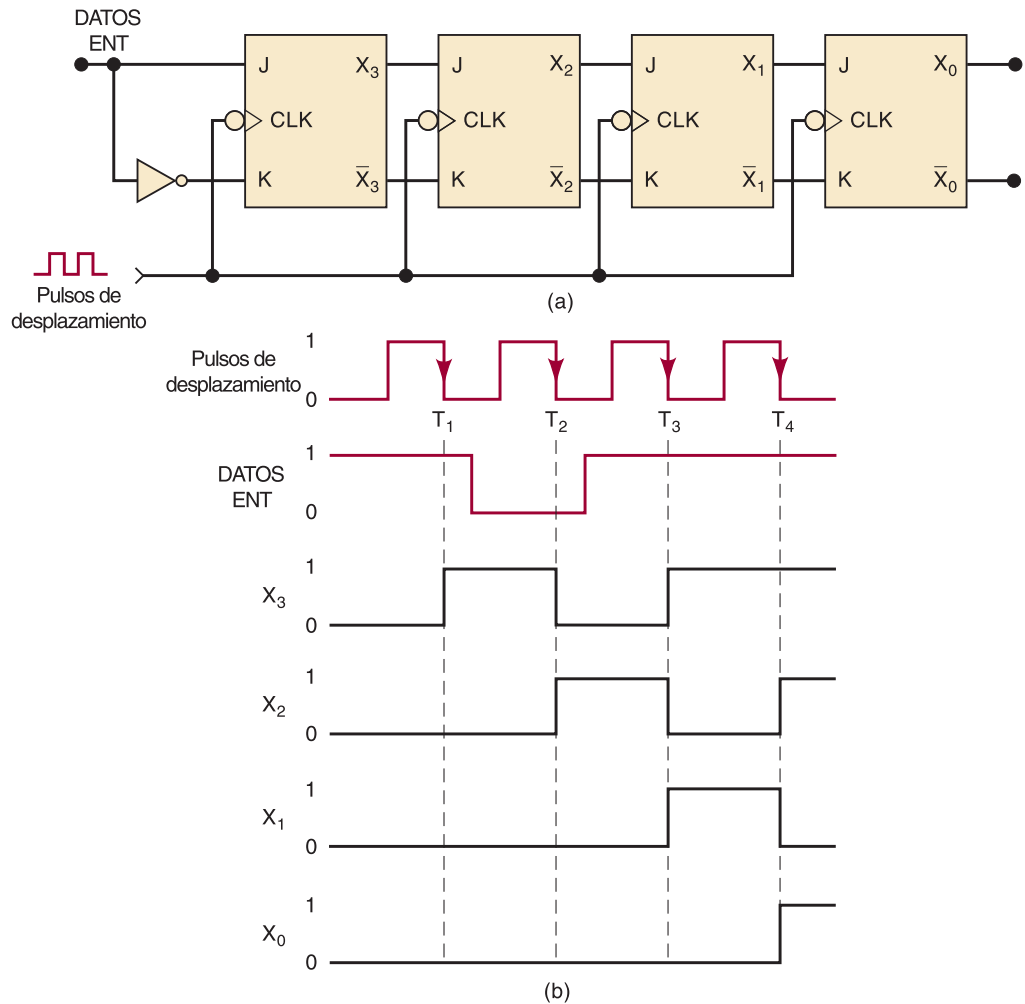


FIGURA 5-45 Registro de desplazamiento de cuatro bits.

Requerimiento de tiempo de retención

En este arreglo de registros de desplazamiento es necesario que los FFs tengan un requerimiento de tiempo de retención muy corto, debido a que hay veces en las que las entradas J , K están cambiando casi al mismo tiempo que la transición en CLK . Por ejemplo, la salida X_3 cambia de 1 a 0 en respuesta a la NGT en T_2 , lo cual provoca que las entradas J , K de X_2 cambien mientras su entrada CLK está cambiando. En realidad, debido al retraso de propagación de X_3 las entradas J , K de X_2 no cambiarán sino hasta un corto tiempo después de la NGT. Por esta razón, un registro de desplazamiento debe implementarse utilizando FFs disparados por flanco que tengan un valor de t_H menor que un retraso de propagación de CLK a la salida. Este último requerimiento lo cumplen con facilidad la mayoría de los FFs disparados por flanco.

Transferencia en serie entre registros

La figura 5-46(a) muestra dos registros de desplazamiento de tres bits, conectados de forma tal que el contenido del registro X se transferirá en serie (se desplazará) hacia el registro Y . Estamos usando flip-flops D para cada registro de desplazamiento, ya que se requieren menos conexiones que con los flip-flops J-K. Observe cómo X_0 , el último FF del registro X , está conectado a la entrada D de Y_2 , el primer FF del registro Y . Por ende, a medida que se aplican los pulsos la transferencia de información se realiza de la siguiente manera: $X_2 \rightarrow X_1 \rightarrow X_0 \rightarrow Y_2 \rightarrow Y_1 \rightarrow Y_0$. El

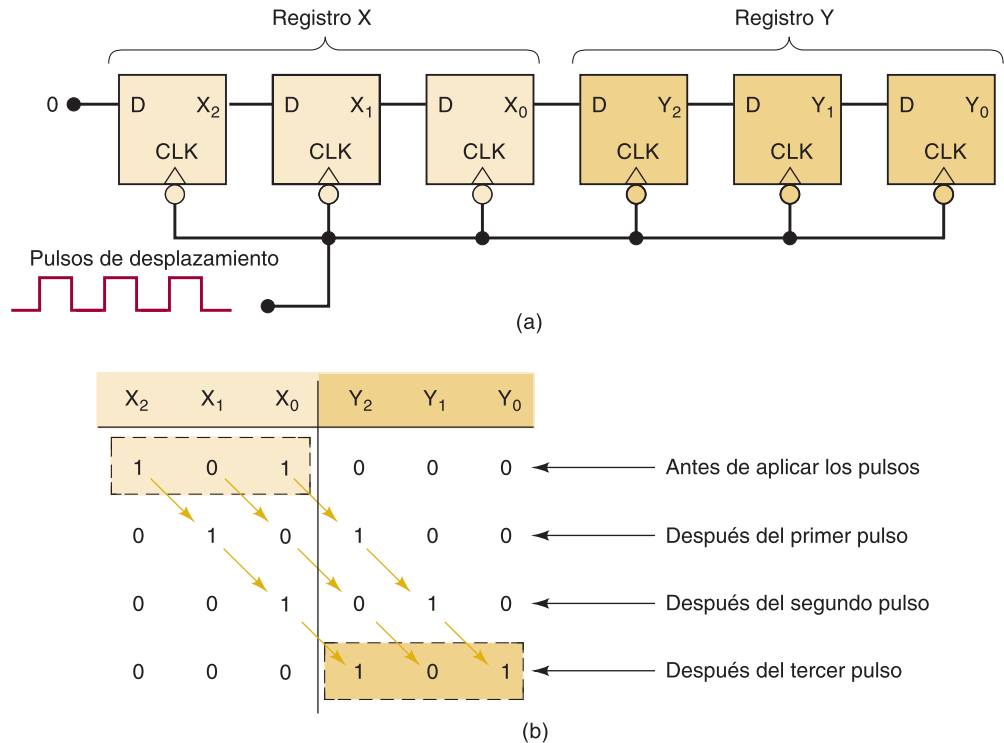


FIGURA 5-46 Transferencia en serie de información, del registro X al registro Y.

FF X_2 cambiará a un estado determinado con base en su entrada D . Por ahora, D se mantendrá en BAJO, de manera que X_2 cambiará a BAJO en el primer pulso y permanecerá ahí.

Para ilustrar lo anterior supongamos que antes de aplicar cualquier pulso de desplazamiento el contenido del registro X es 101 (es decir, $X_2 = 1$, $X_1 = 0$, $X_0 = 1$) y que el registro Y está en 000. Consulte la tabla en la figura 5-46(b), la cual muestra cómo cambian los estados de cada FF a medida que se aplican pulsos de desplazamiento. Hay que recalcar los siguientes puntos:

1. En la NGT de cada pulso, cada FF toma el valor que estaba almacenado en el FF a su izquierda, antes de que ocurriera el pulso.
2. Después de *tres* pulsos, el 1 que estaba al principio en X_2 está ahora en Y_2 , el 0 que estaba al principio en X_1 se encuentra en Y_1 y el 1 que estaba al principio en X_0 está en Y_0 . En otras palabras, el 101 que estaba almacenado en el registro X se ha desplazado ahora hacia el registro Y. El registro X está en 000; ha perdido sus datos originales.
3. La transferencia completa de los *tres* bits de datos requiere *tres* pulsos de desplazamiento.

EJEMPLO 5-13

Asuma el mismo contenido inicial de los registros X y Y en la figura 5-46. ¿Cuál será el contenido de cada FF después de que ocurra el sexto pulso de desplazamiento?

Solución

Si continuamos el proceso que se muestra en la figura 5-46(b) para tres pulsos de desplazamiento más, encontraremos que todos los FFs estarán en el estado 0 después del sexto pulso. Otra forma de llegar a este resultado es razonar de la siguiente manera: el nivel 0 constante en la entrada D de X_2 desplaza un nuevo 0 en cada pulso, de manera que después de seis pulsos los registros se llenan con 0s.

Operación de desplazamiento a la izquierda

Los FFs en la figura 5-46 pueden conectarse con igual facilidad de manera que la información se desplace de derecha a izquierda. No hay alguna ventaja particular de desplazar hacia una dirección o hacia la otra; como veremos más adelante la dirección que elija el diseñador lógico a menudo dependerá de la naturaleza de la aplicación.

Comparación entre transferencia en paralelo y en serie

En la transferencia en paralelo, toda la información se transfiere en forma simultánea cuando ocurre un pulso de comando de transferencia *individual* (figura 5-44), sin importar cuántos bits se estén transfiriendo. En la transferencia en serie, según el ejemplo de la figura 5-46, la transferencia completa de N bits de información requiere N pulsos de reloj (tres bits requieren tres pulsos, cuatro bits requieren cuatro pulsos, etc.). Es obvio que la transferencia en paralelo es mucho más rápida que la transferencia en serie cuando se utilizan registros de desplazamiento.

En la transferencia en paralelo, la salida de cada FF en el registro X está conectada a la correspondiente entrada FF en el registro Y . En la transferencia en serie, sólo el último FF en el registro X está conectado al registro Y . En general, la transferencia en paralelo requiere más interconexiones entre el registro emisor (X) y el receptor (Y) que la transferencia en serie. Esta diferencia se vuelve crítica cuando se transfiere un número mayor de bits de información. Ésta es una importante consideración cuando los registros emisor y receptor están alejados uno del otro, ya que determina cuántas líneas (alambres) se necesitan para la transición de la información.

La elección de transmisión en serie o en paralelo depende de la aplicación de un sistema en particular y de sus especificaciones. A menudo se utiliza una combinación de los dos tipos para aprovechar la *velocidad* de la transferencia en paralelo y la *economía y simpleza* de la transferencia en serie. Más adelante veremos otros detalles acerca de la transferencia de información.

PREGUNTAS DE REPASO

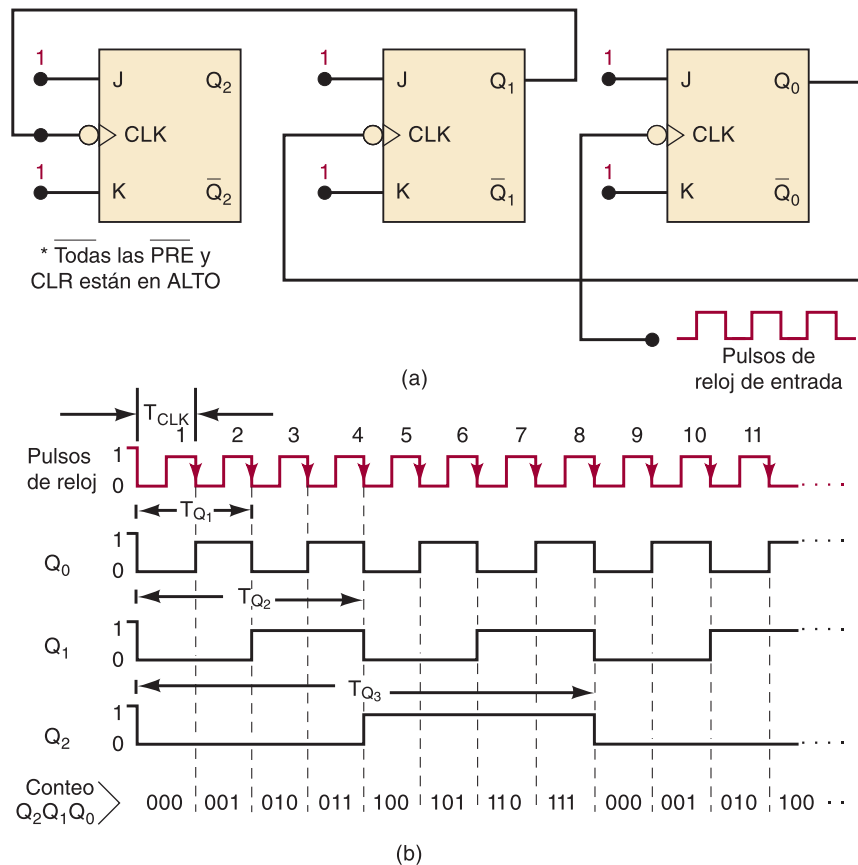
1. *Verdadero o falso*: el método más rápido para transferir datos de un registro a otro es la transferencia en paralelo.
2. ¿Cuál es la principal ventaja de la transferencia en serie, en comparación con la transferencia en paralelo?
3. Consulte la figura 5-46. Suponga que el contenido inicial de los registros es $X_2 = 0$, $X_1 = 1$, $X_0 = 0$, $Y_2 = 1$, $Y_1 = 1$, $Y_0 = 0$. Suponga además que la entrada D de X_2 se mantiene en ALTO. Determine el valor de la salida de cada FF después de que se produce el cuarto pulso de desplazamiento.
4. ¿En cuál de las formas de transferencia de datos el registro de origen no pierde su información?

5-19 DIVISIÓN Y CONTEO DE FRECUENCIA

Consulte la figura 5-47(a). Cada FF tiene sus entradas J y K en el nivel 1, por lo que cambiará de estado (conmutará) cada vez que la señal en su entrada CLK cambie de ALTO a BAJO. Los pulsos de reloj sólo se aplican a la entrada CLK del FF Q_0 . La salida Q_0 está conectada a la entrada CLK del FF Q_1 , y la salida Q_1 está conectada a la entrada CLK del FF Q_2 . Las formas de onda de la figura 5-47(b) muestran cómo los FFs cambian de estado a medida que se aplican los pulsos. Hay que recalcar los siguientes puntos importantes:

1. El flip-flop Q_0 conmuta en la transición de pendiente negativa de cada pulso de reloj de entrada. Por ende, la forma de onda de la salida Q_0 tiene una frecuencia exactamente igual a la mitad de la frecuencia del pulso de reloj.

FIGURA 5-47 Flip-flops J-K conectados como un contador binario de tres bits (MOD-8).



- El flip-flop Q_1 conmuta cada vez que la salida Q_0 cambia de ALTO a BAJO. La forma de onda de Q_1 tiene una frecuencia exactamente igual a la mitad de la frecuencia de la salida Q_0 y que, por lo tanto, es igual a un cuarto del valor de la frecuencia del reloj.
- El flip-flop Q_2 conmuta cada vez que la salida Q_1 cambia de ALTO a BAJO. Por ende, la forma de onda de Q_2 tiene la mitad de la frecuencia de Q_1 y en consecuencia, un octavo de la frecuencia del reloj.
- La salida de cada FF es una onda cuadrada (ciclo de trabajo del 50%).

Como se describió antes, cada FF divide la frecuencia de su entrada por 2. De esta forma, si agregáramos un cuarto FF a la cadena, éste tendría una frecuencia igual a $1/16$ de la frecuencia del reloj, y así en lo sucesivo. Si utilizamos el número de FFs apropiados, este circuito podría dividir una frecuencia por cualquier potencia de 2. De manera específica, si se utilizan N flip-flops se producirá una frecuencia de salida del último FF, la cual equivale a $1/2^N$ de la frecuencia de entrada.

A esta aplicación de los flip-flops se le conoce como **división de frecuencia**. Muchas aplicaciones requieren una división de frecuencia. Por ejemplo, su reloj de pulso es sin duda un reloj de “cuarzo”. El término *reloj de cuarzo* significa que se utiliza un cristal de cuarzo para generar una frecuencia de oscilación muy estable. La frecuencia natural resonante del cristal de cuarzo en su reloj tal vez sea de 1 MHz o más. Para poder avanzar el indicador de los “segundos” una vez cada segundo, la frecuencia de oscilación se *divide* entre un valor que producirá una frecuencia de salida de 1 Hz bastante estable y precisa.

Operación de conteo

Además de funcionar como divisor de frecuencia, el circuito de la figura 5-47 también opera como **contador binario**. Para demostrarlo podemos examinar la secuencia

FIGURA 5-48 La tabla de estados del flip-flop muestra la secuencia de conteo binario.

2^2 Q_2	2^1 Q_1	2^0 Q_0	
0	0	0	Antes de aplicar los pulsos del reloj
0	0	1	Después del pulso #1
0	1	0	#2
0	1	1	#3
1	0	0	#4
1	0	1	#5
1	1	0	#6
1	1	1	#7
0	0	0	#8 000
0	0	1	#9
0	1	0	#10
0	1	1	#1
.	.	.	.
.	.	.	.
.	.	.	.

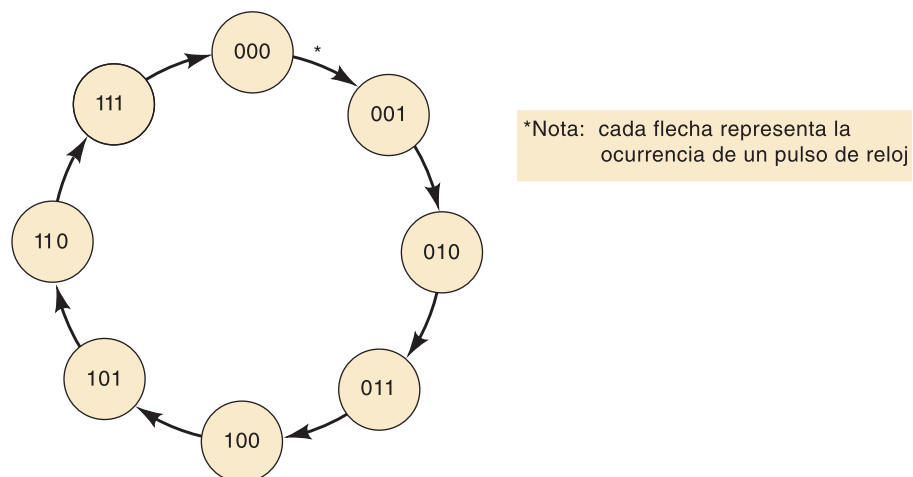
de estados de los FFs después de que ocurre cada pulso de reloj. La figura 5-48 presenta los resultados en una **tabla de estados**. Dejemos que los valores $Q_2Q_1Q_0$ representen un número binario, en donde Q_2 está en la posición 2^2 , Q_1 está en la posición 2^1 y Q_0 está en la posición 2^0 . Los primeros ocho estados de $Q_2Q_1Q_0$ en la tabla deben reconocerse como la secuencia de conteo binario de 000 a 111. Después de la primera NGT, los FFs se encuentran en el estado 001 ($Q_2 = 0$, $Q_1 = 0$, $Q_0 = 1$), la cual representa a 001_2 (que equivale al 1 decimal); después de la segunda NGT los FFs representan el valor 010_2 , que equivale a 2_{10} ; después de tres pulsos, $011_2 = 3_{10}$; después de cuatro pulsos $100_2 = 4_{10}$ y así en lo sucesivo, hasta que después de siete pulsos $111_2 = 7_{10}$. En la octava NGT, los FFs regresan al estado 000, y la secuencia binaria se repite a sí misma para los siguientes pulsos.

Por ende, para los primeros siete pulsos de entrada el circuito funciona como un contador binario, en el cual los estados de los FFs representan a un número binario equivalente al número de pulsos que se han producido. Este contador tiene una capacidad de hasta $111_2 = 7_{10}$ antes de regresar a 000.

Diagrama de transición de estados

Otra manera de mostrar cómo cambian los estados de los FFs con cada pulso de reloj que se aplica es mediante el uso de un **diagrama de transición de estados**, como se muestra en la figura 5-49.

FIGURA 5-49 El diagrama de transición de estados muestra cómo cambian los estados de los flip-flops contadores con cada pulso de reloj que se aplica.



Cada círculo representa uno de los posibles estados, según lo indica el número binario dentro del círculo. Por ejemplo, el círculo que contiene el número 100 representa el estado 100 (es decir, $Q_2 = 1$, $Q_1 = Q_0 = 0$).

Las flechas que conectan un círculo con otro muestra la forma en que cambia un estado a otro, a medida que se aplica un pulso de reloj. Si analizamos un círculo de estado específico, podremos ver cuál estado va antes y cuál va después. Por ejemplo, si analizamos el estado 000 podemos ver que se llega a él cada vez que el contador se encuentra en el estado 111 y se aplica un pulso de reloj. De igual forma, podemos ver que el estado 000 siempre va antes del estado 001.

Utilizaremos los diagramas de transición de estados para ayudar a describir, analizar y diseñar contadores y otros circuitos secuenciales.

Número MOD

El contador de la figura 5-47 tiene $2^3 = 8$ estados distintos (del 000 al 111). A este contador se le denomina *contador MOD-8*, en donde el **número MOD** indica la cantidad de estados en la secuencia de conteo. Si se agregara un cuarto FF, la secuencia de estados contaría en binario desde 0000 hasta 1111, un total de 16 estados. A esto se le llamaría *contador MOD-16*. En general, si se conectan N flip-flops en el arreglo de la figura 5-47, el contador tendrá 2^N estados distintos, por lo cual será un contador MOD- 2^N . Sería capaz de contar hasta $2^N - 1$ antes de regresar a 0.

El número MOD de un contador también indica la división de frecuencia que se obtiene del último FF. Por ejemplo, un contador de cuatro bits tiene cuatro FFs, cada uno de los cuales representa un dígito binario (bit) y, por lo tanto, es un contador MOD- $2^4 = \text{MOD-16}$. Debido a esto, puede contar hasta 15 ($= 2^4 - 1$). También puede usarse para dividir la frecuencia del pulso de entrada por un factor de 16 (el número MOD).

Hasta ahora sólo hemos visto el contador binario de FF básico. En el capítulo 7 examinaremos los contadores con mucho más detalle.

EJEMPLO 5-14

Suponga que el contador MOD-8 de la figura 5-47 se encuentra en el estado 101. ¿Cuál será el estado (cuenta) después de aplicar 13 pulsos?

Solución

Localice el estado 101 en el diagrama de transición de estados. Proceda alrededor del diagrama de estados a través de ocho cambios de estado, y deberá encontrarse de vuelta en el estado 101. Ahora continúe a través de cinco cambios de estado más (para un total de 13) y deberá terminar en el estado 010.

Como éste es un contador MOD-8 con ocho estados, requiere de ocho transiciones de estado para realizar una excursión completa alrededor del diagrama y de vuelta al estado inicial.

EJEMPLO 5-15

Considere un circuito contador que contiene seis FFs conectados como se muestra en la figura 5-47 (es decir, $Q_5, Q_4, Q_3, Q_2, Q_1, Q_0$).

- Determine el número MOD del contador.
- Determine la frecuencia en la salida del último FF (Q_5) cuando la frecuencia del reloj de entrada es de 1 MHz.
- ¿Cuál es el intervalo de estados de conteo para este contador?
- Suponga un estado inicial (conteo) de 000000. ¿Cuál será el estado del contador después de 129 pulsos?

Solución

- (a) Número MOD = $2^6 = 64$.
- (b) La frecuencia en el último FF será igual a la frecuencia del reloj de entrada, dividida entre el número MOD. Esto es,

$$f(\text{en } Q_5) = \frac{1 \text{ MHz}}{64} = 15.625 \text{ kHz}$$

- (c) El contador contará desde 000000_2 hasta 111111_2 (de 0 a 63_{10}) para un total de 64 estados. Observe que el número de estados es el mismo que el número MOD.
- (d) Como éste es un contador MOD-64, cada 64 pulsos de reloj regresarán el contador de vuelta a su estado inicial. Por lo tanto, después de 128 pulsos el conteo se regresa a 000000. El pulso 129 lleva el contador a la cuenta 000001.

PREGUNTAS DE REPASO

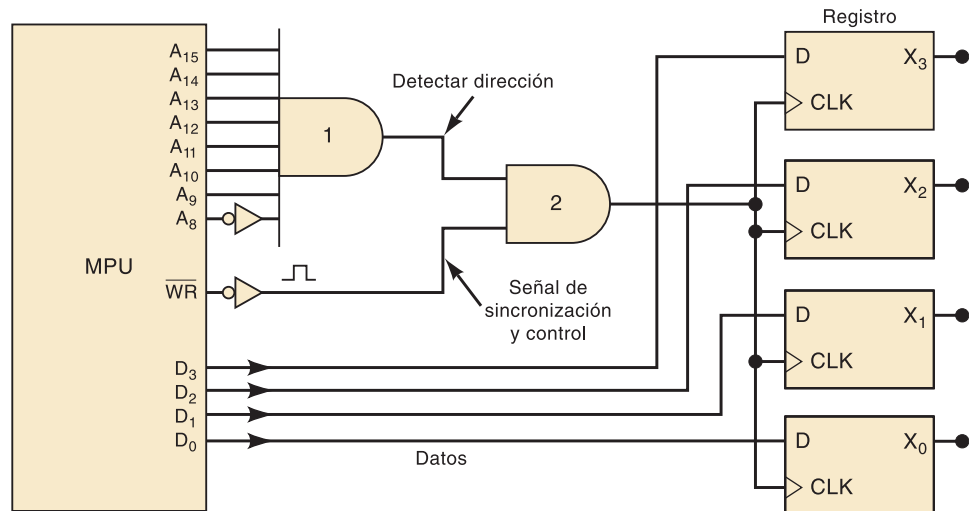
- Una señal de reloj de 20 kHz se aplica a un flip-flop J-K, en donde $J = K = 1$. ¿Cuál es la frecuencia de la forma de onda de salida del FF?
- ¿Cuántos FFs se requieren para un contador que cuente de 0 a 255_{10} ?
- ¿Cuál es el número MOD del contador en la pregunta 2?
- ¿Cuál es la frecuencia de la salida del octavo FF cuando la frecuencia de reloj es de 512 kHz?
- Si este contador empieza en 00000000, ¿cuál será su estado después de 520 pulsos?

5-20 APLICACIÓN DE MICROCOMPUTADORA

Su estudio sobre los sistemas digitales aún se encuentra en una etapa relativamente temprana y, por lo tanto, no ha aprendido mucho todavía sobre los microprocesadores y las microcomputadoras. No obstante, puede darse una idea básica acerca de cómo se emplean los FFs en una aplicación ordinaria controlada por microprocesador, sin preocuparse por todos los detalles que veremos después.

La figura 5-50 muestra una unidad de microprocesador (MPU) en la cual su salida se utiliza para transferir datos binarios al registro X, el cual consiste de cuatro flip-flops D identificados como X_3, X_2, X_1, X_0 . Un conjunto de salidas de la MPU es el *código de dirección*, que está compuesto por las ocho salidas $A_{15}, A_{14}, A_{13}, A_{12}, A_{11}, A_{10}, A_9, A_8$. La mayoría de las MPUs tienen cuando menos 16 salidas de dirección

FIGURA 5-50 Ejemplo de un microprocesador que transfiere datos binarios a un registro externo.



disponibles, pero no siempre se utilizan todas. Un segundo conjunto de salidas de la MPU consiste de las cuatro *líneas de datos* D_3, D_2, D_1, D_0 . La mayoría de las MPUs tienen cuando menos ocho líneas de datos disponibles. La otra salida de la MPU es una señal de control de sincronización $\overline{WR}$, la cual cambia a BAJO cuando la MPU está lista para escribir.

Recuerde que la MPU es la unidad central de procesamiento de una microcomputadora, y su función principal es ejecutar un programa de instrucciones almacenadas en la memoria de la computadora. Una de las instrucciones que podría ejecutar sería la que indique a la MPU que transfiera un número binario desde un registro de almacenamiento dentro de la MPU, hacia el registro externo X. A esto se le conoce como *ciclo de escritura*. Al ejecutar esta instrucción, la MPU realizaría los siguientes pasos:

1. Colocar el número binario en sus líneas de salida de datos D_3 a D_0 .
2. Colocar el código de dirección apropiado en sus líneas de salida A_{15} a A_8 para seleccionar el registro X como el receptor de los datos.
3. Una vez que se hayan estabilizado las salidas de datos y de dirección, la MPU genera el pulso de escritura $\overline{WR}$ para sincronizar el registro y completar la transferencia de datos en paralelo hacia X.

Existen muchas situaciones en las que una MPU, bajo el control de un programa, enviará datos a un registro externo para poder controlar eventos externos. Por ejemplo, los FFs individuales en el registro pueden controlar el estado ENCENDIDO/APAGADO de dispositivos electromecánicos tales como solenoides, relevadores, motores, etcétera (a través de circuitos de interfaz apropiados, desde luego). Los datos que se envíen de la MPU al registro determinarán cuáles dispositivos deben estar ENCENDIDOS y cuáles APAGADOS. Otro ejemplo común es cuando el registro se utiliza para almacenar un número binario para enviarlo como entrada a un convertidor digital-analógico (DAC). La MPU envía el número binario al registro y el DAC lo convierte en un voltaje analógico que puede usarse para controlar algo tal como la posición de un haz de electrones en una pantalla CRT, o la velocidad de un motor.

EJEMPLO 5-16

- (a) ¿Qué código de dirección debe generar la MPU para que los datos puedan transferirse hacia X?
- (b) Suponga que $X_3 - X_0 = 0110$, $A_{15} - A_8 = 1111111$ y $D_3 - D_0 = 1011$. ¿Qué habrá en X después de que ocurra un pulso en $\overline{WR}$?

Solución

- (a) Para poder transferir los datos hacia X, el pulso de reloj debe pasar a través de la compuerta AND 2 y hacia las entradas CLK de los FFs. Esto ocurrirá sólo si la entrada superior de la compuerta AND 2 se encuentra en ALTO. Esto significa que todas las entradas para la compuerta AND 1 deben estar en ALTO; es decir, desde A_{15} hasta A_9 deben ser 1 y A_8 debe ser 0. Por ende, se necesita la presencia del código de dirección 11111110 para permitir la transferencia de los datos hacia X.
- (b) Con $A_8 = 1$, el nivel BAJO de la compuerta AND 1 inhibirá a $\overline{WR}$ de manera que no pueda pasar hacia la compuerta AND 2, y los FFs no se sincronizarán. Por lo tanto, el contenido del registro X no cambiará de 0110.

PREGUNTAS DE REPASO

1. Muestre cómo puede usarse el CI 74HC175 de la figura 5-34 para el registro X de la figura 5-50.

5-21 DISPOSITIVOS DISPARADORES DE SCHMITT

Un **circuito disparador de Schmitt** no se clasifica como un flip-flop, pero exhibe un tipo de memoria característico, que lo hace útil en ciertas situaciones especiales. Una de estas situaciones se muestra en la figura 5-51(a). Aquí un INVERSOR estándar está siendo controlado por una entrada lógica que tiene tiempos de transición relativamente lentos. Cuando estos tiempos de transición se exceden de los valores máximos permitidos (esto depende de la familia lógica específica), las salidas de la compuertas lógicas y de los INVERSORES pueden producir oscilaciones a medida que pasa la señal de entrada a través del intervalo indeterminado. Las mismas condiciones de entrada también pueden producir un disparo errático de los FFs.

Un dispositivo que tiene un tipo de entrada de disparador de Schmitt está diseñado para aceptar señales ruidosas con cambios lentos, y producir una salida con transiciones libres de oscilaciones. Por lo general, la salida tendrá tiempos de transición muy rápidos (por lo general de 10 ns) que son independientes de las características de la señal de entrada. La figura 5-51(b) muestra un INVERSOR disparador de Schmitt y su respuesta a una entrada con cambios lentos.

Si analiza las formas de onda en la figura 5-51(b), podrá observar que la salida no cambia de ALTO a BAJO sino hasta que la entrada se excede del voltaje *umbral de pendiente positiva*, V_{T+} . Una vez que la salida cambia a BAJO, permanecerá ahí incluso aunque la entrada vuelva a caer por debajo de V_{T+} (ésta es su característica de memoria), hasta que caiga por debajo del voltaje *umbral de pendiente negativa*, V_{T-} . Los valores de los dos voltajes de umbral variarán de una familia lógica a otra, pero V_{T-} siempre será menor que V_{T+} .

El INVERSOR disparador de Schmitt y todos los demás dispositivos con entradas de disparador de Schmitt utilizan el símbolo distintivo que se muestra en la figura 5-51(b) para indicar que pueden responder en forma confiable a las señales de entrada con cambios lentos. Los diseñadores lógicos utilizan CIs con entradas tipo disparador de Schmitt para convertir las señales con cambios lentos en señales limpias con cambios rápidos que puedan controlar entradas de CIs estándar.

Hay varios CIs disponibles con entradas tipo disparador de Schmitt. Los CIs 7414, 74LS14 y 74HC14 son CIs INVERSORES hexadecimales con entradas tipo disparador de Schmitt. Los CIs 7413, 74LS13 y 74HC13 son dos compuertas NAND de cuatro entradas tipo disparador de Schmitt.

PREGUNTAS DE REPASO

1. ¿Qué ocurre cuando se aplica una señal con cambios lentos a un CI lógico estándar?
2. ¿Cómo opera un dispositivo lógico disparador de Schmitt en forma distinta de un dispositivo lógico estándar?

5-22 ONE-SHOT (MULTIVIBRADOR MONOESTABLE)

Un circuito digital que está en parte relacionado con el FF es el **one-shot (OS)**. Al igual que el FF, el OS tiene dos entradas Q y $\bar{Q}$ que son el inverso una de otra. A diferencia del FF, el OS sólo tiene un estado de salida *estable* (por lo general $Q = 0$, $\bar{Q} = 1$), en donde permanece hasta que se dispara mediante una señal de entrada. Una vez disparado, las salidas del OS cambian al estado opuesto ($Q = 1$, $\bar{Q} = 0$). Permanece en este **estado casi estable** durante un periodo fijo (t_p), que, por lo general, se determina mediante una constante de tiempo RC que resulta de los valores de los componentes externos conectados al OS. Después de un tiempo t_p , las salidas del OS regresan a su estado inactivo hasta que se vuelven a disparar.

La figura 5-52(a) muestra el símbolo lógico para un OS. A menudo el valor de t_p se indica en alguna parte del símbolo del OS. En la práctica, t_p puede variar de varios nanosegundos hasta varias décadas de segundos. El valor exacto de t_p es variable y se determina mediante los valores de los componentes externos R_T y C_T .

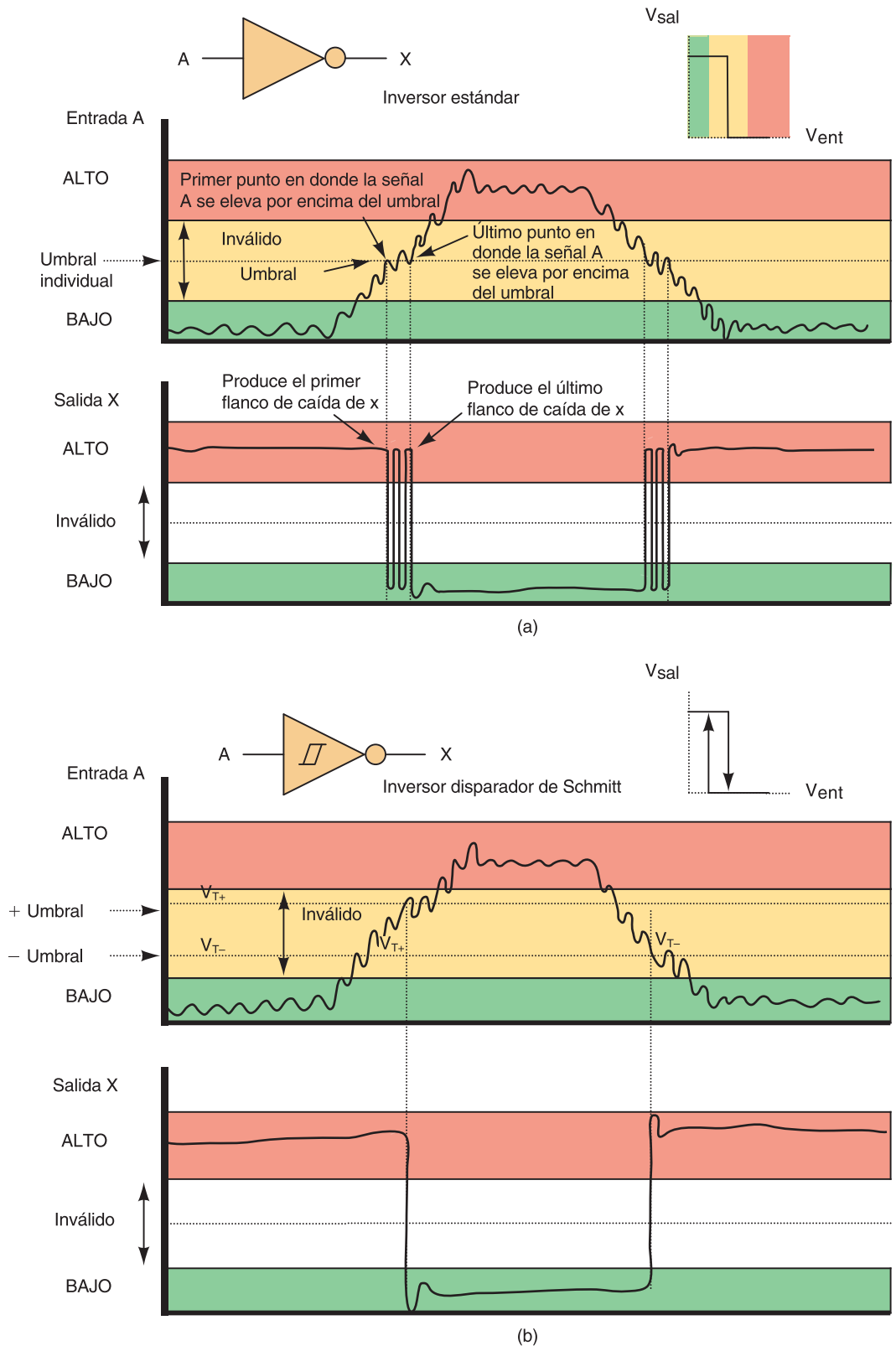
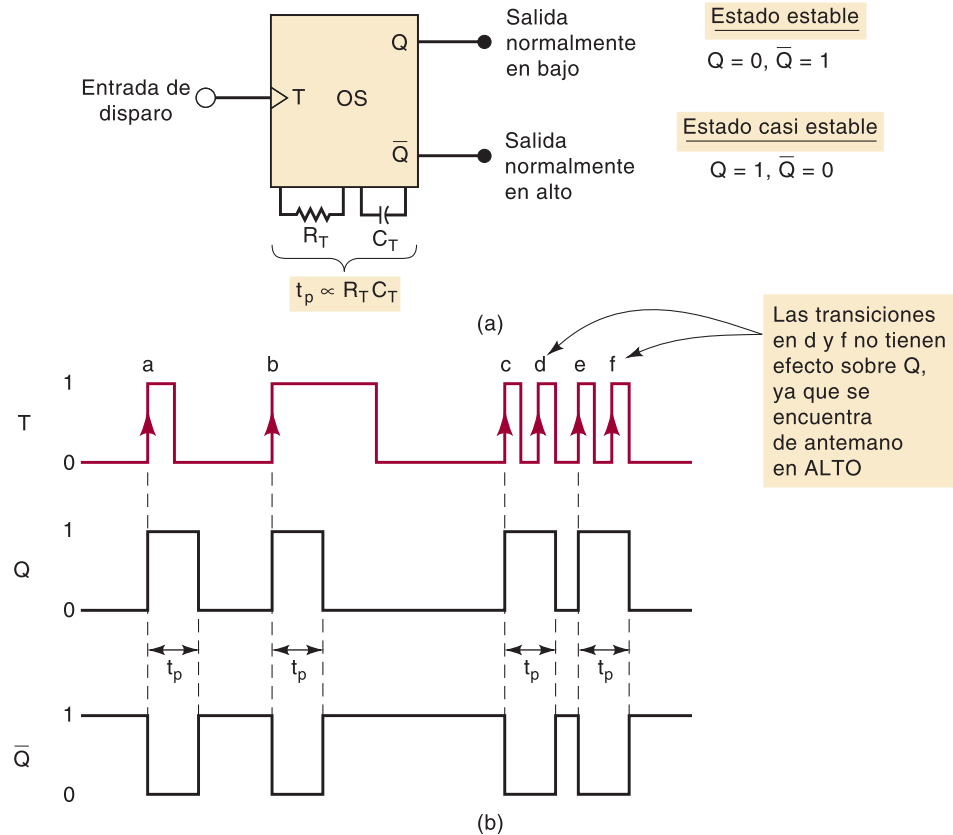


FIGURA 5-51 (a) Respuesta del inversor estándar a una entrada ruidosa y lenta, y (b) respuesta del disparador de Schmitt a una entrada ruidosa y lenta.

FIGURA 5-52 Símbolo del OS y formas de onda típicas para la operación no redispensible.



Hay dos tipos de one-shot disponibles en forma de CI: el **OS no redispensible** y el **OS redispensible**.

One-shot no redispensible

Las formas de onda de la figura 5-12(b) ilustran la operación de un OS no redispensible que se dispara en las transiciones de pendiente positiva en su entrada de disparo (T). Los puntos importantes a observar son:

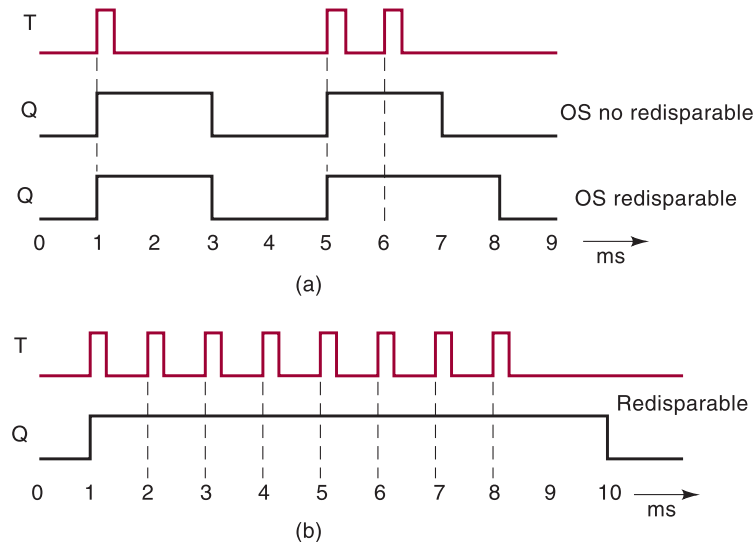
1. Las PGTs en los puntos *a*, *b*, *c* y *e* dispararán el OS a su estado casi estable durante un tiempo t_p , después del cual regresará en forma automática a su estado estable.
2. Las PGTs en los puntos *d* y *f* no tienen efecto sobre el OS, ya que de antemano se ha disparado al estado casi estable. El OS debe regresar al estado estable antes de que pueda dispararse.
3. La duración del pulso de salida del OS es siempre la misma, sin importar la duración de los pulsos de entrada. Como se dijo antes, t_p sólo depende de R_T y de C_T , además de los circuitos internos del OS. Un OS ordinario puede tener un t_p dado por $t_p = 0.693 R_T C_T$.

One-shot redispensible

El OS redispensible opera en forma muy parecida al OS no redispensible, con una diferencia importante: *puede redispensarse mientras se encuentra en el estado casi estable, y comenzará un nuevo intervalo t_p* . La figura 5-53(a) compara la respuesta de ambos tipos de OS mediante el uso de un t_p de 2 ms. Examinemos estas formas de onda.

FIGURA 5-53

(a) Comparación de las respuestas del OS no redispensible y del OS redispensible para $t_p = 2$ ms. (b) El OS redispensible comienza un nuevo intervalo t_p cada vez que recibe un pulso de disparo.



Ambos tipos de OS responden al primer pulso de disparo en $t = 1$ ms, cambiando a nivel ALTO durante 2 ms y después regresando a BAJO. El segundo pulso de disparo en $t = 5$ ms dispara ambos monoestables al estado ALTO. El tercer pulso de disparo en $t = 6$ ms no tiene efecto sobre el OS no redispensible, ya que se encuentra de antemano en su estado casi estable. No obstante, este pulso de disparo *redispensará* el OS redispensible para comenzar un nuevo intervalo $t_p = 2$ ms. Por ende, permanecerá en ALTO durante 2 ms *después* de este tercer pulso de disparo.

En efecto, entonces un OS redispensible comienza un nuevo intervalo t_p cada vez que se aplica un pulso de disparo, sin importar el estado actual de su salida Q . De hecho, pueden aplicarse pulsos de disparo a una velocidad lo bastante rápida como para que el OS siempre se redispense antes de que termine el intervalo t_p y Q permanecerá en ALTO. Esto se muestra en la figura 5-53(b), en donde se aplican ocho pulsos cada 1 ms. Q no regresa a BAJO sino hasta 2 ms después del último pulso de disparo.

Dispositivos reales

Hay varios CIs monoestables disponibles tanto en la versión redispensible como en la no redispensible. El 74121 es un CI con one-shot individual no redispensible; el 74221, 74LS221 y 74HC221 son CIs con dos one-shot no redispensibles; el 74122 y 74LS122 son CIs con one-shot individual redispensible; el 74123, 74LS123 y 74HC123 son CIs con dos one-shot redispensibles.

La figura 5-54(a) muestra el símbolo tradicional para el CI 74121 con one-shot no redispensible. Observe que contiene compuertas lógicas internas para permitir que las entradas A_1 , A_2 y B disparen el OS en una variedad de formas. La entrada B es tipo disparador de Schmitt y se le permite tener tiempos de transición lentos sin que pierda la capacidad de disparar en forma confiable el OS. Las terminales etiquetadas como R_{INT} , R_{EXT}/C_{EXT} y C_{EXT} se utilizan para conectar una resistencia y un capacitor externos para lograr la duración deseada en el pulso de salida. La figura 5-54(b) utiliza el símbolo IEEE/ANSI para el OS no redispensible 74121. Observe cómo este símbolo representa las compuertas lógicas. Observe además la presencia de un pequeño pulso con 1 enfrente del mismo. Esto indica que el dispositivo es un OS no redispensible. El símbolo IEEE/ANSI para un OS redispensible no tendría el 1 enfrente del pulso.

Multivibrador monoestable

Otro nombre para el one-shot es *multivibrador monoestable*, ya que sólo tiene un estado estable. Los monoestables encuentran una aplicación limitada en la mayoría

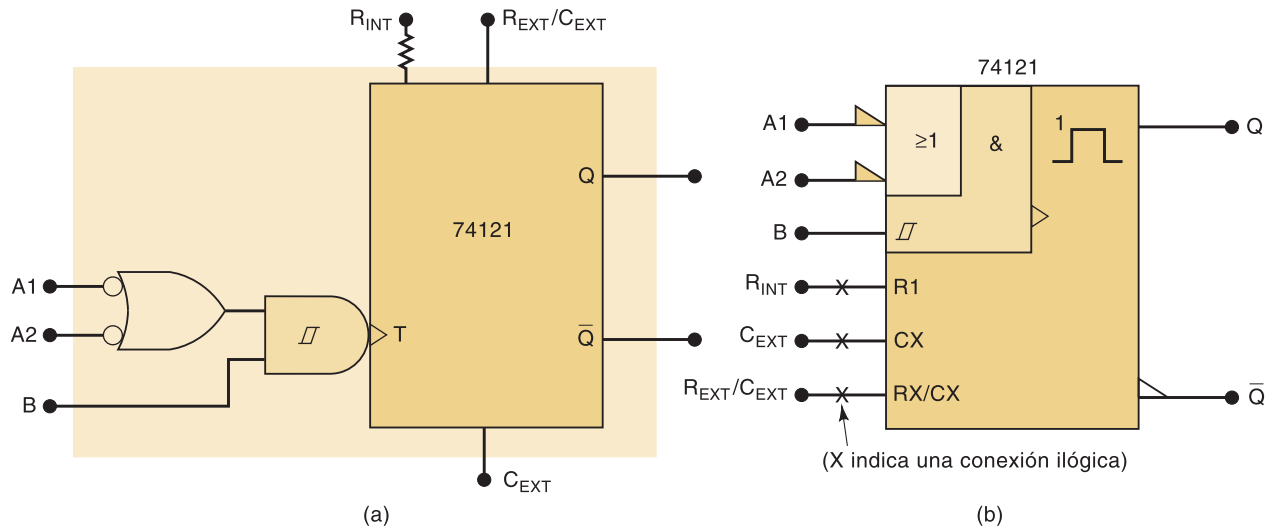


FIGURA 5-54 Símbolos lógicos para el monoestable no redisparable 74121: (a) tradicional; (b) IEEE/ANSI.

de los sistemas controlados por reloj; los diseñadores experimentados evitan usarlos ya que están propensos a falsos disparos debido al ruido espurio. Cuando se llegan a utilizar, por lo general, es en aplicaciones de sincronización simples que utilizan el intervalo t_p predeterminado. Varios de los problemas al final del capítulo ilustrarán la manera en que se utiliza un OS.

PREGUNTAS DE REPASO

1. En la ausencia de un pulso de disparo, ¿cuál será el estado de la salida de un OS?
2. *Verdadero o falso:* cuando se aplica un pulso a un OS no redisparable mientras se encuentra en su estado casi estable, la salida no se ve afectada.
3. ¿Qué es lo que determina el valor de t_p para un OS?
4. Describa cómo opera un OS redisparable en forma distinta de un OS no redisparable.

5-23 CIRCUITOS GENERADORES DE RELOJ

Los flip-flops tienen dos estados estables; por lo tanto, podemos decir que son *multivibradores biestables*. Los monoestables (one-shot) tienen sólo un estado estable, por lo que podemos llamarlos *multivibradores monoestables*. Un tercer tipo de multivibrador no tiene estados estables; se le llama **multivibrador astable o de funcionamiento libre**. Este tipo de circuito lógico alterna (oscila) entre dos estados de salida inestables. Es útil para generar señales de reloj para circuitos digitales síncronos.

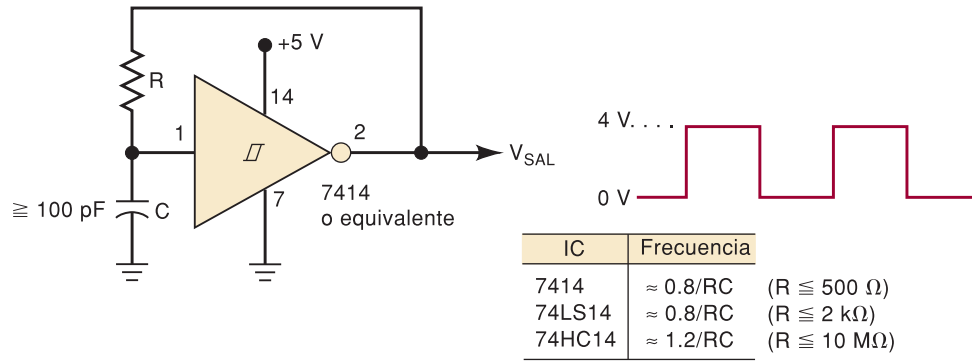
Hay varios tipos de multivibradores astables de uso común. Presentaremos tres de ellos sin tratar de analizar su operación. Los presentamos aquí para que usted pueda construir un circuito generador de reloj si lo necesita para un proyecto o para probar circuitos digitales en el laboratorio.

Oscilador disparador de Schmitt

La figura 5-55 muestra cómo puede conectarse un INVERSOR disparador de Schmitt como un oscilador. La señal en V_{SAL} es una onda cuadrada aproximada con una frecuencia que depende de los valores de R y de C . La relación entre la frecuencia y

FIGURA 5-55

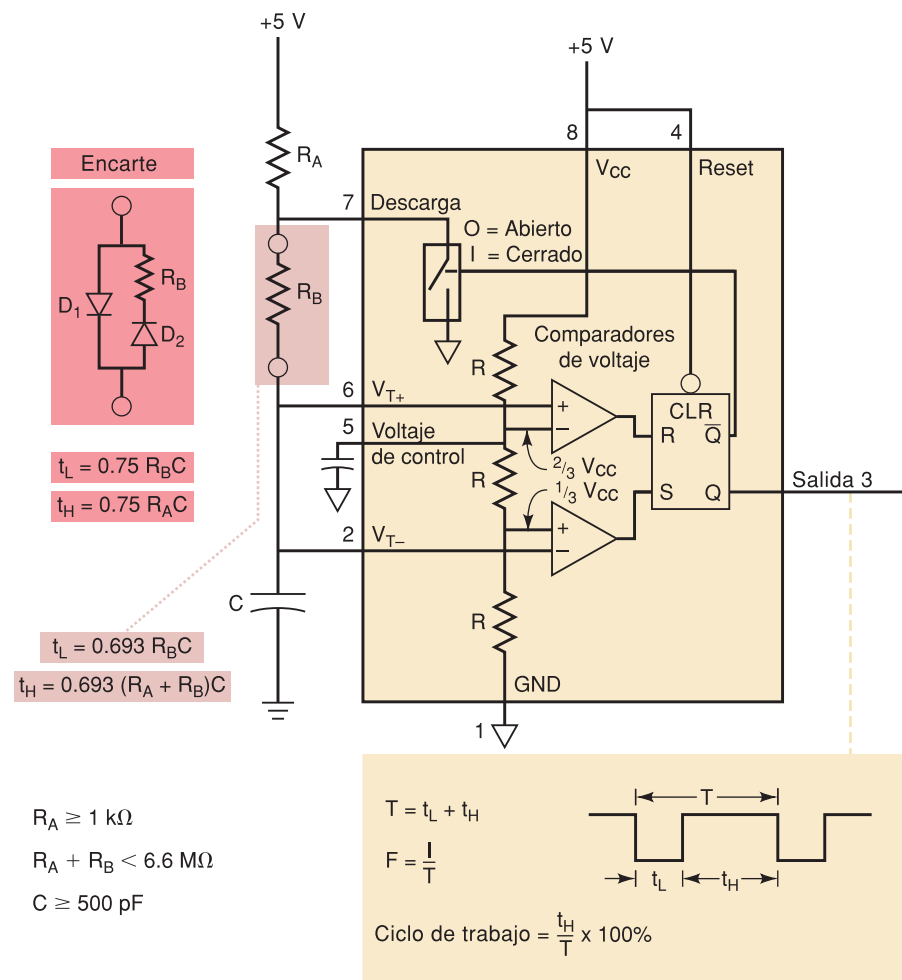
Oscilador disparador de Schmitt que utiliza un INVERSOR 7414. También puede usarse una compuerta NAND tipo disparador de Schmitt 7413.



los valores RC se muestra en la figura 5-55 para tres INVERSORES disparadores de Schmitt distintos. Observe los límites máximos en el valor de resistencia para cada dispositivo. El circuito no oscilará si R no se mantiene por debajo de estos límites.

Temporizador 555 utilizado como multivibrador astable

El **CI temporizador 555** es un dispositivo compatible con TTL que puede operar en varios modos. La figura 5-56 muestra cómo pueden conectarse componentes externos a un 555 de manera que opere como un oscilador de funcionamiento libre. Su salida es una forma de onda rectangular repetitiva que alterna entre dos niveles

FIGURA 5-56 CI temporizador 555 utilizado como multivibrador astable.

lógicos, en donde los espacios de tiempo en cada nivel lógico se determinan con base en los valores de R y de C .

El corazón del temporizador 555 está compuesto de dos comparadores de voltaje y un latch SR, como se muestra en la figura 5-56. Los comparadores de voltaje son dispositivos que producen una salida en ALTO cada vez que el voltaje en la entrada + es mayor que el voltaje en la entrada -. El capacitor externo (C) se carga hasta que su voltaje excede de $\frac{2}{3} \times V_{CC}$, según lo determinado por el comparador de voltaje superior que monitorea a V_{T+} . Cuando la salida de este comparador cambia a ALTO, restablece el latch SR, provocando que la terminal de salida (3) cambie a BAJO. Al mismo tiempo, $\bar{Q}$ cambia a ALTO y cierra el interruptor de descarga, haciendo que el capacitor comience a descargarse a través de R_B . Continuará descargándose hasta que su voltaje caiga por debajo de $\frac{1}{3} \times V_{CC}$, según lo determinado por el comparador de voltaje inferior que monitorea a V_{T-} . Cuando la salida de este comparador cambia a ALTO establece el latch SR, haciendo que la terminal de salida cambie a ALTO y se abre el interruptor de descarga, con lo cual el capacitor puede empezar a cargarse otra vez, a medida que se repite el ciclo.

Las fórmulas para estos intervalos (t_L y t_H) y para el periodo total de las oscilaciones (T) se proporcionan en la figura. Desde luego que la frecuencia de las oscilaciones es el recíproco de T . Como indican las fórmulas en el diagrama, los intervalos t_L y t_H no pueden ser iguales a menos que R_A se haga cero. Esto no puede hacerse sin producir corriente en exceso a través del dispositivo. Esto significa que con este circuito es imposible producir una salida de onda cuadrada perfecta con un ciclo de trabajo del 50 por ciento. No obstante, es posible llegar muy cerca del 50 por ciento del ciclo de trabajo si hacemos que $R_B \gg R_A$ (mientras R_A se mantenga mayor que 1 k Ω), de manera que $t_L \approx t_H$.

EJEMPLO 5-17

Calcule la frecuencia y el ciclo de trabajo de la salida del multivibrador astable 555 para $C = 0.001 \mu\text{F}$, $R_A = 2.2 \text{ k}\Omega$ y $R_B = 100 \text{ k}\Omega$.

Solución

$$\begin{aligned} t_L &= 0.693(100 \text{ k}\Omega)(0.001 \mu\text{F}) = 69.3 \mu\text{s} \\ t_H &= 0.693(102.2 \text{ k}\Omega)(0.001 \mu\text{F}) = 70.7 \mu\text{s} \\ T &= 69.3 + 70.7 = 140 \mu\text{s} \\ f &= 1/140 \mu\text{s} = 7.29 \text{ kHz} \\ \text{ciclo de trabajo} &= 70.7/140 = 50.5\% \end{aligned}$$

Observe que el ciclo de trabajo está cerca del 50 por ciento (onda cuadrada) ya que R_B es mucho mayor que R_A . Puede hacerse aún más cercano al 50 por ciento si R_B se hace aún más grande en comparación con R_A . Por ejemplo, debería verificar que si cambiamos R_A a 1 k Ω (su valor mínimo permitido), los resultados son $f = 7.18 \text{ kHz}$ y el ciclo de trabajo = 50.3 por ciento.

Puede hacerse una simple modificación a este circuito para permitir un ciclo de trabajo de menos del 50 por ciento. La estrategia es permitir que el capacitor se llene (cargue) con partículas cargadas que fluyan sólo a través de R_A y se vacíe (descargue) a medida que las partículas cargadas fluyen sólo a través de R_B . Para lograr esto basta con conectar un diodo (D_2) en serie con R_B y otro diodo (D_1) en paralelo con R_B y D_2 , como muestra el encarte de la figura 5-56. El circuito del encarte sustituye a R_B en el dibujo. Los diodos son dispositivos que permiten que las partículas cargadas fluyan a través de ellos sólo en una dirección, según lo que indique la punta de la flecha. El diodo D_1 permite que toda la corriente de carga que ha pasado a través de R_A se desvíe de R_B , y D_2 asegura que la corriente de carga no pueda fluir a través de R_B . Toda la corriente de descarga fluye a través

de D_2 y de R_B cuando se cierra el interruptor de descarga. Las ecuaciones para el tiempo alto y el tiempo bajo para este circuito son

$$\begin{aligned}t_L &= 0.75 R_B C \\t_H &= 0.75 R_A C\end{aligned}$$

Nota: La constante 0.75 es correcta sólo para $V_{CC} = 5\text{ V}$.

EJEMPLO 5-18

Utilice los diodos junto con R_B , como se muestra en la figura 5-56, para calcular los valores de R_A y R_B necesarios para obtener una forma de onda de 1 kHz con un ciclo de trabajo del 25 por ciento como salida de un 555. Suponga que C es un capacitor de $0.1\text{ }\mu\text{F}$.

Solución

$$\begin{aligned}T &= \frac{1}{F} = \frac{1}{1000} = 0.001\text{ s} = 1\text{ ms} \\t_H &= 0.25 \times T = 0.25 \times 1\text{ ms} = 250\text{ }\mu\text{s} \\R_A &= \frac{250\text{ }\mu\text{s}}{0.75 \times C} = \frac{250\text{ }\mu\text{s}}{0.75 \times 0.1\text{ }\mu\text{F}} = 3.3\text{ k}\Omega \\R_B &= \frac{750\text{ }\mu\text{s}}{0.75 \times C} = \frac{750\text{ }\mu\text{s}}{0.75 \times 0.1\text{ }\mu\text{F}} = 10\text{ k}\Omega\end{aligned}$$

Generadores de reloj controlados por cristal

Las frecuencias de salida de las señales de los circuitos generadores de reloj antes descritos dependen de los valores de las resistencias y los capacitores, por lo cual no son en extremo precisos o estables. Aún si se utilizaran resistencias variables, de manera que la frecuencia deseada pudiera ajustarse mediante un ajuste en los valores de la resistencia, se producirían cambios en los valores de R y C debido a los cambios en la temperatura ambiental y al desgaste, con lo cual la frecuencia ajustada variaría. Si es imprescindible tener una frecuencia precisa y estable, podemos utilizar otro método para generar señales de reloj: un **generador de reloj controlado por cristal**. Este tipo de generador emplea un componente de alta estabilidad y precisión, al cual se le conoce como *cristal de cuarzo*. Puede cortarse una pieza de cristal de cuarzo a un tamaño y forma específicos para que vibre (resuene) a una frecuencia precisa que es en extremo estable en relación con los cambios de temperatura y el desgaste; pueden lograrse frecuencias desde 10 kHz hasta 80 MHz. Cuando se coloca un cristal en ciertas configuraciones de circuitos, puede producir oscilaciones a una frecuencia precisa y estable, igual a la frecuencia resonante del cristal. Los osciladores de cristal están disponibles como encapsulados de CI.

Los circuitos generadores de reloj controlados por cristal se utilizan en todos los sistemas basados en microprocesador y en las microcomputadoras, y en cualquier aplicación en la que se utilice una señal de reloj para generar intervalos de sincronización precisos. En los siguientes capítulos veremos su funcionamiento en algunas de las aplicaciones de ejemplo.

PREGUNTAS DE REPASO

1. Determine la frecuencia aproximada de un oscilador disparador de Schmitt que utiliza un 74HC14 con $R = 10\text{ k}\Omega$ y $C = 0.005\text{ }\mu\text{F}$.
2. Determine la frecuencia y el ciclo de trabajo aproximados del oscilador 555 para $R_A = R_B = 2.2\text{ k}\Omega$ y $C = 2000\text{ pF}$.
3. ¿Cuál es la ventaja de los circuitos generadores de reloj controlados por cristal, en comparación con los circuitos controlados por RC?

5-24 DIAGNÓSTICO DE FALLAS EN CIRCUITOS CON FLIP-FLOPS

Los CIs con flip-flops son susceptibles a los mismos tipos de fallas internas y externas que ocurren en los circuitos lógicos combinacionales. Todas las ideas sobre diagnóstico de fallas que vimos en el capítulo 4 pueden aplicarse también a los circuitos que contienen FFs, así como compuertas lógicas.

Debido a su característica de memoria y a su operación sincronizada, los circuitos con FFs están sujetos a varios tipos de fallas y síntomas asociados que no se presentan en los circuitos combinacionales. En especial, los circuitos con FFs son susceptibles a los problemas de sincronización que, por lo general, no son de cuidado en los circuitos combinacionales. A continuación describiremos los tipos más comunes de fallas en los circuitos con FFs.

Entradas abiertas

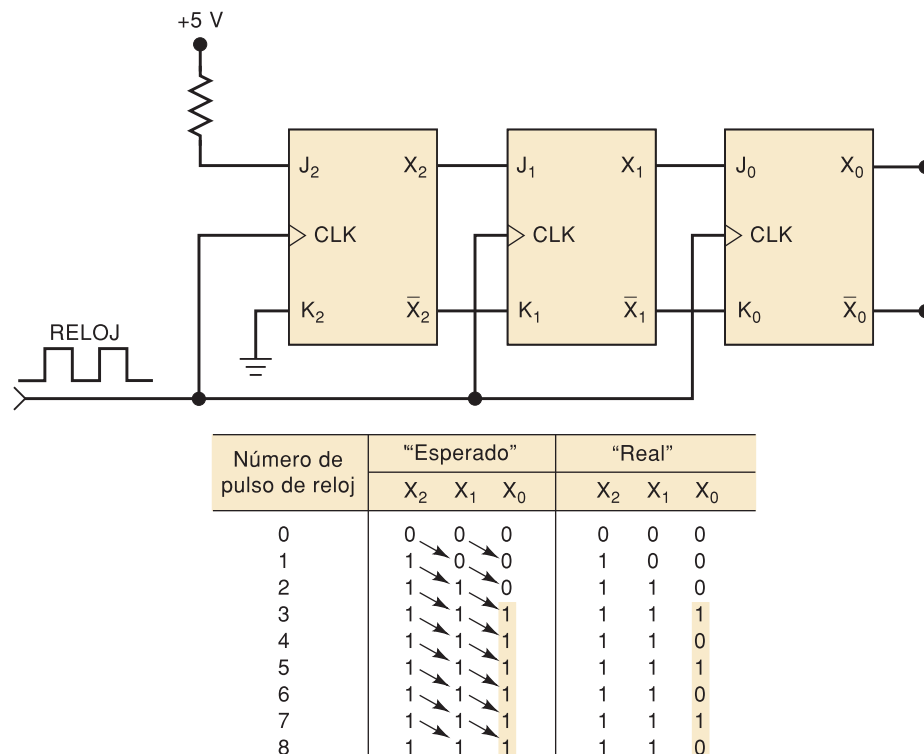
Las entradas desconectadas o flotantes de cualquier circuito lógico son en especial susceptibles a la atracción de fluctuaciones espurias de voltaje, conocidas como *ruido*. Si el ruido es lo suficientemente grande en amplitud y largo en duración, la salida del circuito lógico puede llegar a cambiar de estado en respuesta a ese ruido. En una compuerta lógica, la salida regresará a su estado original cuando la señal de ruido se detenga. No obstante, en un FF la salida permanecerá en su nuevo estado debido a su característica de memoria. Por ende, el efecto de atraer ruido en cualquier entrada abierta es, por lo general, más crítico para un FF o latch que para una compuerta lógica.

Las entradas más susceptibles en un FF son las que pueden dispararlo a un estado distinto, tales como *CLK*, *PRESET* y *CLEAR*. Cada vez que vea que la salida de un FF cambia de estado en forma errónea, deberá considerar la posibilidad de una conexión abierta en una de estas entradas.

EJEMPLO 5-19

La figura 5-57 muestra un registro de desplazamiento de tres bits compuesto de flip-flops TTL. En un principio todos los FFs se encuentran en el estado BAJO, antes de aplicar pulsos de reloj.

FIGURA 5-57 Ejemplo 5-19.



A medida que se apliquen pulsos de reloj, cada PGT hará que la información se desplace desde cada FF hacia el que se encuentra a su derecha. El diagrama muestra la secuencia “esperada” de estados de los FFs después de cada pulso de reloj. Como $J_2 = 1$ y $K_2 = 0$, el flip-flop X_2 cambiará a ALTO en el pulso de reloj 1 y permanecerá ahí durante todos los siguientes pulsos. Este nivel ALTO se desplazará hacia X_1 y después hacia X_0 en los pulsos 2 y 3, respectivamente. En consecuencia, después del tercer pulso todos los FFs estarán en ALTO y deberán permanecer ahí a medida que se apliquen pulsos en forma continua.

Ahora suponga que la respuesta “real” de los estados de los FFs es como se muestra en el diagrama. Aquí los FFs cambian según lo esperado durante los primeros tres pulsos de reloj. De ahí en adelante el flip-flop X_0 , en vez de permanecer en ALTO alterna entre ALTO y BAJO. ¿Cuál de las posibles fallas del circuito pueden producir esta situación?

Solución

En el segundo pulso X_1 cambia a ALTO, con lo cual se debe producir la condición $J_0 = 1$, $K_0 = 0$ de manera que todos los pulsos de reloj subsiguientes deberán establecer la condición $X_0 = 1$. En vez de ello, vemos que X_0 cambia de estado (conmuta) en todos los pulsos después del segundo. Esta operación de conmutación se produciría si J_0 y K_0 estuvieran ambas en ALTO. La falla más probable es una interrupción en la conexión entre $\bar{X}_1$ y K_0 . Recuerde que un dispositivo TTL responde a una entrada abierta como si fuera un nivel ALTO lógico, por lo que un circuito abierto en K_0 es lo mismo que un nivel ALTO.

Salidas en corto

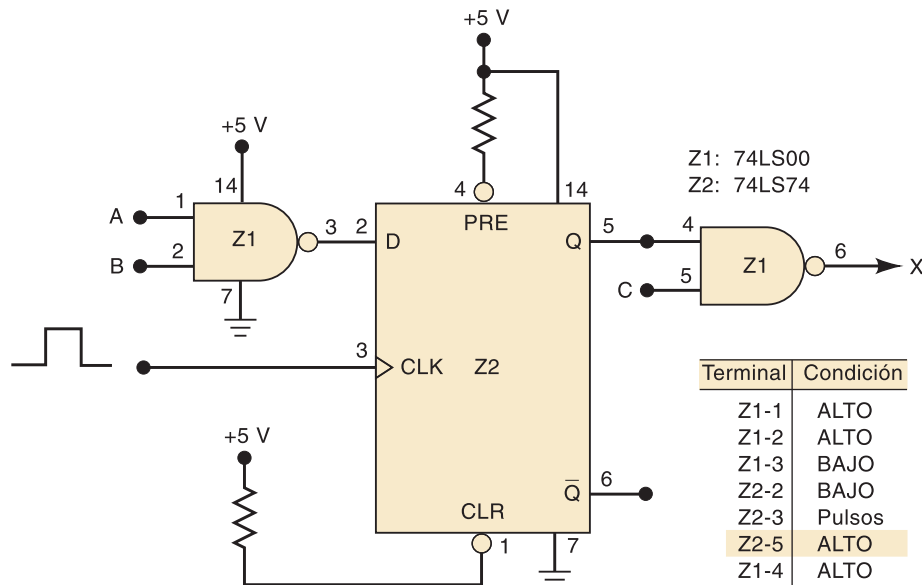
El siguiente ejemplo ilustra cómo una falla en un circuito con FF puede provocar un síntoma confuso, por el cual se requiera más tiempo para aislar la falla.

EJEMPLO 5-20

Considere el circuito de la figura 5-58 y examine las indicaciones de la sonda lógica que se muestran en la tabla que lo acompaña. Hay un nivel BAJO en la entrada D del FF cuando se aplican pulsos a su entrada CLK , pero la salida Q no cambia al estado BAJO. El técnico que está probando este circuito considera cada una de las posibles fallas:

1. Que Z2-5 esté en corto interno con V_{CC} .
2. Que Z1-4 esté en corto interno con V_{CC} .

FIGURA 5-58 Ejemplo 5-20.



3. Que Z2-5 o Z1-4 estén en corto externo con V_{CC} .
4. Que Z2-4 esté en corto interno o externo con TIERRA. Con esto se mantendría la entrada *PRE* activa y se ignoraría la entrada *CLK*.
5. Que haya una falla interna en Z2, la cual evite que *Q* responda en forma apropiada a sus entradas.

Después de realizar las comprobaciones necesarias con el óhmetro, el técnico descarta las primeras cuatro posibilidades. También comprueba las terminales V_{CC} y TIERRA de Z2 y encuentra que tienen los voltajes apropiados. Está renuente a desoldar Z2 del circuito hasta que esté seguro de que está defectuoso, por lo que decide analizar la señal del reloj. Utiliza un osciloscopio para revisar su amplitud, frecuencia, anchura de pulso y tiempos de transición. Encuentra que todos están dentro de las especificaciones para el 74LS74. Por último concluye que Z2 está defectuoso.

Quita el chip 74LS74 y lo sustituye con otro. Para su sorpresa, el circuito con el nuevo chip se comporta de igual forma. Después de rascarse la cabeza, decide cambiar el chip de la compuerta NAND aunque no sabe por qué. Como lo esperaba, no ve ningún cambio en la operación del circuito.

Quedando aún más desconcertado, recuerda que su instructor del laboratorio de electrónica enfatizaba el valor de realizar una comprobación visual detallada en el tablero de circuitos, por lo que comienza a examinarlo con cuidado. Mientras lo hace, detecta un puente de soldadura entre las terminales 6 y 7 de Z2. Lo elimina, prueba el circuito y éste funciona de manera correcta. Explique cómo esta falla produjo la operación observada.

Solución

El puente de soldadura estaba poniendo en corto la salida $\bar{Q}$ con TIERRA. Esto significa que $\bar{Q}$ se mantuvo siempre en el nivel BAJO. Recuerde que en todos los latches y FFs las salidas $\bar{Q}$ y *Q* en su interior están conectadas en forma transversal, de manera que el nivel en una afectará al nivel en la otra. Por ejemplo, de otro vistazo a los circuitos internos para un flip-flop J-K en la figura 5-25. Observe que un nivel BAJO constante en $\bar{Q}$ mantendría un nivel BAJO en una entrada de la compuerta NAND 3, por lo que *Q* tendría que mantenerse en ALTO, sin importar las condiciones en *J*, *K* y *CLK*.

El técnico aprendió una valiosa lección acerca del diagnóstico de fallas en circuitos con FF. Aprendió que deben comprobarse ambas salidas para ver si hay fallas, aún aquellas que no están conectadas a otros dispositivos.

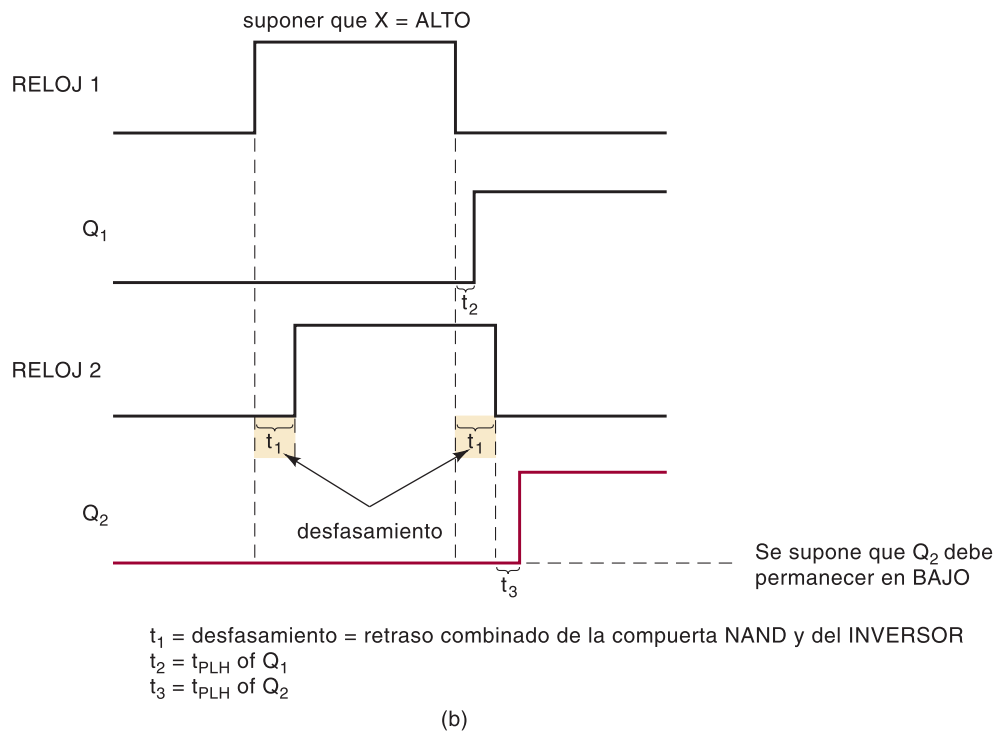
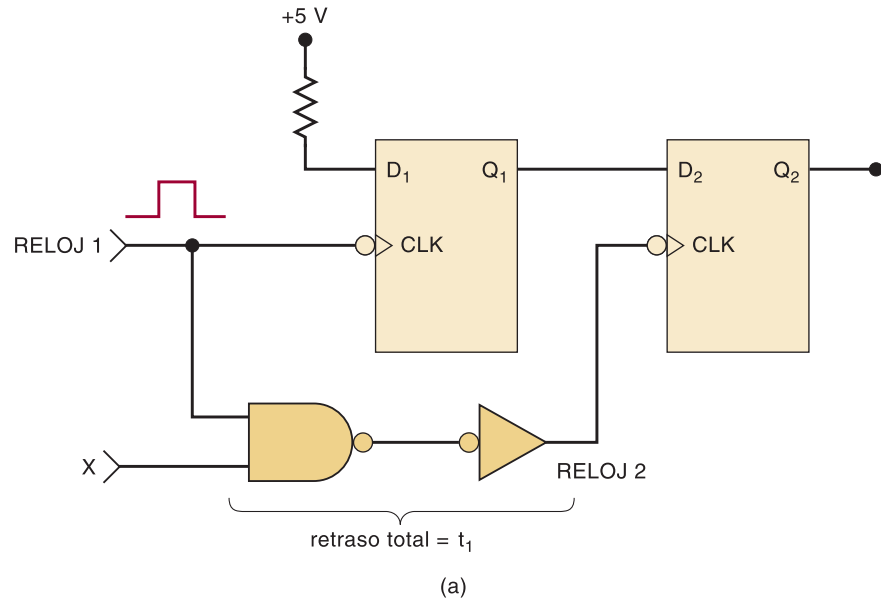
Desfasamiento de reloj

Uno de los problemas de sincronización más comunes en los circuitos secuenciales es el **desfasamiento de reloj**. Esto ocurre cuando una señal de reloj, debido a los tiempos de propagación, llega a las entradas *CLK* de distintos FFs en distintos tiempos. En muchas situaciones el desfasamiento puede provocar que un FF cambie a un estado incorrecto. Esto se ilustra mejor con un ejemplo.

Consulte la figura 5-59(a), en donde la señal *RELOJ1* está conectada en forma directa al FF Q_1 y en forma indirecta a Q_2 por medio de una compuerta NAND y un INVERSOR. Se supone que ambos FFs deben estar sincronizados por la ocurrencia de una NGT de *RELOJ1*, siempre y cuando *X* esté en ALTO. Si suponemos que al principio $Q_1 = Q_2 = 0$ y *X* = 1, la NGT de *RELOJ1* debería hacer que $Q_1 = 1$ y no tendría efecto sobre Q_2 . Las formas de onda en la figura 5-59(b) muestran cómo el desfasamiento del reloj puede producir un disparo incorrecto de Q_2 .

Debido a la combinación de los tiempos de propagación de la compuerta NAND y del INVERSOR, las transiciones de la señal *RELOJ2* se retrasan con respecto a *RELOJ1* por una cantidad de tiempo t_1 . La NGT de *RELOJ2* llega a la entrada *CLK* de Q_2 un tiempo t_1 después de que la NGT de *RELOJ1* aparece en la entrada *CLK* de Q_1 . Este t_1 es el desfasamiento del reloj. La NGT de *RELOJ1* hará que Q_1

FIGURA 5-59 El desfasamiento del reloj se produce cuando dos flip-flops que se supone deben estar sincronizados en forma simultánea se sincronizan en tiempos ligeramente distintos debido a un retraso en la llegada de la señal de reloj en el segundo flip-flop. (a) Circuitos de compuerta adicionales que pueden provocar un desfasamiento del reloj; (b) sincronización que muestra la llegada tardía de RELOJ 2.



cambie a ALTO después de un tiempo t_2 , el cual es igual al retraso de propagación t_{PLH} de Q_1 . Si t_2 es menor que el desfasamiento t_1 , Q_1 estará en ALTO cuando ocurra la NGT de RELOJ2 y esto podría provocar que Q_2 se estableciera de forma incorrecta en 1 ($Q_2 = 1$) si se cumple su requerimiento de tiempo de estabilización, t_s .

Por ejemplo, suponga que el desfasamiento del reloj es de 40 ns y que el t_{PLH} de Q_1 es de 25 ns. Por ende, Q_1 cambiará a ALTO 15 ns antes de la NGT de RELOJ2. Si el requerimiento de tiempo de estabilización de Q_2 es menor de 15 ns, Q_2 responderá al nivel ALTO en su entrada D cuando ocurra la NGT de RELOJ2, y Q_2 cambiará a ALTO. Desde luego que ésta no es la respuesta esperada de Q_2 . Se supone que debe permanecer en BAJO.

Los efectos del desfase del reloj no siempre son fáciles de detectar, ya que la respuesta del FF afectado puede ser intermitente (algunas veces funciona bien, otras no). Esto se debe a que la situación es dependiente de los tiempos de propagación del circuito y de los parámetros de sincronización del FF, los cuales varían con la temperatura, la longitud de las conexiones, el voltaje de suministro de energía y la carga. Algunas veces con sólo conectar una sonda de osciloscopio a la salida de un FF o de una compuerta se agrega la suficiente capacitancia de carga como para incrementar el retraso de propagación del dispositivo de manera que el circuito funcione en forma correcta; después cuando se quita la sonda, reaparece la operación incorrecta. Éste es el tipo de situación que explica por qué algunos técnicos envejecen en forma prematura.

Los problemas ocasionados por el desfase del reloj pueden eliminarse si se ecualizan los retrasos en las diversas rutas de la señal del reloj, de manera que la transición activa llegue a cada FF casi al mismo tiempo. En el problema 5-52 examinaremos esta situación.

PREGUNTAS DE REPASO

1. ¿Qué es el desfase del reloj? ¿Cómo puede ocasionar un problema?

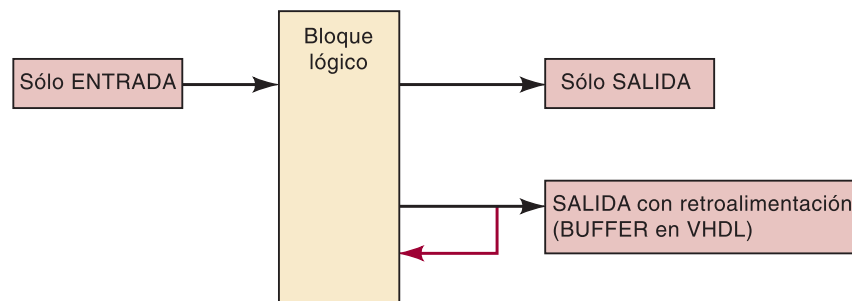
5-25 CIRCUITOS SECUENCIALES MEDIANTE EL USO DE HDL*

En los capítulos 3 y 4 utilizamos el HDL para programar circuitos lógicos combinacionales simples. En este capítulo hemos estudiado circuitos lógicos tipo latch y circuitos de flip-flop sincronizado por reloj que avanzan en secuencia a través de varios estados, en respuesta a un flanco del reloj. Estos circuitos de latch y secuenciales también pueden implementarse mediante el uso de los PLDs, y también pueden describirse mediante el uso de HDL.

En la sección 5-1 de este capítulo describimos un match de compuerta NAND. Recordará que la característica única de este circuito es el hecho de que sus salidas están conectadas en forma transversal de vuelta con las entradas de sus compuertas. Esto hace que el circuito responda de manera distinta, dependiendo del estado en el que se encuentre su salida. Para describir circuitos que tienen salidas que se *retroalimentan* a la entrada con ecuaciones booleanas o mediante HDL se requiere utilizar las variables de salida en la porción condicional de la descripción. Con las ecuaciones booleanas esto significa incluir los términos de salida en el lado derecho de la ecuación. Al utilizar instrucciones IF/THEN se deben incluir las variables de salida en la cláusula IF. La mayoría de los PLDs tienen la habilidad de retroalimentar la señal de salida a los circuitos de entrada, para poder acomodar la acción del latch.

Al escribir ecuaciones que utilizan retroalimentación, algunos lenguajes tales como el VHDL requieren una designación especial para el puerto de salida. En estos casos, el bit de puerto no es sólo una salida; es una salida con retroalimentación. La diferencia se muestra en la figura 5-60.

FIGURA 5-60 Tres modos de entrada/salida.



* Como se dijo en el capítulo 3, puede omitir esta sección y todas las secciones que tratan acerca de PLDs y HDLs si lo desea.

En vez de describir la operación de un latch mediante ecuaciones Booleanas, tratemos de pensar en una descripción del comportamiento en cuanto a la operación del latch. Las situaciones que necesitamos abordar son cuando se activa SBAR, cuando se activa RBAR y cuando no se activa ninguna de las dos. Recuerde que el estado inválido ocurre cuando ambas entradas se activan al mismo tiempo. Si podemos describir un circuito que siempre reconoce una de las entradas como la ganadora cuando ambas están activas, podremos evitar los resultados indeseables de tener una condición de entrada inválida. Para describir tal circuito, tenemos que hacernos la pregunta acerca de bajo qué condiciones debe establecerse el latch ($Q = 1$). Es evidente que el latch debe establecerse si la entrada SET está activa, pero ¿qué pasa después de que SET regresa a su nivel inactivo? ¿Cómo sabe el latch que debe permanecer en el estado SET? La descripción necesita utilizar la *condición de la salida en estos momentos* para determinar la condición *futura* de la misma. La siguiente instrucción describe las condiciones que deben hacer que la salida cambie a ALTO en un latch SR:

SI (IF) ESTABLECER está activa, ENTONCES (THEN) Q debe estar en ALTO.

¿Qué condiciones deben hacer que la salida cambie a BAJO?

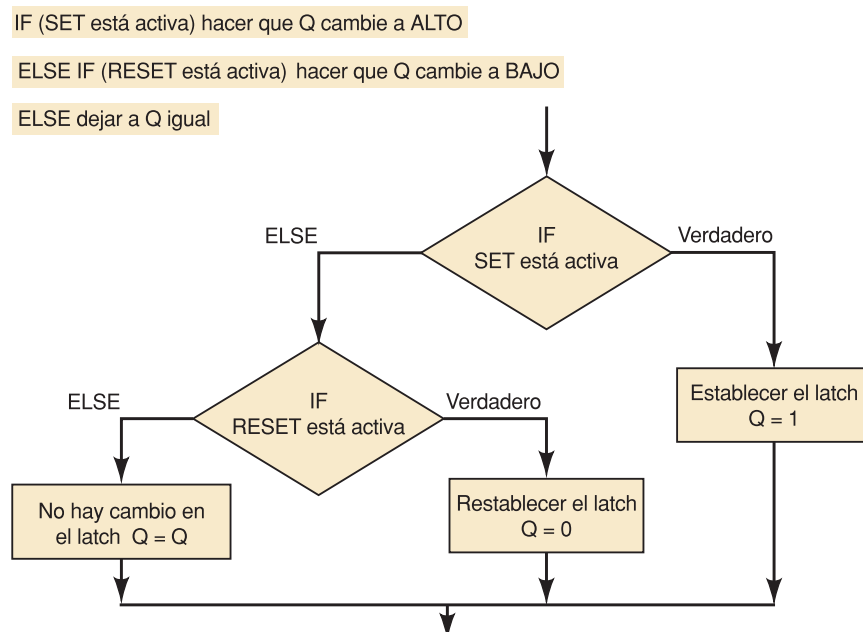
SI (IF) RESET está activa, ENTONCES (THEN) Q debe estar en BAJO.

¿Qué pasa si no se activa ninguna entrada? Entonces la salida deberá permanecer igual y podemos expresar esto como $Q = Q$. Esta expresión proporciona la retroalimentación del estado de salida para combinarla con las condiciones de entrada, con el fin de decidir lo que va a ocurrirle a continuación a la salida.

¿Qué pasa si se activan ambas entradas (es decir, la combinación de entrada inválida)? La estructura de la decisión IF/ELSE que se muestra en forma gráfica en la figura 5-61 se asegura de que el latch nunca trate de responder a ambas entradas. Si SET está activa, sin importar lo que haya en RESET la salida se verá forzada a cambiar a ALTO. De esta forma, la entrada inválida siempre tendrá como valor predeterminado la condición de SET. La cláusula ELSIF se considera sólo cuando SET no está activa. El uso del término de retroalimentación ($Q = Q$) afecta a la operación (acción de retención) sólo cuando ninguna de las entradas están activas.

Cuando diseñamos circuitos secuenciales que alimentan el valor de salida de vuelta a las entradas, existe la posibilidad de crear un sistema inestable. Un cambio en el estado de salida podría alimentarse de vuelta a las entradas, con lo

FIGURA 5-61 La lógica de una descripción del comportamiento de un latch SR.



cual se cambiaría de nuevo el estado de la salida, el cual se retroalimentaría a las entradas, con lo que se cambiaría la salida de nuevo. Es obvio que esta oscilación es indeseable, por lo cual es muy importante asegurarnos de que ninguna combinación de entradas y salidas pueda hacer que esto ocurra. Debemos utilizar un proceso cuidadoso de análisis, simulación y prueba para asegurarnos que nuestro circuito sea estable bajo todas las condiciones. Para este circuito es necesario habilitar la síntesis de multinivel para la serie MAX 7000 de componentes antes de realizar la compilación, con lo cual se evita la oscilación (por lo menos en el simulador) cuando se cambia de la entrada “inválida” a la entrada “sin cambios”.

EJEMPLO 5-21

Describe un match SR con entradas activas en BAJO identificadas como SBAR, RBAR y una salida llamada *Q*. Debe seguir la tabla de funciones de un latch NAND (vea la figura 5-6) y la combinación de entrada inválida debe producir $Q = 1$.

- (a) Use AHDL.
- (b) Use VHDL.

Solución

- (a) La figura 5-62 muestra una posible solución en AHDL. Los puntos importantes a observar son:

1. *Q* se define como una SALIDA, aún y cuando se *retroalimenta* en el circuito. AHDL permite que las salidas se retroalimenten al circuito.
2. La cláusula después del IF determinará el estado de salida que se produzca cuando ambas entradas estén activas (estado inválido). En este código el comando SET es el que predomina.
3. Para evaluar la igualdad se utiliza el doble signo de igual. En otras palabras, $SBAR == 0$ se evalúa como VERDADERO cuando SBAR está activa (BAJO).

```
SUBDESIGN fig5_62
(
    sbar, rbar      :INPUT;
    q               :OUPUT;
)
BEGIN
    IF    sbar == 0 THEN q = VCC;    -- establecer o comando ilegal
    ELSIF rbar == 0 THEN q = GND;    -- restablecer
    ELSE                q = q;      -- retención
    END IF;
END;
```

FIGURA 5-62 Un latch NAND mediante el uso de AHDL.

- (b) La figura 5-63 muestra una posible solución en VHDL. Los puntos importantes a observar son:

1. *Q* se define como BUFFER en lugar de OUTPUT. Esto le permite retroalimentarse al circuito.
2. La palabra PROCESS describe lo que ocurre cuando los valores en la lista de sensibilidad (SBAR, RBAR) cambian de estado.
3. La cláusula después de IF determinará el estado de salida que se produzca cuando ambas entradas estén activas (estado inválido). En este código el comando SET es el que predomina.

```
-- debe compilarse con Síntesis de Multinivel para MAX 7000 habilitada
ENTITY fig5_63 IS
PORT (sbar, rbar      :IN BIT;
      Q                :BUFFER BIT);
END fig5_63;
ARCHITECTURE comportamiento OF fig5_63 IS
BEGIN
  PROCESS (sbar, rbar)
  BEGIN
    IF      sbar = '0' THEN      q <= '1'; -- establecer o comando ilegal
    ELSIF   rbar = '0' THEN      q <= '0'; -- restablecer
    ELSE                                         q <= q;  -- retención
    END IF;
  END PROCESS;
END comportamiento;
```

FIGURA 5-63 Un latch NAND mediante el uso de VHDL.

El latch *D*

El latch *D* transparente también puede implementarse con facilidad mediante los HDLs. El software de Altera cuenta con una primitiva de biblioteca llamada LATCH. El módulo de AHDL que se muestra a continuación ilustra el uso de esta primitiva LATCH. Todo lo que se necesita es conectar los puertos de habilitación (.ena) y de datos (.d) de la primitiva con las señales del módulo apropiado. El módulo VHDL que se muestra también a continuación es una descripción del comportamiento de la función del latch *D*. También puede usar la primitiva LATCH como un componente en VHDL.

AHDL D latch

```
SUBDESIGN latchd_ahdl
(habilita, dent      :INPUT;
 q                  :OUTPUT;)

VARIABLE
  q                  :LATCH;
BEGIN
  q.ena = habilita;
  q.d = dent;
END;
```

VHDL D latch

```
ENTITY latchd_vhdl IS
PORT (habilita, dent      :IN BIT;
      q                  :OUT BIT);
END latchd_vhdl;

ARCHITECTURE v OF latchd_vhdl IS
BEGIN
  PROCESS (habilita, dent)
  BEGIN
    IF habilita = '1' THEN
      q <= dent;
    END IF;
  END PROCESS;
END v;
```

PREGUNTAS DE REPASO

1. ¿Cuál es la característica distintiva de hardware de los circuitos lógicos con latches?
2. ¿Cuál es la principal característica de los circuitos secuenciales?

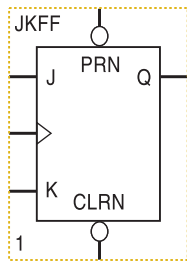


FIGURA 5-64
Primitiva lógica del
flip-flop J-K.

5-26 DISPOSITIVOS DISPARADOS POR FLANCO

En secciones anteriores de este capítulo presentamos los dispositivos disparados por flanco, cuyas salidas responden a las entradas cuando la entrada de reloj ve un “flanco”, el cual lo único que indica es una transición de ALTO a BAJO o viceversa, y a menudo se le llama **evento**. Si en el código vamos a escribir instrucciones concurrentes, ¿cómo pueden cambiar las salidas sólo cuando una entrada de reloj detecta un evento de flanco? La respuesta a esta pregunta difiere en forma sustancial, dependiendo del HDL que usted utilice. En esta sección queremos concentrarnos en la creación de circuitos lógicos sincronizados por reloj en su forma más simple, mediante el uso de HDL. Utilizaremos flip-flops J-K para correlacionar con muchos de los ejemplos que vimos antes en este capítulo.

El flip-flop J-K es un bloque de construcción estándar de circuitos lógicos sincronizados por reloj (secuenciales), al cual se le conoce como **primitiva lógica**. En su forma más común tiene cinco entradas y una salida, como se muestra en la figura 5-64. Se pueden estandarizar los nombres de las entradas/salidas para poder hacer referencia a las conexiones de este circuito primitivo o fundamental. La operación real del circuito primitivo se define en una biblioteca de componentes, la cual está disponible para el compilador de HDL a medida que éste genera un circuito a partir de nuestra descripción. AHDL utiliza primitivas lógicas para describir la operación de un flip-flop. VHDL ofrece algo similar, pero también permite al diseñador describir la operación del circuito lógico sincronizado por reloj de manera explícita en el código.

FLIP-FLOPS EN AHDL

Para utilizar un flip-flop en AHDL hay que declarar un registro (de hecho a un flip-flop se le conoce como registro). Hay varios tipos de primitivas de registros disponibles en AHDL, incluyendo JKFF, DFF, SRFF y latch. Cada tipo distinto de primitiva de registros tiene sus propios nombres oficiales (de acuerdo con el software de Altera) para los puertos de estas primitivas. Para encontrarlos podemos usar el menú HELP (ayuda) en el software de ALTERA y buscar bajo Primitives (primitivas). La tabla 5-3 lista algunos de estos nombres. Los registros que utilizan estas primitivas se declaran en la sección VARIABLE del código. Al registro se le da un nombre de instancia, de la misma forma como hemos nombrado a las variables intermedias o a los nodos ocultos en ejemplos anteriores. No obstante, en vez de declararlo como nodo se declara con base en el tipo de primitiva del registro. Por ejemplo, un flip-flop JK puede declararse como:

```
VARIABLE
    ff1 :JKFF;
```

TABLA 5-3
Identificadores de
puertos de primitiva
de Altera.

Función de pieza estándar	Nombre de puerto de primitiva
Entrada de reloj	clk
PRESET asíncrono (activo en BAJO)	prn
CLEAR asíncrono (activo en BAJO)	clrn
Entradas J, K, S, R, D	j, k, s, r, d
Entrada HABILITAR disparada por nivel	ena
Salida Q	Q

El nombre de instancia es ff1 (usted puede inventarlo) y el tipo de primitiva de registro es JKFF (es requerido por Altera). Una vez declarado un registro, éste se conecta con la demás lógica en el diseño mediante sus nombres de puertos estándar.


```

1  %      JK circuito flip-flop      %
2  SUBDESIGN fig5_65
3  (
4      jent, kent, clkent, preset, clear      :INPUT
5      qsal                                  :OUTPUT;
6  )
7  VARIABLE
8  ff1      :JKFF;      -- define este flip flop como de tipo JKFF
9  BEGIN
10     ff1.prn = preset; -- éstos son opcionales, su valor predeterminado es vcc
11     ff1.clrn = clear;
12     ff1.j = jent;      -- conecta primitiva a la señal de entrada
13     ff1.k = kent;
14     ff1.clk = clkent;
15     qsal = ff1.q;      -- conecta la terminal de salida a la primitiva
16  END;

```

FIGURA 5-65 Flip-flop JK individual mediante el uso de AHDL.

Para hacer referencia a los puertos (o terminales) en el flip-flop se utiliza el nombre de instancia, con una extensión después del punto que designa la entrada o salida específica. La figura 5-65 muestra un ejemplo para un flip-flop JK en AHDL. Observe que hemos creado nuestros propios nombres para las entradas/salidas en este SUBDESIGN, para poder distinguirlos de los nombres de los puertos de las primitivas. El flip-flop individual se declara en la línea 8, como se describió antes. La entrada o puerto J para este dispositivo se etiqueta entonces como *ff1.j*, la entrada K es *ff1.k*, la entrada de reloj es *ff1.clk* y así en lo sucesivo. Cada una de las instrucciones de asignación de puertos dadas realizarán las conexiones de alambrado necesarias para este bloque de diseño. Los puertos *prn* y *clrn* son controles asíncronos activos en BAJO, como los que se encuentran, por lo general, en un flip-flop estándar. De hecho, estos controles asíncronos en una primitiva de FF pueden utilizarse para implementar un latch SR con más eficiencia que el código en la figura 5-62. Los controles *prn* y *clrn* son opcionales en AHDL y su valor predeterminado es una condición deshabilitada (en un 1 lógico) si se omiten de la sección lógica. En otras palabras, si se eliminaran las líneas 10 y 11 los puertos *prn* y *clrn* de *ff1* se enlazarían en forma automática a V_{CC} .

COMPONENTES DE LAS BIBLIOTECAS DE VHDL

El software de Altera incluye varias bibliotecas extensas de componentes y primitivas para que las utilicen los diseñadores. En la figura 5-66(a) se muestra la descripción gráfica de un componente JKFF en la biblioteca Altera. Después de colocar el componente en la hoja de trabajo, cada uno de sus puertos se conecta a las entradas y salidas del módulo. Este mismo concepto puede implementarse en VHDL mediante el uso de un componente de biblioteca. Para encontrar las entradas y salidas de estos componentes de biblioteca hay que buscar bajo el menú HELP/Primitives. La figura 5-66(b) muestra la declaración **COMPONENT** en VHDL para una primitiva de flip-flop JK. Los puntos clave a observar son el nombre del componente (JKFF) y los nombres de los puertos. Son los mismos nombres que los que se utilizan en el símbolo gráfico de la figura 5-66(a). Observe además que el *tipo* de cada una de las variables de entrada y salida es STD_LOGIC. Éste es uno de los tipos de datos del estándar IEEE definidos en la biblioteca, que muchos componentes de la misma utilizan.

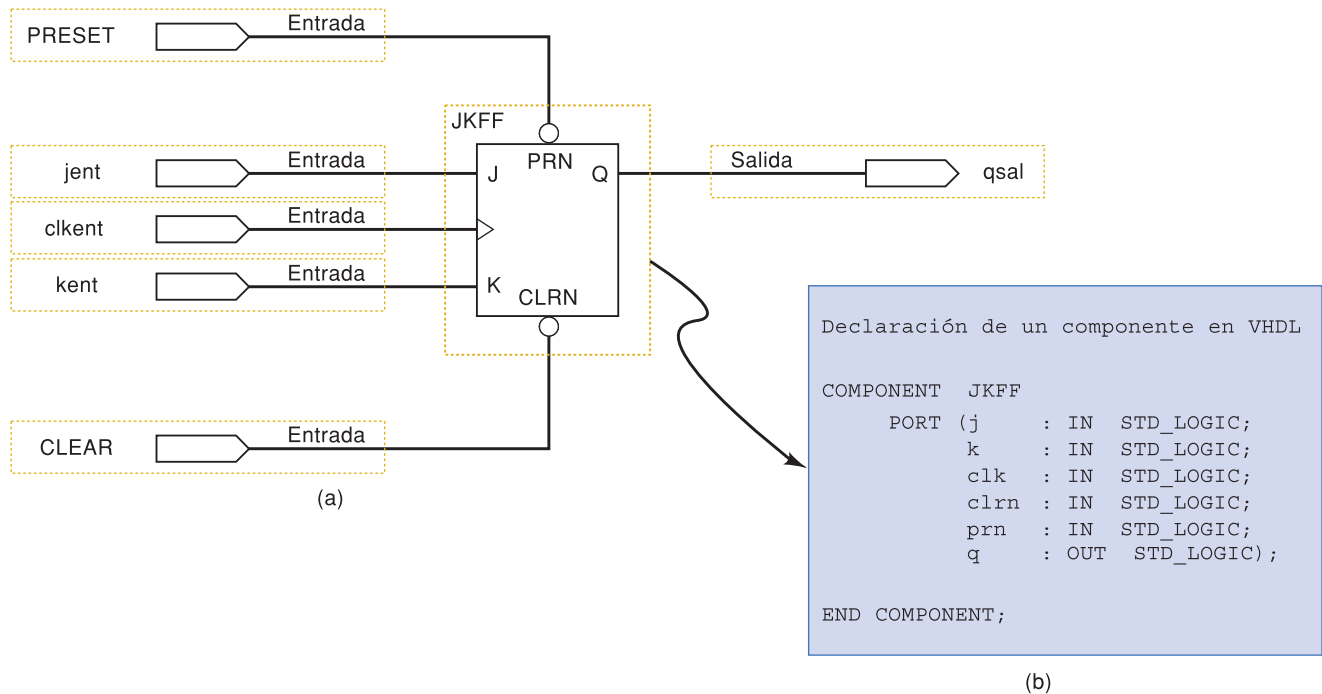


FIGURA 5-66 (a) Representación gráfica mediante el uso de un componente. (b) Declaración de un componente en VHDL.

La figura 5-67 utiliza un componente JKFF de la biblioteca en VHDL para crear un circuito equivalente al diseño gráfico de la figura 5-66(a). Las primeras dos líneas indican al compilador que debe usar la biblioteca IEEE para encontrar las definiciones de los tipos de datos `std_logic`. Las siguientes dos líneas indican al compilador que deberá buscar en la biblioteca Altera cualquier componente de biblioteca estándar que se utilice más adelante en el código. Las entradas y salidas del módulo se declaran como en los ejemplos anteriores, sólo que ahora el tipo es `STD_LOGIC` en vez de `BIT`. Esto se debe a que los tipos de los puertos de los módulos deben concordar con los tipos de los puertos de los componentes. Dentro de la sección de arquitectura se proporciona un nombre (`ff1`) a esta instancia del com-

```

LIBRARY ieee;
USE ieee.std_logic_1164.all; -- define los tipos std_logic
LIBRARY altera;
USE altera.maxplus2.all; -- provee los componentes estándar

ENTITY fig5_67 IS
  PORT( clkent, jent, kent, preset, clear :IN std_logic;
        Qsal :OUT std_logic);
END fig5_67;

ARCHITECTURE a OF fig5_67 IS
  BEGIN
    ff1: JKFF PORT MAP ( clk => clkent,
                          j   => jent,
                          k   => kent,
                          prn => preset,
                          clrn => clear,
                          q   => Qsal);
  END a;

```

FIGURA 5-67 Un flip-flop JK mediante el uso de VHDL.

ponente JKFF. Las palabras clave **PORT MAP** van seguidas de una lista de todas las conexiones que deben hacerse con los puertos del componente. Observe que los puertos del componente (por ejemplo, clk) se listan a la izquierda del símbolo => y que los objetos a los que se conectan (por ejemplo, clkent) se listan a la derecha.

FLIP-FLOPS EN VHDL

Ahora que hemos visto cómo usar los componentes estándar disponibles en la biblioteca, veamos cómo crear nuestro propio componente que pueda usarse una y otra vez. Para simplificar la comparación describiremos el código VHDL para un flip-flop JK que es idéntico al componente de biblioteca JKFF.

VHDL se creó para ser un lenguaje muy flexible, ya que nos permite definir la operación de dispositivos sincronizados por reloj en forma explícita en el código, sin depender de las primitivas lógicas. La clave para los circuitos secuenciales disparados por flanco en VHDL es **PROCESS**. Como recordará, esta palabra clave va seguida de una lista de sensibilidad entre paréntesis. Cada vez que una de las variables de la lista de sensibilidad cambia de estado, el código en el bloque del proceso (**PROCESS**) determina cómo debe responder el circuito. Esto es muy parecido a un flip-flop que no hace nada sino hasta que la entrada de reloj cambie de estado, momento en el cual evalúa sus entradas y actualiza sus salidas. Si el flip-flop necesita responder a entradas distintas de la entrada de reloj (por ejemplo, **PRESET** y **CLEAR**), éstas pueden agregarse a la lista de sensibilidad. El código en la figura 5-68 demuestra un flip-flop JK escrito en VHDL.

En la línea 9 de la figura se declara una señal con el nombre de *gestado*. Las señales pueden considerarse como alambres que conectan dos puntos en la descripción del circuito, pero también tienen características de “memoria” implícita. Esto significa que una vez que se asigne un valor a la señal, permanecerá en ese valor hasta que se le asigne un valor distinto en el código. En VHDL se utiliza a menudo la palabra clave **VARIABLE** para implementar esta característica de “memoria”,

```

1  -- Circuito de Flip-flop JK
2  ENTITY jk IS
3  PORT(
4      clk, j, k, prn, clrn :IN BIT;
5      q                     :OUT BIT);
6  END jk ;
7
8  ARCHITECTURE a OF jk IS
9      SIGNAL gestado :BIT;
10 BEGIN
11     PROCESS(clk, prn, clrn)    -- responde a cualquiera de estas señales
12     BEGIN
13         IF prn = '0' THEN gestado <= '1'; -- PRESET asíncrono
14         ELSIF clrn = '0' THEN gestado <= '0';-- CLEAR asíncrono
15         ELSIF clk = '1' AND clk'EVENT THEN -- en PGT del flanco del reloj
16             IF j = '1' AND k = '1' THEN gestado <= NOT gestado;
17             ELSIF j = '1' AND k = '0' THEN gestado <= '1';
18             ELSIF j = '0' AND k = '1' THEN gestado <= '0';
19             END IF;
20         END IF;
21     END PROCESS;
22     q <= gestado;              -- actualiza terminal de salida
23 END a;
```

FIGURA 5-68 Flip-flop JK individual mediante el uso de VHDL.

pero las variables deben declararse y utilizarse dentro del mismo bloque de descripción. En este ejemplo, si *qestado* se declarara como VARIABLE, tendría que declararse dentro de PROCESS (después de la línea 11) y debería asignarse a *q* antes del final de PROCESS (línea 21). Nuestro ejemplo utiliza una señal mediante la directiva SIGNAL, la cual puede declararse y usarse a lo largo de toda la descripción de la arquitectura.

Observe que la lista de sensibilidad de PROCESS contiene las señales asíncronas PRESET y CLEAR. El flip-flop debe responder a estas entradas tan pronto como se asignen (BAJO), y estas entradas deben predominar sobre las entradas *J*, *K* y *CLK*. Para lograr esto podemos usar la naturaleza secuencial de las instrucciones IF/ELSE. Primero la directiva PROCESS describirá lo que ocurre sólo cuando una de las tres señales (*clk*, *prn* o *clrn*) cambia de estado. En este ejemplo, la entrada de mayor prioridad es *prn* ya que se evalúa primero en la línea 13. Si se asigna, *qestado* cambiará a ALTO y todas las demás entradas ni siquiera se evaluarán, ya que se encuentran en la ramificación ELSE de la decisión. Si *prn* está en ALTO, *clrn* se evaluará en la línea 14 para ver si está en BAJO. De ser así, el flip-flop se borrará y no se evaluará nada más en el bloque PROCESS. La línea 15 se evaluará sólo si tanto *prn* como *clrn* están en ALTO. El término *clk'EVENT* en la línea 5 se evalúa como VERDADERO sólo si se ha producido una transición en *clk*. Como *clk* = '1' debe ser también VERDADERO, esta condición sólo responde a una transición del reloj en el flanco de subida. Las siguientes tres condiciones de las líneas 16, 17 y 18 se evalúan sólo después de un flanco de subida en *clk* y sirven para actualizar el estado del flip-flop. En otras palabras, están **anidadas** dentro de la instrucción ELSIF de la línea 15. Sólo se evalúan los comandos de las entradas JK para conmutación, establecer y restablecer mediante la instrucción IF/ELSIF de las líneas 16-18. Desde luego que en un JK hay un cuarto comando: retención. La condición ELSE "faltante" se interpretará en VHDL como un dispositivo de memoria implícito que retendrá el estado PRESENTE si ninguna de las condiciones del JK dadas es VERDADERA. Observe que cada estructura IF/ELSIF tiene su propia instrucción END IF. La línea 19 termina la estructura de decisión que se encarga de establecer, borrar o conmutar. La línea 20 termina la estructura IF/ELSIF que decide entre las respuestas de PRESET, CLEAR y flanco del reloj. Tan pronto como termina el bloque PROCESS, el estado del flip-flop se transfiere al puerto de salida *q*.

Sin importar si usted desarrolla su descripción en AHDL o en VHDL, puede verificar que el circuito opere en forma apropiada mediante el uso de un simulador. La parte más importante y retadora de la verificación por medio de un simulador es la creación de un conjunto de condiciones de entrada hipotéticas que demostrarán que el circuito hace todo lo que debe hacer. Hay muchas formas de hacer esto; es decisión del diseñador elegir la mejor. En la figura 5-69 se muestra la simulación que se utiliza para verificar la operación de la primitiva JKFF. La entrada *PRESET* se activa en un principio y después, en *t1*, se activa la entrada *CLEAR*. Estas pruebas aseguran que *PRESET* y *CLEAR* funcionen en forma asíncrona. La entrada *jent* está en ALTO en *t2* y *kent* está en ALTO en *t3*. Entre estos puntos, las entradas *jent* y *kent* están en BAJO. Esta porción de la simulación prueba los modos síncronos de establecer, retención y PRESET. Empezando en *t4*, se prueba el comando de conmutación

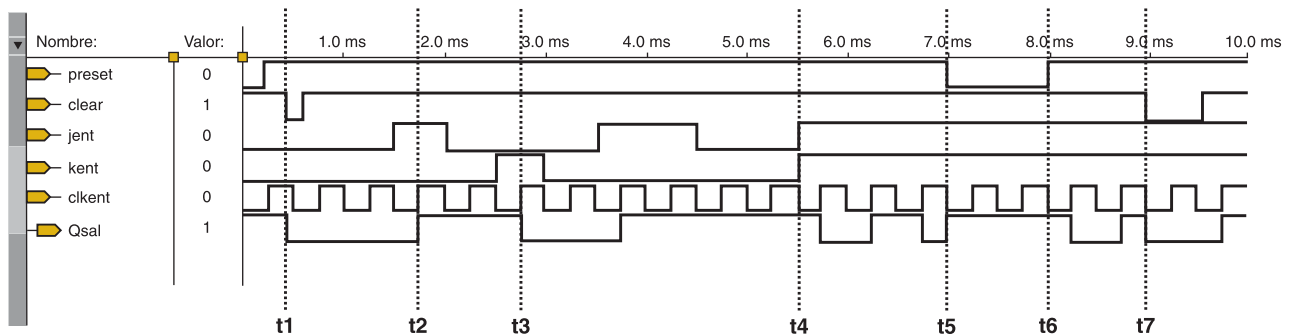


FIGURA 5-69 Simulación del flip-flop JK.

con $jent = kent = 1$. Observe que en $t5$ *PRESET* se asigna (BAJO) para evaluar si *PRESET* predomina sobre el comando de conmutación. Después de $t6$ la salida empieza a conmutar de nuevo, y en $t7$ se muestra cómo la entrada *CLEAR* predomina sobre las entradas síncronas. Es muy importante probar todos los modos de operación y la interacción de varios controles cuando se está simulando un circuito.

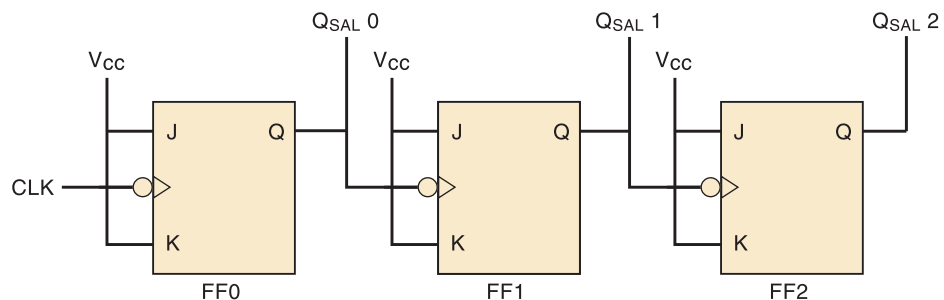
PREGUNTAS DE REPASO

1. ¿Qué es una primitiva lógica?
2. ¿Qué necesita saber el diseñador para poder utilizar una primitiva lógica?
3. En el sistema de Altera, ¿en dónde puede encontrar información acerca de las primitivas y funciones de biblioteca?
4. ¿Cuál es el elemento clave en VHDL que permite la descripción explícita de los circuitos lógicos sincronizados por reloj?
5. ¿Cuál es la biblioteca que define los tipos de datos *std_logic*?
6. ¿Cuál es la biblioteca que define las primitivas lógicas y los componentes comunes?

5-27 CIRCUITOS DE HDL CON VARIOS COMPONENTES

Empezamos este capítulo con el estudio de los latches. Éstos se utilizan para crear flip-flops y éstos, a su vez, para crear muchos circuitos, incluyendo los contadores binarios. En la figura 5-70 se muestra una descripción gráfica (diagrama lógico) de un simple contador binario ascendente. En cuanto a su funcionalidad, este circuito es el mismo que el de la figura 5-47, que se dibujó con el LSB a la derecha para facilitar la visualización del valor numérico del conteo binario. Aquí hemos vuelto a dibujar el circuito para mostrar el flujo de las señales en el formato más convencional, con entradas a la izquierda y salidas a la derecha. Observe que estos símbolos lógicos se disparan en los flancos negativos. Además estos flip-flops tampoco tienen las entradas asíncronas *prn* o *clrn*. Nuestro objetivo es describir este circuito contador mediante el uso de HDL; para ello interconectamos tres instancias del mismo componente flip-flop JK.

FIGURA 5-70
Contador binario de tres bits.



CONTADOR DE RIZO ASCENDENTE

Una descripción basada en texto de este circuito requiere tres flip-flops del mismo tipo, al igual que la descripción gráfica. Consulte la figura 5-71. En la línea 8 de esta figura se utiliza la notación en arreglo de bits para declarar un registro de tres flip-flops JK. El nombre del registro es *q*, al igual que el nombre del puerto de salida. AHDL puede interpretar que esto indica que la salida de cada flip-flop debe

```

1  % Contador de rizados MOD 8 ascendente %
2  SUBDESIGN fig5_71
3  (
4      reloj                :INPUT;
5      q[2..0]              :OUTPUT;
6  )
7  VARIABLE
8      q[2..0] :JKFF;        -- define tres FFs JK
9  BEGIN
10                                     -- nota: valor determinado de prn, clrn es vcc
11      q[2..0].j = VCC;        -- modo de conmutación J=K=1 para todos los FFs
12      q[2..0].k = VCC;
13      q[0].clk = !reloj;
14      q[1].clk = !q[0].q;
15      q[2].clk = !q[1].q;    -- conecta los relojes en forma de rizo
16  END;

```

FIGURA 5-71 Contador de rizados MOD-8 en AHDL.

conectarse al puerto de salida. Cada bit del arreglo *q* tiene todos los atributos de una primitiva JKFF. AHDL es muy flexible en cuanto al uso de los conjuntos indizados como éste. Como ejemplo del uso de esta notación de conjunto, observe cómo las entradas J y K para todos los flip-flops se enlazan a VCC en las líneas 11 y 12. Si los flip-flops se hubieran llamado A, B y C en lugar de usar un arreglo de bits, sería necesario utilizar asignaciones individuales para cada entrada J y K, con lo cual el código sería más extenso. A continuación se hacen las interconexiones clave entre los flip-flops para crear el contador de rizo ascendente. La señal de reloj se invierte y se asigna a la entrada de reloj de FF0 (línea 13), la salida Q de FF0 se invierte y se asigna a la entrada de reloj de FF1 (línea 14) y así en lo sucesivo, para formar un contador de rizo.

CONTADOR DE RIZO ASCENDENTE EN VHDL

En la figura 5-68 describimos el código de VHDL para un JKFF disparado con flanco positivo, con los controles PRESET y CLEAR. El contador de la figura 5-70 se dispara con los flancos negativos y no requiere de los controles asíncronos PRESET o CLEAR. Nuestro objetivo ahora es escribir el código de VHDL para uno de estos flip-flops, representar tres instancias del mismo flip-flop e interconectar los puertos para crear el contador.

Primero vamos a analizar la descripción de VHDL en la figura 5-72, comenzando con la línea 18. Este módulo de VHDL está describiendo la operación de un componente flip-flop JK individual. El nombre del componente es *jk_neg* (línea 18); tiene las entradas *clk*, *j*, *k* (línea 19) y la salida *q* (línea 20). Una señal llamada *qestado* se utiliza para retener el estado del flip-flop y conectarlo a la salida *q*. En la línea 25 el bloque PROCESS sólo tiene a *clk* en su lista de sensibilidad, por lo que sólo responde a los cambios en *clk* (PGTs y NGTs). La instrucción que hace que este flip-flop se dispare con los flancos negativos se encuentra en la línea 27. Si la instrucción IF (*clk* EVENT AND *clk* = '0') es verdadera, entonces se acaba de producir un flanco en *clk* y ahora *clk* está en BAJO, lo cual significa que debió haber sido una NGT de *clk*. Las decisiones IF/ELSE que van después implementan los cuatro estados de un flip-flop JK.


```

1  ENTITY fig5_72 IS
2  PORT (   reloj           :IN  BIT;
3          qsal             :BUFFER BIT_VECTOR (2 DOWNTO 0));
4  END fig5_72;
5  ARCHITECTURE contador OF fig5_72 IS
6      SIGNAL alto          :BIT;
7      COMPONENT jk_neg
8      PORT ( clk, j, k      :IN  BIT;
9              q             :OUT  BIT);
10     END COMPONENT;
11 BEGIN
12     alto <= '1'           -- conectado a VCC
13 ff0: jk_neg      PORT MAP (j => alto, k => alto, clk => reloj, q => qsal(0));
14 ff1: jk_neg      PORT MAP (j => alto, k => alto, clk => qsal(0), q => qsal(1));
15 ff2: jk_neg      PORT MAP (j => alto, k => alto, clk => qsal(1), q => qsal(2));
16 END contador;
17
18 ENTITY jk_neg IS
19 PORT (   clk, j, k      :IN  BIT;
20         q              :OUT  BIT);
21 END jk_neg;
22 ARCHITECTURE simple OF jk_neg IS
23     SIGNAL  gestado      :BIT;
24 BEGIN
25     PROCESS (clk)
26     BEGIN
27         IF (clk'EVENT AND clk = '0') THEN
28             IF j = '1' AND k = '1' THEN gestado <= NOT gestado;    -- conmutación
29             ELSIF j = '1' AND k = '0' THEN gestado <= '1';        -- establecer
30             ELSIF j = '0' AND k = '1' THEN gestado <= '0';        -- restablecer
31             END IF;
32         END IF;
33     END PROCESS;
34     q <= gestado          -- conecta el estado del flip-flop a la salida
35 END simple;;

```

FIGURA 5-72 Contador de rizados MOD-8 en VHDL.

Ahora que sabemos cómo funciona un flip-flop llamado `jk_neg`, veamos cómo podemos usarlo tres veces en un circuito y enlazar todos los puertos. La línea 1 define la entidad (ENTITY) que constituirá el contador de tres bits. Las líneas 2-3 contienen las definiciones de las entradas y las salidas. Observe que las salidas se encuentran en la forma de un arreglo de tres bits (vector de bits). En la línea 6 la señal (SIGNAL) `alto` puede considerarse como un alambre que se utiliza para conectar los puntos en el circuito con V_{CC} . La línea 7 es muy importante, ya que aquí es donde declaramos que planeamos utilizar un componente en nuestro diseño, cuyo nombre es `jk_neg`. En este ejemplo el verdadero código está escrito en la parte inferior de la página, pero podría estar en un archivo separado o incluso en una biblioteca. Esta declaración indica al compilador todos los datos importantes sobre el componente y los nombres de sus puertos.

La parte final de la descripción es la sección concurrente de las líneas 12-15. Primero la señal `alto` se conecta a V_{CC} en la línea 12. En las siguientes tres líneas se crean las instancias de los componentes tipo flip-flop. Las tres instancias se llaman `ff0`, `ff1` y `ff2`. Cada instancia va seguida de una instrucción `PORT MAP`, en la cual se lista cada puerto del componente y se describe con qué está conectado en el módulo.

No es difícil conectar componentes entre sí mediante el uso de HDL, pero es muy tedioso. Como podrá ver, hasta el archivo para un circuito muy simple puede ser bastante extenso. A este método de describir circuitos se le conoce como **nivel de abstracción estructural**. Requiere que el diseñador tome en cuenta cada una de las terminales de cada componente y que defina señales para cada alambre que se utilice para interconectar los componentes. Por lo general, las personas acostumbradas a utilizar diagramas lógicos para describir circuitos encuentran que es fácil comprender el nivel estructural, pero no es tan fácil de leer de un vistazo, en comparación con el diagrama del circuito lógico equivalente. De hecho, podemos decir que si el nivel estructural de descripción fuera lo único que estuviera disponible, la mayoría de las personas preferiría utilizar descripciones gráficas (esquemáticos) en vez de HDL. La verdadera ventaja del HDL está en el uso de niveles más altos de abstracción y la habilidad de manipular los componentes para que se ajusten a las necesidades exactas del proyecto. En los siguientes capítulos exploraremos el uso de estos métodos, así como las herramientas gráficas para conectar los módulos.

PREGUNTAS DE REPASO

1. ¿Puede el mismo componente utilizarse más de una vez en el mismo circuito?
2. En AHDL, ¿dónde se declaran las instancias múltiples de un componente?
3. ¿Cómo podemos diferenciar entre múltiples instancias de un componente?
4. En AHDL, ¿qué operador se utiliza para “conectar” señales?
5. En VHDL, ¿qué sirve como “alambres” para conectar componentes?
6. En VHDL, ¿qué palabra clave identifica la sección de código en donde se especifican las conexiones para las instancias de los componentes?

RESUMEN

1. Un flip-flop es un circuito lógico con una característica de memoria tal que sus salidas Q y $\bar{Q}$ cambiarán a un nuevo estado en respuesta a un pulso de entrada y permanecerán en ese nuevo estado aún después de que termine el pulso de entrada.
2. Un latch NAND y un latch NOR son simples FFs que responden a los niveles lógicos en sus entradas SET y RESET.
3. Borrar (CLEAR, RESET) un FF significa que su salida termina en el estado $Q = 0$ / $\bar{Q} = 1$. Establecer (SET) un FF significa que termina en el estado $Q = 1$ / $\bar{Q} = 0$.
4. Los FFs sincronizados por reloj tienen una entrada de reloj (CLK , CP , CK) que se dispara por flanco, lo cual significa que dispara el FF en una transición de pendiente positiva (PGT) o en una transición de pendiente negativa (NGT).
5. Los FFs disparados por flanco (sincronizados) pueden dispararse a un nuevo estado mediante el flanco activo de la entrada de reloj, de acuerdo con el estado de las entradas de control síncronas del FF (S , R o J , K o D).
6. La mayoría de los FFs sincronizados por reloj tienen entradas asíncronas que pueden establecer o borrar el FF, sin importar el estado de la entrada de reloj.
7. El latch D es un latch NAND modificado que opera como un flip-flop D , con la excepción de que no se dispara por flanco.
8. Algunos de los principales usos de los FFs son: almacenamiento y transferencia de datos, desplazamiento de datos, conteo y división de frecuencia. Se utilizan en circuitos secuenciales que siguen una secuencia predeterminada de estados.

9. Un monoestable (one-shot, OS) es un circuito lógico que puede dispararse desde su estado inactivo normal ($Q = 0$) a su estado activo ($Q = 1$), en donde permanecerá durante un intervalo de tiempo proporcional a una constante de tiempo RC .
10. Los circuitos que tienen una entrada tipo disparador de Schmitt responden en forma confiable a las señales con cambios lentos y producen salidas con flancos limpios y bien definidos.
11. Puede usarse una variedad de circuitos para generar señales de reloj a una frecuencia deseada, incluyendo los osciladores disparadores de Schmitt, un temporizador 555 y un oscilador controlado por cristal.
12. En la contraportada interior encontrará un resumen completo de los diversos tipos de FFs.
13. Los dispositivos lógicos programables pueden programarse para operar como circuitos tipo latch y circuitos secuenciales.
14. Los bloques de construcción fundamentales llamados primitivas lógicas están disponibles en la biblioteca de Altera para ayudar a implementar sistemas más grandes.
15. Los flip-flops sincronizados por reloj están disponibles como primitivas lógicas.
16. Puede escribirse código en VHDL para describir la lógica sincronizada por reloj de manera explícita, sin necesidad de usar primitivas lógicas.
17. VHDL permite utilizar archivos de HDL como componentes en sistemas más grandes. Hay componentes prefabricados en la biblioteca de Altera.
18. Podemos usar HDL para describir componentes interconectados en forma muy parecida a una herramienta gráfica de captura de diagramas esquemáticos.

TÉRMINOS IMPORTANTES

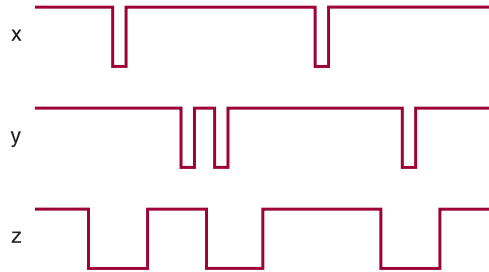
anidadas	flip-flop sincronizado por reloj en D	pulsos
bloque de control común	flip-flop sincronizado por reloj en J-K	rebote de contacto
circuito de conducción de pulso	flip-flop S-R sincronizado por reloj	registro de desplazamiento
circuito detector de flancos	frecuencia	registros
circuito disparador de Schmitt	generador de reloj controlado por cristal	reloj
circuitos secuenciales	ignorar entradas	retrasos por tiempos de propagación
CLEAR (estados/entradas)	latch D	RESET (estados/entradas)
COMPONENT	latch de compuerta NAND	retraso de propagación
contador binario	latch de compuerta NOR	SET (estados/entradas)
desfasamiento de reloj	modo de conmutación monoestable	tabla de estados
diagrama de transición de estados	(one-shot - OS)	temporizador 555
disparo	multivibrador astable o de funcionamiento libre	tiempo de estabilización, t_S
disparada por flanco	nivel estructural de abstracción	tiempo de retención, t_H
división de frecuencia	número MOD	transferencia asíncrona (atascada)
entradas asíncronas	OS no redisparable	transferencia atascada
entradas de control	OS redisparable	transferencia de datos
entradas de control síncronas	periodo	transferencia de datos en paralelo
entradas predominantes	PORT MAP	transferencia de datos en serie
estado casi estable	primitiva lógica	transferencia síncrona
EVENT		transición de pendiente negativa (NGT)
flip-flop		transición de pendiente positiva (PGT)
flip-flop sincronizado por reloj		

PROBLEMAS

SECCIONES 5-1 A 5-3

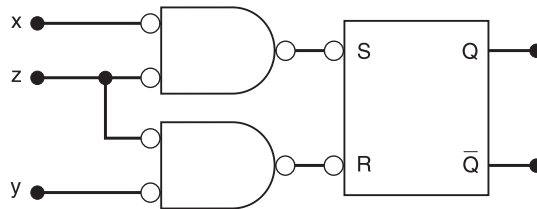
- B** 5-1.* Suponiendo que al principio $Q = 0$, aplique las formas de onda x y y de la figura 5-73 a las entradas SET y RESET de un latch NAND y determine las formas de onda de Q y $\bar{Q}$.

FIGURA 5-73 Problemas 5-1 a 5-3.



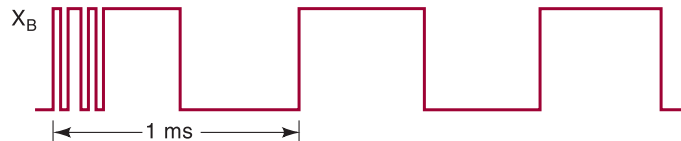
- B** 5-2. Invierta las formas de onda x y y de la figura 5-73, aplíquelas a las entradas SET y RESET de un latch NOR y determine las formas de onda de Q y $\bar{Q}$. Suponga que al principio $Q = 0$.
- 5-3.* Las formas de onda de la figura 5-73 están conectadas al circuito de la figura 5-74. Suponga que al principio $Q = 0$ y determine la forma de onda de Q .

FIGURA 5-74 Problema 5-3.



- D** 5-4. Modifique el circuito de la figura 5-9 para que utilice un latch de compuerta NOR.
- D** 5-5. Modifique el circuito de la figura 5-12 para que utilice un latch de compuerta NAND.
- F** 5-6.* Consulte el circuito de la figura 5-13. Para probar la operación del circuito, un técnico observa las salidas con un osciloscopio de almacenamiento mientras que el interruptor se mueve de A a B . Cuando el interruptor se mueve de A a B , aparece la pantalla de X_B en el osciloscopio según se muestra en la figura 5-75. ¿Qué falla del circuito podría producir este resultado? (Sugerencia: ¿cuál es la función del latch NAND?)

FIGURA 5-75 Problema 5-6.



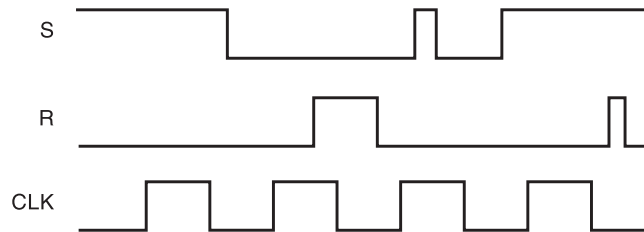
SECCIONES 5-4 A 5-6

- B** 5-7. Cierta FF sincronizado por reloj tiene los valores mínimos $t_S = 20$ ns y $t_H = 5$ ns. ¿Cuánto tiempo deben permanecer estables las entradas de control antes de la transición activa del reloj?

* Encontrará las respuestas a los problemas marcados con un asterisco al final del capítulo.

- B** 5-8. Aplique las formas de onda de S , R y CLK de la figura 5-19 al FF de la figura 5-20 y determine la forma de onda de Q .
- B** 5-9.* Aplique las formas de onda de la figura 5-76 al FF de la figura 5-19 y determine la forma de onda en Q . Repita el proceso para el FF de la figura 5-20. Suponga que al principio $Q = 0$.

FIGURA 5-76
Problema 5-9.

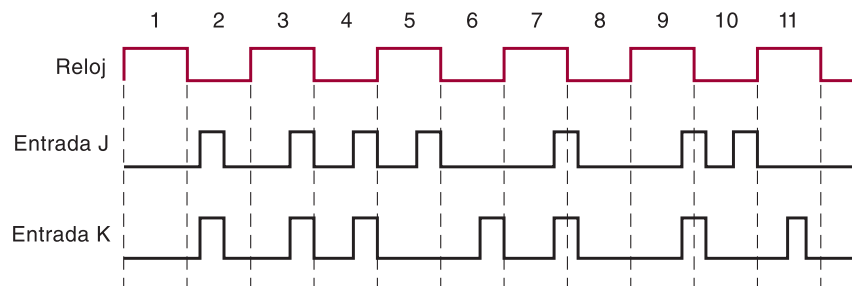


- 5-10. Dibuje las siguientes formas de onda de pulsos digitales. Etiquételas como t_r , t_f y t_w , franco anterior y franco posterior.
- Un pulso TTL negativo con $t_r = 20$ ns, $t_f = 5$ ns y $t_w = 50$ ns.
 - Un pulso TTL positivo con $t_r = 5$ ns, $t_f = 1$ ns y $t_w = 25$ ns.
 - Un pulso positivo con $t_w = 1$ ms, cuyo flanco anterior ocurra cada 5 ms. Proporcione la frecuencia de esta forma de onda.

SECCIÓN 5-7

- B** 5-11.* Aplique las formas de onda de J , K y CLK de la figura 5-23 al FF de la figura 5-24. Suponga que al principio $Q = 1$ y determine la forma de onda de Q .
- D** 5-12. (a)* Muestre cómo un flip-flop J-K puede operar como un FF *conmutador* (que cambia de estado en cada pulso del reloj). Después aplique una señal de reloj de 10-kHz en su entrada CLK y determine la forma de onda en Q .
- (b) Conecte la salida Q de este FF a la entrada CLK de un segundo FF J-K que también tiene $J = K = 1$. Determine la frecuencia de la señal en la salida de este FF.
- B** 5-13. Las formas de onda que se muestran en la figura 5-77 se van a aplicar a dos FFs distintos:
- J-K disparado con flanco positivo.
 - J-K disparado con flanco negativo.
- Dibuje la respuesta de la forma de onda de Q para cada uno de estos FFs, suponiendo que al principio $Q = 0$. Suponga también que cada FF tiene $t_H = 0$.

FIGURA 5-77
Problema 5-13.



SECCIÓN 5-8

- N 5-14. Algunas veces se utiliza un FF D para *retrasar* una forma de onda binaria, de manera que la información binaria aparezca en la salida durante una cierta cantidad de tiempo después de que aparece en la entrada *D*.
- (a)* Determine la forma de onda de *Q* en la figura 5-78 y compárela con la forma de onda de entrada. Observe que está retrasada desde la entrada por un periodo de reloj.
- (b) ¿Cómo puede obtenerse un retraso de dos periodos de reloj?

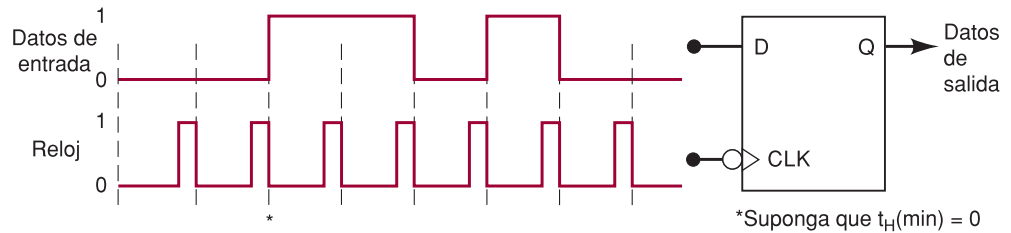
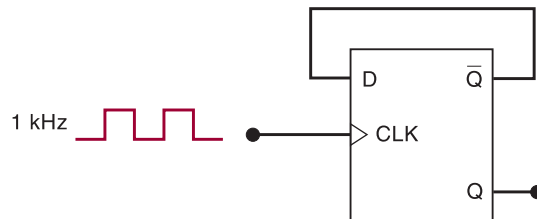


FIGURA 5-78 Problema 5-14.

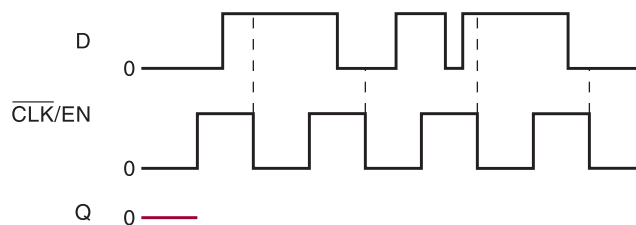
- B 5-15. (a) Aplique las formas de onda de *S* y *CLK* de la figura 5-76 a las entradas *D* y *CLK* de un FF D que se dispara en las PGTs. Después determine la forma de onda en *Q*.
- (b) Repita el procedimiento, utilizando la forma de onda de *C* de la figura 5-76 para la entrada *D*.
- B 5-16.* Se puede hacer que un flip-flop D disparado por flanco opere en el modo de conmutación si se le conecta como se muestra en la figura 5-79. Suponga que al principio $Q = 0$ y determine la forma de onda de *Q*.

FIGURA 5-79 Flip-flop D conectado para conmutar (problema 5-16).



SECCIÓN 5-9

- B 5-17. (a) Aplique las formas de onda de *S* y *CLK* de la figura 5-76 a las entradas *D* y *EN* de un latch D en forma respectiva y determine la forma de onda en *Q*.
- (b) Repita el proceso utilizando la forma de onda de *C* aplicada a *D*.
- 5-18. Compare la operación del latch *D* con un flip-flop *D* disparado por flanco negativo, mediante la aplicación de las formas de onda de la figura 5-80 a cada uno de ellos y determine la forma de onda de *Q*.

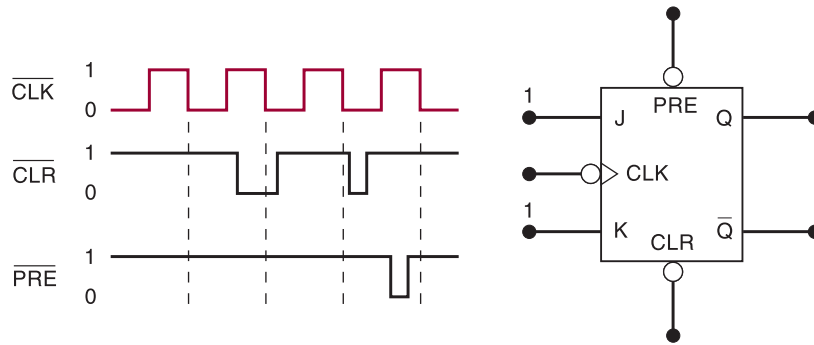
FIGURA 5-80
Problema 5-18.

- 5-19. En el problema 5-16 vimos cómo un flip-flop D disparado por flanco puede operar en el modo de conmutación. Explique por qué la misma idea no funcionará para un latch D.

SECCIÓN 5-10

- B** 5-20. Determine la forma de onda de Q para el FF de la figura 5-81. Suponga que al principio $Q = 0$ y recuerde que las entradas asíncronas predominan sobre todas las demás entradas.

FIGURA 5-81
Problema 5-20.



- B** 5-21.* Aplique las formas de onda de $\overline{CLK}$, $\overline{PRE}$ y $\overline{CLR}$ (figura 5-32) a un flip-flop D disparado por flanco positivo con entradas asíncronas activas en BAJO. Suponga que D se mantiene en ALTO y que al principio Q está en BAJO. Determine la forma de onda de Q .
- B** 5-22. Aplique las formas de onda de la figura 5-81 a un flip-flop D que se dispara con las NGTs y que tiene entradas asíncronas activas en BAJO. Suponga que D se mantiene en BAJO y que al principio Q está en ALTO. Dibuje la forma de onda resultante en Q .

SECCIÓN 5-12

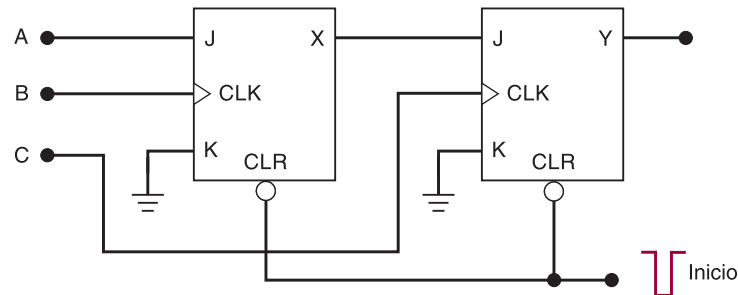
- B** 5-23. Use la tabla 5-2 de la sección 5-12 para determinar lo siguiente:
- * ¿Cuánto puede tardar la salida Q de un 74C74 en cambiar de 0 a 1, en respuesta a una transición activa de CLK ?
 - * ¿Cuál FF en la tabla 5-2 requiere que sus entradas de control permanezcan estables durante el mayor tiempo *después* de la transición activa en CLK ? ¿*Antes* de la transición?
 - ¿Cuál es el pulso más estrecho que puede aplicarse a la entrada $\overline{PRE}$ de un FF 7474?
- B** 5-24. Use la tabla 5-2 para determinar lo siguiente:
- ¿Cuánto tiempo se requiere para borrar un 74LS112 en forma asíncrona?
 - ¿Cuánto tiempo se requiere para establecer un 74HC112 en forma asíncrona?
 - ¿Cuál es el intervalo más corto aceptable entre las transiciones activas de reloj para un 7474?
 - La entrada D de un 74HC112 cambia a ALTO 15 ns antes del flanco activo del reloj. ¿Se guardarán los datos en forma confiable en el flip-flop?
 - ¿Cuánto tiempo se requiere (después del flanco del reloj) para almacenar en forma síncrona un 1 en un flip-flop D 7474 borrado?

SECCIONES 5-15 Y 5-16

- D** 5-25.* Modifique el circuito de la figura 5-40 para que utilice un flip-flop J-K.

- D** 5-26. En el circuito de la figura 5-82, al principio las entradas A , B y C están en BAJO. Se supone que la salida Y debe cambiar a ALTO sólo cuando A , B y C cambien a ALTO en cierta secuencia.
- Determine la secuencia que hará que Y cambie a ALTO.
 - Explique por qué se necesita el pulso INICIO.
 - Modifique este circuito para que utilice FFs D.

FIGURA 5-82
Problema 5-26.



SECCIONES 5-17 Y 5-18

- D** 5-27.* (a) Dibuje un diagrama de circuito para la transferencia síncrona en paralelo de un registro de tres bits a otro registro, mediante el uso de flip-flops J-K.
- (b) Repita el proceso para una transferencia asíncrona en paralelo.
- N, D** 5-28. Un registro de desplazamiento *recirculante* es un registro de desplazamiento que mantiene circulando la información binaria a través del registro, a medida que se aplican pulsos de reloj. El registro de desplazamiento de la figura 5-45 puede convertirse en un registro circulante si se conecta X_0 a la línea DATOS ENT. No se utilizan entradas externas. Suponga que este registro circulante empieza con el valor 1011 almacenado en el mismo (es decir, $X_3 = 1$, $X_2 = 0$, $X_1 = 1$ y $X_0 = 1$). Liste la secuencia de estados por los que pasan los FFs del registro a medida que se aplican ocho pulsos de desplazamiento.
- D** 5-29.* Consulte la figura 5-46, en donde un número de tres bits almacenado en el registro X se está desplazando en forma serial hacia el registro Y . ¿Cómo puede modificarse el circuito de manera que al final de la operación de transferencia esté el número original almacenado en X presente en ambos registros? (Sugerencia: vea el problema 5-28.)

SECCIÓN 5-19

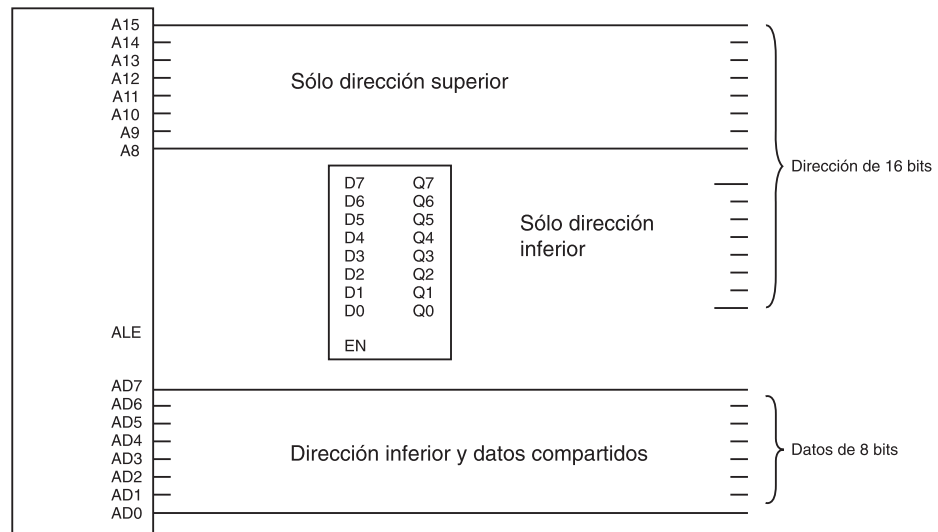
- B** 5-30. Consulte el circuito contador de la figura 5-47 y responda a lo siguiente:
- * Si el contador empieza en 000, ¿cuál será la cuenta después de 13 pulsos de reloj? ¿Después de 99 pulsos? ¿Después de 256 pulsos?
 - Si el contador empieza en 100, ¿cuál será la cuenta después de 13 pulsos? ¿Después de 99 pulsos? ¿Después de 256 pulsos?
 - Conecte un cuarto FF J-K (X_3) a este contador y dibuje el diagrama de transición de estados para este contador de 4 bits. Si la frecuencia del reloj de entrada es de 80 MHz, ¿cuál será la apariencia de la forma de onda en X_3 ?
- B** 5-31. Consulte el contador binario de la figura 5-47. Cámbielo conectando $\overline{X}_0$ a la señal CLK del flip-flop X_1 , y $\overline{X}_1$ a la señal CLK del flip-flop X_2 . Empiece con todos los FFs en el estado 1 y dibuje las diversas formas de onda de salida de los FFs (X_0 , X_1 , X_2) para 16 pulsos de entrada. Después liste la secuencia de estados de los FFs, como se hizo en la figura 5-48. A este contador se le conoce como *contador descendente*. ¿Por qué?
- B** 5-32. Dibuje el diagrama de transición de estados para este contador descendente y compárelo con el diagrama de la figura 5-49. ¿Cuál es la diferencia?

- B** 5-33.* (a) ¿Cuántos FFs se requieren para construir un contador binario que cuente desde 0 hasta 1023?
 (b) Determine la frecuencia en la salida del último FF de este contador, para una frecuencia de entrada de reloj de 2 MHz.
 (c) ¿Cuál es el número MOD del contador?
 (d) Si al principio el contador está en cero, ¿qué cuenta tendrá después de 2060 pulsos?
- B** 5-34. Un contador binario está recibiendo pulsos a través de una señal de reloj de 256-kHz. La frecuencia de salida del último FF es de 2 kHz.
 (a) Determine el número MOD.
 (b) Determine el intervalo de conteo.
- B** 5-35. Un circuito fotodetector se está usando para generar un pulso cada vez que un cliente entra en cierto establecimiento. Los pulsos se alimentan a un contador de ocho bits. Este contador se utiliza para contar estos pulsos como un medio para determinar cuántos clientes han entrado en la tienda. Después de cerrar la tienda, el propietario revisa el contador y descubre que muestra un conteo de $00001001_2 = 9_{10}$. Él sabe que esto es incorrecto, ya que había mucho más de nueve personas en su tienda. Suponiendo que el circuito contador esté funcionando en forma apropiada, ¿cuál podría ser el motivo de la discrepancia?

SECCIÓN 5-20

- D** 5-36.* Modifique el circuito de la figura 5-50 de manera que sólo la presencia del código de dirección 10110110 permita transferir datos al registro X.
- F** 5-37. Suponga que el circuito de la figura 5-50 está funcionando mal, de manera que se transfieran datos a X para cualquiera de los códigos de dirección 11111110 o 11111111. ¿Cuáles son algunas de las fallas de un circuito que podrían estar ocasionando esto?
- N, D** 5-38. Muchos microcontroladores comparten las mismas terminales para enviar de salida la dirección inferior y transferir datos. Para poder mantener la constante de dirección mientras que se transfieren los datos. La información sobre la dirección se almacena en un latch, el cual se habilita mediante la señal de control ALE (habilita latch de dirección), como se muestra en la figura 5-83. Conecte este latch al microcontrolador de tal forma que reciba lo que haya en la dirección inferior y en las líneas de datos mientras que ALE esté en ALTO, y que lo retenga en las líneas de sólo dirección inferior cuando ALE esté en BAJO.

FIGURA 5-83
Problema 5-38.

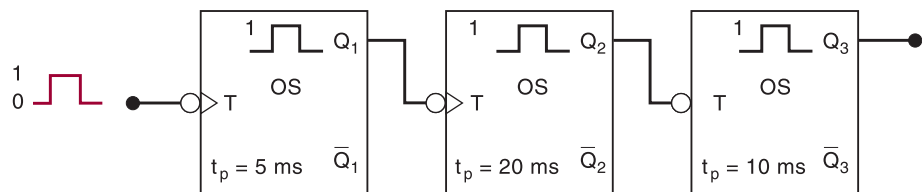


- D** 5-39. Modifique el circuito de la figura 5-50, de forma que la MPU tenga ocho líneas de salida de datos conectadas para transferir ocho bits de datos a un registro de ocho bits formado por dos CIs 74HC175 [Figura 5-34(b)]. Muestre todas las conexiones del circuito.

SECCIÓN 5-22

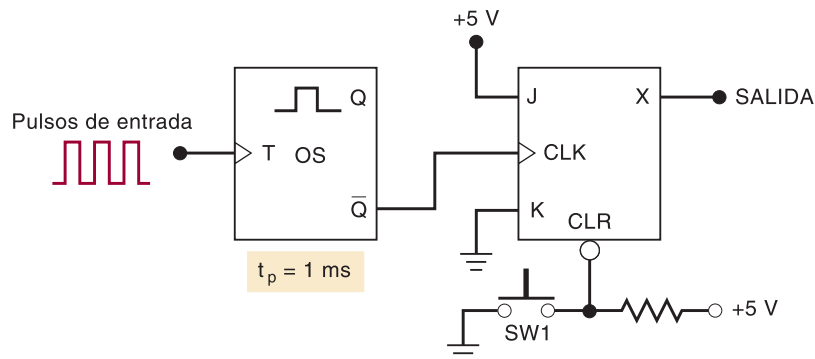
- B** 5-40. Consulte las formas de onda de la figura 5-53(a). Cambie la duración del pulso del OS a 0.5 ms y determine la salida Q para ambos tipos de OS. Después repita el proceso utilizando una duración de pulso de OS de 1.5 ms.
- N** 5-41.* La figura 5-84 muestra tres monoestables no redispables conectados en una cadena de sincronización que produce tres pulsos de salida secuenciales. Observe el “1” enfrente del pulso en cada símbolo de OS para indicar la operación no redispable. Dibuje un diagrama de tiempo que muestre la relación entre el pulso de entrada y las salidas de los tres OS. Suponga una duración de 10 ms para el pulso de entrada.

FIGURA 5-84
Problema 5-41.



- 5-42. Un OS *redispable* puede utilizarse como detector de frecuencia de pulso que detecte cuando la frecuencia de un pulso de entrada esté por debajo de un valor predeterminado. En la figura 5-85 se muestra un ejemplo simple de esta aplicación. La operación empieza con el cierre momentáneo del interruptor SW1.
- Describa cómo responde el circuito a las frecuencias de entrada por encima de 1 kHz.
 - Describa cómo responde el circuito a las frecuencias de entrada por debajo de 1 kHz.
 - ¿Cómo modificaría el circuito para detectar cuando la frecuencia de entrada cae por debajo de 50 kHz?

FIGURA 5-85
Problema 5-42.



- 5-43. Consulte el símbolo lógico para un monoestable no redispable 74121 en la figura 5-54(a).
- ¿Qué condiciones de entrada se necesitan para que el OS se dispare mediante una señal en la entrada B ?
 - ¿Qué condiciones de entrada se necesitan para que el OS se dispare mediante una señal en la entrada A_1 ?

- A, D** 5-44. La anchura del pulso de salida de un OS 74121 se obtiene mediante la fórmula aproximada

$$t_p \approx 0.7 R_T C_T$$

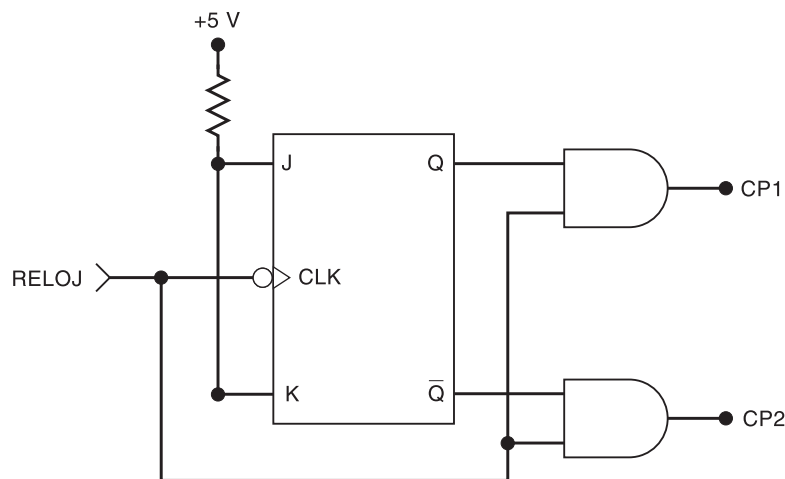
en donde R_T es la resistencia conectada entre la terminal R_{EXT}/C_{EXT} y V_{CC} , y C_T es la capacitancia conectada entre la terminal C_{EXT} y la terminal R_{EXT}/C_{EXT} . El valor de R_T puede variar entre 2 y 40 k Ω , y C_T puede ser de hasta 1000 μ F.

- (a) Muestre cómo puede conectarse un 74121 para producir un pulso de pendiente negativa con una duración de 5 ms, cada vez que cualquiera de dos señales lógicas (E o F) produce una NGT. Tanto E como F se encuentran, por lo general, en el estado ALTO.
- (b) Modifique el circuito de manera que una señal de entrada de control (G) pueda deshabilitar el pulso de salida del OS, sin importar lo que ocurra en E o en F .

SECCIÓN 5-23

- B, N** 5-45.* Muestre cómo usar un INVERSOR disparador de Schmitt 74LS14 para producir una onda cuadrada aproximada con una frecuencia de 10 kHz.
- B, D** 5-46. Diseñe un oscilador 555 de funcionamiento libre para producir una onda cuadrada aproximada a 40 kHz. C deberá mantenerse en 500 pF o más.
- D** 5-47. Un oscilador 555 puede combinarse con un flip-flop J-K para producir una onda cuadrada perfecta (ciclo de trabajo del 50 por ciento). Modifique el circuito del problema 5-46 para que incluya un flip-flop J-K. La salida final tiene que seguir siendo una onda cuadrada de 40 kHz.
- 5-48. Diseñe un circuito temporizador 555 que produzca una forma de onda de 5 kHz con un ciclo de trabajo del 10 por ciento. Seleccione un capacitor mayor de 500 pF y resistencias menores de 100 k Ω . Dibuje el diagrama del circuito con los números de las terminales etiquetados.
- A, N** 5-49. El circuito de la figura 5-86 puede usarse para generar dos señales de reloj que no se traslapen entre sí, a la misma frecuencia. Estas señales de reloj se utilizaban en los primeros sistemas de microprocesador, los cuales requerían cuatro transiciones de reloj distintas para sincronizar sus operaciones.
- (a) Dibuje las formas de onda de sincronización CP1 y CP2 si $RELOJ$ es una onda cuadrada de 1 MHz. Suponga que t_{PLH} y t_{PHL} son de 20 ns para el FF y de 10 ns para las compuertas AND.

FIGURA 5-86
Problema 5-49.



- (b) Este circuito tendría un problema si el FF se cambiara por uno que respondiera a una PGT en CLK . Dibuje las formas de onda de CP1 y CP2 para esa situación. Ponga especial atención a las condiciones que pueden producir deformaciones.

SECCIÓN 5-24

- F** 5-50. Consulte el circuito contador de la figura 5-47. Suponga que todas las entradas asíncronas se conectan a V_{CC} . Al probarlo, las formas de onda del circuito aparecen como se muestra en la figura 5-87. Considere la siguiente lista de posibles fallas. Para cada una, indique “sí” o “no” en cuanto a si pueden producir los resultados observados. Explique cada respuesta.
- (a)* La entrada CLK de X_2 está abierta.
 - (b)* Los tiempos de transición de la salida X_1 son demasiado largos, tal vez debido a la carga.
 - (c) La salida de X_2 está en corto con tierra.
 - (d) No se cumple con el requerimiento de tiempo de retención de X_2 .

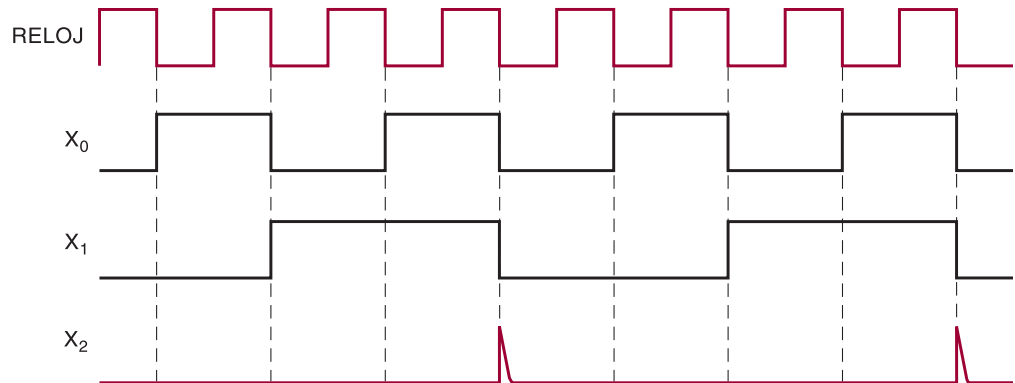
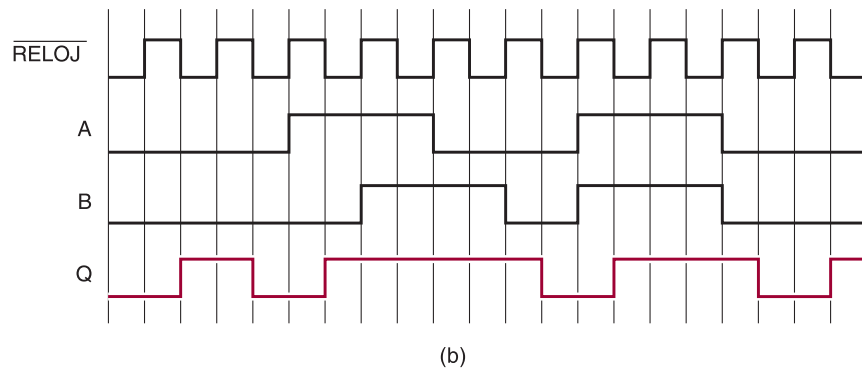
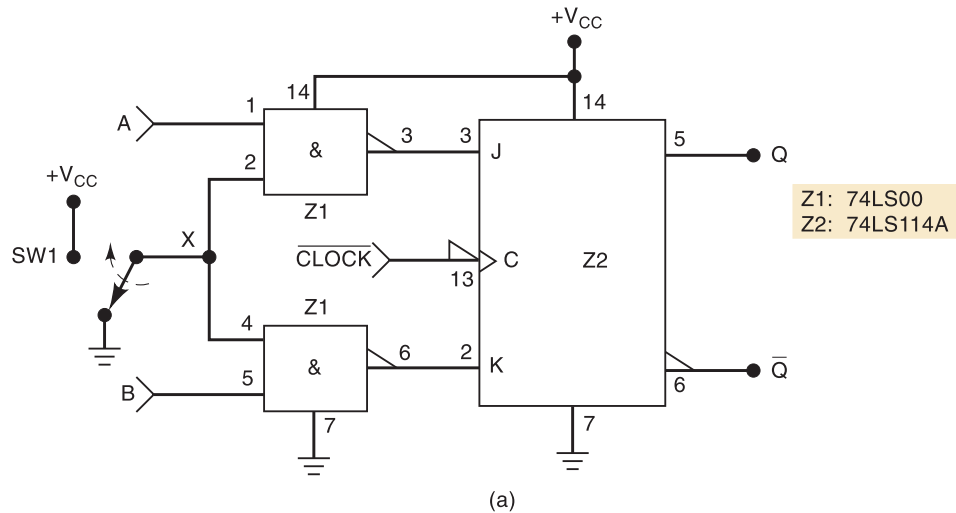


FIGURA 5-87 Problema 5-50.

- A, F** 5-51. Considere la situación de la figura 5-59 para cada uno de los siguientes conjuntos de valores de sincronización. Para cada uno de ellos indique si el flip-flop Q_2 responderá o no en forma correcta.
- (a)* Cada FF: $t_{PLH} = 12$ ns; $t_{PHL} = 8$ ns; $t_S = 5$ ns; $t_H = 0$ ns;
 Compuerta NAND: $t_{PLH} = 8$ ns; $t_{PHL} = 6$ ns;
 INVERSOR: $t_{PLH} = 7$ ns; $t_{PHL} = 5$ ns;
 - (b) Cada FF: $t_{PLH} = 10$ ns; $t_{PHL} = 8$ ns; $t_S = 5$ ns; $t_H = 0$ ns;
 Compuerta NAND: $t_{PLH} = 12$ ns; $t_{PHL} = 10$ ns;
 INVERSOR: $t_{PLH} = 8$ ns; $t_{PHL} = 6$ ns;
- D** 5-52. Muestre y explique cómo el problema de desfase de reloj de la figura 5-59 puede eliminarse si se insertan dos INVERSORES en forma apropiada.
- F** 5-53. Consulte el circuito de la figura 5-88. Suponga que los CIs son de la familia lógica TTL. La forma de onda de Q se obtuvo cuando se probó el circuito con las señales de entrada que se muestran, y con el interruptor en la posición “arriba”; no es correcta. Considere la siguiente lista de fallas y para cada una indique “sí” o “no” en cuanto a cuál podría ser la verdadera falla. Explique cada respuesta.
- (a)* El punto X siempre está en BAJO debido a un interruptor defectuoso.
 - (b)* La terminal 1 de Z1 está en corto interno con V_{CC} .
 - (c) La conexión de Z1-3 a Z2-3 está rota.
 - (d) Hay un puente de soldadura entre las terminales 6 y 7 de Z1.

FIGURA 5-88
Problema 5-53.



- A** 5-54. El circuito de la figura 5-89 funciona como un candado de combinación secuencial. Para operar el candado, proceda de la siguiente manera:
1. Active por un momento el interruptor RESET.
 2. Ajuste los interruptores SWA, SWB y SWC con la primera parte de la combinación. Después conmute por un momento el interruptor INTRODUCIR hacia delante y hacia atrás.
 3. Ajuste los interruptores con la segunda parte de la combinación y conmute el interruptor INTRODUCIR de nuevo. Esto deberá producir un nivel ALTO en Q_2 para abrir el candado.
- Si se introduce la combinación incorrecta en cualquiera de los pasos, el operador deberá comenzar la secuencia de nuevo. Analice el circuito y determine la secuencia correcta de combinaciones que abrirán el candado.
- A, F** 5-55.* Cuando se prueba el candado de combinación de la figura 5-89, se descubre que al introducir la combinación correcta no se abre el candado. Una comprobación con la sonda lógica muestra que al introducir la primera combinación correcta Q_1 se establece en ALTO, pero al introducir la segunda combinación correcta sólo se produce un pulso momentáneo en Q_2 . Considere cada una de las siguientes fallas e indique cuál(es) podría(n) producir la operación observada. Explique cada una de las opciones.
- (a) Rebote de interruptor en SWA, SWB o SWC.
 - (b) La entrada CLR de Q_2 está abierta.
 - (c) La conexión de la salida de la compuerta NAND 4 a la entrada de la compuerta NAND 3 está abierta.

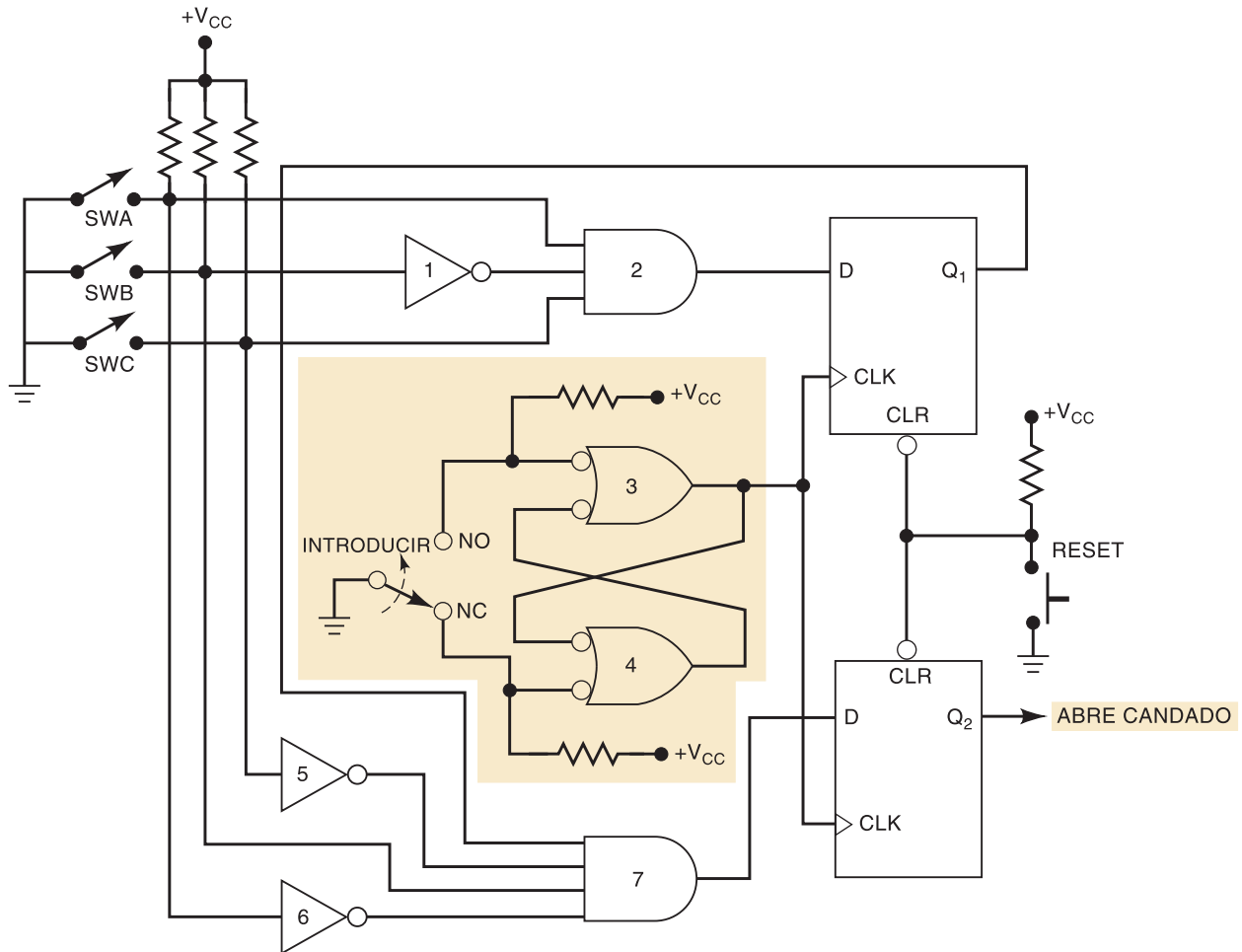


FIGURA 5-89 Problemas 5-54 y 5-55.

PREGUNTAS DE PRÁCTICA

- B** 5-56. Para cada una de las siguientes aseveraciones, indique qué tipo de FF se está describiendo.
- (a)* Tiene una entrada SET y una entrada CLEAR pero no tiene entrada *CLK*.
 - (b)* Conmutará en cada pulso de *CLK* cuando ambas entradas de control estén en ALTO.
 - (c)* Tiene una entrada HABILITAR en vez de una entrada *CLK*.
 - (d)* Se utiliza para transferir datos con facilidad de un registro de FF a otro.
 - (e) Sólo tiene una entrada de control.
 - (f) Tiene dos salidas que son complementos una de la otra.
 - (g) Puede cambiar de estado sólo en la transición activa de *CLK*.
 - (h) Se utiliza en los contadores binarios.
- B** 5-57. Defina los siguientes términos.
- (a) Entradas asíncronas.
 - (b) Disparado por flanco.
 - (c) Registro de desplazamiento.
 - (d) División de frecuencia.

- (e) Transferencia asíncrona (atascada).
- (f) Diagrama de transición de estados.
- (g) Transferencia de datos en paralelo.
- (h) Transferencia de datos en serie.
- (i) Monoestable (one-shot) redisparable.
- (j) Entradas tipo disparador de Schmitt.

SECCIÓN 5-25

- B** 5-58. Simule el diseño de HDL para un latch NAND, el cual se muestra en la figura 5-62 (AHDL) o en la figura 5-63 (VHDL). ¿Qué hace este latch SR si se aplica un comando de entrada “inválido”? Como sabemos que cualquier latch SR puede tener un resultado de salida inusual cuando se le aplica un comando de entrada inválido, debe simular esa condición de entrada junto con los comandos normales establecer, restablecer y retener del latch. Algunos diseños de latches pueden tener una tendencia en la que la salida oscila cuando un comando inválido va seguido de un comando de retención, por lo que debe revisar eso en su simulación.
- B, H** 5-59.* Escriba el código de HDL para un latch SR con entradas activas en ALTO.
- B, H** 5-60. Modifique la descripción del latch que se muestra en la figura 5-62 (AHDL) o en la figura 5-63 (VHDL) para hacer que el SR se restablezca si se le aplica una entrada inválida. Simule el diseño.
- B, H** 5-61.* Agregue salidas invertidas a los diseños en HDL del latch NAND que se muestran en la figura 5-62 o en la figura 5-63. Verifique su correcta operación por medio de la simulación.
- B** 5-62. Simule el diseño en AHDL o en VHDL para un latch D que aparece en la sección 5-25.
- D, H** 5-63. Cree un latch transparente de cuatro bits con una entrada *habilitar*, con base en el diseño en AHDL o VHDL para un dispositivo latch D individual que aparece en la sección 5-25. Simule el latch de cuatro bits.
- D, H, N** 5-64. Un flip-flop de conmutación (T) tiene una sola entrada de control (T). Cuando $T = 0$, el flip-flop se encuentra en el estado sin cambio, algo similar a un FF J-K con $J = K = 0$. Cuando $T = 1$ el flip-flop se encuentra en el modo de conmutación, algo similar a un JKFF con $J = K = 1$. Escriba el archivo de diseño en:
- (a) AHDL.
 - (b) VHDL.
- H** 5-65. (a) Escriba el código en AHDL para el registro de desplazamiento que se muestra en la figura 5-45.
- (b) Escriba el código en VHDL para el registro de desplazamiento que se muestra en la figura 5-45.
- H** 5-66. (a)* Escriba el código en AHDL para el registro de desplazamiento que se muestra en la figura 5-46.
- (b) Escriba el código en VHDL para el registro de desplazamiento que se muestra en la figura 5-46.
- H** 5-67. (a) Escriba el código en AHDL para el circuito de FF que se muestra en la figura 5-59.
- (b) Escriba el código en VHDL para el circuito de FF que se muestra en la figura 5-59.
- H** 5-68. (a) Escriba el código en AHDL para implementar todo el circuito de la figura 5-89.
- (b) Escriba el código en VHDL para implementar todo el circuito de la figura 5-89.

RESPUESTAS A LAS PREGUNTAS DE REPASO DE LAS SECCIONES

SECCIÓN 5-1

1. ALTO; BAJO
2. $Q = 0, \overline{Q} = 1$
3. Verdadero
4. Aplicar un nivel BAJO momentáneo en la entrada $\overline{SET}$.

SECCIÓN 5-2

1. BAJO; ALTO
2. $Q = 1$ y $\overline{Q} = 0$
3. Hacer que $CLEAR = 1$
4. $\overline{SET}$ y $\overline{RESET}$ por lo general, estarían en su estado activo en BAJO.

SECCIÓN 5-4

1. Entradas de control síncronas y entrada de reloj sólo cuando ocurre la transición de reloj apropiada.
2. La salida del FF puede cambiar sólo cuando ocurre la transición de reloj apropiada.
3. Falso.
4. El tiempo de estabilización es el intervalo requerido justo antes del flanco activo de la señal CLK , durante la cual las entradas de control deben mantenerse estables. El tiempo de retención es el intervalo requerido justo después del flanco activo de CLK , durante el cual las entradas de control deben mantenerse estables.

SECCIÓN 5-5

1. ALTO; BAJO; ALTO
2. Porque CLK^* está en ALTO sólo durante unos cuantos nanosegundos.

SECCIÓN 5-6

1. Verdadero
2. No
3. $J = 1, K = 0$

SECCIÓN 5-7

1. Q cambiará a BAJO en el punto a y permanecerá en BAJO.
2. Falso. La entrada D puede cambiar sin afectar a Q , ya que ésta puede cambiar sólo en el flanco activo de CLK .
3. Sí, convirtiendo a FFs D (figura 5-25).

SECCIÓN 5-8

1. En un latch D , la salida Q puede cambiar mientras EN está en ALTO. En un flip-flop D , la salida puede cambiar sólo en el flanco activo de CLK .
2. Falso.
3. Verdadero.

SECCIÓN 5-9

1. Las entradas asíncronas funcionan en forma independiente de la entrada CLK .
2. Sí, ya que $\overline{PRE}$ es activa en BAJO.
3. $J = K = 1, \overline{PRE} = \overline{CLR} = 1$, y una PGT en CLK .

SECCIÓN 5-10

1. El triángulo dentro del rectángulo indica la operación de disparo por flanco; el triángulo recto fuera del rectángulo indica el disparo en una NGT.
2. Se utiliza para indicar la función de las entradas que son comunes para más de un circuito en el chip.

SECCIÓN 5-11

1. t_{PLH} y t_{PHL}
2. Falso; la forma de onda también debe satisfacer los requerimientos de $t_w(L)$ y $t_w(H)$.

SECCIÓN 5-17

1. Falso.
2. Flip-flop D .
3. Seis.
4. Verdadero.

SECCIÓN 5-18

1. Verdadero
2. Menos interconexiones entre registros
3. $X_2X_1X_0 = 111; Y_2Y_1Y_0 = 101$
4. En la transmisión de datos en paralelo.

SECCIÓN 5-19

1. 10 kHz 2. Ocho 3. 256 4. 2 kHz 5. $00001000_2 = 8_{10}$

SECCIÓN 5-21

1. La salida puede contener oscilaciones. 2. Producirá señales de salida más rápidas y limpias, incluso hasta para las señales de entrada con cambios lentos.

SECCIÓN 5-22

1. $Q = 0, \bar{Q} = 1$ 2. Verdadero. 3. Los valores externos de R y C . 4. Para un OS redispensible, cada nuevo pulso de disparo comienza un nuevo intervalo t_p sin importar el estado de la salida Q .

SECCIÓN 5-23

1. 24 kHz 2. 109.3 kHz; 66.7 por ciento. 3. Estabilidad de frecuencia.

SECCIÓN 5-24

1. El desfase de reloj es la llegada de una señal de reloj a las entradas CLK de distintos FFs en tiempos distintos. Puede hacer que un FF cambie a un estado incorrecto.

SECCIÓN 5-25

1. Retroalimentación: las salidas se combinan con las entradas para determinar el siguiente estado de las salidas. 2. Progresar a través de una secuencia predeterminada de estados en respuesta a una señal de reloj de entrada.

SECCIÓN 5-26

1. Un bloque de construcción estándar de una biblioteca de componentes que realice cierta función lógica fundamental. 2. Los nombres de cada entrada y salida, y el nombre de la primitiva que reconoce el sistema de desarrollo. 3. Bajo el menú HELP (AYUDA). 4. El bloque PROCESS permite las instrucciones IF secuenciales y el atributo EVENT detecta las transiciones. 5. `ieee.std_logic_1164`. 6. `altera.maxplus2`

SECCIÓN 5-27

1. Sí 2. En la sección VARIABLE. 3. A cada una se le asigna un nombre de variable. 4. = 5. Las señales (SIGNAL). 6. PORT MAP

CONTADORES Y REGISTROS

CONTENIDO

Parte 1

- 7-1 Contadores asíncronos (de rizo)
- 7-2 Retraso de propagación en contadores de rizo
- 7-3 Contadores síncronos (en paralelo)
- 7-4 Contadores con números $MOD < 2^N$
- 7-5 Contadores síncronos descendentes y ascendentes/ descendentes
- 7-6 Contadores preajustables
- 7-7 Contadores síncronos de CI
- 7-8 Decodificación de un contador
- 7-9 Análisis de contadores síncronos
- 7-10 Diseño de un contador síncrono
- 7-11 Contadores básicos mediante el uso de HDL
- 7-12 Contadores con todas las características en HDL

- 7-13 Cómo alambrar módulos de HDL en conjunto

- 7-14 Máquina de estados

Parte 2

- 7-15 Registros de circuito integrado
- 7-16 Entrada en paralelo/salida en paralelo: 74ALS174/74HC174
- 7-17 Entrada en serie/salida en serie: 74ALS166/74HC166
- 7-18 Entrada en paralelo/salida en serie: 74ALS165/74HC165
- 7-19 Entrada en serie/salida en paralelo: 74ALS164/74HC164
- 7-20 Contadores de registro de desplazamiento
- 7-21 Diagnóstico de fallas
- 7-22 Registros en HDL
- 7-23 Contadores de anillo en HDL
- 7-24 Monoestables en HDL

■ OBJETIVOS

Al terminar este capítulo, usted podrá:

- Comprender la operación y las características de los contadores síncronos y asíncronos.
- Construir contadores con números MOD menores de 2^N .
- Construir contadores tanto ascendentes como descendentes.
- Conectar contadores con múltiples etapas.
- Analizar y evaluar varios tipos de contadores.
- Diseñar contadores síncronos de secuencia arbitraria.
- Comprender varios tipos de esquemas utilizados para decodificar distintos tipos de contadores.
- Describir circuitos de contadores mediante el uso de distintos niveles de abstracción en HDL.
- Comparar las principales diferencias entre los contadores de anillo y Johnson.
- Reconocer y comprender la operación de varios tipos de registros de CI.
- Describir los registros de desplazamiento y los contadores de registro de desplazamiento mediante el uso de HDL.
- Aplicar las técnicas existentes de diagnóstico de fallas que se utilizan en los sistemas lógicos combinacionales para diagnosticar fallas en sistemas lógicos secuenciales.

■ INTRODUCCIÓN

En el capítulo 5 vimos cómo podían conectarse los flip-flops para funcionar como contadores y registros. En ese momento sólo estudiamos los circuitos básicos de contadores y registros. Los sistemas digitales emplean muchas variaciones de estos circuitos básicos, la mayoría en forma de circuitos integrados. En este capítulo veremos cómo pueden combinarse los FFs y las compuertas lógicas para producir diferentes tipos de contadores y registros.

Debido al extenso número de temas en este capítulo, lo hemos dividido en dos partes. En la **PARTE 1** cubriremos los principios de la operación de los contadores, los diversos arreglos de circuitos de contadores y los contadores de CI representativos. En la **PARTE 2** presentaremos varios tipos de registros de CI, contadores de registro de desplazamiento y el diagnóstico de fallas. Cada parte incluye una sección que contiene descripciones en HDL de los contadores y registros.

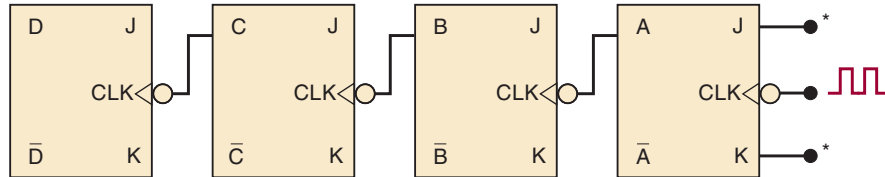
A medida que vaya avanzando en el capítulo tendrá que recurrir con frecuencia al material que cubrimos en los capítulos anteriores. Regrese y repase el material que aprendió con anterioridad cada vez que lo necesite.

PARTE 1

7-1 CONTADORES ASÍNCRONOS (DE RIZO)

La figura 7-1 muestra un circuito contador binario de cuatro bits tal como el que vimos en el capítulo 5. Recuerde los siguientes puntos con respecto a su operación:

1. Los pulsos de reloj se aplican sólo a la entrada *CLK* del flip-flop *A*. En consecuencia, el flip-flop *A* conmutará (cambiará a su estado opuesto) cada vez que los pulsos de reloj hagan una transición negativa (de ALTO a BAJO). Observe que $J = K = 1$ para todos los FFs.
2. La salida normal del flip-flop *A* actúa como la entrada *CLK* para el flip-flop *B*, por lo que el flip-flop *B* conmutará cada vez que la salida de *A* cambie de 1 a 0. De manera similar, el flip-flop *C* conmutará cuando *B* cambie de 1 a 0 y el flip-flop *D* conmutará cuando *C* cambie de 1 a 0.
3. Las salidas de los FFs *D*, *C*, *B* y *A* representan un número binario de cuatro bits, en donde *D* es el MSB. Vamos a suponer que todos los FFs se han borrado para quedar en el estado 0 (no se muestran las entradas de LIMPIA). Las formas de onda en la figura 7-1 muestran que se sigue una secuencia de conteo binario de 0000 a 1111 a medida que se aplican pulsos de reloj en forma continua.
4. Después de que ocurre la NGT (transición negativa) del quinceavo pulso de reloj, los FFs del contador se encuentran en la condición 1111. En la diecisei-



* Se asume que todas las entradas J y K son 1.

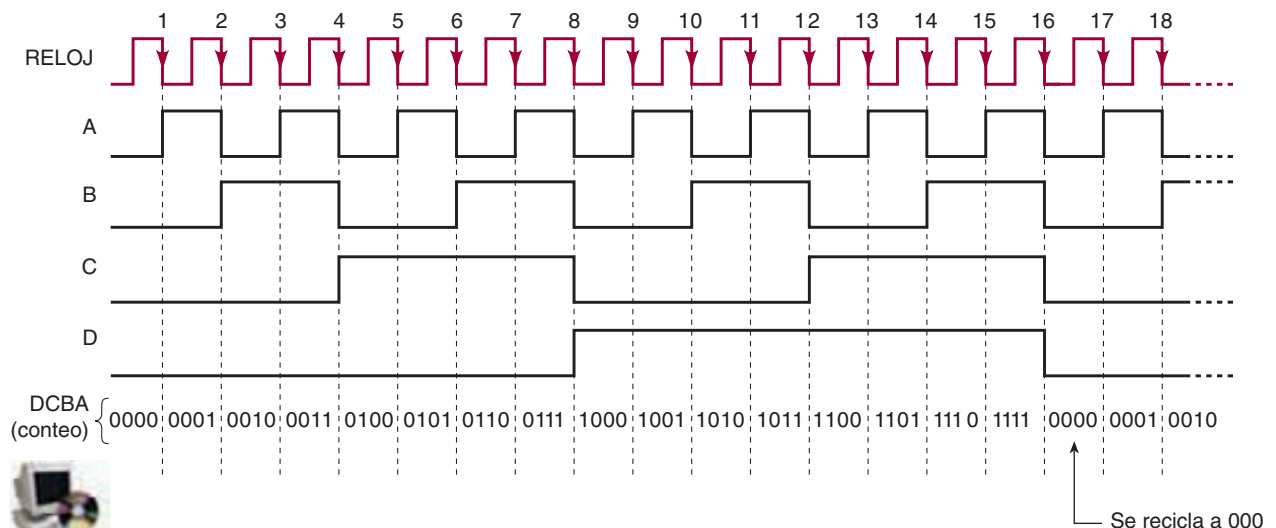


FIGURA 7-1 Contador asíncrono (de rizo) de cuatro bits.

sava NGT el flip-flop *A* cambia de 1 a 0, lo cual hace que el flip-flop *B* cambie de 1 a 0 y así en lo sucesivo, hasta que el contador queda en el estado 0000. En otras palabras, el contador ha pasado por un ciclo completo (de 0000 a 1111) y ha *reciclado* de vuelta a 0000. A partir de este punto empezará un nuevo ciclo de conteo, a medida que se apliquen los subsiguientes pulsos de reloj.

En este contador, cada salida de los FFs excita la entrada *CLK* del siguiente FF. A este tipo de arreglo de contador se le conoce como **contador asíncrono**, ya que los FFs no cambian de estado en sincronía exacta con los pulsos de reloj que se aplican; sólo el flip-flop *A* responde a los pulsos de reloj. El FF *B* debe esperar a que el FF *A* cambie de estado para poder conmutar; el FF *C* debe esperar al FF *B*, y así en lo sucesivo. Por ende, hay un retraso entre las respuestas de los FFs sucesivos. Por lo general este retraso es de 5 a 20 ns por cada FF. Como veremos, en algunos casos este retraso puede ser problemático. A este tipo de contador también se le conoce comúnmente como **contador de rizo** debido a la forma en que responden los FFs, uno después de otro, en un tipo de efecto de rizo. Utilizaremos los términos *contador asíncrono* y *contador de rizo* con el mismo fin, de manera indistinta.

Flujo de la señal

Es una convención en los diagramas esquemáticos dibujar los circuitos (siempre que sea posible) de manera que el flujo de la señal sea de izquierda a derecha, con las entradas a la izquierda y las salidas a la derecha. En este capítulo quebrantaremos con frecuencia esta convención, en especial en los diagramas que muestren contadores. Por ejemplo, en la figura 7-1 las entradas *CLK* de cada FF están a la derecha, las salidas están a la izquierda y la señal de reloj de entrada se muestra saliendo de la derecha. Utilizaremos este arreglo debido a que facilita la comprensión y el seguimiento de la operación del contador (pues el orden de los FFs es el mismo que el orden de los bits en el número binario que representa el contador). En otras palabras, el FF *A* (que es el LSB) es el FF más a la derecha y el FF *D* (que es el MSB) es el FF más a la izquierda. Si nos apegáramos al flujo de señal convencional de izquierda a derecha tendríamos que colocar el FF *A* a la izquierda y el FF *D* a la derecha, lo opuesto a sus posiciones en el número binario que representa al contador. En algunos de los diagramas de contadores que veremos más adelante en este capítulo emplearemos el flujo de señal convencional de izquierda a derecha, para que usted pueda acostumbrarse a verlo.

EJEMPLO 7-1

El contador de la figura 7-1 empieza en el estado 0000 y después se aplican pulsos de reloj. Cierta tiempo después se quitan los pulsos de reloj y los FFs del contador indican 0011. ¿Cuántos pulsos de reloj han transcurrido?

Solución

La respuesta aparente parece ser 3, ya que 0011 es el equivalente binario de 3. Sin embargo, con la información que se proporciona no hay manera de saber si el contador ha reciclado o no. Esto significa que podrían haber transcurrido 19 pulsos de reloj; los primeros 16 regresan el contador de vuelta a 0000 y los últimos 3 lo llevan hasta 0011. O podrían haber transcurrido 35 pulsos (dos ciclos completos y después tres conteos más), o 51 pulsos, etcétera.

Número MOD

El contador de la figura 7-1 tiene 16 estados diferentes (de 0000 a 1111). Por ende, es un *contador de rizo MOD 16*. Recuerde que, por lo general, el **número MOD** es igual al número de estados por los que pasa el contador en cada ciclo completo,

antes de que recicle de vuelta a su estado inicial. El número MOD puede aumentarse con sólo agregar más FFs al contador. Esto es,

$$\text{Número MOD} = 2^N \quad (7-1)$$

en donde N es el número de FFs conectados en el arreglo de la figura 7-1.

EJEMPLO 7-2

Se necesita un contador que cuente el número de elementos que pasan a través de una banda transportadora. Se utiliza una combinación de fotocelda y fuente de luz para generar un pulso individual cada vez que un elemento cruza su camino. El contador debe ser capaz de contar hasta mil elementos. ¿Cuántos FFs se requieren?

Solución

Es muy sencillo determinar qué valor de N se necesita para que $2^N \geq 1000$. Como $2^9 = 512$, 9 FFs no serán suficientes. $2^{10} = 1024$, por lo que 10 FFs producirían un contador que podría contar hasta $1111111111_2 = 1023_{10}$. Por lo tanto, utilizaríamos 10 FF; podríamos usar más de 10 pero sería un desperdicio, ya que cualquier FF más allá del diez no se necesitaría.

División de frecuencia

En el capítulo 5 vimos que en el contador básico cada FF proporciona una forma de onda de salida que es la *mitad* exacta de la frecuencia de la forma de onda en su entrada CLK . Para ilustrar esto, suponga que la señal de reloj en la figura 7.1 es de 16 kHz. La figura 7.2 muestra las formas de onda de salida. La forma de onda en la salida A es una *onda cuadrada* de 8 kHz, en la salida B es de 4 kHz, en la salida C es de 2 kHz y en la salida D es de 1 kHz. Observe que la salida del flip-flop D tiene una frecuencia igual a la original del reloj, dividida entre 16. En general,

En cualquier contador, la señal en la salida del último FF (es decir, el MSB) tendrá una frecuencia igual a la frecuencia del reloj de entrada dividida entre el número MOD del contador.

Por ejemplo, en un contador MOD-16 la salida del último FF tendrá una frecuencia igual a 1/16 de la frecuencia del reloj de entrada. Por ende, también se le puede llamar *contador de división entre 16*. De igual forma, un contador MOD 8 tiene una frecuencia de salida igual a 1/8 de la frecuencia de entrada; es un *contador de división entre 8*.

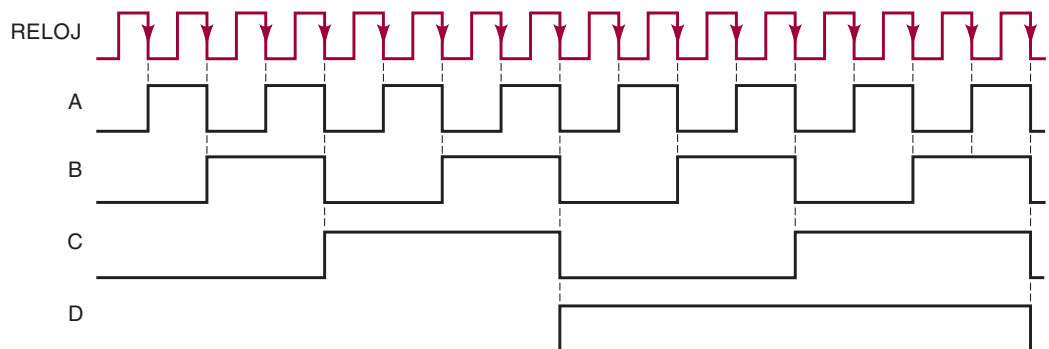
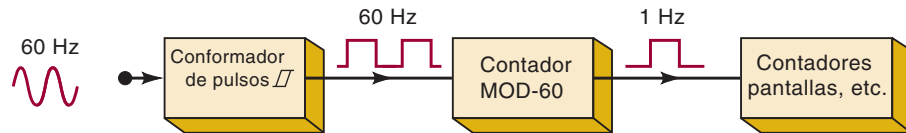


FIGURA 7-2 Formas de onda del contador que muestran la división entre 2 para cada FF.

EJEMPLO 7-3

El primer paso relacionado con la construcción de un reloj digital es tomar la señal de 60 Hz y alimentarla a un circuito de conformación de pulsos tipo monoestable de Schmitt* para producir una onda cuadrada, como se ilustra en la figura 7-3. La onda cuadrada de 60 Hz se aplica después a un contador MOD-60, el cual se utiliza para dividir la frecuencia de 60 Hz entre un valor exacto de 60 para producir una forma de onda de 1 Hz. Esta forma de onda de 1 Hz se alimenta a una serie de contadores, los cuales a su vez se encargan de contar en segundos, minutos, horas, etcétera. ¿Cuántos FFs se requieren para el contador MOD-60?

FIGURA 7-3 Ejemplo 7-3.

**Solución**

No hay una potencia entera de 2 que sea igual a 60. La más cercana es $2^6 = 64$. Por lo tanto, un contador que utilice seis FFs actuará como un contador MOD-64. Es obvio que esto no cumple con el requerimiento. Parece que no hay solución si se utiliza un contador del tipo que se muestra en la figura 7-1. En parte esto es verdad; en la sección 7-4 veremos cómo modificar los contadores binarios básicos para que pueda obtenerse casi *cualquier* número MOD y así no estaremos limitados a los valores de 2^N .

PREGUNTAS DE REPASO

1. Verdadero o falso: en un contador asíncrono todos los FFs cambian de estado al mismo tiempo.
2. Suponga que el contador en la figura 7-1 tiene el conteo 0101. ¿Cuál será el conteo después de 27 pulsos de reloj?
3. ¿Cuál sería el número MOD del contador si se agregaran tres FFs más?

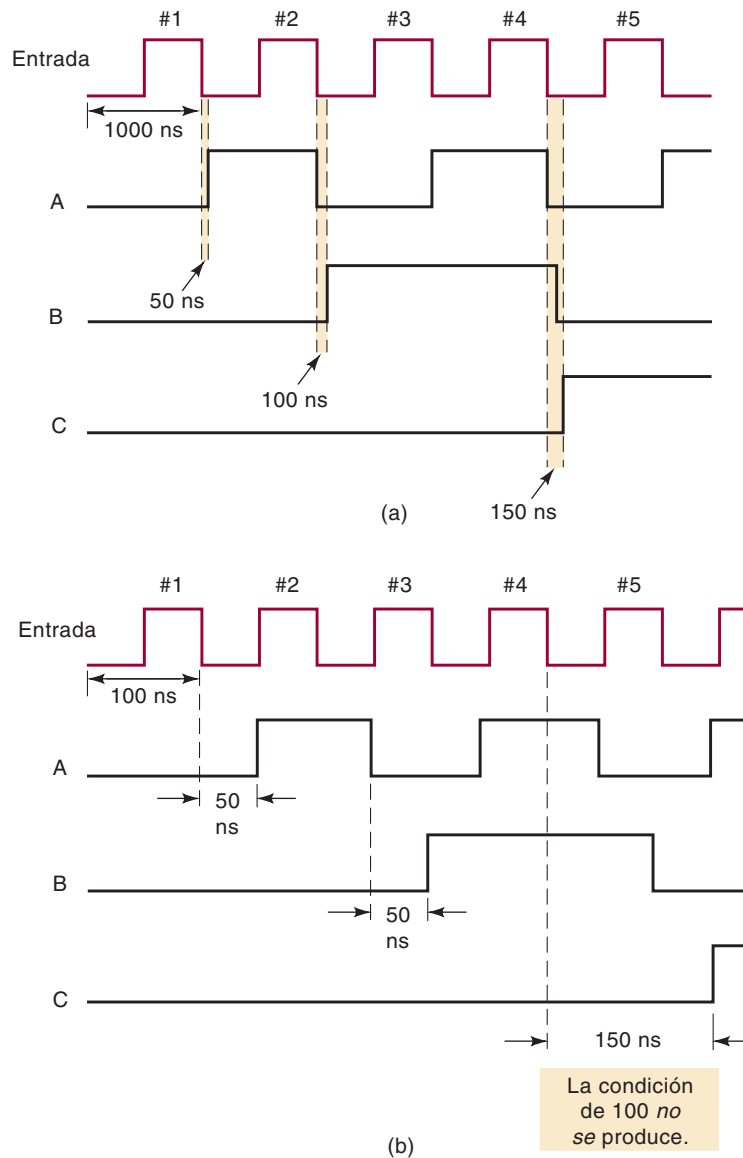
7-2 RETRASO DE PROPAGACIÓN EN CONTADORES DE RIZO

Los contadores de rizo son el tipo más simple de contadores binarios, ya que requieren la menor cantidad de componentes para producir una operación de conteo dada. No obstante tienen una gran desventaja debido a su principio básico de operación: cada FF se dispara mediante la transición en la salida del FF anterior. Debido al tiempo de retraso de propagación inherente (t_{pd}) de cada FF, esto significa que el segundo FF no responderá sino hasta un tiempo t_{pd} después de que el primer FF reciba una transición activa de reloj; el tercer FF no responderá sino hasta un tiempo igual a $2 \times t_{pd}$ después de esa transición del reloj; y así en lo sucesivo. En otras palabras, los retrasos de propagación de los FFs se acumulan de manera que el N -ésimo FF no puede cambiar de estado sino hasta un tiempo igual a $N \times t_{pd}$ después de que ocurre la transición del reloj. Esto se ilustra en la figura 7-4, en donde se muestran las formas de onda para un contador de rizo de tres bits.

El primer conjunto de formas de onda en la figura 7-4(a) muestra una situación en la que se produce un pulso de entrada cada 1000 ns (el periodo del reloj $T = 1000$ ns) y se supone que cada FF tiene un retraso de propagación de 50 ns ($t_{pd} = 50$

* Consulte la sección 5-21.

FIGURA 7-4 Formas de onda de un contador de rizo de tres bits, en donde se ilustran los efectos de los retrasos de propagación de los FFs para distintas frecuencias del pulso de entrada.



ns). Observe que la salida del flip-flop *A* conmuta 50 ns después de la NGT de cada pulso de entrada. De manera similar, *B* conmuta 50 ns después de que *A* cambia de 1 a 0 y *C* conmuta 50 ns después de que *B* cambia de 1 a 0. Como resultado, cuando ocurre la cuarta NGT de entrada la salida *C* cambia a ALTO después de un retraso de 150 ns. En esta situación el contador opera de manera apropiada en el sentido en que en cierto momento los FFs llegan a sus estados correctos para representar el conteo binario. No obstante, la situación se empeora si los pulsos de entrada se aplican a una frecuencia mucho mayor.

Las formas de onda en la figura 7-4(b) muestran lo que ocurre si los pulsos de entrada se producen una vez cada 100 ns. De nuevo, cada salida de FF responde 50 ns después de la transición de 1 a 0 en su entrada *CLK* (observe el cambio en la escala de tiempo relativa). Es de particular interés la situación después del flanco de caída del *cuarto* pulso de entrada, en donde la salida *C* no cambia a ALTO sino hasta 150 ns después, que viene siendo el mismo tiempo que transcurre hasta que la salida *A* cambia a ALTO en respuesta al *quinto* pulso de entrada. En otras palabras, la condición $C = 1, B = A = 0$ (conteo de 100) nunca aparece, ya que la frecuencia de entrada es demasiado alta. Esto podría provocar un grave problema si esta condición fuera a utilizarse para controlar alguna otra operación en un sistema digital.

Se pueden evitar problemas como éste si el periodo entre los pulsos de entrada se alarga más que el retraso de propagación total del contador. Esto es, para que el contador opere en forma correcta necesitamos que

$$T_{\text{reloj}} \geq N \times t_{\text{pd}} \quad (7-2)$$

en donde N = el número de FFs. Dicho en términos de la frecuencia del reloj de entrada, la frecuencia máxima que puede utilizarse se da mediante la siguiente ecuación:

$$f_{\text{máx}} = \frac{1}{N \times t_{\text{pd}}} \quad (7-3)$$

Suponga, por ejemplo, que se construye un contador de rizo de cuatro bits utilizando el flip-flop J-K 74LS112. La tabla 5-2 muestra que el 74LS112 tiene los retrasos de propagación $t_{\text{PLH}} = 16$ ns y $t_{\text{PHL}} = 24$ ns de CLK a Q . Para calcular $f_{\text{máx}}$ supondremos el “peor caso”; es decir, utilizaremos $t_{\text{pd}} = t_{\text{PLH}} = 24$ ns, para que

$$f_{\text{máx}} = \frac{1}{4 \times 24 \text{ ns}} = 10.4 \text{ MHz}$$

Es evidente que a medida que aumenta el número de FFs en el contador, se incrementa el tiempo de propagación total y se decrementa $f_{\text{máx}}$. Por ejemplo, un contador de rizo que utiliza seis FFs 74LS112 tendrá

$$f_{\text{máx}} = \frac{1}{6 \times 24 \text{ ns}} = 6.9 \text{ MHz}$$

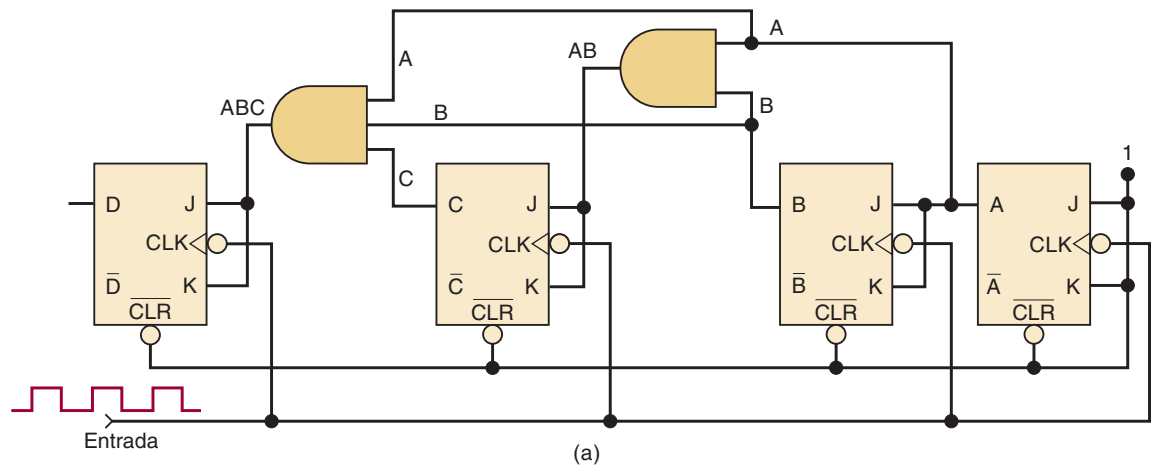
Por lo tanto, los contadores asíncronos no son útiles a frecuencias muy altas, en especial para los contadores con grandes números de bits. Otro problema que ocasionan los retrasos de propagación en los contadores asíncronos ocurre cuando tratamos de detectar por medio electrónico (*decodificar*) los estados de salida del contador. Si analiza con detalle la figura 7-4(a), durante un breve tiempo (50 ns en nuestro ejemplo) justo después del estado 011 verá que el estado 010 ocurre antes que el estado 100. Es obvio que ésta no es la secuencia correcta de conteo binario, y aunque el ojo humano es demasiado lento como para ver este estado temporal, nuestros circuitos digitales serán lo bastante rápidos como para detectarlo. Estos patrones de conteo erróneos pueden generar lo que se conoce como **deformaciones** en las señales producidas por sistemas digitales que utilizan contadores asíncronos. A pesar de su simpleza, estos problemas limitan la utilidad de los contadores asíncronos en aplicaciones digitales.

PREGUNTAS DE REPASO

1. Explique por qué la limitación de frecuencia máxima de un contador de rizo disminuye a medida que se agregan más FFs al contador.
2. Cierta flip-flop J-K tiene $t_{\text{pd}} = 12$ ns. ¿Cuál es el contador MOD más grande que puede construirse a partir de estos FFs y que pueda operar hasta 10 MHz?

7-3 CONTADORES SÍNCRONOS (EN PARALELO)

Los problemas que se encuentran en los contadores de rizo son ocasionados por los retrasos de propagación acumulados en los FF; dicho de otra forma, los FFs no cambian de estado todos al mismo tiempo, en sincronía con los pulsos de entrada. Estas limitaciones pueden resolverse con el uso de los **contadores en paralelo o síncronos**,



Conteo	D	C	B	A
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1
0	0	0	0	0
.	.	.	.	.
.	.	.	.	.
.	.	etc.	.	.

(b)



FIGURA 7-5 Contador MOD-16 síncrono. Cada FF está sincronizado por la NGT de la señal de entrada de reloj, de manera que todas las transiciones de los FFs ocurran al mismo tiempo.

en los cuales todos los FFs se disparan al mismo tiempo (en paralelo) mediante los pulsos de entrada de reloj.

Debido a que los pulsos de entrada se aplican a todos los FFs, debe utilizarse algún medio para controlar cuándo debe conmutar un FF y cuándo no debe ser afectado por un pulso de reloj. Esto se logra mediante el uso de las entradas J y K , y se ilustra en la figura 7-5 para un contador síncrono MOD-16 de cuatro bits.

Si comparamos el arreglo del circuito para este contador síncrono con su contraparte asíncrona de la figura 7-1, podremos ver las siguientes diferencias notables:

- Las entradas CLK de todos los FFs están conectadas en conjunto, de manera que la señal de reloj de entrada se aplique a cada FF al mismo tiempo.
- Sólo el flip-flop A , el LSB, tiene sus entradas J y K de manera permanente en el nivel ALTO. Las entradas J , K de los demás FFs se excitan mediante cierta combinación de salidas de los FFs.
- El contador síncrono requiere más circuitos que el contador asíncrono.

Operación del circuito

Para que este circuito pueda contar en forma apropiada, en una NGT dada del reloj sólo los FFs que se supone deben conmutar en esa NGT tienen la condición $J = K = 1$ cuando ocurre esa NGT. Analicemos la secuencia de conteo en la figura 7-5(b) para ver lo que esto significa para cada FF.

La secuencia de conteo muestra que el flip-flop *A* debe cambiar de estado en cada NGT. Por esta razón sus entradas *J* y *K* se mantienen en ALTO de manera permanente, para que conmute en cada NGT de la entrada de reloj.

La secuencia de conteo muestra que el flip-flop *B* debe cambiar de estado en cada NGT que se produzca mientras que $A = 1$. Por ejemplo, cuando el conteo es 0001 la siguiente NGT debe conmutar a *B* al estado 1; cuando el conteo es 0011, la siguiente NGT debe conmutar a *B* al estado 0; y así en lo sucesivo. Para lograr esta operación se conecta la salida de *A* con las entradas *J* y *K* del flip-flop *B*, de manera que $J = K = 1$ sólo cuando $A = 1$.

La secuencia de conteo muestra que el flip-flop *C* debe cambiar de estado en cada NGT que se produzca mientras que $A = B = 1$. Por ejemplo, cuando el conteo es 0011 la siguiente NGT debe conmutar a *C* al estado 1; cuando el conteo es 0111 la siguiente NGT debe conmutar a *C* al estado 0; y así en lo sucesivo. Al conectar la señal lógica *AB* a las entradas *J* y *K* del FF *C*, este FF conmutará sólo cuando $A = B = 1$.

De manera similar, podemos ver que el flip-flop *D* debe conmutar en cada NGT que se produzca mientras que $A = B = C = 1$. Cuando el conteo es 0111, la siguiente NGT debe conmutar a *D* al estado 1; cuando el conteo es 1111 la siguiente NGT debe conmutar a *D* al estado 0. Al conectar la señal lógica *ABC* a las entradas *J* y *K* del FF *D*, este FF conmutará sólo cuando $A = B = C = 1$.

Por lo tanto, podemos establecer el principio básico para construir un contador de la siguiente manera:

Cada FF deberá tener sus entradas *J* y *K* conectadas de manera que estén en ALTO sólo cuando las salidas de todos los FFs de menor orden se encuentren en el estado ALTO.

Ventaja de los contadores síncronos en comparación con los contadores asíncronos

En un contador en paralelo todos los FFs cambiarán de estado al mismo tiempo; esto es, todos están sincronizados con las NGTs de los pulsos de entrada del reloj. En consecuencia y a diferencia de los contadores asíncronos, los retrasos de propagación de los FFs no se acumulan para producir el retraso total, sino que el tiempo de respuesta total de un contador síncrono como el de la figura 7-5 es el tiempo que requiere un FF para conmutar, más el tiempo para que los nuevos niveles lógicos se propaguen a través de una sola compuerta AND para llegar a las entradas *J*, *K*. Por lo tanto, para un contador síncrono,

$$\text{retraso total} = t_{pd} \text{ del FF} + t_{pd} \text{ de la compuerta AND}$$

Este retraso total es el mismo, sin importar cuántos FFs haya en el contador y, por lo general, será mucho menor que en un contador asíncrono con el mismo número de FFs. Por ende, un contador síncrono puede operar a una frecuencia de entrada mucho más alta. Desde luego que los circuitos del contador síncrono son más complejos que los del contador asíncrono.

CI's reales

Existen muchos contadores de CI síncronos tanto en la familia lógica TTL como en la CMOS. Algunos de los dispositivos que se utilizan con más frecuencia son:

- 74ALS160/162, 74HC160/162: contadores síncronos de décadas.
- 74ALS161/163, 74HC161/163: contadores MOD-16 síncronos.

EJEMPLO 7-4

- Determine el valor de $f_{\text{máx}}$ para el contador de la figura 7-5(a) si el t_{pd} para cada FF es de 50 ns y el t_{pd} para cada compuerta AND es de 20 ns. Compare este valor con $f_{\text{máx}}$ para un contador de rizo MOD-16.
- ¿Qué debe hacerse para convertir este contador en MOD-32?
- Determine el valor de $f_{\text{máx}}$ para el contador MOD-32 en paralelo.

Solución

- El retraso total que debe permitirse entre los pulsos de reloj de entrada es igual al t_{pd} del FF + el t_{pd} de la compuerta AND. Por ende, $T_{\text{reloj}} \geq 50 + 20 = 70$ ns, y en consecuencia el contador en paralelo tiene

$$f_{\text{máx}} = \frac{1}{70 \text{ ns}} = 14.3 \text{ MHz contador en paralelo}$$

Un contador de rizo MOD-16 utiliza cuatro FFs con $t_{\text{pd}} = 50$ ns. Por lo tanto, la $f_{\text{máx}}$ para el contador de rizo es

$$f_{\text{máx}} = \frac{1}{4 \times 50 \text{ ns}} = 5 \text{ MHz contador de rizo}$$

- Debe agregarse un quinto FF, ya que $2^5 = 32$. La entrada CLK de este FF también está enlazada a los pulsos de entrada. Sus entradas J y K se alimentan mediante la salida de una compuerta AND de cuatro entradas A , B , C y D .
- La $f_{\text{máx}}$ se determina igual que en (a), sin importar el número de FFs en el contador en paralelo. Por lo tanto, $f_{\text{máx}}$ sigue siendo 14.3 MHz.

PREGUNTAS DE REPASO

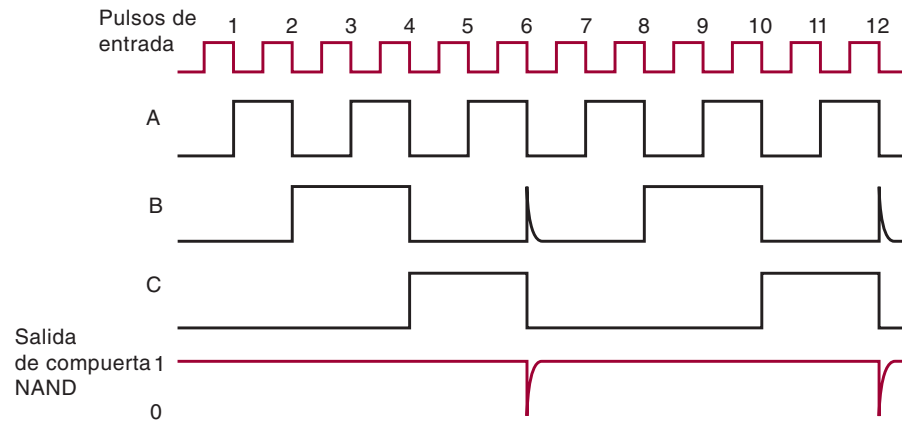
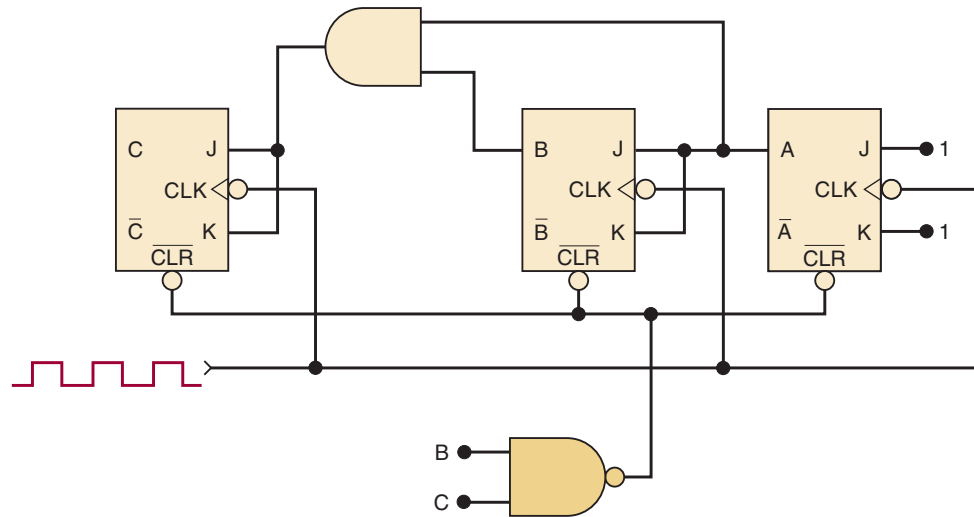
- ¿Cuál es la ventaja de un contador síncrono, en comparación con un contador asíncrono? ¿Cuál es la desventaja?
- ¿Cuántos dispositivos lógicos se requieren para un contador MOD-64 en paralelo?
- ¿Qué señal lógica excita las entradas J , K del flip-flop del MSB para el contador de la pregunta 2?

7-4 CONTADORES CON NÚMEROS MOD < 2^N

El contador síncrono básico de la figura 7-5 está limitado a números MOD que sean iguales a 2^N , en donde N es el número de FFs. En realidad este valor es el número MOD máximo que puede obtenerse mediante el uso de N flip-flops. El contador básico puede modificarse para producir números MOD menores que 2^N si se permite al contador *omitir estados* que, por lo general, forman parte de la secuencia de conteo. Uno de los métodos más comunes para esto se ilustra en la figura 7-6, en donde se muestra un contador de tres bits. Si descartamos la compuerta NAND por un momento, podremos ver que el contador es del tipo binario MOD-8, el cual contará en secuencia desde 0000 hasta 1111. No obstante, la presencia de la compuerta NAND alterará esta secuencia de la siguiente manera:

- La salida NAND se conecta a las entradas LIMPIAR asíncronas de cada FF. Mientras que la salida de la compuerta NAND esté en ALTO no tendrá efecto sobre el contador. Pero cuando cambie a BAJO borrará todos los FFs, lo cual provocará que el contador cambie de inmediato al estado 000.

FIGURA 7-6 Contador MOD-6 que se produce al borrar un contador MOD-8 cuando llega a un conteo de seis (110).



- Las entradas de la compuerta NAND son las salidas de los flip-flops *B* y *C*, por lo que la salida de la compuerta NAND cambiará a BAJO siempre que $B = C = 1$. Esta condición se producirá cuando el contador cambie del estado 101 al 110 en la NGT del pulso de entrada 6. El nivel BAJO en la salida de la compuerta NAND borrará de inmediato (por lo general, en unos cuantos nanosegundos) el contador para que quede en el estado 000. Una vez que se hayan borrado los FFs, la salida de la compuerta NAND cambiará de vuelta al nivel ALTO, ya que la condición $B = C = 1$ dejó de existir.
- Por lo tanto, la secuencia de conteo es

CBA

000	←
001	
010	
011	
100	
101	
110	→ (estado temporal necesario para borrar el contador)

Aunque el contador cambia al estado 110, permanece ahí sólo durante unos cuantos nanosegundos antes de reciclarse a 000. Por lo tanto, en esencia podemos decir que este contador cuenta desde 000 (cero) hasta 101 (cinco) y después se recicla a 000. Omite los estados 110 y 111, de manera que sólo pasa por seis estados distintos; por lo tanto es un contador MOD-6.

Observe que la forma de onda en la salida de *B* contiene un *pico* o *deformación* producida por la ocurrencia momentánea del estado 110 antes de que el FF se borre. Esta deformación es muy estrecha y no producirá ninguna indicación visual en LEDs indicadores o en pantallas numéricas. No obstante, podría provocar un problema si la salida *B* se utilizara para excitar otros circuitos fuera del contador. Hay que observar también que la salida *C* tiene una frecuencia igual a un sexto de la frecuencia de entrada; en otras palabras, este contador MOD-6 ha dividido la frecuencia de entrada entre *seis*. La forma de onda en *C* *no es* una onda cuadrada simétrica (ciclo de trabajo del 50 por ciento) debido a que está en ALTO sólo durante dos ciclos de reloj, mientras que está en BAJO durante cuatro ciclos.

Diagrama de transición de estados

La figura 7-7(a) es el diagrama de transición de estados para el contador MOD-6 de la figura 7-6. En este diagrama se muestra cómo los FFs *C*, *B* y *A* cambian de estado a medida que se aplican pulsos en la entrada *CLK* del flip-flop *A*. Recuerde que cada círculo representa uno de los posibles estados del contador y que las flechas indican cómo un estado cambia a otro en respuesta a un pulso de reloj de entrada.

Si suponemos un conteo inicial de 000, el diagrama nos muestra que los estados del contador cambian, por lo general, en forma ascendente hasta llegar al conteo de 101. Cuando se produce el siguiente pulso de reloj, el contador cambia de manera temporal al conteo 110 antes de pasar al conteo 000 estable. Las líneas punteadas indican la naturaleza temporal del estado 110. Como dijimos antes, la duración de este estado temporal es tan breve que para la mayoría de los fines podemos considerar que el contador cambia en forma directa de 101 a 000 (flecha sólida).

Observe que no hay una flecha que vaya al estado 111, ya que el contador nunca podrá avanzar hasta ese estado. No obstante, puede producirse el estado 111 al momento de encender el circuito, cuando los FFs prenden en estados aleatorios. Si esto ocurre, la condición 111 producirá un nivel BAJO en la salida de la compuerta NAND y de inmediato se borrará el contador para quedar en 000. Por ende, el estado 111 también es una condición temporal que termina en 000.

Visualización de los estados de un contador

Algunas veces durante la operación normal, y con mucha frecuencia durante la prueba, es necesario tener una pantalla visual de la manera en que el contador cambia de estados en respuesta a los pulsos de entrada. Más adelante veremos con detalle varias formas de hacer esto. Por ahora, la figura 7-7(b) muestra uno de los métodos más simples, en el que se utilizan LEDs indicadores individuales para cada salida de FF. Cada salida se conecta a un INVERSOR cuya salida proporciona la ruta de la corriente para el LED. Por ejemplo, cuando la salida *A* está en ALTO, la salida del INVERSOR cambia a BAJO y el LED se enciende. Un LED encendido indica que $A = 1$. Cuando la salida *A* está en BAJO, la salida del INVERSOR está en ALTO y el LED se apaga. Cuando pasa esto, nos indica que $A = 0$.

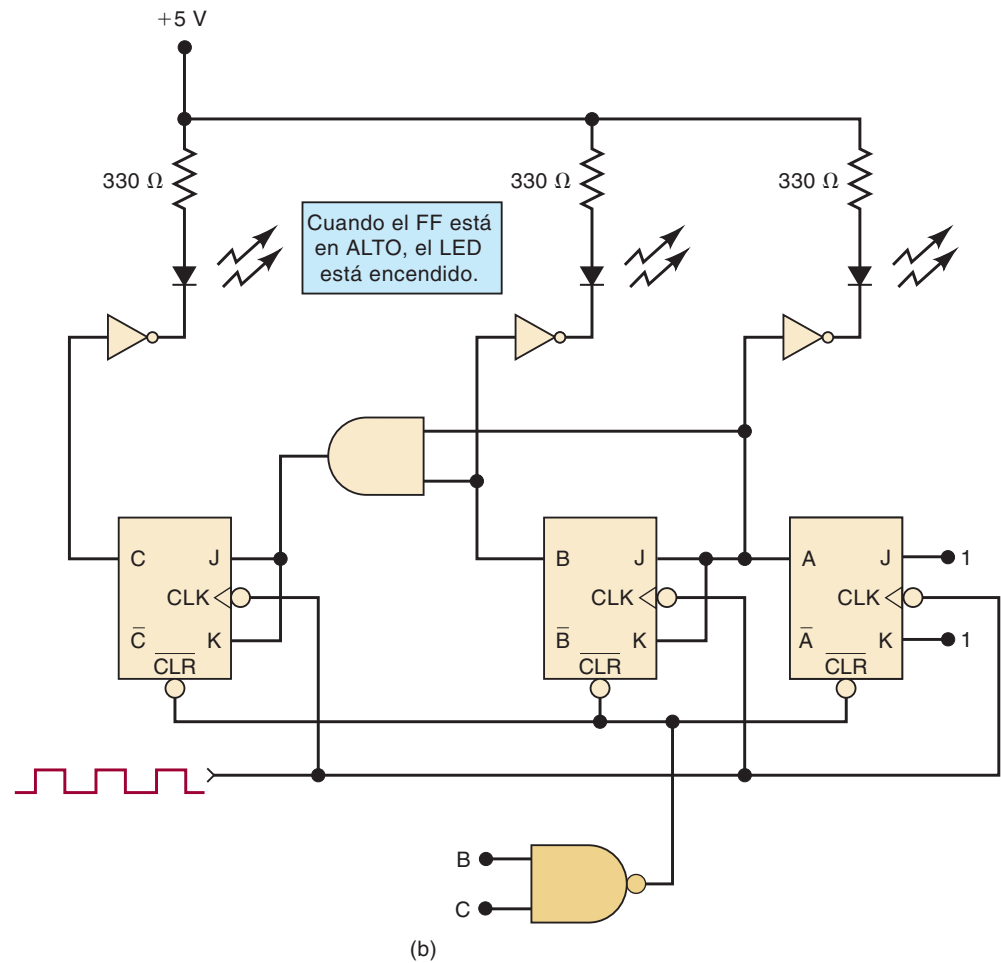
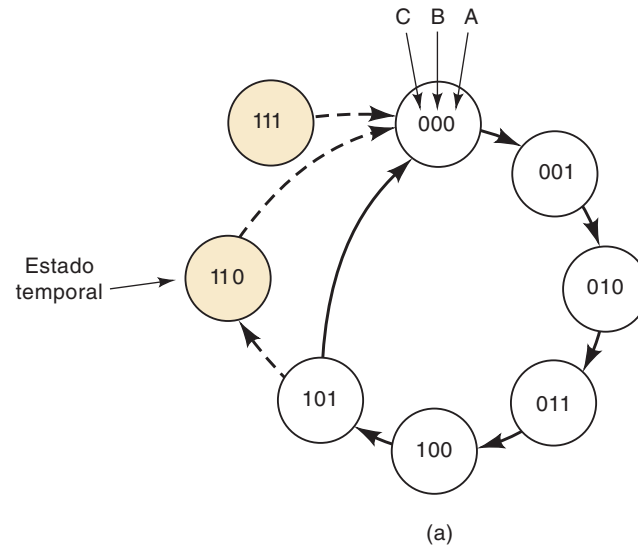


FIGURA 7-7 (a) Diagrama de transición de estados para el contador MOD-6 de la figura 7-6. (b) A menudo se utilizan LEDs para visualizar los estados de un contador.

EJEMPLO 7-6

- (a) ¿Cuál será el estado de los LEDs cuando el contador tenga el conteo de cinco?
- (b) ¿Qué mostrarán los LEDs cuando el contador esté sincronizado por una entrada de 1 kHz?
- (c) ¿Será visible el estado 110 en los LEDs?

Solución

- (a) Como $5_{10} = 101_2$, los LEDs 2^0 y 2^2 estarán encendidos y el LED 2^1 estará apagado.
- (b) A 1 kHz, los LEDs estarán cambiando entre encendido y apagado tan rápido que parecerá (al ojo humano) que están encendidos todo el tiempo, con un brillo aproximado a la mitad del normal.
- (c) No; el estado 110 persistirá durante sólo unos cuantos nanosegundos, hasta que el contador se recicle a 000.

Cambiar el número MOD

El contador de las figuras 7-6 y 7-7 es del tipo MOD-6 debido a la elección de entradas que van a la compuerta NAND. Si se cambian estas entradas, podrá obtenerse cualquier número MOD que se desee. Por ejemplo, si utilizamos una compuerta NAND de tres entradas A , B y C , el contador funcionaría en forma normal hasta que llegara a la condición 111, punto en el cual se restablecería de inmediato al estado 000. Si ignoramos la excursión temporal al estado 111, el contador iría de 000 a 110 y después se reciclaría de nuevo a 000, lo cual nos lleva a deducir que es un contador MOD-7 (7 estados).

EJEMPLO 7-6

Determine el número MOD del contador en la figura 7-8(a). Determine además la frecuencia en la salida D .

Solución

Éste es un contador de cuatro bits, que, por lo general, contaría desde 0000 hasta 1111. Las entradas NAND son D , C y B , lo cual significa que el contador se reciclará de inmediato a 0000 cuando se llegue al conteo de 1110 (14 decimal). En consecuencia, el contador en realidad tiene 14 estados estables (de 0000 a 1101) y es, por lo tanto, del tipo MOD-14. Como la frecuencia de entrada es de 30 kHz, la frecuencia en la salida D será

$$\frac{30 \text{ kHz}}{14} = 2.14 \text{ kHz}$$

Procedimiento general

Para construir un contador que empiece desde 0 y que tenga un número MOD X :

1. Busque el número más pequeño de FFs tal que $2^N \geq X$, y conéctelos para formar un contador. Si $2^N = X$, no realice los pasos 2 y 3.
2. Conecte una compuerta NAND a las entradas LIMPIAR asíncronas de todos los FFs.
3. Determine cuáles FFs estarán en el estado ALTO cuando el conteo sea igual a X ; después conecte las salidas normales de estos FFs a las entradas de la compuerta NAND.

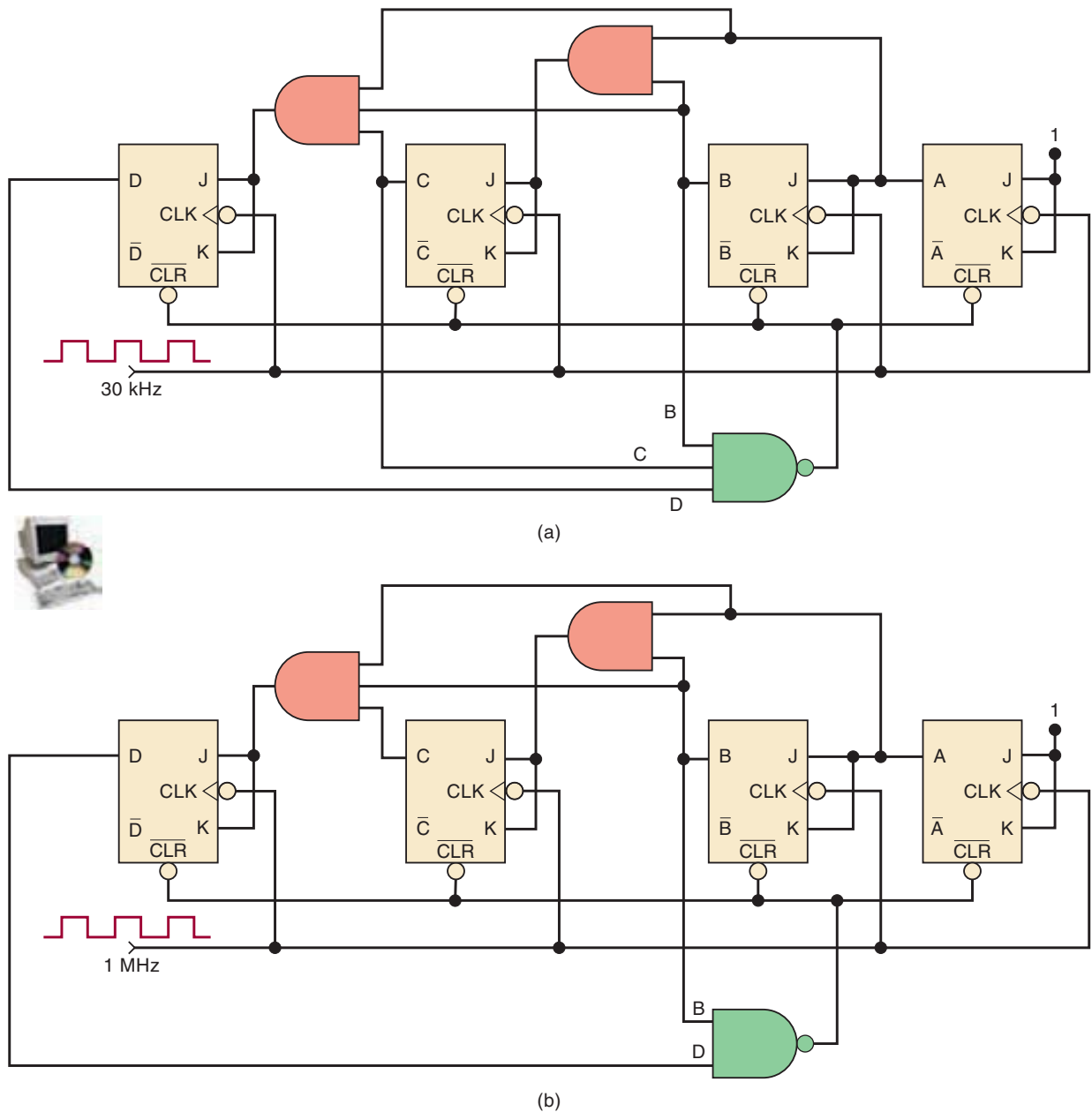


FIGURA 7-8 (a) Contador de rizo MOD-14; (b) Contador de rizo MOD-10 (de décadas).

EJEMPLO 7-7

Construya un contador MOD-10 que vaya desde 0000 (cero) hasta 1001 (9 decimal).

Solución

$2^3 = 8$ y $2^4 = 16$; entonces se requieren cuatro FFs. Como el contador debe tener una operación estable hasta el conteo de 1001, debe restablecerse a cero cuando llegue al conteo de 1010. Por lo tanto, las salidas de los FFs *D* y *B* deberán conectarse como entradas para la compuerta NAND. La figura 7-8(b) muestra el arreglo.

Contadores de décadas/contadores BCD

Al contador MOD-10 del ejemplo 7-7 también se le conoce como **contador de décadas**. De hecho, un contador de décadas es cualquiera que tenga 10 estados distintos,

sin importar la secuencia. A un contador de décadas como el de la figura 7-8(b), que cuenta en secuencia desde 0000 (cero) hasta 1001 (9 decimal), también se le conoce comúnmente como **contador BCD** ya que sólo utiliza los 10 grupos de código BCD 0000, 0001, . . . , 1000, y 1001. Para reiterar, cualquier contador MOD-10 es de décadas y cualquier contador de décadas que cuente en binario desde 0000 hasta 1001 es del tipo BCD.

Los contadores de décadas (en especial el tipo BCD) tienen un amplio uso en aplicaciones en las que se van a contar pulsos o eventos y los resultados se mostrarán en algún tipo de indicador numérico decimal. Más adelante examinaremos esto con mayor detalle. Un contador de décadas también se utiliza a menudo para dividir la frecuencia de un pulso *exactamente* entre 10. Los pulsos de entrada se aplican a las entradas de reloj en paralelo, y los pulsos de salida se toman de la salida del flip-flop *D*, que tiene una frecuencia igual a un décimo de la frecuencia de la señal de entrada.

EJEMPLO 7-8

En el ejemplo 7-3 se necesitaba un contador MOD-60 para dividir la frecuencia de línea de 60 Hz hasta 1 Hz. Construya un contador MOD-60 apropiado.

Solución

$2^5 = 32$ y $2^6 = 64$, por lo que necesitamos seis FFs, como muestra la figura 7-9. El contador deberá borrarse cuando llegue a 60 (111100). Entonces, las salidas de los flip-flops Q_5 , Q_4 , Q_3 y Q_2 deberán conectarse a la compuerta NAND. La salida del flip-flop Q_5 tendrá una frecuencia de 1 Hz.

PREGUNTAS DE REPASO

1. ¿Cuáles salidas de los FFs deberán conectarse a la compuerta NAND que se encarga de borrar los FFs, para formar un contador MOD-13?
2. Verdadero o falso: todos los contadores BCD son de décadas.
3. ¿Cuál es la frecuencia de salida de un contador de décadas que se sincroniza a partir de una señal de 50 kHz?

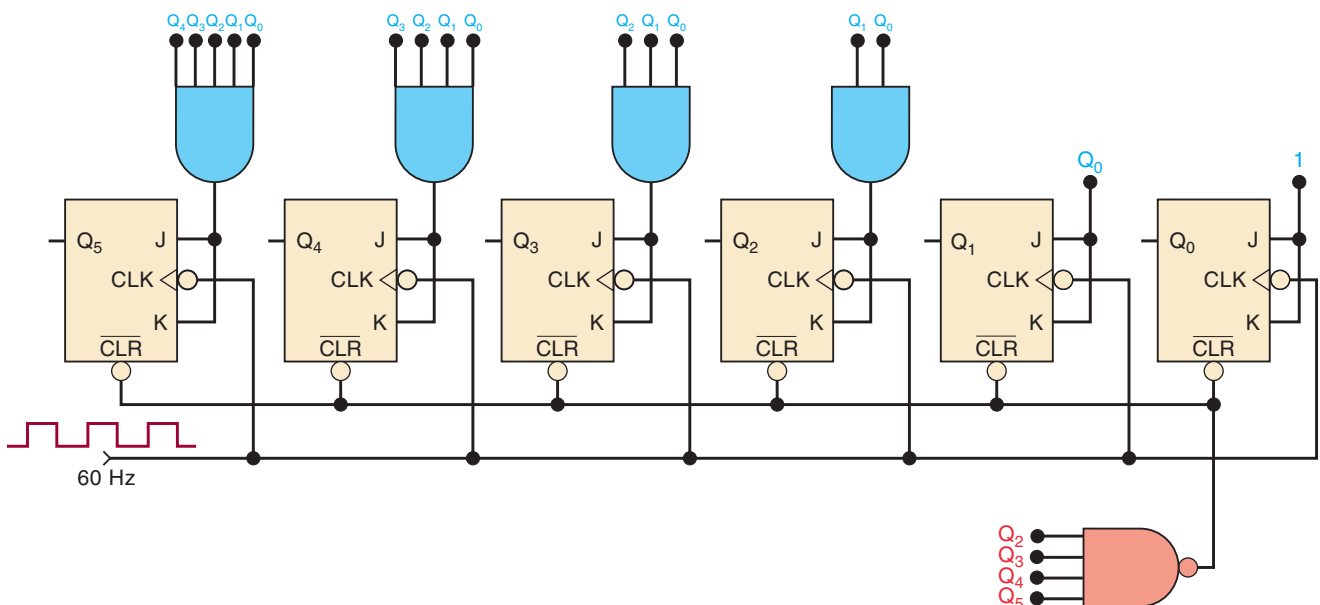


FIGURA 7-9 Contador MOD-60.

7-5 CONTADORES SÍNCRONOS DESCENDENTES Y ASCENDENTES/DESCENDENTES

En la sección 7-3 vimos que al utilizar la salida de los FFs de menor orden para controlar la conmutación de cada FF se crea un **contador ascendente** síncrono. Un **contador descendente** síncrono se construye en forma similar, sólo que utilizamos las salidas de los FF invertidas para controlar las entradas J , K de mayor orden. Si comparamos el contador descendente síncrono MOD-16 de la figura 7-10 con el contador ascendente de la figura 7-5 veremos que sólo necesitamos sustituir la correspondiente salida invertida del FF en vez de las salidas A , B y C . Para una secuencia de conteo descendente, el FF (A) del LSB aún necesita conmutar con cada NGT de la señal de entrada del reloj. El flip-flop B debe cambiar de estado en la siguiente NGT del reloj cuando $A = 0$ ($\bar{A} = 1$). El flip-flop C cambia de estado cuando $\bar{A} = \bar{B} = 0$ ($\bar{A} \cdot \bar{B} = 1$) y el flip-flop D cambia de estado cuando $\bar{A} = \bar{B} = \bar{C} = 0$ ($\bar{A} \cdot \bar{B} \cdot \bar{C} = 1$). Esta configuración del circuito producirá la secuencia de conteo: 15, 14, 13, 12, ..., 3, 2, 1, 0, 15, 14..., como se muestra en el diagrama de sincronización.

La figura 7-11(a) muestra cómo formar un **contador ascendente/descendente** en paralelo. La entrada de control Arriba/Abajo controla si se alimentan las salidas normales o las salidas invertidas del FF a las entradas J y K de los FFs sucesivos. Cuando Arriba/Abajo se mantiene en ALTO se habilitan las compuertas AND 1 y 2 mientras que se deshabilitan las compuertas AND 3 y 4 (observe el inversor). Esto permite que las salidas A y B pasen a través de las compuertas 1 y 2 a las entradas J y K de los FFs B y C . Cuando Arriba/Abajo se mantiene en BAJO, las compuertas AND 1 y 2 se deshabilitan mientras que las compuertas AND 3 y 4 se habilitan. Esto permite que las salidas A y B invertidas pasen a través de las compuertas 3 y 4 a las entradas J y K de los FFs B y C . Las formas de onda en la figura 7-11(b) ilustran la operación. Observe que para los primeros cinco pulsos de reloj, Arriba/Abajo = 1 y el contador cuenta hacia arriba; para los últimos cinco pulsos Arriba/Abajo = 0 y el contador cuenta hacia abajo.

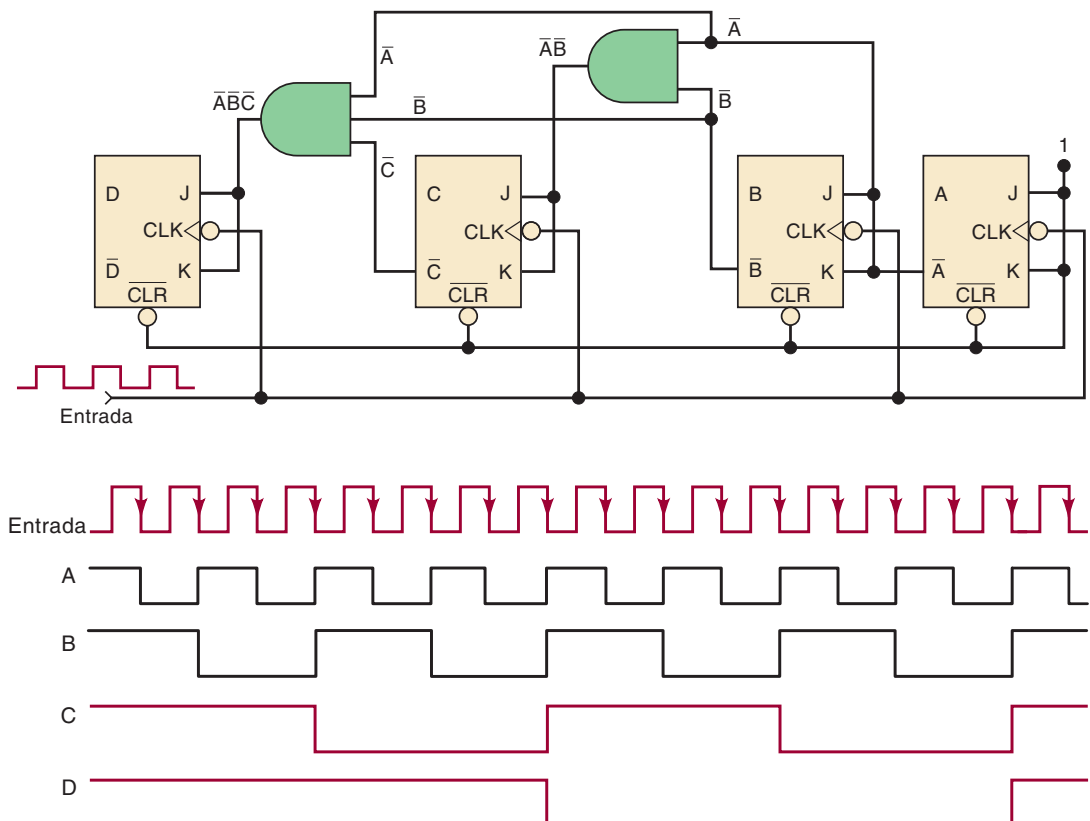


FIGURA 7-10 El contador descendente MOD-16 síncrono y las formas de onda de salida.

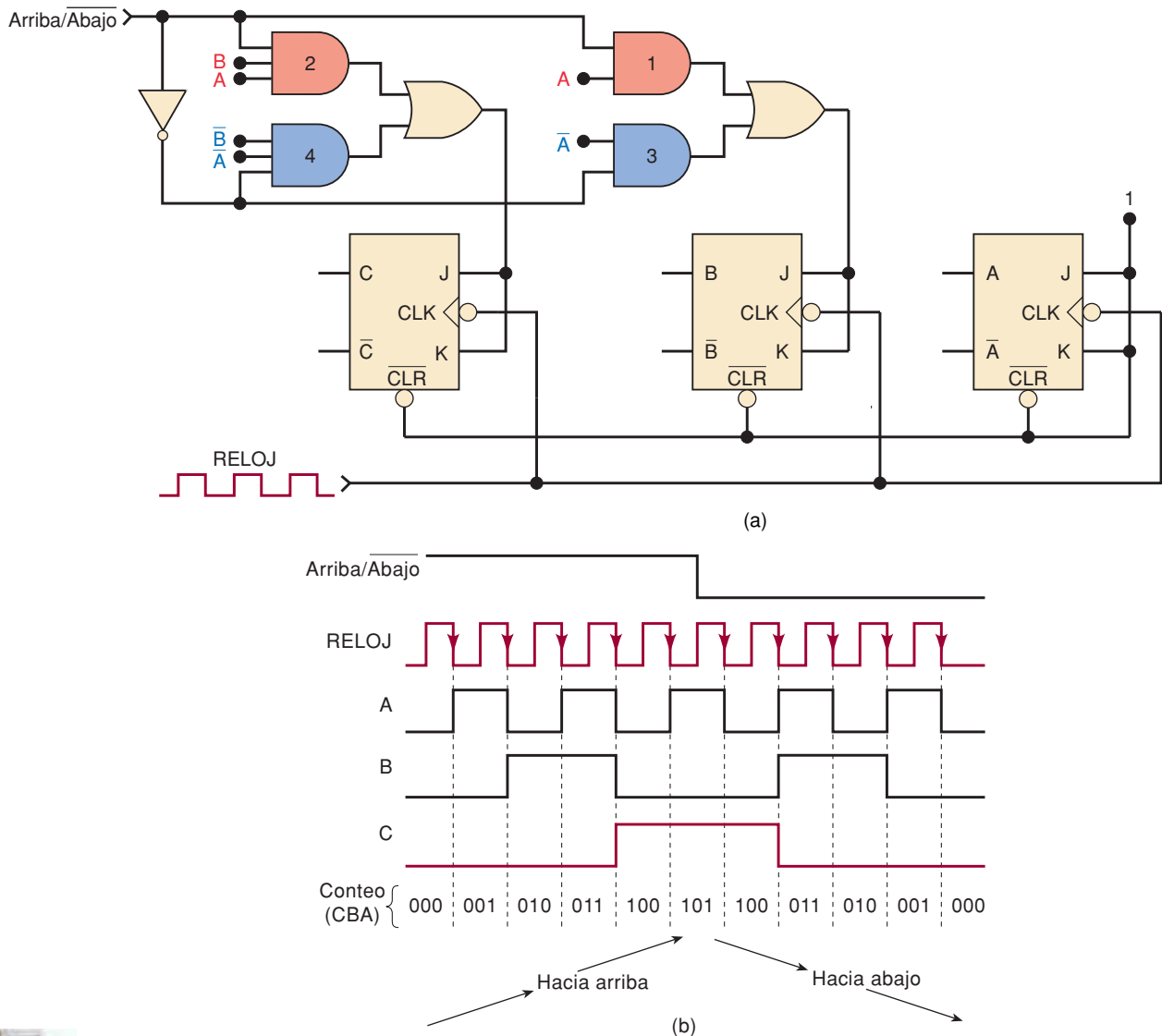


FIGURA 7-11 (a) Contador ascendente/descendente MOD-8 síncrono. (b) El contador cuenta hacia arriba cuando la entrada de control Arriba/Abajo = 1; cuenta hacia abajo cuando la entrada de control Arriba/Abajo = 0.

La nomenclatura utilizada para la señal de control ($\overline{\text{Arriba/Abajo}}$) se eligió de manera que nos haga ver claro la forma en que afecta al contador. La operación de conteo ascendente es activa en ALTO; la operación de conteo descendente es activa en BAJO.

EJEMPLO 7-9

¿Qué problemas podrían ocasionarse si la señal Arriba/Abajo cambiara de niveles en la NGT del reloj?

Solución

Los FFs podrían operar en forma impredecible, ya que en algunos de ellos sus entradas J y K estarían cambiando casi al mismo tiempo que se produjera una NGT en su entrada CLK . No obstante, los efectos del cambio en la señal de control deben propagarse a través de dos compuertas antes de llegar a las entradas J, K , por lo que es más probable que los FFs respondan en forma predecible a los niveles que hay en J, K antes de la NGT de CLK .

PREGUNTAS DE REPASO

1. ¿Cuál es la diferencia entre la secuencia de conteo de un contador ascendente y de un contador descendente?
2. ¿Qué cambios en el circuito convertirán a un contador binario síncrono ascendente en un contador binario descendente?

7-6 CONTADORES PREAJUSTABLES

Muchos contadores síncronos (en paralelo) que están disponibles como CIs están diseñados para ser **preajustables**; en otras palabras, pueden preajustarse a cualquier conteo inicial deseado, ya sea en forma asíncrona (independiente de la señal del reloj) o síncrona (en la transición activa de la señal del reloj). A esta operación de preajuste se le conoce también como **carga en paralelo** del contador.

La figura 7-12 muestra el circuito lógico para un contador ascendente en paralelo preajutable de 3 bits. Las entradas J , K y CLK se conectan para que opere como un contador ascendente en paralelo. Las entradas asíncronas PRESTABLECER y LIMPIAR se conectan para realizar un ajuste asíncrono. El contador se carga con cualquier conteo deseado en cualquier momento mediante el siguiente procedimiento:

1. Se aplica el conteo deseado a las entradas de datos en paralelo P_2 , P_1 y P_0 .
2. Se aplica un pulso BAJO a la entrada de CARGA EN PARALELO ($\overline{PL}$).

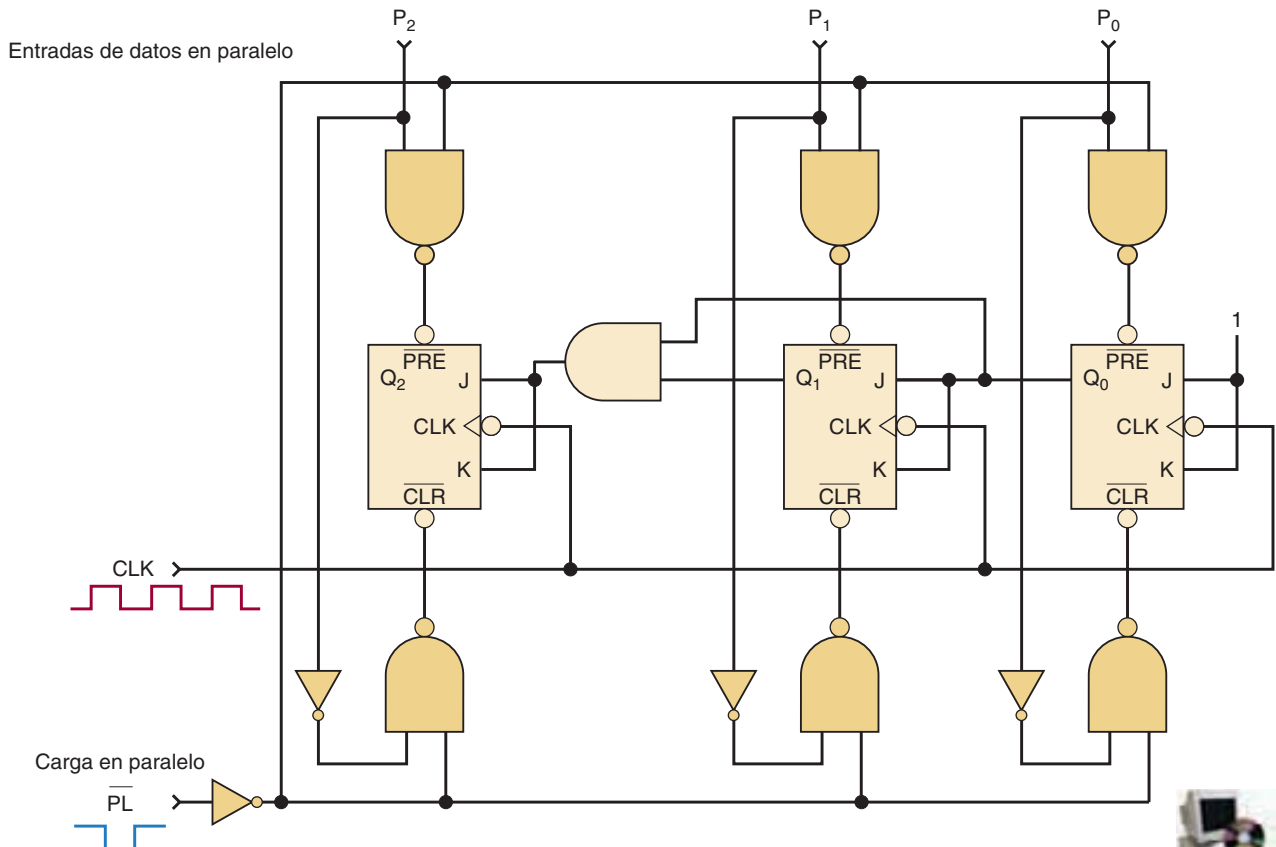


FIGURA 7-12 Contador síncrono con carga en paralelo asíncrona.

Este procedimiento realizará una transferencia asíncrona de los niveles P_2 , P_1 y P_0 hacia los flip-flops Q_2 , Q_1 y Q_0 , respectivamente (sección 5-17). Esta *transferencia forzada* ocurre de manera independiente de las entradas J , K y CLK . El efecto de la entrada CLK quedará deshabilitado mientras $\overline{PL}$ se encuentre en su estado activo en BAJO, ya que cada FF tendrá activada una de sus entradas asíncronas mientras $\overline{PL} = 0$. Una vez que $\overline{PL}$ regrese a nivel ALTO, los FFs podrán responder con sus entradas CLK y resumir la operación de conteo ascendente, empezando en la cuenta que se cargó en el contador.

Por ejemplo, digamos que $P_2 = 1$, $P_1 = 0$ y $P_0 = 1$. Mientras $\overline{PL}$ esté en ALTO, estas entradas de datos en paralelo no tendrán efecto. Si hay pulsos de reloj presentes, el contador realizará la operación normal de conteo ascendente. Ahora digamos que $\overline{PL}$ cambia a BAJO cuando el contador está en el conteo 010 (es decir, $Q_2 = 0$, $Q_1 = 1$ y $Q_0 = 0$). Este nivel BAJO en $\overline{PL}$ producirá niveles BAJO en la entrada CLR de Q_1 y en las entradas PRE de Q_2 y Q_0 , de manera que el contador cambiará al conteo 101 *sin importar lo que esté ocurriendo en la entrada CLK*. El conteo se mantendrá en 101 hasta que $\overline{PL}$ se desactive (regrese a ALTO); en ese momento el contador seguirá contando hacia arriba con cada pulso de reloj, empezando desde el conteo de 101.

Este preajuste asíncrono lo utilizan varios contadores de CI, tales como los chips TTL 74ALS190, 74ALS191, 74ALS192 y 74ALS193, y sus equivalentes en CMOS, 74HC190, 74HC191, 74HC192 y 74HC193.

Preajuste síncrono

Muchos contadores en paralelo de CI utilizan el *preajuste síncrono*, en el cual el contador se preestablece en la transición activa de la misma señal de reloj que se utiliza para el conteo. El nivel lógico en la entrada de control de carga en paralelo determina si el contador se va a preestablecer con los datos de entrada aplicados en la siguiente transición activa del reloj.

Algunos ejemplos de contadores de CI que utilizan el preajuste síncrono son: De la familia TTL 74ALS160, 74ALS161, 74ALS162 y 74ALS163, y sus equivalentes en la familia CMOS 74HC160, 74HC161, 74HC162 y 74HC163.

PREGUNTAS DE REPASO

1. ¿Qué queremos decir cuando expresamos que un contador es preajustable?
2. Describa la diferencia entre preajuste asíncrono y síncrono.

7-7 CONTADORES SÍNCRONOS DE CI

La serie 74ALS160-163/74HC160-163

La figura 7-13 muestra el símbolo lógico, el módulo y la tabla de funciones para la serie 74ALS160 a 74ALS163 de contadores de CI (y sus contrapartes equivalentes en CMOS, 74HC160 a 74HC163). Estos contadores reciclables de cuatro bits tienen sus salidas etiquetadas como QD, QC, QB, QA, en donde QA es el LSB y QD es el MSB. Están sincronizados mediante una señal PGT que se aplica a CLK. Cada uno de estos circuitos integrados tiene una combinación distinta de dos características diferentes. Como puede ver en la figura 7-13(b), dos de los contadores son MOD-10 (74ALS160 y 74ALS162), mientras que los otros dos son contadores binarios MOD-16 (74ALS161 y 74ALS163). La otra variación para estas piezas está en la operación de la función de limpiar [como se resalta en la figura 7-13(c)]. El 74ALS160 y el 74ALS161 tienen, cada uno, una entrada para limpiar asíncrona. Esto significa que tan pronto como $\overline{CLR}$ cambie a BAJO ($\overline{CLR}$ es activa en BAJO en los circuitos mencionados), la salida del contador se restablecerá a 0000. Por otro lado, los contadores de CI 74ALS162 y 74ALS163 se borran en forma síncrona. Para ello la entrada CLR debe estar en BAJO y debe aplicarse una señal PGT a la entrada de reloj. La entrada limpiar tiene prioridad sobre todas las demás funciones en esta serie de contadores de CI. Limpiar predominará sobre todas las demás entradas de control, como se indica mediante las X en la tabla de funciones de la figura 7-13(c).

FIGURA 7-13
Contadores síncronos de la serie 74ALS160-74ALS163: (a) símbolo lógico, (b) módulos; (c) tabla de funciones.

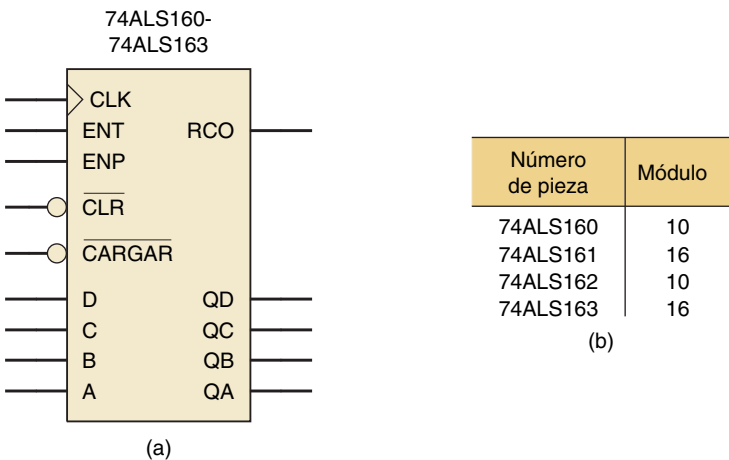


Tabla de funciones de los CIs 74ALS160-74ALS163						
CLR	CARGAR	ENP	ENT	CLK	Función	Números de parte
L	X	X	X	X	Limpiar asíncrona	74ALS160 & 74ALS161
L	X	X	X	↑	Limpiar síncrona	74ALS162 & 74ALS163
H	L	X	X	↑	Cargar síncrona	Todos
H	H	H	H	↑	Conteo ascendente	Todos
H	H	L	X	X	Sin cambios	Todos
H	H	X	L	X	Sin cambios	Todos

(c)

La función de segunda prioridad disponible en esta serie de contadores de CI es la carga en paralelo de datos en los flip-flops del contador. Para preestablecer un valor de datos, la entrada limpiar se pone inactiva (ALTO), se aplica el valor de cuatro bits deseado a las terminales de entrada de datos D, C, B, A, (A es el LSB y D es el MSB), se aplica un nivel BAJO a la entrada $\overline{\text{CARGAR}}$ y después se pone la señal de reloj al circuito por medio de una PGT. Por lo tanto, la función de carga es síncrona y tiene prioridad sobre el conteo, por lo que no importa qué niveles lógicos se apliquen a ENT o ENP. Para contar a partir del estado preestablecido será necesario deshabilitar la carga (con un nivel ALTO) y habilitar la función de conteo. Si la función de carga está inactiva, no importa lo que se aplique a las terminales de entrada de datos.

Para habilitar el conteo, la función de menor prioridad, las entradas de control CLR y CARGAR deben estar inactivas. Además hay dos controles de habilitación de conteo activos en ALTO, ENT y ENP. En esencia, se aplica una operación AND a ENT y ENP para controlar la función de conteo. Si cualquiera o ambos controles de **habilitación de conteo** están inactivos (BAJO), el contador mantendrá el estado actual. Por lo tanto, para incrementar el conteo en cada PGT de CLK las cuatro entradas de control deben estar en ALTO. Al contar, los contadores de décadas (74ALS160 y 74ALS162) se reciclarán de manera automática a 000 después del estado 1001 (9) y los contadores binarios (74ALS161 y 74ALS163) se reciclarán de manera automática después de 1111 (15).

Esta serie de contadores de CI tiene una terminal más de salida: RCO. La función de esta salida activa en ALTO es detectar (*decodificar*) el último estado (o estado terminal) del contador. El estado terminal para un contador de décadas es 1001 (9), mientras que el estado terminal para un contador MOD-16 es 1111 (15). La entrada de habilitación de conteo primaria ENT también controla la operación de RCO. ENT debe estar en ALTO para que el contador indique con la salida RCO que ha llegado a su estado terminal. Esta característica es muy útil para conectar dos o más circuitos integrados tipo contador entre sí en un arreglo de múltiples etapas para crear contadores más grandes.

EJEMPLO 7-10

Consulte la figura 7-14, en donde un 74HC163 tiene las señales de entrada que se proporcionan en el diagrama de sincronización que se aplica. Las entradas de datos en paralelo están conectadas en forma permanente como 1100. Suponga que al principio el contador se encuentra en el estado 0000 y determine las formas de onda de salida de éste.

Solución

Al principio (en t_0) todos los FFs del contador están en BAJO. Como éste no es el estado terminal para el contador, la salida RCO también estará en BAJO. La primera PGT en la entrada $\overline{\text{CLK}}$ ocurre en t_1 y, como todas las entradas de control están en ALTO, el contador se incrementará a 0001. El contador seguirá contando en forma ascendente con cada PGT hasta t_2 . En este tiempo, la entrada $\overline{\text{CLR}}$ está en BAJO. Debido a esto, el contador se restablecerá en forma síncrona al conteo 0000 en t_2 . Después de t_2 , la entrada $\overline{\text{CLR}}$ se vuelve inactiva (ALTO), por lo que el contador empezará a contar de nuevo hacia arriba, partiendo desde 0000, con cada PGT subsiguiente. En t_3 , la entrada $\overline{\text{CARGAR}}$ está en BAJO. Debido a esto, en t_3 se

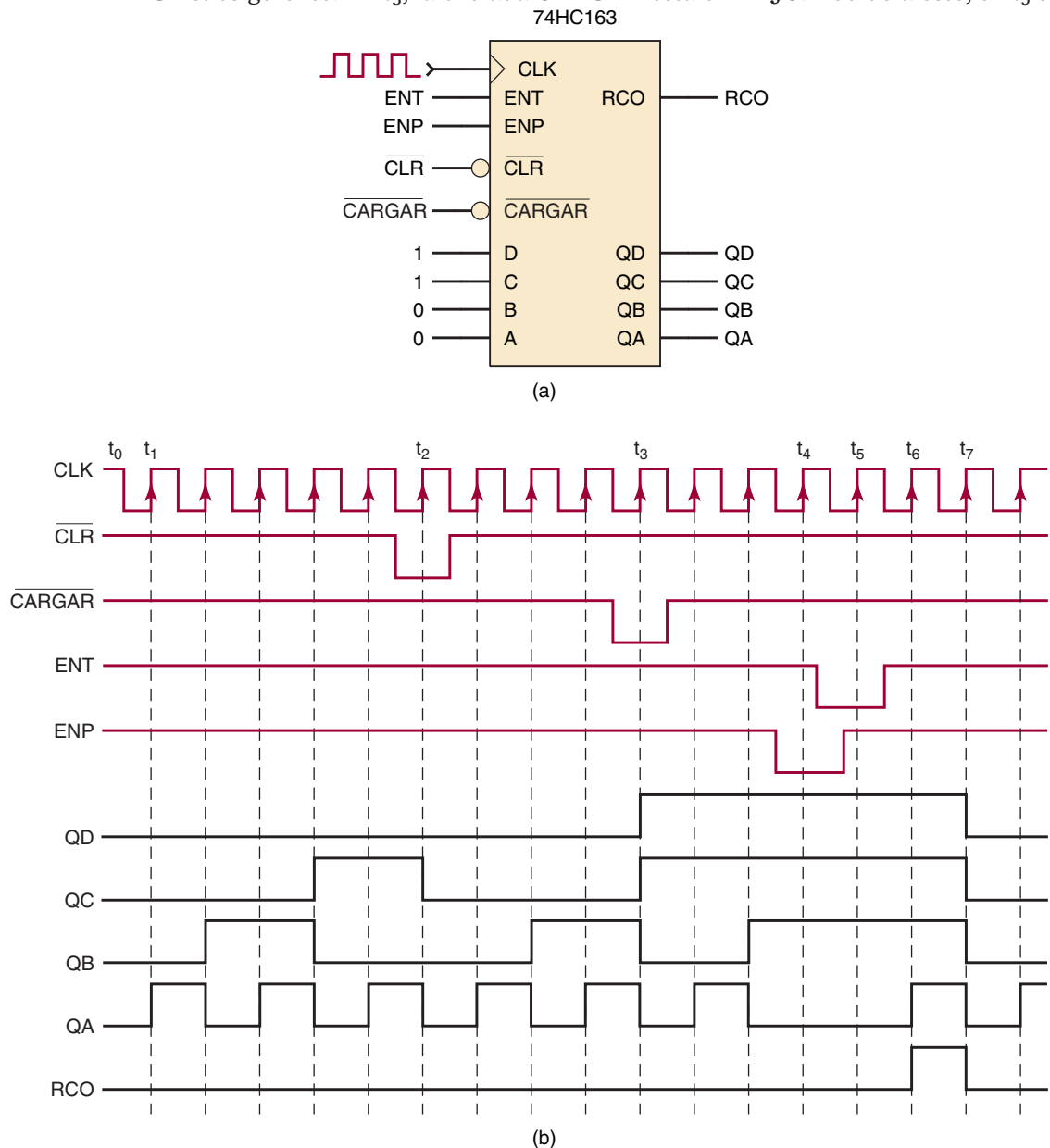


FIGURA 7-14
Ejemplo 7-10.

cargará en forma síncrona el valor de datos 1100 (12) aplicado al contador. Después de t_3 la entrada $\overline{\text{CARGAR}}$ se vuelve inactiva (ALTO), por lo que el contador seguirá contando hacia arriba, partiendo desde 1100 con cada PGT subsiguiente hasta t_4 . La salida del contador no cambia en t_4 ni en t_5 , ya que ENP o ENT (las entradas de habilitación de conteo) están en BAJO. Esto mantiene el conteo en 1110 (14). En t_6 el contador se habilita de nuevo y cuenta hasta 1111 (15), su estado terminal. Como resultado, ahora la salida RCO cambia a ALTO. En t_7 otra PGT en CLK hará que el contador se recicle a 0000 y RCO regresa al nivel BAJO.

EJEMPLO 7-11

Consulte la figura 7-15, ahí se muestra un 74HC160 al que se le aplican las señales de entrada indicadas en el diagrama de tiempo. Las entradas de datos en paralelo se conectan de manera permanente como 0111. Suponga que al principio el conta-

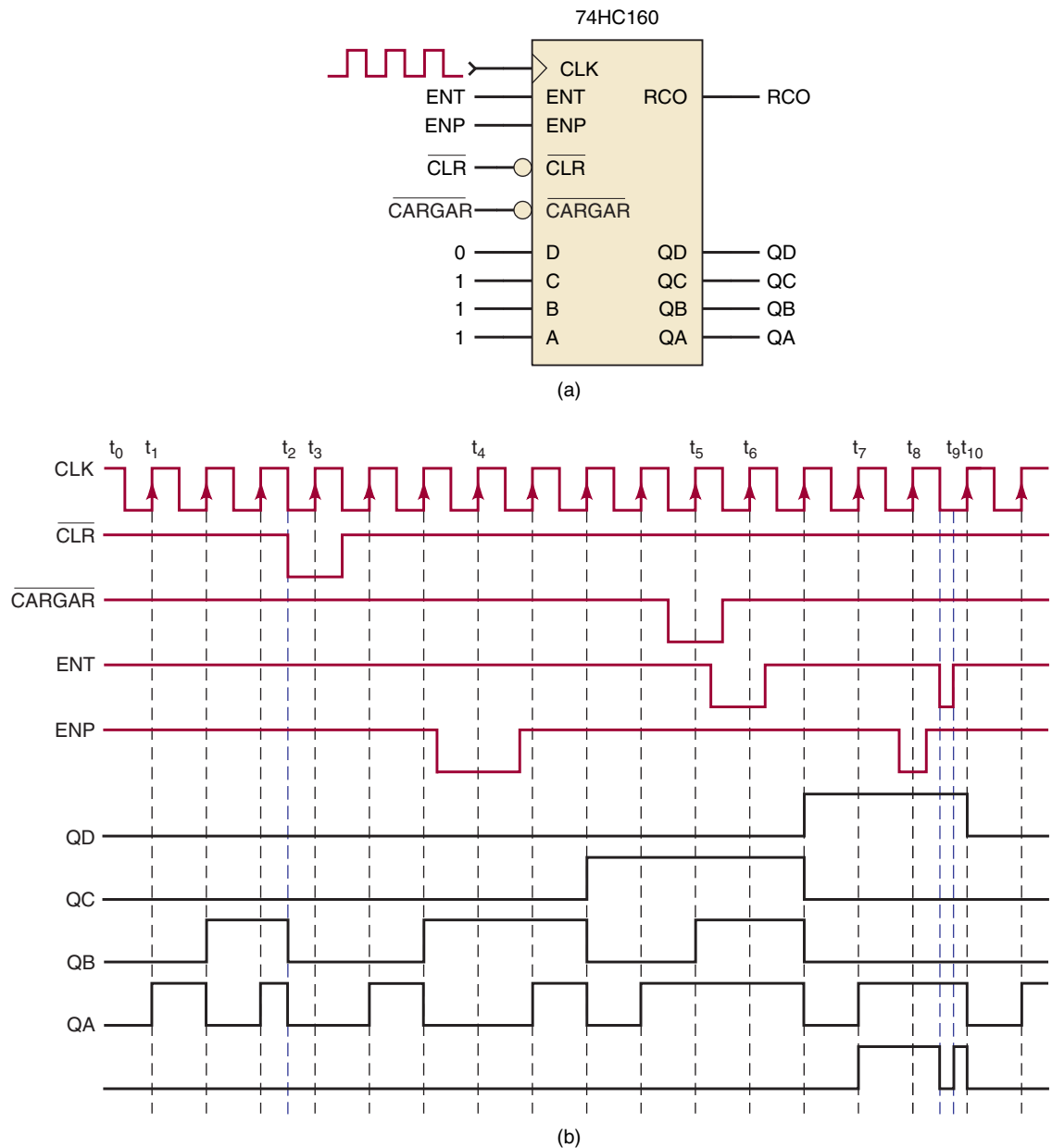


FIGURA 7-15 Ejemplo 7-11.

dor se encuentra en el estado 0000 y determine las formas de onda de salida del contador.

Solución

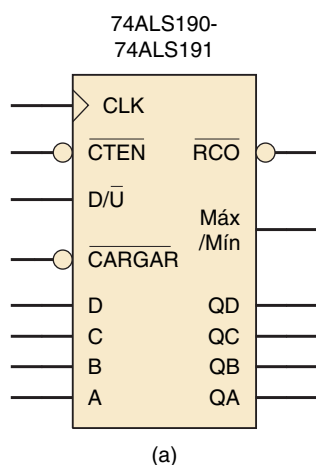
Al principio (en t_0) los FFs del contador están todos en BAJO. Como éste no es el estado terminal para el contador BCD, la salida RCO también estará en BAJO. La primera PGT en la entrada CLK ocurre en t_1 , y como todas las entradas de control están en ALTO, el contador se incrementará a 0001. El contador seguirá contando en forma ascendente con cada PGT hasta t_2 . La entrada $\overline{\text{CLR}}$ asíncrona cambia a BAJO en t_2 , con lo cual el contador se restablecerá de inmediato a 0000 en ese punto. En t_3 la entrada CLR sigue activa (BAJO), por lo que se ignorará la PGT de la entrada CLK y el contador permanecerá en 0000. Más adelante la entrada CLR se vuelve inactiva de nuevo y el contador seguirá contando hasta 0001 y después hasta 0010. En t_4 la señal de habilitación de conteo ENP está en BAJO, por lo que el conteo se mantiene en 0010. En las PGTs subsiguientes de la entrada CLK, el contador se habilita y cuenta hacia arriba hasta t_5 . La entrada $\overline{\text{CARGAR}}$ está en bajo para t_5 . Esto cargará en forma síncrona el valor de datos aplicado 0111 (7) en el contador, en t_5 . En t_6 la señal de habilitación de conteo ENT está en BAJO, por lo que el conteo se mantiene en 0111. Para las dos PGTs subsiguientes después de t_6 , el contador seguirá contando hacia arriba ya que está rehabilitado. En t_7 , el contador BCD llega a su estado terminal 1001 (9) y la salida RCO cambia a ALTO. En t_8 , ENP está en BAJO y el contador se detiene (se queda en 1001). En t_9 , mientras ENT esté en BAJO la salida RCO se deshabilitará, de forma que regrese a BAJO aún y cuando el contador siga en su estado terminal (1001). Recuerde que sólo ENT controla la salida RCO. Cuando ENT regresa a ALTO durante el estado terminal del contador, RCO cambia a ALTO de nuevo. En t_{10} el contador se habilita y se recicla a 0000, después cuenta hasta 0001 en la última PGT.

La serie 74ALS190-191/74HC190-191

La figura 7-16 muestra el símbolo lógico, el módulo y la tabla de funciones para las series de circuitos integrados contadores 74ALS190 y 74ALS191 (y sus equivalentes en CMOS, 74HC190 y 74HC191). Estos contadores reciclables de cuatro bits tienen salidas etiquetadas como QD, QC, QB, QA, en donde QA es el LSB y QD es el MSB. Se sincronizan mediante una señal PGT que se aplica en CLK. La única diferencia entre dichos circuitos es el módulo del contador. El 74ALS190 es un contador MOD-10 y el 74ALS191 es un contador binario MOD-16. Ambos circuitos son contadores ascendentes/descendentes y tienen una entrada de carga asíncrona, activa en BAJO. Esto significa que, tan pronto como $\overline{\text{CARGAR}}$ cambie a BAJO, el contador

FIGURA 7-16

Contadores síncronos de la serie 74ALS190-74ALS191: (a) símbolo lógico; (b) módulo; (c) tabla de funciones.



Número de parte	Módulo
74ALS190	10
74ALS191	16

(b)

Tabla de funciones de los circuitos 74ALS190-74ALS191

CARGAR	CTEN	D/U	CLK	Función
L	X	X	X	Carga asíncrona
H	L	L	↑	Conteo ascendente
H	L	H	↑	Conteo descendente
H	H	X	X	Sin cambio

(c)

se preestablecerá (en paralelo) con los datos disponibles en las terminales de entrada D, C, B, A (A es el LSB y D es el MSB). Si la función de carga está inactiva, no importa lo que se aplique a las terminales de datos de entrada. La entrada de carga tiene prioridad sobre la función de conteo.

Para contar, la entrada de control $\overline{\text{CARGAR}}$ debe estar inactiva (ALTO) y la entrada de control de habilitación de conteo ($\overline{\text{CTEN}}$) debe estar en BAJO. La dirección del conteo se controla mediante la entrada de control $\overline{\text{D/U}}$. Si $\overline{\text{D/U}}$ está en BAJO, el conteo se incrementa con cada PGT de CLK, mientras que un nivel ALTO en $\overline{\text{D/U}}$ disminuirá el conteo. Ambos contadores se reciclan de manera automática en cualquier dirección de conteo. El contador de décadas se recicla a 0000 después del estado 1001 (9) cuando cuenta hacia arriba, o a 1001 después del estado 0000 cuando cuenta hacia abajo. El contador binario se recicla a 0000 después de 1111 (15) cuando cuenta hacia arriba, o a 1111 después del estado 0000 cuando cuenta hacia abajo.

Estos circuitos tipo contador tienen dos terminales más de salida, MÁX/MÍN y RCO. La primera es una salida activa en ALTO que detecta (decodifica) el estado terminal del contador. Como son contadores ascendentes/descendentes, el estado terminal depende de la dirección del conteo. El estado terminal (MÍN) para cualquiera de los contadores cuando el conteo es descendente es 0000 (0). No obstante, cuando el conteo es ascendente el estado terminal (MÁX) para un contador de décadas es 1001 (9), mientras que el estado terminal para un contador MOD-16 es 1111 (15). Observe que MÁX/MÍN sólo detecta un estado en la secuencia de conteo; sólo depende de si está contando hacia arriba o hacia abajo. La salida $\overline{\text{RCO}}$ activa en BAJO también detecta el estado terminal apropiado para el contador, pero es más complicada. En primer lugar, sólo se habilita cuando $\overline{\text{CTEN}}$ está en BAJO. Además, $\overline{\text{RCO}}$ sólo estará en BAJO mientras que la entrada CLK se encuentre también en BAJO. Así que, en esencia, $\overline{\text{RCO}}$ imitará la forma de onda de CLK sólo durante el estado terminal mientras el contador esté habilitado.

EJEMPLO 7-12

Consulte la figura 7-17, en donde se muestra un circuito 74HC190 al que se le aplican las señales de entrada mostradas en el diagrama de tiempo. Las entradas de datos en paralelo se conectan de manera permanente como 0111. Suponga que al principio el contador se encuentra en el estado 0000 y determine las formas de onda de salida del contador.

Solución

Al principio (en t_0) los FFs del contador están todos en BAJO. Como el contador está habilitado ($\overline{\text{CTEN}} = 0$) y la entrada de control de dirección de conteo $\overline{\text{D/U}} = 0$, el contador BCD empezará a contar en forma ascendente en la primera señal PGT que se aplique a CLK en t_1 , y continuará contando hacia arriba con cada PGT hasta t_2 , en donde el conteo ha llegado a 0101. La entrada $\overline{\text{CARGAR}}$ asíncrona cambia a BAJO en t_2 y cargará de inmediato el valor 0111 en el contador en ese punto. En t_3 la entrada $\overline{\text{CARGAR}}$ sigue activa (BAJO), por lo que se ignorará la señal PGT en la entrada CLK y el contador permanecerá en 0111. Más adelante la entrada $\overline{\text{CARGAR}}$ cambia a ALTO de nuevo y el contador avanzará el conteo ascendente hasta 1000 en la siguiente PGT. En t_4 el contador se incrementa a 1001, que es el estado terminal para un contador BCD ascendente y en consecuencia la salida MÁX/MÍN cambiará a ALTO. Durante t_5 el contador se encuentra en su estado terminal y la entrada CLK está en BAJO, por lo que $\overline{\text{RCO}}$ cambia a BAJO. Para las PGTs subsiguientes de la entrada CLK, el contador se recicla a 0000 y continúa contando hacia arriba hasta t_6 . Justo antes de t_6 , el control $\overline{\text{D/U}}$ cambia al nivel ALTO. Esto hará que el contador cuente hacia abajo en t_6 y de nuevo en t_7 , en donde quedará en el estado 0000, que ahora es el estado terminal ya que está contando hacia abajo, y MÁX/MÍN tendrá un nivel ALTO. Durante t_8 , cuando la entrada CLK cambia a BAJO, la salida $\overline{\text{RCO}}$ estará de nuevo en BAJO. En t_9 el contador se deshabilita con $\overline{\text{CTEN}} = 1$ y mantendrá el conteo en 1001. Para los siguientes pulsos en CLK, el contador seguirá contando hacia abajo.

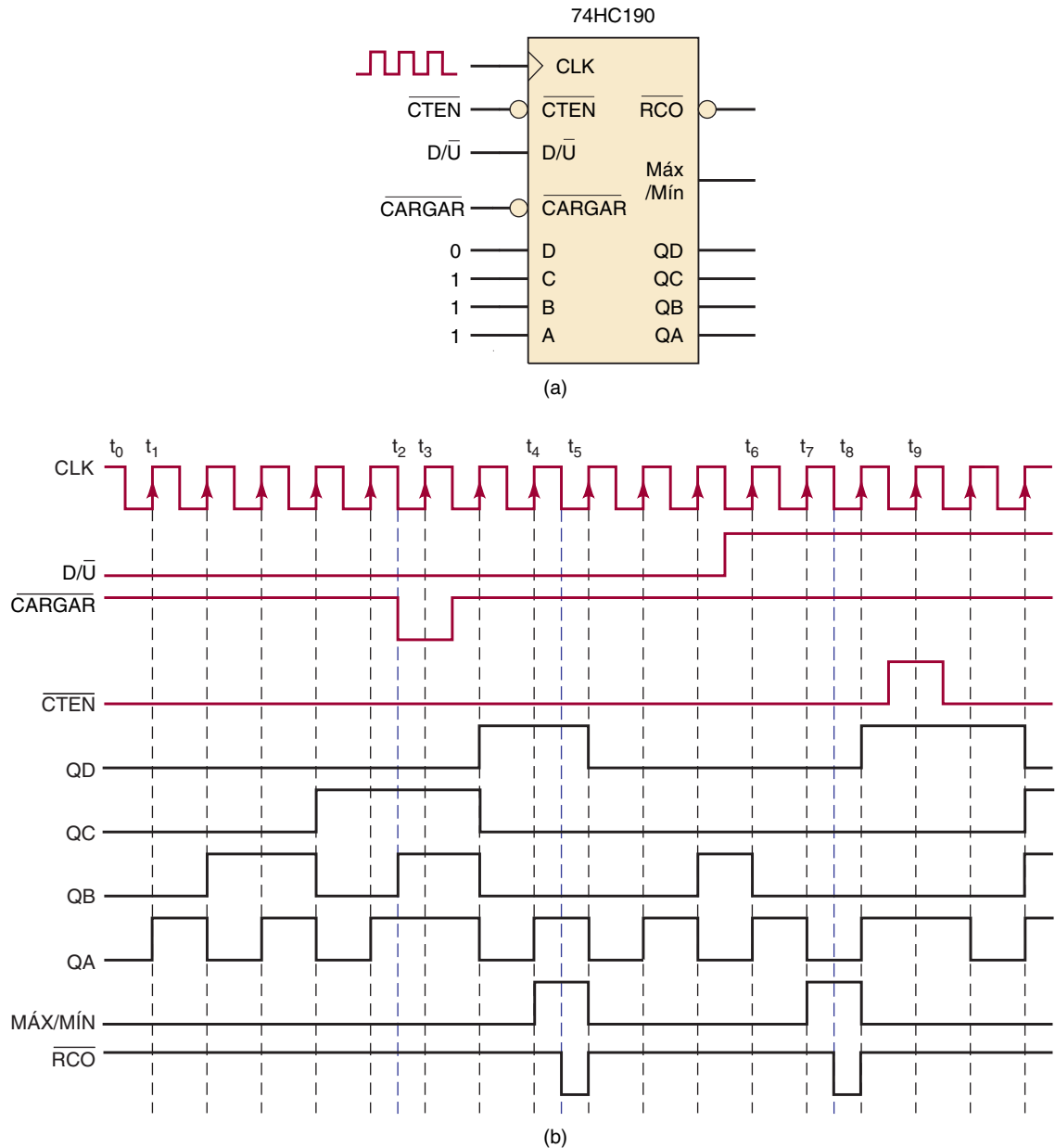


FIGURA 7-17 Ejemplo 7-12.

EJEMPLO 7-13

Compare la operación de dos contadores: uno con carga síncrona y el otro con carga asíncrona. Consulte la figura 7-18(a), en donde un 74ALS163 y un 74ALS191 se han conectado en forma similar para contar hacia arriba en binario. Ambos circuitos se controlan mediante la misma señal de reloj y sus salidas QD y QC están conectadas a una compuerta NAND para controlar la entrada de control CARGAR respectiva. Suponga que al principio ambos contadores se encuentran en el estado 0000.

- Determine la forma de onda de salida para cada contador.
- ¿Cuál es la secuencia de conteo de reciclaje y el módulo para cada contador?
- ¿Por qué tienen distintas secuencias de conteo?

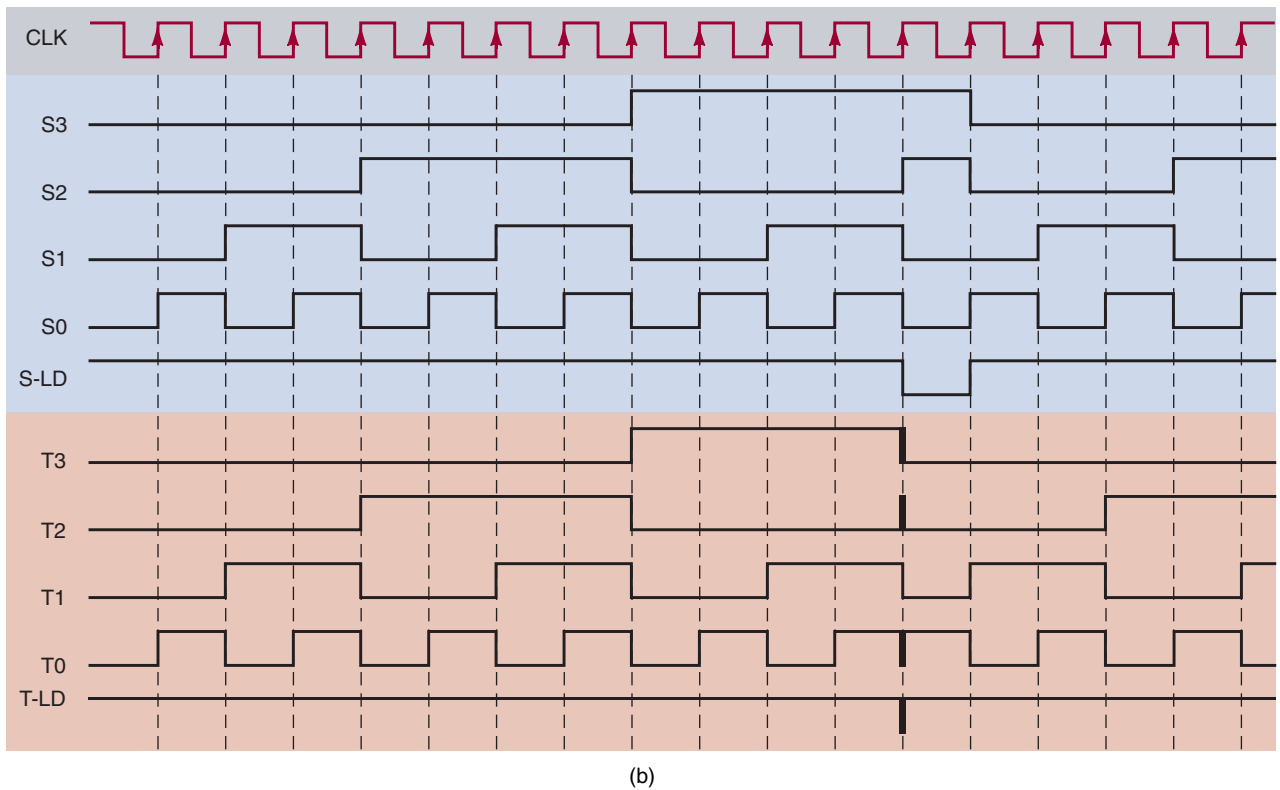
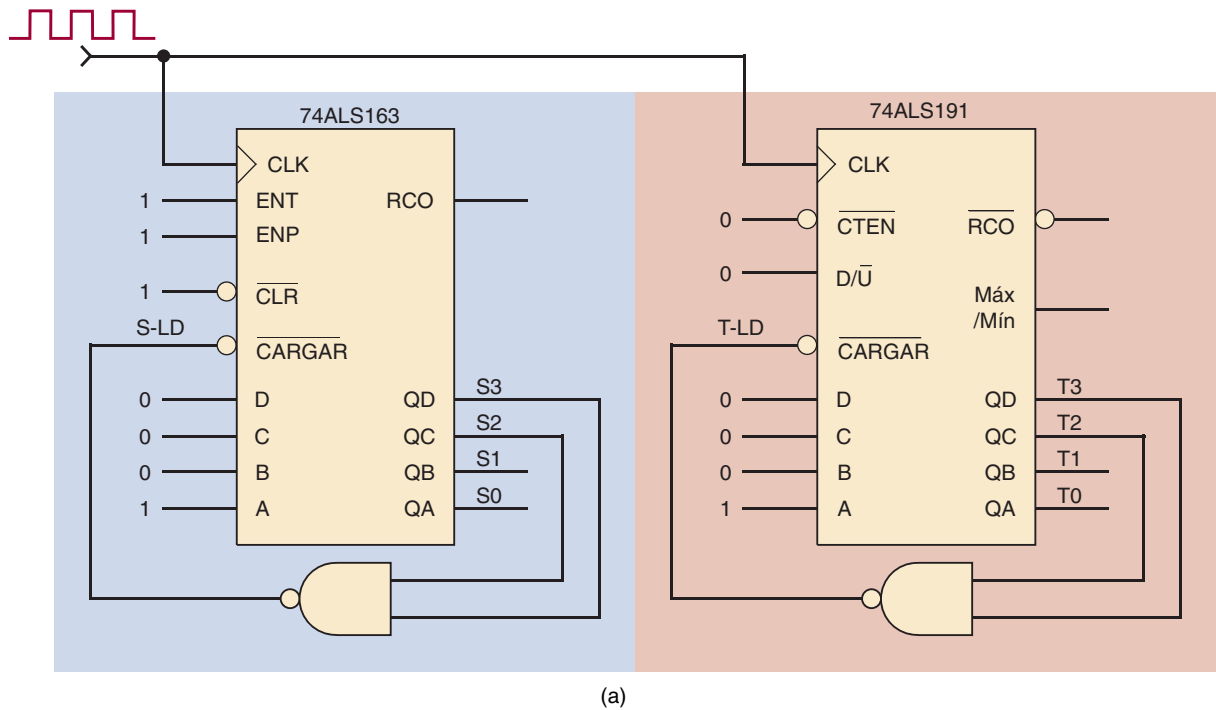


FIGURA 7-18 Ejemplo 7-13.

Solución

- (a) Comenzando en el estado 0000, cada contador contará hacia arriba hasta llegar al estado 1100 (12), como muestra la figura 7-18(b). La salida de cada compuerta NAND aplicará un nivel BAJO a la entrada $\overline{\text{CARGAR}}$ respectiva en ese tiempo. El 74ALS163 tiene una entrada $\overline{\text{CARGAR}}$ síncrona, por lo que esperará

hasta la siguiente PGT en CLK para cargar los datos de entrada 0001 en el contador. El 74ALS191 tiene una entrada $\overline{\text{CARGAR}}$ asíncrona, por lo que cargará de inmediato los datos de entrada 0001 en el contador. Esto convertirá al estado 1100 en temporal o transiente para el 74ALS191. El estado transiente producirá algunos picos o deformaciones en algunas de las salidas del contador, debido a la rápida acción de alternar hacia atrás y hacia delante.

- (b) El circuito 74ALS163 tiene una secuencia de conteo de reciclaje de 0001 hasta 1100 y es un contador MOD-12. El circuito 74ALS191 tiene una secuencia de conteo de reciclaje de 0001 hasta 1011 y es un contador MOD-11. No se incluyen los estados transientes al determinar el módulo para un contador.
- (c) Los circuitos tipo contador tienen distintas secuencias de conteo debido a que uno de ellos tiene una carga síncrona y el otro tiene una carga asíncrona.

Arreglo de múltiples etapas

Muchos contadores de CI estándar se han diseñado para facilitar la conexión de varios circuitos integrados en conjunto para crear circuitos con un rango de conteo más alto. Todos los circuitos tipo contador que se presentan en esta sección pueden conectarse en un arreglo de **múltiples etapas** o **en cascada**. En la figura 7-19, dos circuitos 74ALS163 se conectan en un arreglo de contadores de dos etapas que produce una secuencia binaria con reciclaje de 0 a 255, para un módulo máximo de 256. Al aplicar un nivel BAJO en la entrada CLR se borrarán en forma síncrona ambas etapas de contador, y al aplicar un nivel BAJO a LD el contador de ocho bits se preestablecerá en forma síncrona al valor binario en las entradas D7, D6, D5, D4, D3, D2, D1, D0 (D0 = LSB). El bloque a la izquierda (etapa 1) es la etapa de menor orden y proporciona las salidas menos significativas del contador: Q3, Q2, Q1, Q0 (en donde Q0 = LSB). La etapa 2 a la derecha proporciona las salidas más significativas del contador: Q7, Q6, Q5, Q4 (en donde Q7 = MSB).

EN, la entrada de habilitación para el contador de ocho bits, se conecta a la entrada ENT en la etapa 1. Observe que debemos usar la entrada ENT y no ENP, ya que sólo ENT controla la salida RCO. El uso de ENT y RCO facilita en forma considerable la conexión en cascada. Ambos bloques del contador se coordinan en forma síncrona, pero el bloque a la derecha (etapa 2) está deshabilitado hasta que el nibble de salida menos significativo haya llegado a su estado terminal, el cual se indica mediante la salida TC1. Cuando Q3, Q2, Q1, Q0 llegan a 1111 y si EN está en ALTO, entonces TC1 producirá un nivel ALTO. Esto permitirá que ambas etapas del contador cuenten uno hacia arriba con la siguiente PGT en el reloj. La etapa 1 se

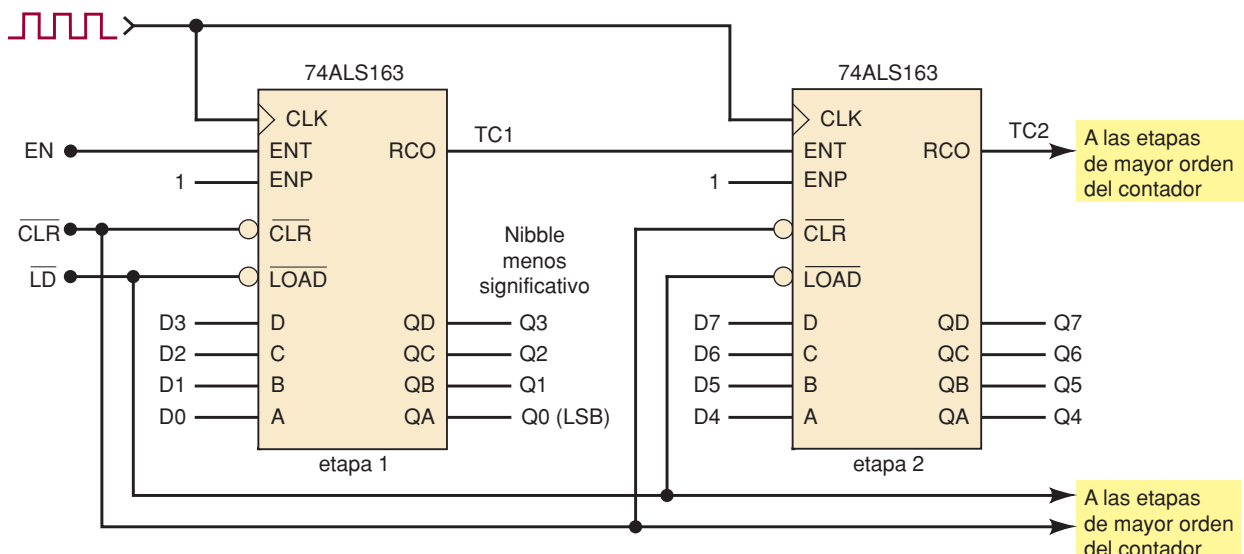


FIGURA 7-19 Dos chips 74ALS163 conectados en un arreglo de dos etapas, para extender el rango máximo de conteo.

reciclará de vuelta a 0000 y la etapa 2 se incrementará a partir de su estado de salida anterior. TC1 regresará a un nivel BAJO ya que la etapa 1 no se encuentra más en su estado terminal. Con los subsiguientes ciclos de reloj, la etapa 1 continuará contando hacia arriba si $EN = 1$, hasta que llegue de nuevo a 1111 y se repite el proceso. Cuando el contador de ocho bits llega a 11111111, se reciclará de vuelta a 00000000 en el siguiente pulso de reloj.

De la misma forma, pueden conectarse en cascada circuitos contador 74ALS163 adicionales. TC2 se conectaría al control ENT en el siguiente circuito, y así en lo sucesivo. TC2 estará en ALTO cuando Q7, Q5, Q4 sea igual a 1111 y TC1 esté en ALTO, lo cual a su vez significa que Q3, Q2, Q1, Q0 son igual a 1111 y EN está en ALTO. Esta técnica en cascada funciona para todos los circuitos (familia TTL o CMOS) en esta serie, incluso para los contadores BCD. La serie 74ALS190-191 (o 74HC190-191) también puede conectarse en cascada de manera similar si se utilizan las terminales $\overline{CTEN}$ y $\overline{RCO}$ activas en BAJO. Un contador de múltiples etapas que utilice circuitos 74ALS190-191 conectado de esta forma podrá contar hacia arriba o hacia abajo.

PREGUNTAS DE REPASO

1. Describa la función de las entradas $\overline{CARGAR}$ y D, C, B, A.
2. Describa la función de la entrada $\overline{CLR}$.
3. *Verdadero o falso:* el 74HC161 no puede preestablecerse mientras $\overline{CLR}$ está activa.
4. ¿Qué niveles lógicos deben estar presentes en las entradas de control para que el 74ALS162 pueda contar pulsos que aparezcan en CLK?
5. ¿Qué niveles lógicos deben estar presentes en las entradas de control para que el 74HC190 pueda contar hacia abajo con pulsos que aparezcan en CLK?
6. ¿Cuál sería el rango máximo de conteo para un contador de cuatro etapas formado por CIs 74HC163? ¿Cuál es el rango máximo de conteo para CIs 74ALS190?

7-8 DECODIFICACIÓN DE UN CONTADOR

Los contadores digitales se utilizan con frecuencia en aplicaciones en las que el conteo representado por los estados de los FFs debe determinarse o visualizarse de alguna forma. Uno de los medios más simples para visualizar el contenido de un contador implica sólo conectar la salida de cada FF a un pequeño LED indicador [vea la figura 7-7(b)]. De esta manera, los estados de los FFs se representan en forma visual mediante los LEDs (encendido = 1, apagado = 0) el conteo se puede determinar mentalmente, **decodificando** los estados binarios de los LEDs. Por ejemplo, suponga que se utiliza este método para un contador BCD y que los estados de los LEDs son apagado-encendido-encendido-apagado, respectivamente. Esto representaría el valor 0110, el cual decodificaríamos mentalmente como el 6 decimal. Otras combinaciones de los estados de los LEDs representarían los demás posibles conteos.

El método del LED indicador se vuelve inconveniente a medida que aumenta el tamaño (número de bits) del contador, ya que es demasiado difícil decodificar mentalmente los resultados visualizados. Por esta razón es preferible desarrollar un medio para decodificar por medios *electrónicos* el contenido de un contador y visualizar los resultados en un formato que se reconozca de inmediato y no requiera operaciones mentales.

Una razón aún más importante para la decodificación electrónica de un contador se debe a las diversas aplicaciones en las que se utilizan los contadores para controlar la sincronización o la secuencia de las operaciones en forma *automática*, sin intervención humana. Por ejemplo, la operación de cierto sistema tal vez tenga que iniciarse cuando un contador llegue al estado 101100 (conteo de 44_{10}). Puede utilizarse un circuito lógico para decodificar o detectar el momento en el que se presente este conteo específico y después iniciar la operación. En un sistema digi-

tal, muchas operaciones tienen que controlarse de esta forma. Es evidente que no sería conveniente la intervención humana en este proceso, excepto en sistemas en extremo lentos.

Decodificación activa en ALTO

Un contador MOD-X tiene X estados distintos; cada estado es un patrón específico de 0s y 1s que se almacenan en los FFs del contador. Una red decodificadora es un circuito lógico que genera X salidas distintas, cada una detecta (decodifica) la presencia de un estado específico del contador. Las salidas del decodificador pueden diseñarse para producir ya sea un nivel ALTO o BAJO cuando ocurre la detección. Un decodificador activo en ALTO produce salidas en ALTO para indicar la detección. La figura 7-20 muestra la lógica de decodificación activa en ALTO completa

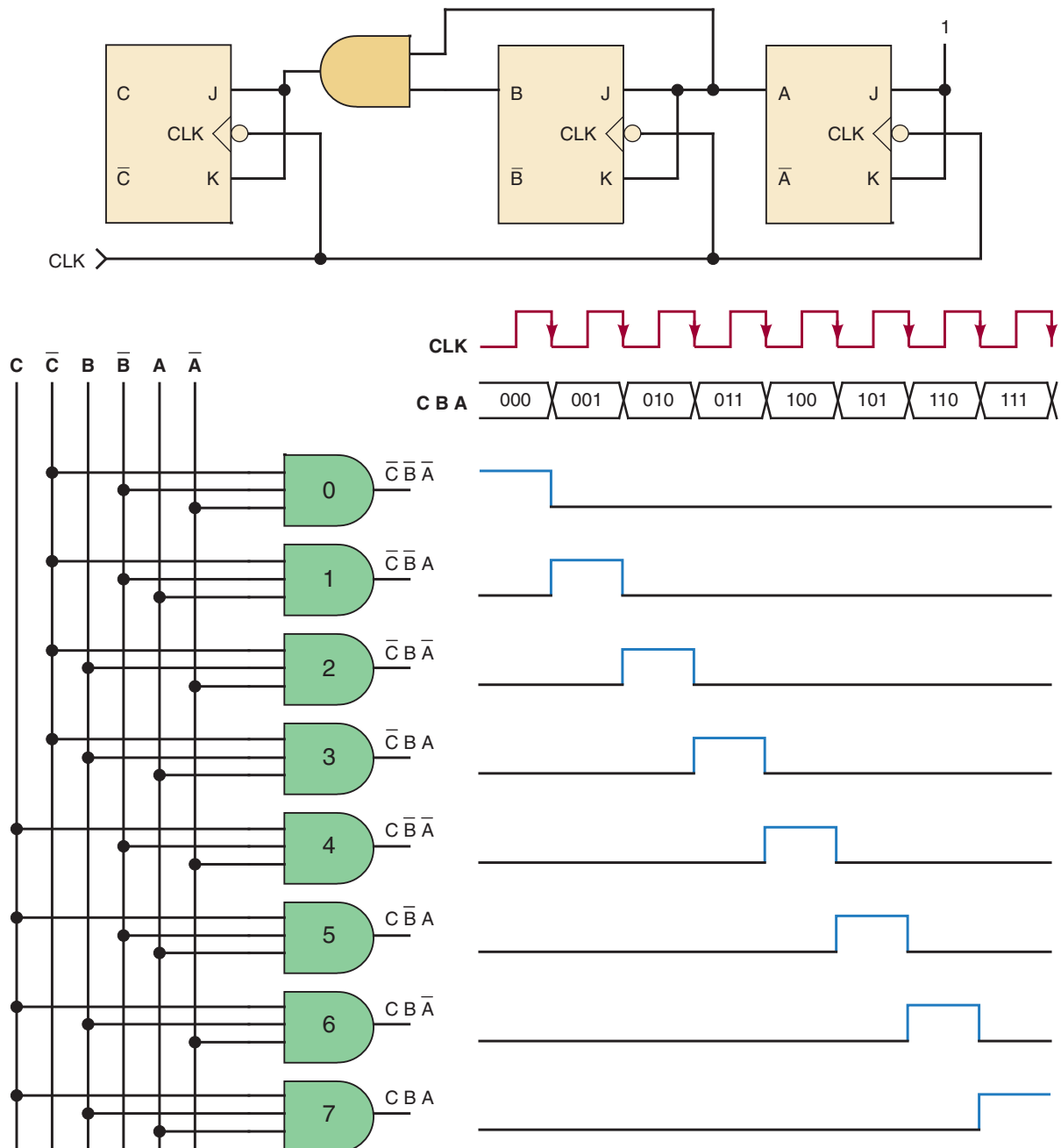


FIGURA 7-20 Uso de compuertas AND para decodificar un contador MOD-8.

para un contador MOD-8. El decodificador consiste de ocho compuertas AND de tres entradas; cada una estas compuertas produce una salida en ALTO para un estado específico del contador.

Por ejemplo, la compuerta AND 0 tiene como entradas las salidas $\bar{C}$, $\bar{B}$ y $\bar{A}$ de los FFs. Por ende, su salida estará en BAJO en todo momento, *excepto* cuando $A = B = C = 0$; esto es, en el conteo de 000 (cero). De manera similar, la compuerta AND 5 tiene como entradas las salidas C , B y A de los FFs, por lo que su salida cambiará a ALTO sólo cuando $C = 1$, $B = 0$ y $A = 1$; esto es, en el conteo de 101 (5 decimal). El resto de las compuertas AND funciona de la misma forma para los demás posibles conteos. En cualquier momento dado, sólo la salida de una compuerta AND está en ALTO: la que está decodificando para el conteo específico presente en el contador. Las formas de onda de la figura 7-20 muestran esto con claridad.

Las salidas de las ocho compuertas AND pueden utilizarse para controlar ocho LEDs, los cuales representan los números decimales del 0 al 7. Sólo un LED estará encendido en cualquier momento dado, indicando el conteo apropiado.

El decodificador de compuerta AND puede extenderse para funcionar con contadores con cualquier número de estados. Esto se ilustra mediante el siguiente ejemplo.

EJEMPLO 7-14

¿Cuántas compuertas AND se requieren para decodificar por completo todos los estados de un contador binario MOD-32? ¿Cuáles son las entradas para la compuerta que decodifica el conteo de 21?

Solución

Un contador MOD-32 tiene 32 posibles estados. Se necesita una compuerta AND para decodificar para cada estado; por lo tanto, el decodificador requiere 32 compuertas AND. Como $32 = 2^5$, el contador contiene cinco FFs. Por ende, cada compuerta tendrá cinco entradas, una de cada FF. Para decodificar el conteo de 21 (esto es, 10101_2) se requieren las entradas de E , D , C , B y A en la compuerta AND, en donde E es el flip-flop del MSB.

Decodificación activa en BAJO

Si se utilizan compuertas NAND en vez de compuertas AND, las salidas del decodificador producen una señal que está, por lo general, en nivel ALTO, la cual cambia a BAJO sólo cuando ocurre el número que se está decodificando. Se utilizan ambos tipos de decodificadores, dependiendo del tipo de circuitos que se van a controlar mediante las salidas del decodificador.

EJEMPLO 7-15

La figura 7-21 muestra una situación común en la que se utiliza un contador para generar una forma de onda de control, la cual podría usarse para controlar dispositivos tales como un motor, una válvula solenoide o un calentador. El contador MOD-16 cicla y recicla a través de su secuencia de conteo. Cada vez que llega al conteo de 8 (1000), la compuerta NAND superior produce una salida en BAJO que establece el flip-flop X para que quede en el estado 1. El flip-flop X permanece en ALTO hasta que el contador llega al conteo de 14 (1110), tiempo en el cual la compuerta NAND inferior la decodifica y produce una salida en BAJO para borrar X y que quede en el estado 0. En consecuencia, la salida X está en ALTO entre los conteos de 8 y 14 para cada ciclo del contador.

Decodificación de contadores BCD

Un contador BCD tiene 10 estados que pueden decodificarse mediante el uso de las técnicas antes descritas. Los decodificadores BCD proporcionan 10 salidas que corresponden a los dígitos decimales del 0 al 9 y se representan mediante los esta-

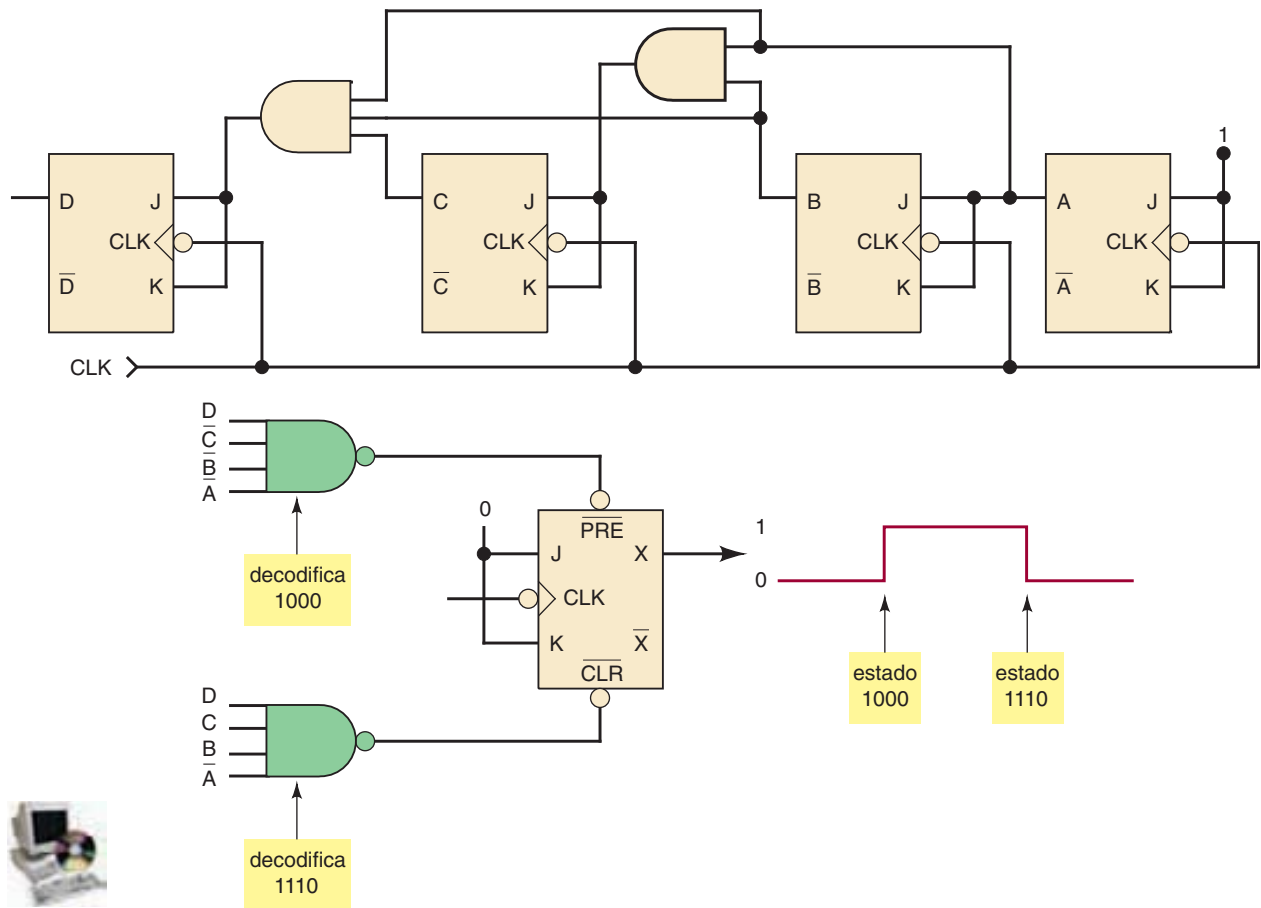


FIGURA 7-21 Ejemplo 7-15.

dos de los FFs del contador. Estas 10 salidas pueden utilizarse para controlar 10 LEDs indicadores individuales para un despliegue visual. Es más común utilizar un solo dispositivo de visualización para mostrar los números decimales del 0 al 9 en lugar de 10 LEDs separados. Una clase de visualizadores decimales contiene siete pequeños segmentos formados de un material (por lo general, LEDs o pantallas de cristal líquido) que emiten luz o reflejan la luz ambiental. Las salidas del decodificador BCD controlan qué segmentos se iluminan para producir un patrón que represente uno de los dígitos decimales.

En el capítulo 9 veremos con más detalle todos estos tipos de decodificadores y pantallas. No obstante y como los contadores BCD y sus decodificadores y exhibidores asociados son muy comunes, utilizaremos la unidad decodificadora/pantalla (vea la figura 7-22) para representar los circuitos completos que se utilizan para mostrar en forma visual el contenido de un contador BCD como un dígito decimal.

FIGURA 7-22 Por lo general, los contadores BCD muestran su conteo en un solo dispositivo de visualización.

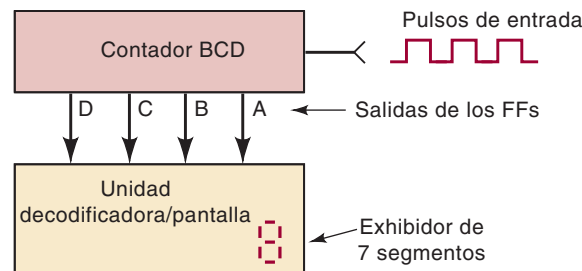


TABLA 7-1

Estado PRESENTE			Entradas de control						Estado SIGUIENTE		
C	B	A	J_C	K_C	J_B	K_B	J_A	K_A	C	B	A
0	0	0	0	0	0	0	1	1	0	0	1
0	0	1	0	0	1	1	1	1	0	1	0
0	1	0	0	0	0	0	1	1	0	1	1
0	1	1	1	0	1	1	1	1	1	0	0
1	0	0	0	1	0	0	0	0	0	0	0
1	0	1	0	1	1	1	0	0	0	1	1
1	1	0	0	1	0	0	0	0	0	1	0
1	1	1	1	1	1	1	0	0	0	0	1

introducido esta información en la primera línea de la tabla de estado PRESENTE / estado SIGUIENTE que se muestra en la tabla 7-1. A continuación podemos utilizar el estado 001 como nuestro estado PRESENTE. Si analizamos las expresiones de control con esta nueva combinación se producirá ahora la condición $J_C K_C = 0\ 0$, $J_B K_B = 1\ 1$ y $J_A K_A = 1\ 1$, lo cual nos da un comando de retención para el FF C y comandos de conmutación para los FFs B y A. Esto producirá el estado SIGUIENTE de 010 para CBA, que hemos listado en la segunda línea de la tabla 7-1. Si continuamos con este proceso se producirá una secuencia de conteo reciclable de 000, 001, 010, 011, 100, 000. Esta sería una secuencia de conteo MOD-5. Podemos predecir los estados SIGUIENTE para las tres posibles combinaciones de estado restantes de la misma forma. Al hacerlo podremos determinar si el diseño del contador es *autocorregible*. Un **contador autocorregible** es aquel en el que, por lo general, los estados sin utilizar regresarán de alguna forma a la secuencia de conteo normal. Si cualquiera de estos estados no utilizados no puede regresar a la secuencia normal, se dice que el contador no es autocorregible. Hemos introducido en la tabla 7-1 nuestras predicciones del estado SIGUIENTE para todos los posibles estados. La información resaltada indica que el diseño de este contador es autocorregible. La figura 7-24 muestra completos el diagrama de transición de estados y el diagrama de tiempos para este contador.

De igual forma podemos analizar la operación de los circuitos tipo contador que utilizan flip-flops D para almacenar el estado presente del contador. Por lo

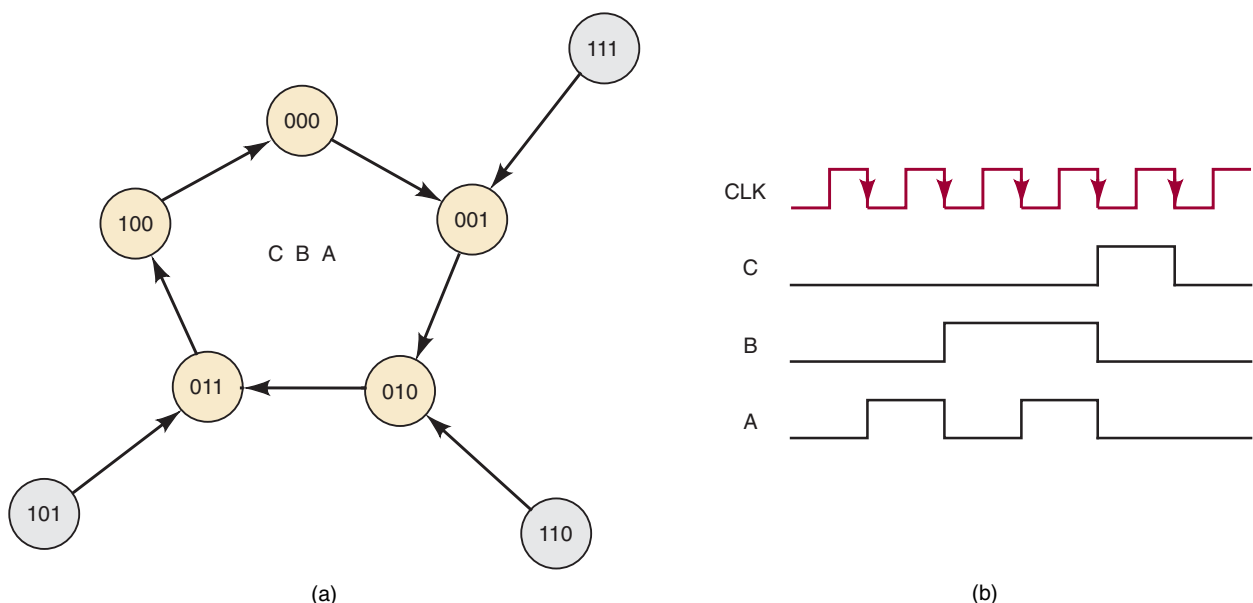
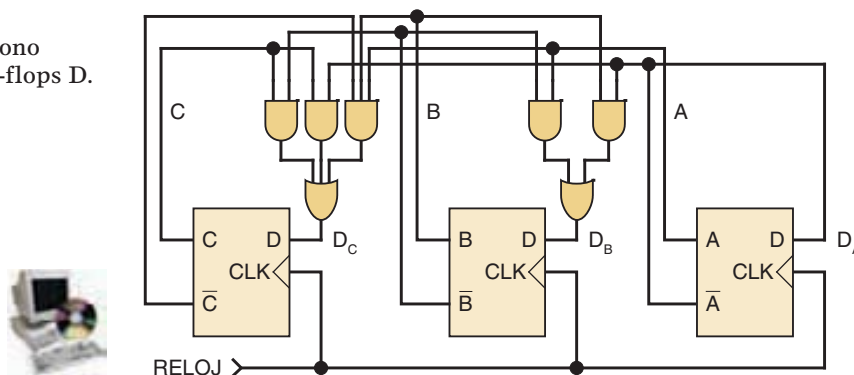


FIGURA 7-24 (a) Diagrama de transición de estados y (b) diagrama de sincronización para el contador asíncrono de la figura 7-23.

FIGURA 7-25

Contador síncrono
que utiliza flip-flops D.



general, los circuitos de control para un flip-flop tipo *D* serán más complejos que para un contador tipo JK equivalente que produzca la misma secuencia de conteo, pero también tendremos que controlar la mitad del número de entradas síncronas. La mayoría de los PLDs utilizan flip-flops *D* para sus elementos de memoria, por lo que el análisis de este tipo de circuito contador nos proporcionará algo de detalle sobre la forma en que se programan los contadores dentro de un PLD.

La figura 7-25 muestra un contador síncrono diseñado con flip-flops *D*. El primer paso es escribir las expresiones lógicas para las entradas *D*:

$$\begin{aligned} D_C &= C\bar{B} + C\bar{A} + \bar{C}BA \\ D_B &= \bar{B}A + B\bar{A} \\ D_A &= \bar{A} \end{aligned}$$

Después determinaremos la tabla del estado PRESENTE / estado SIGUIENTE para el circuito contador; para ello debemos suponer un estado y aplicar ese conjunto de valores de bits a las expresiones de entrada antes descritas. Si elegimos CBA = 000 para el estado inicial del contador obtendremos como resultado $D_C = 0$, $D_B = 0$ y $D_A = 1$. Con una PGT en RELOJ los flip-flops “cargarán” el valor 001, que se convertirá en el estado SIGUIENTE del contador. Si utilizamos 001 como el estado PRESENTE se producirán entradas de $D_C = 0$, $D_B = 1$ y $D_A = 0$, de manera que 010 será el estado SIGUIENTE, y así en lo sucesivo. La tabla del estado PRESENTE / estado SIGUIENTE que se muestra en la tabla 7-2 indica que este circuito es un contador binario MOD-8 reciclable. Si aplicamos un poco de álgebra booleana a las expresiones de entrada, podremos ver que en realidad hay un patrón de circuito bastante simple para crear contadores binarios a partir de flip-flops *D*:

$$\begin{aligned} D_C &= C\bar{B} + C\bar{A} + \bar{C}BA = C(\bar{B} + \bar{A}) + \bar{C}BA \\ &= C\bar{B}\bar{A} + \bar{C}(BA) = C \oplus (AB) \\ D_B &= \bar{B}A + B\bar{A} = B \oplus A \\ D_A &= \bar{A} \end{aligned}$$

TABLA 7-2

Estado PRESENTE			Entradas de control			Estado SIGUIENTE		
<i>C</i>	<i>B</i>	<i>A</i>	D_C	D_B	D_A	<i>C</i>	<i>B</i>	<i>A</i>
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	0
0	1	0	0	1	1	0	1	1
0	1	1	1	0	0	1	0	0
1	0	0	1	0	1	1	0	1
1	0	1	1	1	0	1	1	0
1	1	0	1	1	1	1	1	1
1	1	1	0	0	0	0	0	0

Es importante observar que los recursos de compuertas para la mayoría de los PLDs en realidad consisten de conjuntos de arreglos de circuitos AND-OR y que la expresión lógica SOP (Suma de Productos) describe con más precisión la implementación interna del circuito. No obstante, podemos ver que las expresiones se han simplificado de manera considerable mediante el uso de la función XOR. Esto nos lleva a predecir de manera correcta que para crear un contador binario MOD-16 con flip-flops D necesitaríamos un cuarto FF con:

$$D_D = D \oplus (A B C)$$

PREGUNTAS DE REPASO

1. ¿Por qué es conveniente evitar tener controles asíncronos en los contadores?
2. ¿Qué herramienta es útil para el análisis de los contadores síncronos?
3. ¿Qué es lo que determina la secuencia de conteo para un circuito tipo contador?
4. ¿Qué característica del contador se describe diciendo que es autocorregible?

7-10 DISEÑO DE UN CONTADOR SÍNCRONO*

Hay muchos arreglos de contador en forma de CIs: asíncronos, síncronos y combinación de asíncrono/síncrono. La mayoría cuenta con una secuencia binaria normal o en BCD, aunque sus secuencias de conteo pueden alterarse en cierta forma mediante los métodos para borrar o cargar que demostramos para las series de CIs 74ALS160-163 y 74ALS190-191. No obstante, hay situaciones en las que se requiere un contador personalizado que siga una secuencia que no sea un patrón de conteo binario regular, por ejemplo: 000, 010, 101, 001, 110, 000

Existen varios métodos para diseñar contadores que siguen secuencias arbitrarias. Vamos a presentar los detalles para un método común que utiliza flip-flops J-K en una configuración de contador síncrono. El mismo método puede utilizarse en diseños con flip-flops D . La técnica es uno de varios procedimientos de diseño que forman parte de un área del diseño de circuitos digitales, conocida como **diseño de circuitos secuenciales**, que, por lo general, es parte de un curso avanzado.

Idea básica

En los contadores síncronos todos los FFs se sincronizan al mismo tiempo. Antes de cada pulso de reloj, las entradas J y K de cada FF en el contador deben estar en el nivel correcto para asegurar que el FF cambie al estado correcto. Por ejemplo, considere la situación en la que el estado 101 para el contador CBA debe ir seguido del estado 011. Cuando ocurre el siguiente pulso de reloj, las entradas J y K de los FFs deben estar en los niveles correctos que hagan que el flip-flop C cambie de 1 a 0, que el flip-flop B cambie de 0 a 1 y el flip-flop A de 1 a 1 (es decir, no cambie).

En consecuencia, el proceso de diseñar un contador síncrono se convierte en el proceso de diseñar los circuitos lógicos que *decodifiquen* los diversos estados del contador para suministrar los niveles lógicos apropiados a cada entrada J y K en el tiempo correcto. Las entradas para estos circuitos decodificadores provendrán de las salidas de uno o más FFs. Para ilustrar esto, en el contador síncrono de la figura 7-5 la compuerta AND que alimenta las entradas J y K del flip-flop C decodifica los estados de los flip-flops A y B . De igual forma, la compuerta AND que alimenta las entradas J y K del flip-flop D decodifica los estados de A , B y C .

* Puede omitirse este tema sin afectar la continuidad del resto del libro.

Tabla de excitación J-K

Antes de comenzar el proceso de diseñar los circuitos decodificadores para cada entrada J y K , primero debemos repasar la operación del flip-flop J - K utilizando un enfoque distinto, conocido como *tabla de excitación* (tabla 7-3). La columna más a la izquierda en esta tabla lista cada posible transición de salida de un FF. Las columnas segunda y tercera listan el estado PRESENTE del FF (simbolizado como Q_n) y el estado SIGUIENTE (simbolizado como Q_{n+1}) para cada transición. Las últimas dos columnas listan los niveles de J y K requeridos para producir cada transición. Ahora examinemos cada caso.

TRANSICIÓN 0 → 0 El estado PRESENTE del FF es 0 y deberá permanecer en 0 cuando se aplique un pulso de reloj. De lo que conocemos acerca de la forma en que funciona un flip-flop J - K , esto puede ocurrir cuando $J = K = 0$ (condición sin cambio) o cuando $J = 0$ y $K = 1$ (condición de limpiar). Por ende, J debe estar en 0 pero K puede estar en cualquier nivel. La tabla indica esto con un “0” bajo J y una “X” bajo K . Recuerde que “X” representa la condición de “no importa”.

TRANSICIÓN 0 → 1 El estado PRESENTE es 0 y va a cambiar a 1, lo cual puede ocurrir cuando $J = 1$ y $K = 0$ (condición de establecer) o cuando $J = K = 1$ (condición de conmutación). En consecuencia, J debe ser 1 pero K puede estar en cualquier nivel para que ocurra la transición.

TRANSICIÓN 1 → 0 El estado PRESENTE es 1 y va a cambiar a 0, lo cual puede ocurrir cuando $J = 0$ y $K = 1$, o cuando $J = K = 1$. Por lo tanto, K debe ser 1 pero J puede estar en cualquier nivel.

TRANSICIÓN 1 → 1 El estado PRESENTE es 1 y va a permanecer en 1, lo cual puede ocurrir cuando $J = K = 0$ o cuando $J = 1$ y $K = 0$. Por ende, K debe ser 0 mientras que J puede estar en cualquier nivel.

El uso de esta **tabla de excitación J-K** (tabla 7-3) es una parte imprescindible del procedimiento de diseño de contadores síncronos.

TABLA 7-3 Tabla de excitación de un flip-flop J-K.

Transición en la salida del FF	Estado PRESENTE Q_n	Estado SIGUIENTE Q_{n+1}	J	K
0 → 0	0	0	0	x
0 → 1	0	1	1	x
1 → 0	1	0	x	1
1 → 1	1	1	x	0

Procedimiento de diseño

Ahora veremos todo un procedimiento de diseño de un contador síncrono completo. Aunque lo haremos para una secuencia de conteo específica, pueden seguirse los mismos pasos para cualquier secuencia deseada.

Paso 1. Determine el número deseado de bits (FFs) y la secuencia de conteo deseada.

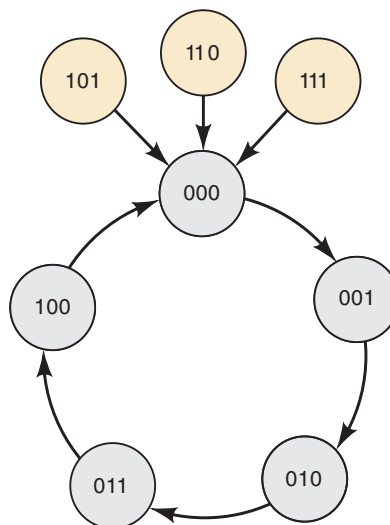
Para nuestro ejemplo, diseñaremos un contador de tres bits que pasa por la secuencia mostrada en la tabla 7-4. Observe que esta secuencia no incluye los estados 101, 110 y 111. Nos referiremos a éstos como *estados indeseables*.

Paso 2. Dibuje el diagrama de transición de estados que muestre *todos* los estados posibles, incluyendo aquellos que no forman parte de la secuencia de conteo deseada.

TABLA 7-4

C	B	A
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
0	0	0
0	0	1
etc.		

FIGURA 7-26 Diagrama de transición de estados para el ejemplo de diseño de un contador síncrono.



Para nuestro ejemplo el diagrama de transición de estados aparece como se muestra en la figura 7-26. Los estados 000 a 100 se conectan en la secuencia esperada. También hemos incluido un estado SIGUIENTE definido para cada uno de los estados indeseables. Esto se hizo en caso de que el contador llegue por accidente a uno de estos estados al momento del encendido o debido al ruido. El diseñador del circuito puede optar por hacer que cada uno de estos estados indeseables vaya a cualquier estado al momento en el que se aplique el siguiente pulso de reloj. De manera alternativa, el diseñador puede optar por no definir la acción del contador para los estados indeseables. En otras palabras, podríamos no preocuparnos por el estado SIGUIENTE para cualquier estado indeseable. Por lo general, el uso del método de diseño “no importa” antes mencionado producirá un diseño más simple, pero puede ser un problema potencial en la aplicación en donde se vaya a utilizar este contador. Para nuestro ejemplo de diseño optaremos por hacer que todos los estados indeseables se vayan al estado 000. Esto hará que nuestro diseño sea auto-correctible pero algo distinto del ejemplo del contador MOD-5 que analizamos en la sección 7-9.

Paso 3. Utilice el diagrama de transición de estados para preparar una tabla que liste *todos* los estados PRESENTES y sus estados SIGUIENTES.

Para nuestro ejemplo, la tabla 7-5 muestra la información. La porción de la izquierda lista *todos* los posibles estados, aún aquellos que no forman parte de la secuencia. Vamos a etiquetarlos como los estados PRESENTES. La porción de la derecha lista el estado SIGUIENTE para cada estado PRESENTE. Éstos se obtienen del diagrama de transición de estados de la figura 7-26. Por ejemplo, la línea 1 muestra que el estado

TABLA 7-5

	Estado PRESENTE			Estado SIGUIENTE		
	C	B	A	C	B	A
Línea 1	0	0	0	0	0	1
2	0	0	1	0	1	0
3	0	1	0	0	1	1
4	0	1	1	1	0	0
5	1	0	0	0	0	0
6	1	0	1	0	0	0
7	1	1	0	0	0	0
8	1	1	1	0	0	0

PRESENTE 000 tiene el estado SIGUIENTE 001, y la línea 5 muestra que el estado PRESENTE 100 tiene el estado SIGUIENTE 000. Las líneas 6, 7 y 8 muestran que los estados PRESENTES indeseables 101, 110 y 111 tienen todos el estado SIGUIENTE 000.

Paso 4. Añada una columna a esta tabla para cada entrada J y K . Para cada estado PRESENTE, indique los niveles requeridos en cada entrada J y K para poder producir la transición al estado SIGUIENTE.

Nuestro ejemplo de diseño utiliza tres FFs (C , B y A) y cada uno de ellos tiene una entrada J y K . Por lo tanto, debemos agregar seis nuevas columnas como se muestra en la tabla 7-6. A esta tabla completa se le conoce como la **tabla de excitación del circuito**. Las seis nuevas columnas son las entradas J y K de cada FF. Las entradas bajo cada J y K se obtienen de la tabla 7-3, la tabla de excitación del flip-flop J-K que desarrollamos antes. Demostraremos esto para varios de los casos y usted podrá verificar el resto.

Veamos la línea 1 en la tabla 7-6. El estado PRESENTE 000 va a cambiar al estado SIGUIENTE 001 cuando ocurra un pulso de reloj. Para esta transición de estado, el flip-flop C cambia de 0 a 0. De la tabla de excitación J-K podemos ver que J_C debe estar en 0 y K_C en "X" para que ocurra esta transición. El flip-flop B también cambia de 0 a 0, por lo que $J_B = 0$ y $K_B = X$. El flip-flop A cambia de 0 a 1. También de la tabla 7-3 podemos ver que $J_A = 1$ y $K_A = x$ para esta transición.

En la línea 4 de la tabla 7-6, el estado PRESENTE 011 tiene un estado SIGUIENTE de 100. Para esta transición de estado, el flip-flop C cambia de 0 a 1, para lo cual requiere que $J_C = 1$ y $K_C = x$. Los flip-flops A y B cambian de 1 a 0. La tabla de excitación J-K indica que estos dos FFs necesitan que $J = X$ y $K = 1$ para que esto ocurra.

Los niveles requeridos de J y K para las demás líneas de la tabla 7-6 pueden determinarse de la misma forma.

Paso 5. Diseñe los circuitos lógicos necesarios para generar los niveles requeridos en cada entrada J y K .

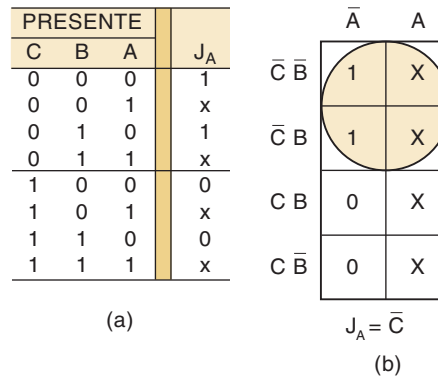
La tabla 7-6 (la tabla de excitación del circuito) lista seis entradas J , K : J_C , K_C , J_B , K_B , J_A y K_A . Debemos considerar cada una de éstas como una salida de su propio circuito lógico con entradas de los flip-flops C , B y A . Después debemos diseñar el circuito para cada una. Vamos a diseñar el circuito para J_A .

Para ello, necesitamos analizar los estados PRESENTES de C , B y A , y los niveles deseados en J_A para cada caso. Hemos extraído esta información de la tabla 7-6 y la presentamos en la figura 7-27(a). La tabla de verdad muestra los niveles deseados en J_A para cada estado PRESENTE. Desde luego que para algunos de los casos, J_A es

TABLA 7-6
Tabla de excitación del
circuito.

	Estado PRESENTE			Estado SIGUIENTE									
	C	B	A	C	B	A	J _C	K _C	J _B	K _B	J _A	K _A	
Línea 1	0	0	0	0	0	1	0	x	0	x	1	x	
2	0	0	1	0	1	0	0	x	1	x	x	1	
3	0	1	0	0	1	1	0	x	x	0	1	x	
4	0	1	1	1	0	0	1	x	x	1	x	1	
5	1	0	0	0	0	0	x	1	0	x	0	x	
6	1	0	1	0	0	0	x	1	0	x	x	1	
7	1	1	0	0	0	0	x	1	x	1	0	x	
8	1	1	1	0	0	0	x	1	x	1	x	1	

FIGURA 7-27 (a) Porción de la tabla de excitación del circuito que muestra a J_A para cada estado PRESENTE; (b) mapa K utilizado para obtener la expresión simplificada para J_A .



una condición de “no importa”. Para desarrollar el circuito lógico para J_A primero debemos determinar su expresión en términos de C , B y A . Para ello vamos a transferir la información de la tabla de verdad a un mapa de Karnaugh de tres variables y realizaremos la simplificación del mapa K, como se muestra en la figura 7-27(b).

Sólo hay dos 1s en este mapa K, y éstos pueden agruparse para obtener el término $\bar{A} \bar{C}$ pero si utilizamos las condiciones de “no importa” en $A \bar{B} \bar{C}$ y en $A \bar{B} C$ como 1s, podremos agrupar un cuádruple para obtener el término $\bar{C}$ más simple. Por ende, la expresión final es

$$J_A = \bar{C}$$

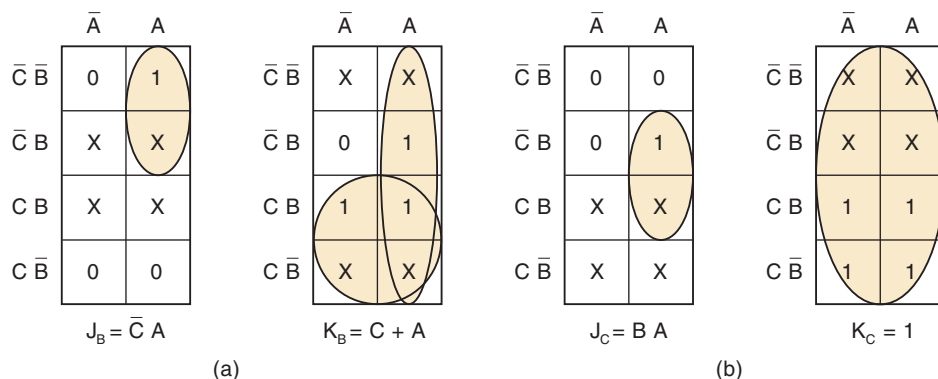
Ahora vamos a considerar a K_A . Podemos seguir los mismos pasos que para J_A . No obstante, un análisis de las entradas bajo K_A en la tabla de excitación del circuito nos muestra sólo valores de 1 y “no importa”. Si cambiamos todos los valores “no importa” por 1s, entonces K_A siempre será 1. Por lo tanto, la expresión final es

$$K_A = 1$$

De manera similar, podemos derivar las expresiones para J_C , K_C , J_B y K_B . Los mapas K para estas expresiones se proporcionan en la figura 7-28. Tal vez usted quiera confirmar que estén correctos, comparándolos con la tabla de excitación del circuito.

FIGURA 7-28

(a) Mapas K para los circuitos lógicos J_B y K_B ; (b) mapas K para los circuitos lógicos J_C y K_C .

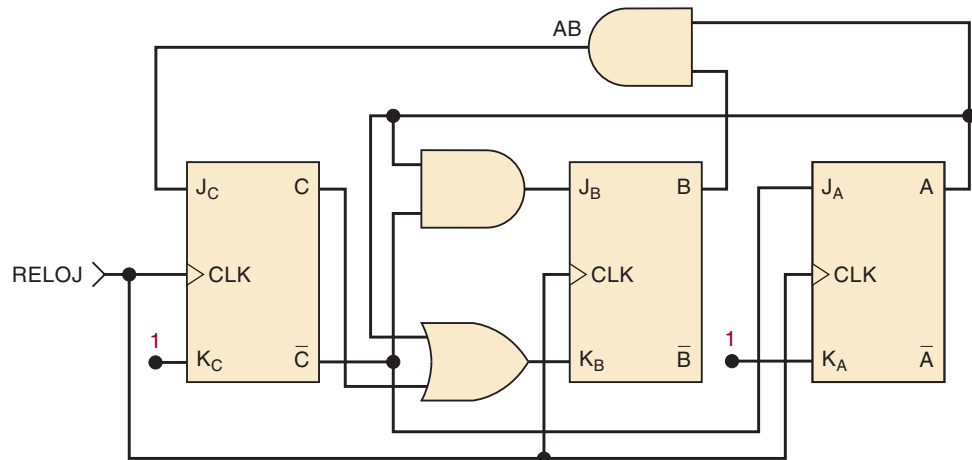


Paso 6. Implemente las expresiones finales.

Los circuitos lógicos para cada entrada J y K se implementan a partir de las expresiones que se obtienen del mapa K. El diseño completo del contador síncrono se implementa en la figura 7-29. Observe que todos los FFs están sincronizados en paralelo. Tal vez quiera verificar que la lógica para las entradas J y K concuerde con las figuras 7-27 y 7-28.

FIGURA 7-29

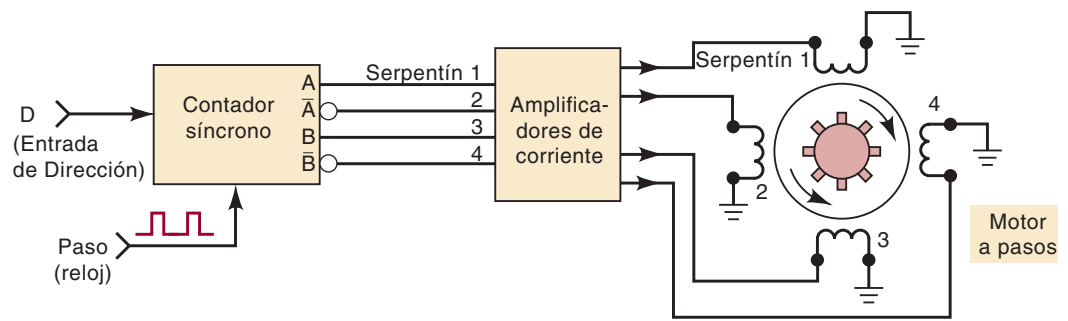
Implementación final del ejemplo del diseño del contador síncrono.



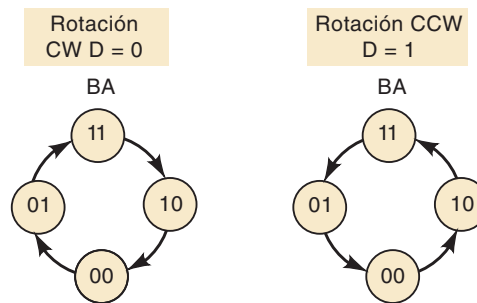
Control de un Motor a Pasos

Ahora aplicaremos este procedimiento de diseño a una situación práctica: el control de un *motor a pasos*. Este tipo de motor gira en pasos, por lo general, de 15° por paso, en vez de girar en un movimiento continuo. Los serpentines o devanados magnéticos dentro del motor deben energizarse y desenergizarse en una secuencia específica para poder producir esta acción de movimiento paso a paso. Por lo general, se utilizan señales digitales para controlar la corriente en cada uno de los serpentines del motor. Los motores a pasos se utilizan mucho en situaciones en las que se requiere un control preciso de posición, como en el posicionamiento de las cabezas de lectura/escritura en los discos magnéticos, en el control de las cabezas de impresión en impresoras, y en los robots.

La figura 7-30(a) es un diagrama de un motor a pasos ordinario con cuatro serpentines. Para que el motor gire en forma apropiada, los serpentines 1 y 2 deben



(a)



(b)

FIGURA 7-30 (a) Un contador síncrono suministra las salidas secuenciales apropiadas para controlar un motor a pasos; (b) diagramas de transición de estados para ambos estados de la entrada de Dirección, D .

estar siempre en estados opuestos; esto es, cuando el serpentín 1 está energizado, el serpentín 2 no lo está y viceversa. De igual forma, los serpentines 3 y 4 deben estar siempre en estados opuestos. Las salidas de un contador síncrono de dos bits se utilizan para controlar la corriente en los cuatro serpentines; A y $\bar{A}$ controlan los serpentines 1 y 2; B y $\bar{B}$ controlan los serpentines 3 y 4. Los amplificadores de corriente son necesarios ya que las salidas de los FFs no pueden suministrar la cantidad de corriente que requieren los serpentines.

Como este motor a pasos puede girar ya sea en sentido a favor de las manecillas del reloj (CW) o en contra de las manecillas del reloj (CCW), tenemos una entrada de Dirección (D) que se utiliza para controlar la dirección de rotación. Los diagramas de estado en la figura 7-30(b) muestran los dos casos. Para que ocurra la rotación CW debemos tener $D = 0$, y el estado del contador BA debe seguir la secuencia 11, 10, 00, 01, 11, 10,..., y así en lo sucesivo, a medida que se sincroniza mediante la señal de entrada Paso. Para la rotación CCW, $D = 1$ y el contador debe seguir la secuencia 11, 01, 00, 10, 11, 01,..., y así en lo sucesivo.

TABLA 7-7

Estado PRESENTE			Estado SIGUIENTE		Entradas de control			
D	B	A	B	A	J _B	K _B	J _A	K _A
0	0	0	0	1	0	x	1	x
0	0	1	1	1	1	x	x	0
0	1	0	0	0	x	1	0	x
0	1	1	1	0	x	0	x	1
1	0	0	1	0	1	x	0	x
1	0	1	0	0	0	x	x	1
1	1	0	1	1	x	0	1	x
1	1	1	0	1	x	1	x	0

FIGURA 7-31 (a) Mapas K para J_B y K_B ; (b) mapas K para J_A y K_A .

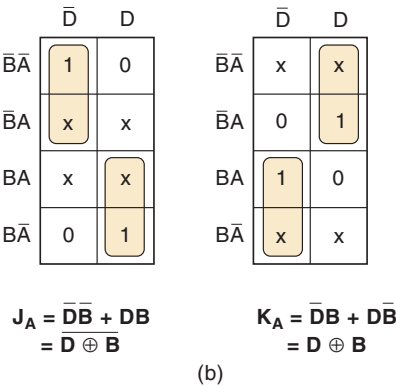
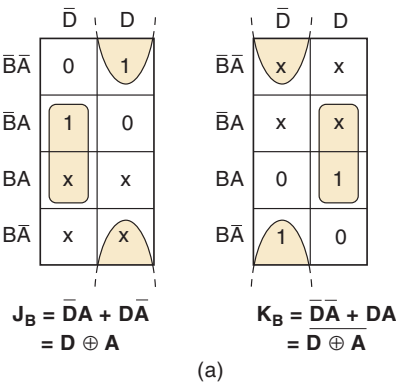
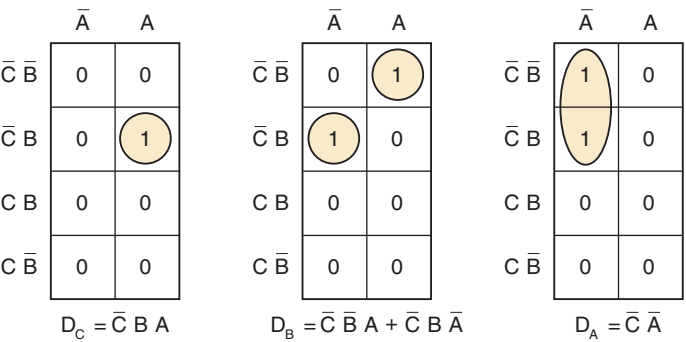


TABLA 7-8

Estado PRESENTE			Estado SIGUIENTE			Entradas de control		
C	B	A	C	B	A	D _C	D _B	D _A
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	0
0	1	0	0	1	1	0	1	1
0	1	1	1	0	0	1	0	0
1	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0

los mapas K y las expresiones lógicas simplificadas. Por último, para el paso 6 puede implementarse el contador con el circuito que se muestra en la figura 7-34.

FIGURA 7-33 Los mapas K y las expresiones lógicas simplificadas para el diseño de contadores MOD-5 con flip-flops.



7-11 CONTADORES BÁSICOS MEDIANTE EL USO DE HDL

En el capítulo 5 estudiamos los flip-flops y los métodos que se utilizan con los HDLs para representar los circuitos con flip-flops. La última sección en el capítulo 5 ilustró cómo conectar componentes de FF de una forma muy parecida a como se conectan los circuitos integrados entre sí. Al conectar la salida Q de un FF con la entrada de reloj del siguiente FF descubrimos que podía crearse un circuito. Al proceso de utilizar un HDL para describir las conexiones de los componentes se le conoce como el nivel estructural de abstracción. Es obvio que sería muy tedioso construir un circuito complicado mediante el uso de los métodos estructurales; además sería muy difícil de leer y de interpretar. En esta sección ampliaremos nuestro uso de HDL para describir circuitos mediante el uso de métodos que se consideran niveles más altos de abstracción. Este término suena intimidante, pero sólo significa que hay muchas formas más concisas y sensibles de describir lo que queremos que haga un contador, sin necesidad de preocuparnos por los detalles acerca de cómo cablear circuitos de flip-flops para hacerlo.

Aún es vital que comprendamos los principios fundamentales de la operación de los flip-flops, en comparación con las compuertas lógicas combinacionales. Como podrá recordar, los flip-flops tienen las siguientes características únicas. Por lo general, la salida se actualiza de acuerdo con la condición de las entradas de control síncronas cuando se produce el *flanco activo* del reloj, lo cual significa que hay un estado lógico en la salida Q antes del flanco del reloj (estado PRESENTE) y es posible que haya un estado distinto en la salida Q después del flanco del reloj (estado SIGUIENTE). Un flip-flop “recuerda” o retiene su estado entre los pulsos de reloj, sin importar los cambios en las entradas de control síncronas (por ejemplo, J y K).

Los circuitos tipo contador que utilizan HDL se basan en esta comprensión básica de un circuito que pasa a través de una secuencia de estados en respuesta al evento de un flanco de reloj. Los contadores de rizo proporcionan un circuito fácil de analizar y de comprender. También son mucho menos complicados de construir mediante el uso de flip-flops y compuertas lógicas, en comparación con sus contrapartes síncronos. El problema con los contadores de rizo es la combinación del retraso de tiempo y los estados temporales espurios que ocurren cuando el contador cambia de estado. Cuando avancemos al siguiente nivel de abstracción y planeemos usar PLDs para implementar nuestro diseño, ya no nos enfocaremos en las cuestiones relacionadas con el cableado, sino en describir la operación del circuito en forma concisa. En consecuencia, los métodos que utilizamos para describir circuitos tipo contador mediante el uso de HDL utilizan en su mayor parte técnicas síncronas, en las que todos los flip-flops se actualizan simultáneamente, en respuesta al mismo evento de reloj. Todos los bits en una secuencia de conteo pasan de su estado PRESENTE a su estado SIGUIENTE prescrito simultáneamente, previniendo, por lo tanto, cualquier estado espurio inmediato.

Métodos de descripción de transición de estados

El siguiente método para describir circuitos que necesitamos examinar utiliza tablas. Este método no se relaciona con la conexión de los puertos de los componentes, sino con la asignación de valores a objetos tales como puertos, señales y variables. En otras palabras, describe la manera en que los datos de salida se relacionan con los datos de entrada en el circuito. Ya hemos utilizado este método en varios de los circuitos introductorios en los capítulos 3 y 4, en forma de tablas de verdad. Con los circuitos tipo contador secuencial el equivalente de la tabla de verdad es la tabla de estado PRESENTE / estado SIGUIENTE, como vimos en la sección anterior. En esencia, podemos utilizar el HDL para describir la tabla de estado PRESENTE / estado SIGUIENTE y evitar así los tediosos detalles de generar las ecuaciones booleanas, como lo hicimos en la sección 7-10 para diseñar usando dispositivos lógicos estándar.

DESCRIPCIONES DE ESTADO EN AHDL

Como ejemplo de un circuito contador simple, implementaremos el contador MOD-5 de la figura 7-26 en AHDL. Las entradas y las salidas se definen en la sección SUBDESIGN de la figura 7-35, como siempre. En la sección VARIABLE de la línea 7 hemos declarado (o instanciado) un arreglo de tres bits de primitivas DFF, las cuales reciben el nombre de instancia *conteo[]*. Este arreglo se tratará como un registro de tres bits en el diseño; en esencia, definiremos qué valor debe almacenarse para cada estado SIGUIENTE. Como éste es un contador síncrono, necesitamos enlazar todas las entradas *clk* de cada DFF con la entrada *reloj* de SUBDESIGN. Esto se logra en AHDL mediante la siguiente instrucción en la sección lógica:

```
conteo[ ].clk = reloj;
```

Las primitivas de flip-flop que se proporcionan en AHDL tienen entradas y salidas estándar, a las cuales se les conoce como “puertos”. Estos puertos se etiquetan con base en un nombre estándar de puerto que se adjunta al nombre de instancia de los flip-flops. Como puede ver en la tabla 5-3, el nombre del puerto de reloj es *.clk*, una entrada *D* se llama *.d* y la salida del FF tiene el nombre *.q*. Para implementar la tabla de estado PRESENTE / estado SIGUIENTE se utiliza una instrucción CASE. Para cada uno de los posibles valores del registro *conteo[]* determinamos el valor que debe colocarse en las entradas *D* de los flip-flops, el cual determinará el estado SIGUIENTE del contador. La instrucción en la línea 21 asigna el valor en *conteo[]* a las terminales de salida. Sin esta línea el contador quedaría “oculto” en el bloque SUBDESIGN y no sería visible para el mundo exterior.

En la figura 7-36 se muestra una solución de diseño alternativa. Hay dos modificaciones de la figura 7-35. La primera se ve en la línea 7, en donde el nombre del arreglo para los flip-flops D es ahora el mismo que el puerto de salida para el bloque SUBDESIGN.

```

1  SUBDESIGN fig7 35
2  (
3      reloj      :INPUT;
4      q[2..0]    :OUTPUT;
5  )
6  VARIABLE
7      conteo[2..0] :DFF;      crea un registro de 3 bits
8  BEGIN
9      conteo[ ].clk = reloj;   conecta todos los relojes en paralelo
10
11      CASE conteo[] IS
12          Presente           Siguiete
13
14          WHEN 0    =>    conteo[ ].d = 1;
15          WHEN 1    =>    conteo[ ].d = 2;
16          WHEN 2    =>    conteo[ ].d = 3;
17          WHEN 3    =>    conteo[ ].d = 4;
18          WHEN 4    =>    conteo[ ].d = 0;
19          WHEN OTHERS =>    conteo[ ].d = 0;
20      END CASE;
21      q[] = conteo[];         asigna el registro a las terminales de salida
22  END;
```

FIGURA 7-35 Contador MOD-5 en AHDL.

FIGURA 7-36 Otra versión del contador MOD-5 descrito en la figura 7-26.

```

1  SUBDESIGN fig7 36
2  (
3      reloj      :INPUT;
4      q[2..0]    :OUTPUT;
5  )
6  VARIABLE
7      q[2..0]    :DFF;      crea un registro de 3 bits
8  BEGIN
9      q[].clk = reloj;      conecta todos los relojes en paralelo
10     TABLE
11         q[].q =>    q[].d;
12         0      =>    1;
13         1      =>    2;
14         2      =>    3;
15         3      =>    4;
16         4      =>    0;
17         5      =>    0;
18         6      =>    0;
19         7      =>    0;
20     END TABLE;
21 END;
```

Esto conectará automáticamente las salidas del flip-flop a las salidas de SUBDESIGN y eliminará la necesidad de incluir una instrucción de asignación como la línea 21 en la primera solución. La segunda modificación es el uso de una instrucción TABLE de AHDL en vez de la instrucción CASE utilizada en la figura 7-35. En la línea 11, el puerto *.q* en el arreglo DFF *q[]* representa el lado del estado PRESENTE de la tabla, mientras que el puerto *.d* para *q[]* representa el estado SIGUIENTE que se introducirá en el conjunto de entradas D del arreglo, cuando se aplique una PGT a *reloj*.

DESCRIPCIONES DE ESTADO EN VHDL

Como ejemplo de un circuito de contador simple, implementaremos el contador MOD-5 de la figura 7-26 en VHDL. Nuestro fin en este ejemplo es demostrar un contador mediante una estructura de control similar a una tabla de estado PRESENTE / estado SIGUIENTE. En VHDL se deben realizar dos tareas clave: detectar el flanco de reloj deseado y asignar el estado SIGUIENTE apropiado al contador. En capítulos anteriores sobre flip-flops vimos que puede usarse un bloque PROCESS para responder a una transición de una señal de entrada. Además aprendimos que una instrucción CASE puede evaluar una expresión y, para cualquier valor de entrada válido, asignar un valor correspondiente a otra señal. El código de la figura 7-37 utiliza una instrucción PROCESS y una instrucción CASE para implementar este contador. Las entradas y salidas están definidas en la declaración ENTITY, como en ejemplos pasados.

Cuando se utiliza VHDL para describir un contador, debemos encontrar una manera de “almacenar” el estado del contador entre los pulsos de reloj (es decir, la acción de un flip-flop). Esto se hace en una de dos formas: mediante señales (instrucción SIGNAL) o mediante variables (instrucción VARIABLE). Hemos usado señales SIGNAL con frecuencia en ejemplos anteriores que operaban en forma concurrente. Una señal en VHDL retiene el último valor que se le asignó, en forma muy parecida a un flip-flop. En consecuencia, podemos usar una señal como objeto de datos para representar el valor del contador. Después, esta señal puede usarse para

```

1  ENTITY fig7 37 IS
2  PORT (
3      reloj  :IN BIT;
4      q      :OUT BIT VECTOR(2 DOWNT0 0)
5  );
6  END fig7 37 ;
7
8  ARCHITECTURE a OF fig7 37  IS
9  BEGIN
10     PROCESS (reloj)
11         VARIABLE conteo: BIT VECTOR(2 DOWNT0 0);
12     BEGIN
13         IF (reloj = '1' AND reloj'EVENT) THEN
14             CASE conteo IS
15                 Presente           Siguiente
16
17                 WHEN "000"    =>    conteo := "001";
18                 WHEN "001"    =>    conteo := "010";
19                 WHEN "010"    =>    conteo := "011";
20                 WHEN "011"    =>    conteo := "100";
21                 WHEN "100"    =>    conteo := "000";
22                 WHEN OTHERS    =>    conteo := "000";
23             END CASE;
24         END IF;
25         q <= conteo;
26     END PROCESS;
27 END a;

```

FIGURA 7-37 Contador MOD-5 en VHDL.

conectar el valor del contador con cualquier otro elemento en la descripción de la arquitectura.

En este diseño hemos optado por usar una **VARIABLE** en lugar de una señal (SIGNAL) como el objeto de datos que almacena el valor del contador. Las **VARIABLES** no son idénticas a las señales porque no se utilizan para conectar diversas partes del diseño. En vez de ello se utilizan como una ubicación local para “almacenar” un valor. Las variables se consideran como objetos de datos locales, ya que se reconocen sólo dentro del bloque **PROCESS** en el que se declaran. En la línea 11 de la figura 7-37 se declara la variable llamada *conteo* dentro del bloque **PROCESO**, antes de la instrucción **BEGIN**. Su tipo es el mismo que el del puerto de salida *q*. La palabra clave **PROCESS** en la línea 10 va seguida de la lista de sensibilidad que contiene la señal de entrada *reloj*. Cada vez que *reloj* cambia de estado se invoca el bloque **PROCESS** y se evalúan las instrucciones dentro de **PROCESS** para producir un resultado. Un atributo **'EVENT'** se evaluará como **VERDADERO** si la señal que va antes de él acaba de cambiar de estado. La línea 13 establece que si *reloj* acaba de cambiar de estado y justo ahora es '1', entonces sabemos que fue un flanco de subida. Para implementar la tabla de estado **PRESENTE/estado SIGUIENTE** se utiliza una instrucción **CASE**. Para cada uno de los posibles valores de la variable *conteo* determinamos el estado **SIGUIENTE** del contador. Observe que se utiliza el operador “=” para asignar un valor a una variable. La línea 25 asigna el valor almacenado en *conteo* a las terminales de salida. Como *conteo* es una variable local, esta asignación debe realizarse antes de la instrucción **END PROCESS** en la línea 26.

Descripción del comportamiento

El **nivel de abstracción de comportamiento** es una manera de describir un circuito mediante la descripción de su comportamiento en términos muy similares a la manera en que podría describirse su operación en español. Piense acerca de la forma en que podría describirse la operación de un circuito contador por alguien que no conozca nada acerca de los flip-flops o las compuertas lógicas. Tal vez la descripción de esa persona sería algo así como: “Cuando la entrada del contador cambie de BAJO a ALTO, el número en la salida cuenta en forma ascendente por 1”. Este nivel de descripción se relaciona más con las relaciones de causa y efecto que con la ruta del flujo de datos o los detalles del cableado. Sin embargo, en realidad no podemos tan solo utilizar cualquier descripción en español para describir el comportamiento del circuito. Debe utilizarse la sintaxis apropiada dentro de las restricciones del HDL.

AHDL

En AHDL, el primer paso importante en este método de descripción es declarar las terminales de salida del contador en forma apropiada. Deben declararse como un arreglo de bits, en donde los índices se decrementen de izquierda a derecha y 0 sea el índice menos significativo en el arreglo, algo opuesto a los bits individuales llamados a, b, c, d, y así en lo sucesivo. De esta forma, el valor numérico asociado con el nombre del arreglo de bits se interpreta como un número binario con base en el cual pueden realizarse ciertas operaciones aritméticas. Por ejemplo, el arreglo de bits *conteo* que se muestra en la figura 7-38 podría contener los bits 1001, como se muestra. El compilador de AHDL interpreta este patrón de bits como si tuviera el valor de 9 en decimal.

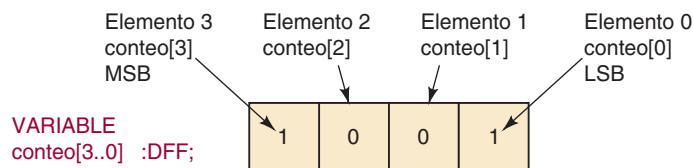
Para poder crear nuestro contador MOD-5 en AHDL, necesitaremos un registro de tres bits que almacene el estado actual del contador. Este arreglo de tres bits, al cual llamaremos *conteo*, se declara mediante el uso de flip-flops D en la línea 7 de la figura 7-39. Si recuerda de la figura 7-36, podemos nombrar el arreglo DFF igual que el puerto de salida *q[2..0]* y en consecuencia eliminar la línea 15, pero también tendríamos que cambiar *conteo[]* por *q[]* en cualquier parte en donde aparezca dentro de la sección lógica. En otras palabras, la instrucción de la línea 7 puede cambiarse por

```
q[2..0] :DFF;
```

Si hiciéramos esto, todas las referencias a *conteo* a partir de ese punto se cambiarían por *q*. Esto puede hacer que el código sea más corto, pero no demuestra los conceptos universales de HDL con tanta claridad. En AHDL todos los relojes pueden especificarse como si estuvieran enlazados entre sí y conectados a una fuente de reloj común, mediante el uso de la instrucción de la línea 10, *conteo[].clk = reloj*. En este ejemplo, *conteo[].clk* hace referencia a la entrada de reloj de cada flip-flop en el arreglo llamado *conteo*.

La descripción del comportamiento de este contador es muy simple. El estado actual del contador se evalúa (*conteo[].q*) en la línea 11, y si es menor que el valor de conteo deseado más alto, utiliza la descripción *conteo[].d = conteo.q + 1* (línea 12). Esto significa que el estado actual de las entradas *D* debe ser igual a un valor que sea un conteo mayor que el estado actual de las salidas *Q*. Cuando el estado actual del contador ha llegado al estado deseado más alto (o mayor), la prueba de la instrucción *IF* será falsa y se producirá un valor de cero en la entrada del estado

FIGURA 7-38 Los elementos de un registro D, en el que se almacena el número 9.




```

1  SUBDESIGN fig7 39
2  (
3      reloj      :INPUT;
4      q[2..0]    :OUTPUT;      declara arreglo de 3 bits de los bits de salida
5  )
6  VARIABLE
7      conteo[2..0] :DFF;      declara un registro de flip flops D.
8
9  BEGIN
10     conteo[].clk = reloj;      conecta todos los relojes a la fuente síncrona
11     IF conteo[].q < 4 THEN      nota; conteo[ ] es lo mismo que conteo[ ].q
12         conteo[].d = conteo[].q + 1;  incrementa en uno el valor actual
13     ELSE conteo[].d = 0;        se recicla a cero: fuerza los estados no usados a 0
14     END IF;
15     q[] = conteo[];            transfiere el contenido del registro a las salidas
16 END;

```

FIGURA 7-39 Descripción del comportamiento de un contador en AHDL

SIGUIENTE (línea 13), con lo cual se reciclará el contador. La última instrucción en la línea 15 sólo conecta el valor del contador con las terminales de salida del dispositivo.

VHDL

En VHDL el primer paso importante en este método de descripción es declarar en forma apropiada el puerto de salida del contador, como se muestra en la figura 7-40. El tipo de datos del puerto de salida (línea 3) debe concordar con el tipo de la

```

1  ENTITY fig7 40 IS
2  PORT( reloj      :IN BIT;
3        q          :OUT INTEGER RANGE 0 TO 7  );
4  END  fig7 40;
5
6  ARCHITECTURE a OF fig7 40 IS
7  BEGIN
8      PROCESS (reloj)
9          VARIABLE conteo: INTEGER RANGE 0 to 7;      define una VARIABLE numérica
10         BEGIN
11             IF (reloj = '1' AND reloj'EVENT) THEN      flanco ascendente?
12                 IF conteo < 4 THEN                      menor que máx?
13                     conteo := conteo + 1;              incrementa el valor
14                 ELSE                                    debe estar en máx o mayor
15                     conteo := 0;                        se recicla a cero
16                 END IF;
17             END IF;
18             q <= conteo;                                transfiere el contenido del registro a las salidas
19         END PROCESS;
20     END a;

```

FIGURA 7-40 Descripción del comportamiento de un contador en VHDL.

variable del contador (línea 9) y debe ser un tipo que permita operaciones aritméticas. Recuerde que VHDL trata a los objetos `BIT_VECTOR` como una cadena de bits, no como una cantidad numérica binaria. Para poder reconocer la señal como una cantidad numérica, el objeto de datos debe escribirse como tipo `INTEGER`. El compilador busca en la cláusula `RANGE 0 TO 7` de la línea 3 y sabe que el contador necesita tres bits. Se necesita una declaración similar para la variable del registro en la línea 9, para que cuente hacia arriba. A ésta se le llamará *conteo*. La primera instrucción después de `BEGIN` en el bloque `PROCESS` responde al flanco de subida del reloj, como en los ejemplos anteriores. Después utiliza métodos de descripción del comportamiento para definir la respuesta del contador al flanco del reloj. Si el contador no ha llegado a su máximo (línea 12) entonces deberá incrementarse (línea 13). En caso contrario (línea 14) deberá reciclar el contador a cero (línea 15). La última instrucción en la línea 18 sólo conecta el valor del contador con las terminales de salida del dispositivo.

Simulación de contadores básicos

La simulación de cualquiera de nuestros diseños de contadores MOD-5 es bastante simple. Los contadores sólo tienen un bit de entrada (*reloj*) y tres bits de salida (*q2 q1 q0*) para mostrar en la simulación. La frecuencia de reloj no se ha especificado, por lo que podemos utilizar cualquier frecuencia que queramos para una simulación funcional; aunque tal vez deberíamos evitar un reloj de alta frecuencia, a menos que se desee investigar los efectos de los retrasos de propagación. La única decisión que debemos hacer es cuántos pulsos aplicar. Como el contador es MOD-5, debemos aplicar por lo menos cinco pulsos de reloj para verificar que el diseño de HDL tenga la secuencia de conteo correcta y que se recicle. Empezaremos la simulación con el estado inicial 000, ya que los PLDs de Altera tienen una característica de reinicio de encendido integrada. No podremos probar cualquiera de los estados no usados, ya que los diseños de HDL no proporcionan la manera de preestablecer el contador a cualquiera de los estados no usados. En la figura 7-41 se muestran los resultados que obtuvimos con nuestra simulación para el diseño en HDL de un contador MOD-5.

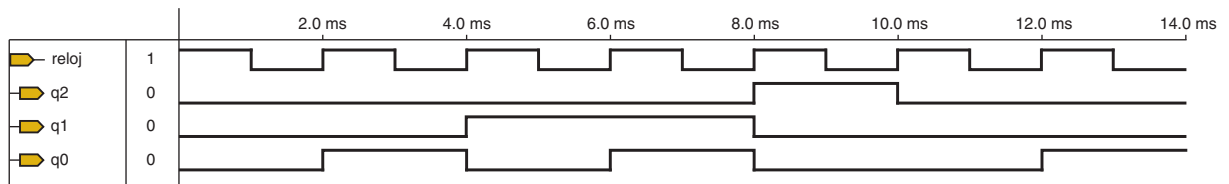


FIGURA 7-41 Resultados de la simulación para el diseño en HDL de un contador MOD-5.

PREGUNTAS DE REPASO

1. ¿Qué tipo de tabla se utiliza para describir la operación de un contador?
2. Al diseñar un contador con flip-flops D, ¿qué se aplica a las entradas D para poder excitarlo hacia el estado SIGUIENTE en el siguiente flanco activo del reloj?
3. ¿Cómo escribiría la descripción en HDL para disparar un dispositivo de almacenamiento (flip-flop) en un flanco de caída, en vez de un flanco de subida del reloj?
4. ¿Qué método describe la operación del circuito mediante el uso de relaciones tipo causa y efecto?

7-12 CONTADORES CON TODAS LAS CARACTERÍSTICAS EN HDL

Los ejemplos que hemos elegido hasta ahora han sido de contadores muy básicos. Todo lo que hacen es contar hasta cuatro y luego se regresan a cero. Los contadores de CI estándar que hemos examinado tienen muchas otras características que los hacen muy útiles para numerosas aplicaciones digitales. Por ejemplo, considere los contadores de CI 74161 y 74191 que vimos en la sección 7-7. Estos dispositivos tienen combinaciones de varias características, incluyendo la habilitación del conteo, el conteo ascendente/descendente, la carga en paralelo (se preestablece a cualquier conteo) y el borrado. Además, estos contadores se han diseñado para conectarse en cascada con facilidad y en forma síncrona para crear contadores más grandes. En esta sección exploraremos las técnicas que nos permitirán incluir esas características en un contador de HDL. Vamos a crear un contador que combinará más características de las que se encuentran ya sea en el 74161 o en el 74191. Utilizaremos este ejemplo para demostrar los métodos de diseño de un contador con capacidades que se adapten en forma específica a nuestras necesidades. Cuando utilizamos HDLs para crear diseños digitales, no estamos limitados a las características que suelen incluirse con cierto CI.

Vamos a revisar las especificaciones para nuestro ejemplo de un contador más complejo. El contador binario MOD-16 reciclable debe cambiar de estado en el flanco de subida de la entrada de reloj cuando el contador esté habilitado en un nivel ALTO. Una entrada de control de dirección hará que el contador cuente en forma ascendente cuando esté en BAJO, o que cuente en forma descendente cuando esté en ALTO. El contador tendrá una señal de limpiar asíncrona, activa en ALTO, para restablecer el contador de inmediato cuando la entrada de control se active. El contador puede cargarse en forma síncrona con un número en las terminales de entrada de datos cuando el control de carga está en ALTO. La prioridad de las funciones de control de entrada, de mayor a menor, serán borrar, cargar y contar. Por último, el contador también incluirá una salida activa en ALTO que detecte su estado terminal cuando esté habilitada la función de conteo. Recuerde que el estado terminal dependerá de la dirección de conteo. Como veremos, la correcta operación de estas características se determina en base a la forma en que escribimos el código de HDL, por lo que tendremos que poner mucha atención a los detalles.

CONTADOR CON TODAS LAS CARACTERÍSTICAS EN AHDL

El código de la figura 7-42 implementa todas las características de las que hemos hablado. Es un contador de cuatro bits, pero puede expandirse en tamaño con facilidad. Analice las entradas y salidas de las líneas 3 y 4 para asegurarse de comprender lo que se supone debe hacer cada una. Si no lo entiende, vuelva a leer los párrafos anteriores de esta sección. La línea 7 define un registro de cuatro bits con flip-flops D, el cual servirá como contador. Debemos observar de nuevo aquí que este registro podría haberse llamado de igual forma que la variable de entrada (*q*). El código está escrito con distintos nombres para diferenciar los puertos (entradas y salidas) del circuito y los dispositivos que operan dentro del mismo. La entrada de reloj está conectada a todas las entradas *clk* de todos los flip-flops D en la línea 10. Todas las entradas borrar activas en BAJO (*clrn*) que van a la primitiva DFF están conectadas al complemento de la señal de entrada *borrar* en la línea 11. Esto hace que el flip-flop se borre de inmediato cuando la entrada *borrar* cambia a ALTO, debido a que las entradas *prn* y *clrn* de la primitiva DFF no dependen del reloj (es decir, son asíncronas).

Para que la función de carga pueda funcionar en forma síncrona deben controlarse las entradas *D* de los flip-flops, de manera que los datos de entrada (*dent*) estén presentes en las entradas *D* cuando la línea de carga esté en ALTO. De esta manera, cuando llegue el siguiente flanco activo del reloj los datos se cargarán en el contador. Esta acción debe ocurrir sin importar que el contador esté habilitado o no. En consecuencia, la primera decisión condicional (IF) en la línea 12 evalúa la entrada de carga. Recuerde del capítulo 4 que la estructura de decisión IF/ELSE da

```

1  SUBDESIGN fig7 42
2  (
3      reloj, borrar, cargar, habilcnt, abajo, dent[3..0]      :INPUT;
4      q[3..0], term ct :OUTPUT;      declara arreglo de 4 bits de salida
5  )
6  VARIABLE
7      conteo[3..0]      :DFF;          declara un registro de flip flops D
8
9  BEGIN
10     conteo[].clk = reloj;              conecta todos los relojes a la fuente de sincronía
11     conteo[].clrn= !borrar;            conecta para borrar asíncrona activa en ALTO
12     IF cargar THEN conteo[].d = dent[]; carga síncrona
13         ELSIF !habilcnt THEN conteo[].d = conteo[].q;      retiene el conteo
14         ELSIF !abajo THEN conteo[].d = conteo[].q + 1;    incremento
15         ELSE conteo[].d = conteo[].q - 1;                  decremento
16     END IF;
17     IF ((conteo[].q == 0) & abajo # (conteo[].q == 15) & !abajo) & habilcnt
18     THEN      ct term = VCC;          señal de salida en cascada síncrona
19     ELSE ct term = GND;
20     END IF;
21     q[] = conteo[];                  transfiere el contenido del registro a las salidas
22 END;
```

FIGURA 7-42 Contador con todas las características en AHDL.

precedencia a la primera condición que se encuentra verdadera ya que, una vez que encuentra una condición verdadera, no sigue evaluando las condiciones en las cláusulas ELSE subsiguientes. En este caso significa que si se activa la línea de carga, no importa si se habilita el conteo o si está tratando de contar hacia arriba o hacia abajo, sino que realizará una carga en paralelo en el siguiente flanco del reloj.

Suponiendo que la línea de carga no esté activa, se evalúa la cláusula ELSIF en la línea 13 para ver si el conteo está deshabilitado. En AHDL es muy importante tener en cuenta que la salida *Q* debe retroalimentarse a la entrada *D* para que en el siguiente flanco del reloj el registro retenga su valor anterior. Si no se inserta esta cláusula las entradas *D* quedarán en cero, con lo cual se restablecerá el contador. Si el contador está habilitado se evalúa la cláusula ELSIF de la línea 14 y *conteo* se incrementa (línea 14) o se decrementa (línea 15). Para resumir estas decisiones, primero se decide si es tiempo de cargar, después se decide si el conteo debe retenerse o cambiar, después se decide si se va a contar hacia arriba o hacia abajo.

La siguiente función descrita es la detección (o decodificación) del conteo terminal. Las líneas 17-20 deciden si se ha llegado al conteo terminal mientras se cuenta hacia arriba o hacia abajo. El operador de doble signo de igual ($=$) es el símbolo que prueba la igualdad entre las expresiones de cada lado del operador. El estado del contador que sea el estado terminal depende de la dirección de conteo. Esto se determina mediante la aplicación de una operación AND entre la detección apropiada del estado terminal de 0 o 15 y la expresión correcta, *abajo* o *!abajo*. *ct_term* producirá como salida un nivel ALTO si se ha llegado al estado correcto, en caso contrario será BAJO. La línea 21 conectará la salida para *conteo* con las terminales de salida para el bloque SUBDESIGN.

Uno de los conceptos clave acerca del uso de HDLs es que, por lo general, es muy sencillo expandir el tamaño de un módulo lógico. Veamos ahora los cambios necesarios a este diseño de AHDL para incrementar el módulo del contador binario a 256. Como $2^8 = 256$, necesitaremos incrementar el número de bits a ocho. Sólo

se requieren cuatro modificaciones a la figura 7-42 para hacer este cambio en el módulo del contador:

# de línea	Modificación
3	dent[3 7..0]
4	q[3 7..0]
7	conteo[3 7..0]
17	(conteo[].q == 15 255)

CONTADOR CON TODAS LAS CARACTERÍSTICAS EN VHDL

El código de la figura 7-43 implementa todas las características de las que hemos hablado. Es un contador de cuatro bits, pero puede expandirse en tamaño con facilidad. Analice las entradas y salidas de las líneas 2 a la 5 para asegurarse de comprender lo que se supone debe hacer cada una. Si no lo entiende, vuelva a leer los párrafos anteriores de esta sección. La instrucción PROCESS en la línea 10 es la clave para todos los circuitos sincronizados por reloj que se describen en VHDL,

```

1  ENTITY fig7 43 IS
2  PORT( reloj, borrar, cargar, habilcnt, descendente   :IN BIT;
3        dent           :IN INTEGER RANGE 0 TO 15;
4        q              :OUT INTEGER RANGE 0 TO 15;
5        ct term        :OUT BIT);
6  END fig7 43;
7
8  ARCHITECTURE a OF fig7 43 IS
9      BEGIN
10     PROCESS ( reloj, borrar, descendente)
11         VARIABLE conteo :INTEGER RANGE 0 to 15;           define una señal numérica
12         BEGIN
13             IF borrar = '1' THEN conteo := 0;             borrar asíncrona
14             ELSIF (reloj = '1' AND reloj'EVENT) THEN      flanco ascendente?
15                 IF cargar = '1' THEN conteo := din;       carga en paralelo
16                 ELSIF habilcnt = '1' THEN                 habilitado?
17                     IF descendente = '0' THEN conteo := conteo + 1;    incremento
18                     ELSE
19                         conteo := conteo - 1;    decremento
20                     END IF;
21                 END IF;
22             END IF;
23             IF (((conteo = 0) AND (descendente = '1')) OR
24                 ((conteo = 15) AND (descendente = '0'))) AND habilcnt = '1'
25                 THEN ct term <= '1';
26             ELSE
27                 ct term <= '0';
28             END IF;
29             q <= conteo;      transfiere el contenido del registro a las salidas
30         END PROCESS;
31     END a;

```

FIGURA 7-43 Contador con todas las características en VHDL.

pero también juega un papel importante en cuanto a determinar si el circuito responde en forma síncrona o asíncrona a sus entradas. Queremos que este circuito responda de inmediato a las transiciones en las entradas *reloj*, *borrar* y *abajo*. Con estas señales en la lista de sensibilidad nos aseguramos que se evalúe el código dentro del bloque PROCESS tan pronto como cualquiera de estas entradas cambie de estado. La variable *conteo* se define en la línea 11 como tipo INTEGER, de forma que pueda incrementarse y decrementarse con facilidad. Las variables se declaran dentro del bloque PROCESS y pueden utilizarse sólo dentro de este bloque.

La entrada *clear* recibe precedencia al evaluarla con la primera instrucción IF en la línea 13. Recuerde del capítulo 4 que la estructura de decisión IF/ELSE da precedencia a la primera condición que se encuentre verdadera, ya que no sigue evaluando las condiciones en las cláusulas ELSE subsiguientes. En este caso, si la señal *borrar* está activa las demás condiciones no importarán. La salida será cero. Para que la función *cargar* pueda operar en forma síncrona, debe evaluarse después de detectar el flanco del reloj. Éste se detecta en la línea 14 y el circuito comprueba de inmediato si *cargar* está activa. De ser así, el *conteo* se carga de *dent*, sin importar que esté habilitado el conteo o no. En consecuencia, la decisión condicional (IF) de la línea 15 evalúa la entrada *cargar*; sólo si está inactiva se evalúa la línea 16 para ver si el contador está habilitado. Si está habilitado, el *conteo* se incrementará o decrementará (líneas 17 y 18, respectivamente).

Lo siguiente es detectar el conteo terminal. Las líneas 22 a 25 deciden si se ha llegado al conteo terminal máximo o mínimo y excitan la salida para que cambie al nivel apropiado. Aquí, la estructura de toma de decisiones es muy importante ya que queremos evaluar esta situación, sin importar que el proceso de toma de decisiones haya sido invocado por *reloj*, *borrar* o *abajo*. Observe que esta decisión no es otra ramificación ELSE de las decisiones IF anteriores, sino que se evalúa para cada señal en la lista de sensibilidad *después* de que se produce el borrado o el conteo. Una vez que se realicen todas estas decisiones, *conteo* deberá tener el valor correcto en el registro y la línea 27 conectará el registro con las terminales de salida.

Uno de los conceptos clave acerca del uso de HDLs es que, por lo general, es muy sencillo expandir el tamaño de un módulo lógico. Veamos ahora los cambios necesarios a este diseño de VHDL para incrementar el módulo del contador binario a 256. Sólo se requieren cuatro modificaciones a la figura 7-42 para hacer este cambio en el módulo del contador:

# de línea	Modificación
3	RANGE 0 TO 15-255
4	RANGE 0 TO 15-255
11	RANGE 0 TO 15-255
23	(conteo = 15-255)

Simulación de un contador con todas las características

Para simular nuestro diseño del contador con todas las características requeriremos de cierta planeación para generar formas de onda de entrada apropiadas. Aunque tal vez no sea necesario simular a detalle todas las combinaciones de entrada imaginables, sí necesitamos probar la suficiente cantidad de condiciones de entrada posibles para convencernos de que funciona en forma apropiada. Esto es exactamente lo que debemos hacer también para probar nuestro diseño de prototipo en el laboratorio. El contador tiene cinco señales de entrada distintas (*reloj*, *borrar*, *cargar*, *habilitar* y *dent*) y dos señales de salida distintas (*q* y *ct_term*) que debemos visualizar en nuestra simulación. Una de las señales de entrada y una de las señales de salida son en realidad de cuatro bits. Elegiremos una frecuencia de reloj conveniente, ya que no se ha especificado ninguna para nuestra simulación funcional del contador. Necesitaremos proveer los suficientes pulsos de reloj como para poder analizar varias condiciones de operación. La simulación deberá evaluar las funciones de habilitar y deshabilitar el contador, contar hacia arriba y hacia abajo, borrar

el contador, cargar un valor en el contador y contar a partir de ese valor, y detectar el estado de conteo terminal.

Hay ciertas cuestiones de simulación en general que debemos considerar al crear nuestras formas de onda de entrada. Como los PLDs de destino se restablecen al momento de encenderse, nuestra simulación empezará con el estado de salida inicial en 0000. Por lo tanto, sería mejor esperar hasta que el contador haya llegado a otro estado antes de aplicar una entrada para borrarlo, de manera que podamos ver un cambio en la salida. De igual forma, cargar el mismo valor que el estado SIGUIENTE del contador no nos convencerá en realidad de que la función *cargar* está operando en forma correcta. Si se cambian las señales de control de entrada al mismo tiempo que se produzca el flanco de reloj se pueden provocar algunos problemas con el tiempo de establecimiento y se obtendrían resultados cuestionables. Los controles asíncronos deben aplicarse en un tiempo distinto al del flanco de reloj apropiado para demostrar con claridad que la acción resultante del circuito es inmediata y no depende del reloj. En general, debemos usar el sentido común para crear nuestras formas de onda de entrada y considerar qué es lo que estamos tratando de verificar con la simulación. Ésta será valiosa en el proceso de diseño sólo si aplicamos las condiciones de entrada apropiadas y evaluamos los resultados de manera crítica.

En la figura 7-44 se muestran algunos resultados de la simulación para el contador con todas las características. La entrada *dent* de cuatro bits y la salida *q* de cuatro bits se muestran en hexadecimal. Al principio el contador está habilitado (*habilcnt* = 1) para contar hacia arriba (*abajo* = 0), y podemos ver que la salida se incrementa en 0, 1, 2, 3, 4, 5. En t_1 el contador responde en forma síncrona (es decir, en la PGT de *reloj*) al nivel ALTO que se aplica en la entrada *cargar*. El contador se prestablece con el valor de 8 de los datos en paralelo (*dent*). Esto también muestra que la carga tiene prioridad sobre el conteo, ya que ambas entradas están activas al mismo tiempo. Después de t_1 , la entrada *cargar* está en BAJO otra vez y el contador sigue contando en forma ascendente a partir de 8. Una entrada en BAJO para *habilcnt* hace que el contador se detenga en el estado 9 durante un ciclo extra de reloj. El conteo se continúa cuando *habilcnt* está en ALTO otra vez hasta t_2 , cuando el contador se borra en forma asíncrona. Observe el tiempo reducido para el estado de salida A debido a que el contador se borra de inmediato. En realidad tendríamos que hacer un acercamiento para ver que se muestra el estado A. También podemos ver que la función *borrar* tiene la prioridad más alta cuando los tres controles (*borrar*, *cargar* y *habilcnt*) están al mismo tiempo en ALTO. La secuencia de conteo ascendente continúa y se recicla a 0 después del estado F, para verificar que el contador sea un contador binario MOD-16. En t_3 el contador llega a su estado terminal F cuando cuenta hacia arriba, y *ct_term* produce un nivel ALTO. En t_4 el contador empieza a contar hacia abajo, ya que *abajo* se cambió a un nivel ALTO. De nuevo, *ct_term* produce un nivel ALTO ya que el contador se encuentra ahora en el estado 0, que es el estado terminal cuando se cuenta hacia abajo. Observe que a través de la acción de *ct_term*, el estado terminal para el contador depende de su dirección de conteo, la cual se controla mediante la entrada *abajo*. El conteo se retiene en el

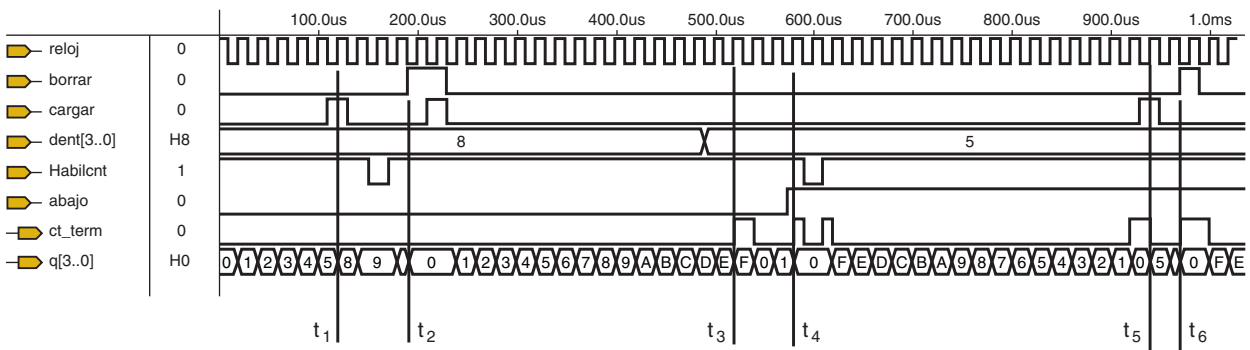


FIGURA 7-44 Resultados de la simulación para el diseño en HDL de un contador con todas las características.

estado 0 durante un periodo de reloj adicional cuando *habilcnt* cambia a BAJO. La salida *ct_term* también se deshabilita mientras que *habilcnt* = 0. La secuencia de conteo descendente continúa en forma correcta cuando *habilcnt* cambia de nuevo a ALTO. En t_5 el contador carga en forma síncrona el valor de datos en paralelo 5. En t_6 el contador se borra en forma asíncrona. De nuevo se verifica que la función de cargar o borrar tenga prioridad sobre la función de conteo descendente en t_5 y en t_6 . ¿Verificamos que nuestro diseño opere en forma correcta, en comparación con las especificaciones? Hicimos un muy buen trabajo, pero hay un par de condiciones de prueba que también podrían agregarse para hacer la simulación más completa. ¿Se borró o se cargó el contador cuando *habilcnt* estaba en BAJO? Parece ser que no verificamos esos casos. Como puede ver, los diseños complejos pueden requerir mucho razonamiento para verificar su operación de una manera adecuada mediante la simulación o la prueba en el laboratorio. ¿Puede pensar en alguna otra prueba que deberíamos realizar?

PREGUNTAS DE REPASO

1. ¿Cuál es la diferencia entre la función borrar asíncrona y la función cargar síncrona?
2. ¿Cómo se crea una función borrar asíncrona en un HDL?
3. ¿Cómo se crea la prioridad de las funciones en una descripción en HDL de un contador?

7-13 CÓMO ALAMBRAR MÓDULOS DE HDL EN CONJUNTO

En las dos secciones anteriores vimos cómo implementar características comunes de un contador mediante HDL. También deberíamos investigar cómo podemos conectar estos circuitos tipo contador a otros módulos digitales para crear sistemas más grandes. El diseño de sistemas digitales extensos se facilita mucho si el sistema se subdivide en módulos más pequeños y manejables que después se interconectan. Ésta es la esencia del concepto del **diseño jerárquico**, del cual veremos sus beneficios mediante proyectos de ejemplo en el capítulo 10. Ahora veamos las técnicas básicas para alambrear módulos entre sí.

DECODIFICACIÓN DEL CONTADOR MOD-5 EN AHDL

En la sección 7-8 dimos un breve vistazo a la idea de decodificar un contador. Como recordará, un circuito decodificador detecta un estado de conteo mediante el patrón de bits único para ese estado. Ahora veamos cómo conectar un circuito decodificador al diseño del contador MOD-5 de la figura 7-35 (o 7-36). Cambiaremos el nombre del contador por SUBDESIGN mod5 para ser un poco más descriptivos en el diagrama de bloques para el circuito general que dibujaremos más adelante. Como el contador no produce los ocho estados posibles para un contador de tres bits, nuestro diseño del decodificador que se muestra en la figura 7-45 sólo decodificará los estados que se utilicen, de 000 hasta 100. Los tres bits de entrada (c = MSB) que se declaran en la línea 3 se conectarán más adelante a las salidas del contador MOD5. En la línea 4, las cinco salidas para el decodificador se llaman *estado0* a *estado4*. Una instrucción CASE (líneas 7 a 14) describe el comportamiento del decodificador mediante la comprobación de la combinación de entradas $c b a$ para determinar cuál de las salidas del decodificador debe estar en ALTO. Cuando la entrada $c b a$ sea 000 sólo la salida *estado0* estará en ALTO, o cuando $c b a$ sea 001 sólo la salida *estado1* estará en ALTO, y así en lo sucesivo. Cualquier valor de entrada mayor de 100, que está cubierto por la instrucción OTHERS y en realidad no debe ocurrir en esta aplicación, producirá niveles BAJO en todas las salidas.

```

1  SUBDESIGN decodifica5
2  (
3      c, b, a          :INPUT;
4      estado[0..4]     :OUTPUT;
5  )
6  BEGIN
7      CASE (c, b, a) IS      -- decodifica el valor binario
8          WHEN B="000" => estado[ ] = B"10000";
9          WHEN B="001" => estado[ ] = B"01000";
10         WHEN B="010" => estado[ ] = B"00100";
11         WHEN B="011" => estado[ ] = B"00010";
12         WHEN B="100" => estado[ ] = B"00001";
13         WHEN OTHERS  => estado[ ] = B"00000";
14     END CASE;
15 END;

```

FIGURA 7-45 Módulo decodificador del contador MOD-5 en AHDL.

Indicaremos al software de Altera que cree símbolos para nuestros dos archivos de diseño, mod5 y decodifica5. Esto nos permitirá dibujar un diagrama de bloques (vea la figura 7-46) para nuestro circuito completo, el cual consistirá de estos dos módulos, los puertos de entrada y salida, y el cableado entre ellos. Cada símbolo se etiqueta con su respectivo nombre mod5 o decodifica5 de SUBDESIGN. Observe que parte del cableado se dibuja con líneas más gruesas. Con el fin de representar un bus, el cual es una colección de líneas de señal. Las líneas más delgadas son señales individuales. En los símbolos creados por Altera se dibujarán de manera automática puertos para indicar si representan señales individuales o buses. Esto se determinará mediante las declaraciones de las señales en la sección SUBDESIGN. Los puertos con nombres de grupo se dibujarán como buses. Como el puerto de salida del contador es un bus pero los puertos de entrada del decodificador son señales individuales, será necesario dividir el bus en líneas de señal individuales para cablear los dos módulos entre sí. Cada vez que se divide un bus, hay que etiquetar tanto el nombre de la señal de grupo del bus como las señales individuales que se vayan a utilizar. Nuestro diagrama de bloques tiene un bus etiquetado como $q[2..0]$ y las correspondientes señales individuales $q2$, $q1$ y $q0$. Los resultados de la simulación para este circuito contador y decodificador se muestran en la figura 7-47.

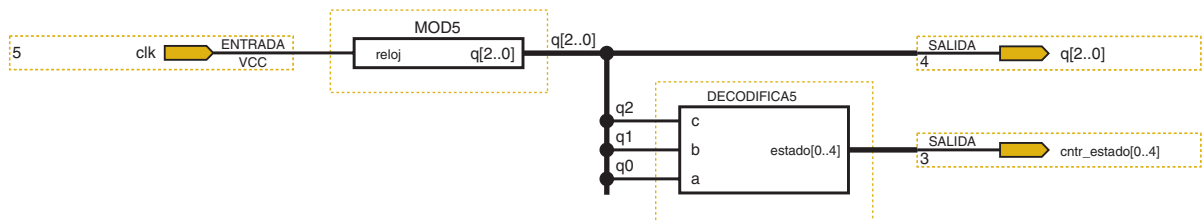


FIGURA 7-46 Diseño de diagrama de bloques para el circuito contador y decodificador MOD-5.

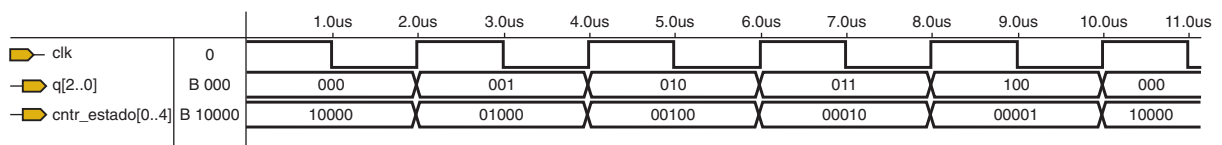


FIGURA 7-47 Simulación del circuito contador y decodificador MOD-5.

DECODIFICACIÓN DEL CONTADOR MOD-5 EN VHDL

En la sección 7-8 dimos un breve vistazo a la idea de decodificar un contador. Como podrá recordar, un circuito decodificador detecta un estado de conteo mediante el patrón de bits único para ese estado. Ahora veamos cómo conectar un circuito decodificador al diseño del contador MOD-5 de la figura 7-37. Cambiaremos el nombre del contador por ENTITY mod5 para facilitar la identificación del módulo en nuestro circuito en general. Como el contador no produce los ocho estados posibles para un contador de tres bits, nuestro diseño del decodificador que se muestra en la figura 7-48 sólo decodificará los estados que se utilicen, de 000 hasta 100. Los tres bits de entrada ($c = \text{MSB}$) que se declaran en la línea 3 se conectarán más adelante a las salidas del contador MOD5. En la línea 4, las cinco salidas para el decodificador se llaman *estado*, un vector de bits. En la línea 9 se declara una señal tipo vector de bits interna llamada *entrada*. Después la línea 11 combina los tres bits de entrada ($c\ b\ a$) para formar un vector de bits llamado *entrada*, el cual puede entonces evaluarse mediante la instrucción CASE en las líneas 14-21. Si cualquiera de los bits de entrada cambia de nivel lógico, se invocará el bloque PROCESS para determinar la salida resultante. Para describir el comportamiento del decodificador, la instrucción CASE comprueba la combinación de *entrada* (que representa $c\ b\ a$) para determinar cuál de las salidas del decodificador debe estar en ALTO. Cuando la *entrada* sea 000 sólo la salida *estado*(0) estará en ALTO; cuando *entrada* sea 001 sólo la salida *estado*(1) estará en ALTO; y así en lo sucesivo. Cualquier valor de entrada mayor de 100, que está cubierto por la instrucción OTHERS y en realidad no debe ocurrir en esta aplicación, producirá niveles BAJO en todas las salidas.

Como estamos usando el software de desarrollo de PLDs de Altera, podemos conectar los dos módulos en forma gráfica. Para ello necesitará instruir al software para que cree símbolos para nuestros dos archivos de diseño, mod5 y decodifica5. Esto nos permitirá dibujar un diagrama de bloques (vea la figura 7-46) para nuestro circuito completo, el cual consistirá de dos módulos, puertos de entrada y salida y el cableado entre ellos. Observe que parte del cableado se dibuja con líneas más gruesas. Para representar un bus, el cual es una colección de líneas de señal. Las líneas más delgadas son señales individuales. En los símbolos creados por Altera se dibujarán automáticamente puertos para indicar si representan señales individuales o buses. Esto se determinará mediante las declaraciones del tipo de datos

```

1  ENTITY decodifica5 IS
2  PORT (
3      c, b, a      :IN BIT;
4      estado      :OUT BIT_VECTOR (0 TO 4)
5  );
6  END decodifica5;
7
8  ARCHITECTURE a OF decodifica5 IS
9  SIGNAL  entrada  : BIT_VECTOR (2 DOWNT0 0);
10 BEGIN
11     entrada <= (c & b & a);      -- combina las entradas en un vector de bits
12     PROCESS (c, b, a)
13     BEGIN
14         CASE entrada IS
15             WHEN "000"          =>      estado <= "10000";
16             WHEN "001"          =>      estado <= "01000";
17             WHEN "010"          =>      estado <= "00100";
18             WHEN "011"          =>      estado <= "00010";
19             WHEN "100"          =>      estado <= "00001";
20             WHEN OTHERS         =>      estado <= "00000";
21         END CASE;
22     END PROCESS;
23 END a;
```

FIGURA 7-48 Módulo decodificador del contador MOD-5 en VHDL.

para cada puerto del bloque ENTITY. Los puertos BIT_VECTOR se dibujarán como buses y los puertos tipo BIT se dibujarán como líneas de señal individuales. Como el puerto de salida del contador es un bus pero los puertos de entrada del decodificador son señales individuales, será necesario dividir el bus en líneas de señal individuales para cablear los dos módulos entre sí. Cada vez que se divide un bus se debe etiquetar tanto el nombre de la señal de grupo del bus como las señales individuales que se vayan a utilizar. Nuestro diagrama de bloques tiene un bus etiquetado como $q[2..0]$ y las señales individuales correspondientes $q2$, $q1$ y $q0$. Los resultados de la simulación para este circuito contador y decodificador se muestran en la figura 7-47.

La técnica de VHDL estándar (y una alternativa con el software de Altera) para conectar módulos de diseño es utilizar VHDL para describir las conexiones entre los módulos en un archivo de texto. Se crean instancias de los módulos deseados en un archivo de diseño de mayor nivel mediante el uso de componentes (COMPONENT), en los cuales se declaran los puertos (PORT) del módulo. Las conexiones de cableado para cada instancia en donde se utiliza el módulo se listan en una instrucción PORT MAP. En la figura 7-49 se muestra un archivo de VHDL que conecta los módulos mod5 y decodifica5 entre sí. Aún y cuando q es un puerto de salida para nuestro archivo de diseño de nivel superior, se escribe como BUFFER en la línea 4 debido al hecho de que es necesario “leer” el arreglo de vectores de bit para una entrada al componente (COMPONENT) *decodifica5* en su mapa de puertos (PORT MAP, línea 25). VHDL no permite utilizar puertos de salida como entradas. La declaración del tipo de datos BUFFER proporciona un puerto que puede usarse tanto para entrada como para salida. El módulo mod5 se declara en las líneas 10 a 15 y el módulo decodifica5 se declara en las líneas 16 a 21. Las descripciones ENTITY/ARCHITECTURE de mod5 y decodifica5 pueden incluirse dentro del archivo de diseño de nivel superior, o en su defecto pueden guardarse en la misma carpeta que el archivo de nivel superior, como lo hicimos aquí. La instrucción PORT MAP para cada instancia de los módulos se lista en las líneas 23 a 25. La palabra a la izquierda de los dos puntos es una etiqueta única para cada instancia y el nombre del módulo va a la derecha, después las palabras clave PORT MAP y por último, entre paréntesis, van las asociaciones con nombre entre las señales y los puertos del diseño. El

```

1  ENTITY mod5decodifica1 IS
2  PORT (
3      clk                :IN BIT;
4      q                  :BUFFER BIT_VECTOR (2 DOWNTO 0);
5      cntr_estado        :OUT BIT_VECTOR (0 TO 4)
6  );
7  END mod5decodifica1
8
9  ARCHITECTURE nivelsuperior OF mod5decodifica1 IS
10 COMPONENT mod5
11     PORT (
12         reloj           :IN BIT;
13         q               :OUT BIT_VECTOR (2 DOWNTO 0)
14     );
15 END COMPONENT;
16 COMPONENT decodifica5
17     PORT (
18         c, b, a         :IN BIT;
19         estado          :OUT BIT_VECTOR (0 TO 4)
20     );
21 END COMPONENT;
22 BEGIN
23     contador:          mod5      PORT MAP (reloj => clk, q => q);
24     decodificador:    decodifica5  PORT MAP
25         (c => q(2), b => q(1), a => q(0), estado => cntr_estado);
26 END nivelsuperior;

```

FIGURA 7-49 Archivo en VHDL de mayor nivel para conectar mod5 y decodifica5 entre sí.

operador => indica qué puertos del módulo (a la izquierda) están conectadas con cuáles señales del sistema de mayor nivel (a la derecha). Este circuito produce los resultados de la simulación que se muestran en la figura 7-47.

Contador BCD MOD-100

Deseamos diseñar un contador BCD MOD-100 reciclable que tenga una entrada *borrar* asíncrona. La manera más sencilla de hacerlo es crear un módulo de contador BCD MOD-10 y conectar en cascada de manera síncrona dos de estos módulos entre sí, en un archivo de diseño de mayor nivel. Las entradas de reloj para los dos módulos MOD-10 estarán conectadas al reloj del sistema para lograr la conexión en cascada síncrona de los dos módulos de contador. Recuerde que hay beneficios considerables en cuanto al uso del diseño de contadores síncronos en vez de las técnicas asíncronas de sincronización por reloj. Además, si no empleáramos la sincronización por reloj en forma síncrona, la señal borrar síncrona no funcionaría en forma apropiada. Aún y cuando las especificaciones de diseño no requieren una señal de habilitación de conteo o la detección del conteo terminal para el contador MOD-100, será necesario incluir estas características en nuestro diseño. Para poder conectar en cascada en forma síncrona dos contadores, se necesitarán las características de habilitación y decodificación. La entrada de habilitación de conteo hace que el contador ignore los flancos del reloj hasta que esté habilitado. La salida de conteo terminal indica que la secuencia de conteo ha llegado a su límite y volverá a empezar en el siguiente pulso de reloj. Para conectar en cascada las etapas de contadores en forma síncrona, la salida de conteo terminal se conecta a la entrada de habilitación de la siguiente etapa de mayor orden. Al utilizar la entrada de habilitación de conteo para controlar también la decodificación del conteo terminal, podemos usar nuestro módulo MOD-10 para crear contadores BCD aún más grandes.

CONEXIÓN EN CASCADA DE CONTADORES BCD EN AHDL

El bloque SUBDESIGN de nuestro contador BCD MOD-10 se muestra en la figura 7-50. El estado terminal para un contador BCD es 9. Las líneas 10 a 13 detectarán

```

1  SUBDESIGN  mod10
2  (
3      reloj, habilita, borrar          :INPUT;
4      contador[3..0], tc              :OUTPUT;
5  )
6  VARIABLE
7      contador[3..0]                  :DFF;
8  BEGIN
9      contador[ ].clk = reloj;
10     IF contador[ ].q == 9 & habilita == VCC THEN
11         tc = VCC;                    -- detecta el conteo terminal
12     ELSE tc = GND;
13     END IF;
14     IF borrar THEN
15         contador[ ].d = B"0000";    -- borrar síncrona
16     ELSIF enable THEN                -- borrar tiene prioridad
17         IF contador[ ].q == 9 THEN -- comprueba el último estado
18             contador[ ].d = B"0000";
19         ELSE
20             contador[ ].d = contador[ ].q + 1; -- incremento
21         END IF;
22     ELSE                             -- retiene el conteo cuando está deshabilitado
23         contador[ ].d = contador[ ].q;
24     END IF;
25 END;
```

FIGURA 7-50 Contador BCD MOD-10 en AHDL.

este estado terminal sólo cuando el contador esté habilitado con un nivel ALTO. Si se aplica un AND al control *habilita* en la función de decodificación se permitirá que dos o más módulos de contador se conecten en cascada de manera síncrona si es necesario, con lo cual nuestro diseño mod10 será más versátil. La función *borrar* operará en forma síncrona en AHDL si se incluye en la instrucción IF, como se muestra en las líneas 14 a 15. Si *borrar* está inactiva, el contador comprobará mediante una instrucción IF anidada en las líneas 17 a 21 si se ha llegado al último estado 9. Después del estado 9 el contador se recicla en forma síncrona a 0. En caso contrario, el contador se incrementará. Si el contador se deshabilita, las líneas 22 a 23 retendrán el valor de conteo actual mediante el proceso de alimentar la salida actual de regreso a la entrada del contador. Esta acción de retención será necesaria en el contador MOD-100 en cascada para que el dígito de los 10s retenga su estado actual mientras que el dígito de los 1s progrese a través de su secuencia de conteo. Una estrategia de diseño apropiada para nosotros sería simular este módulo para determinar si funciona en forma correcta, antes de usarlo en una aplicación de circuito más compleja. De los resultados de la simulación para mod10, que se proporcionan en la figura 7-51, podemos ver que la secuencia de conteo es correcta, la entrada *borrar* es síncrona y tiene prioridad, y *habilita* controla tanto la función de conteo como la salida de decodificación *tc*.

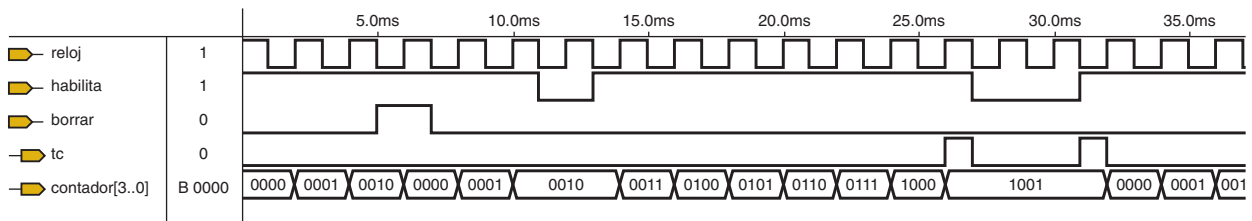


FIGURA 7-51 Resultados de la simulación del contador MOD-10.

Después de crear un símbolo predeterminado para nuestro módulo de contador mod10, podemos ahora dibujar el diagrama de bloques para la aplicación del contador BCD MOD-100. También se han agregado los puertos de entrada, los puertos de salida y el cableado para crear el diseño de la figura 7-52. Observe que las salidas del contador que representan los dígitos de los 1s y los 10s se dibujan como buses. Los módulos mod10 están sincronizados por reloj en forma síncrona. Están conectados en cascada mediante el uso de la salida de conteo terminal del dígito de los 1s para controlar la entrada de habilitación en el dígito de los 10s. El puerto de entrada *en* controla la habilitación/deshabilitación de todo el circuito completo del contador MOD-100. El diseño del contador BCD puede expandirse con facilidad mediante una etapa mod10 adicional, conectando la salida *tc* con la siguiente entrada *habilita* para cada dígito que se necesite. En la figura 7-53 puede verse una muestra de los resultados de la simulación. Ésta nos muestra que el contador MOD-100 tiene una secuencia de conteo BCD correcta y que puede borrarse en forma síncrona.

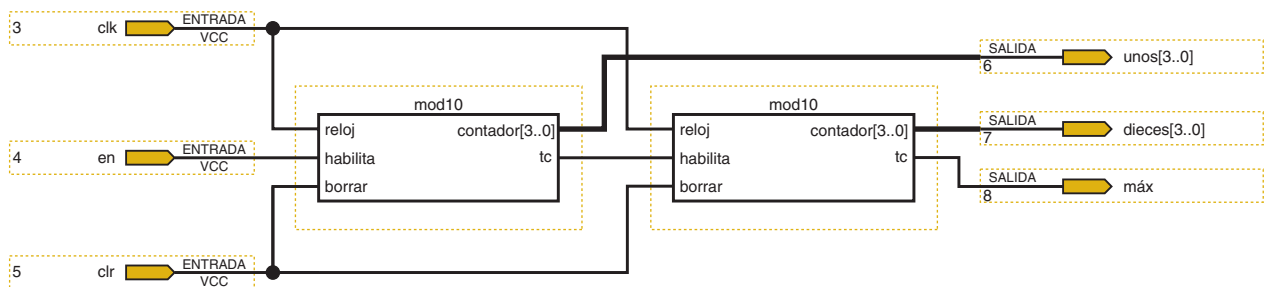


FIGURA 7-52 Diseño de diagrama de bloques para un contador BCD MOD-100.

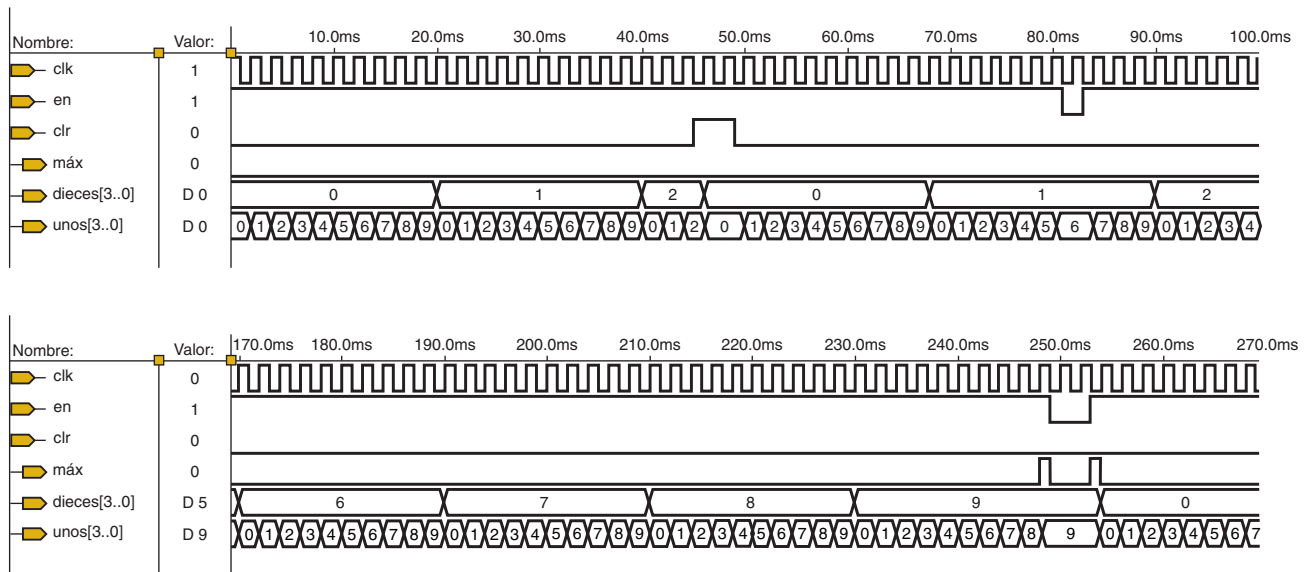


FIGURA 7-53 Resultados de la simulación para el diseño del contador BCD MOD-100.

CONEXIÓN EN CASCADA DE CONTADORES BCD EN AHDL

Los bloques ENTITY y ARCHITECTURE para nuestro contador BCD MOD-10 se muestran en las líneas 26 a 51 de la figura 7-54. El estado terminal para un contador BCD es 9. Las líneas 38 a 40 detectarán este estado terminal sólo cuando el contador se habilite con un nivel ALTO. Si se aplica un AND al control *habilita* en la función de decodificación se permitirá que se conecten en cascada más de dos módulos de contador en forma síncrona si es necesario; además, nuestro diseño mod10 será más versátil. La función *borrar* será síncrona en VHDL si se coloca en la instrucción IF anidada (línea 42) una vez que se detecte el flanco de reloj en la línea 41. Si *borrar* está inactiva, a continuación revisamos si el contador está habilitado (línea 43). Si *habilita* está en ALTO, el contador comprueba mediante otra instrucción IF anidada en las líneas 44 a 46 para ver si se ha llegado al último estado 9. Después del estado 9 el contador se recicla en forma síncrona a 0. En caso contrario se incrementará el conteo. Si el contador está deshabilitado, VHDL retendrá en forma automática el valor de conteo actual. Esta acción de retención será necesaria en el contador MOD-100 en cascada para que el dígito de los 10s retenga su estado actual mientras que el dígito de los 1s progresa a través de su secuencia de conteo. Una estrategia de diseño apropiada para nosotros sería que simuláramos este módulo como un bloque ENTITY separado para determinar si funciona en forma correcta, antes de usarlo en una aplicación de circuito más compleja. Los resultados de la simulación para el bloque ENTITY mod10, que se muestran en la figura 7-51, muestran que la secuencia de conteo es correcta, que la entrada *borrar* es síncrona y tiene prioridad, y que *habilita* controla tanto la función de conteo como la entrada de decodificación.

Tenemos dos opciones para implementar el contador MOD-10. Una técnica es representar el diseño en forma gráfica en un diagrama de bloques, como puede verse en la figura 7-52. También hemos agregado los módulos del contador mod10, los puertos de entrada, de salida y el cableado para crear el contador MOD-100. Observe que las salidas del contador que representan los dígitos de los 1s y los 10s se dibujan como buses. Los módulos mod10 se sincronizan por reloj en forma síncrona. Están conectados en cascada mediante el uso de la salida de conteo terminal del dígito de los 1s para controlar la entrada de habilitación en el dígito de los 10s. El puerto de entrada *en* controla la habilitación/deshabilitación de todo el circuito completo del contador MOD-100. El diseño del contador BCD puede expandirse con facilidad mediante una etapa mod10 adicional, conectando la salida *tc* con la


```

1  ENTITY mod100 IS
2  PORT (
3      clk, en, clr                                :IN BIT;
4      unos                                         :OUT INTEGER RANGE 0 TO 15;
5      dieces                                       :OUT INTEGER RANGE 0 TO 15;
6      max                                          :OUT BIT
7  );
8  END mod100;
9  ARCHITECTURE nivelsuperior OF mod100 IS
10 COMPONENT mod10
11     PORT (
12         reloj, habilita, borrar                 :IN BIT;
13         q                                         :OUT INTEGER RANGE 0 TO 15;
14         tc                                       :OUT BIT;
15     );
16 END COMPONENT;
17 SIGNAL rco                                     :BIT;
18 BEGIN
19     digito1:  mod10  PORT MAP (reloj => clk, habilita => en,
20                               borrar => clr, q => unos, tc => rco);
21     digito2:  mod10  PORT MAP (reloj => clk, habilita => rco,
22                               borrar => clr, q => dieces, tc => max);
23 END nivelsuperior;
24
25
26 ENTITY mod10 IS
27 PORT (
28     reloj, habilita, borrar                     :IN BIT;
29     q                                             :OUT INTEGER RANGE 0 TO 15;
30     tc                                           :OUT BIT
31 );
32 END mod10;
33 ARCHITECTURE bloqinf OF mod10 IS
34 BEGIN
35     PROCESS (reloj, habilita)
36         VARIABLE contador                       :INTEGER RANGE 0 TO 15;
37     BEGIN
38         IF ((contador = 9) AND (habilita = '1')) THEN tc <= '1';
39         ELSE tc <= '0';
40         END IF;
41         IF (reloj'EVENTO AND reloj = '1') THEN
42             IF (borrar = '1') THEN contador := 0;
43             ELSIF (habilita = '1') THEN
44                 IF (contador = 9) THEN contador := 0;
45                 ELSE
46                     contador := contador + 1;
47                 END IF;
48             END IF;
49             q <= contador;
50         END PROCESS;
51 END bloqinf;

```

FIGURA 7-54 Contador BCD MOD-100 en VHDL.

siguiente entrada *habilita* para cada dígito que se necesite. En la figura 7-53 puede verse una muestra de los resultados de la simulación.

La simulación muestra que el contador MOD-100 tiene una secuencia de conteo BCD correcta y puede borrarse en forma síncrona.

La segunda técnica para crear el contador MOD-100 es hacer las conexiones necesarias entre los módulos de diseño, describiendo la estructura del circuito con VHDL. La figura 7-54 muestra el listado para el archivo de diseño de este sistema. La descripción ENTITY/ARCHITECTURE para el subbloque mod10 está conteni-

da dentro del archivo de diseño mod100 general (pero podría estar en un archivo separado dentro de la carpeta de este proyecto). El archivo de diseño mod100 sería el nivel superior para el diseño jerárquico de este sistema. Contiene subbloques de nivel inferior, los cuales son en realidad dos copias del contador mod10 de nivel inferior. El bloque COMPONENT de mod10 se declara en este archivo de diseño de nivel superior (líneas 10 a 16). Las conexiones del cableado para cada instancia en la que se utiliza el módulo se listan en una instrucción PORT MAP. Como necesitamos dos instancias de mod10, hay una instrucción PORT MAP para cada instancia (líneas 19 a 20 y 21 a 22). Cada instancia debe tener una etiqueta única (*digito1* o *digito2*) para diferenciarlos una de la otra. Las instrucciones PORT MAP contienen asociaciones con nombre entre los puertos del módulo de nivel inferior, las cuales se dan a la izquierda, y las señales de nivel superior a las cuales están conectadas, se dan a la derecha. Este circuito produce los mismos resultados de simulación que se muestran en la figura 7-53.

PREGUNTAS DE REPASO

1. Describa cómo conectar módulos de HDL entre sí para crear un sistema digital.
2. ¿Qué es un bus y cómo se representa en un archivo de diseño de diagrama de bloques gráfico en Altera?
3. ¿Qué características de contador deben incluirse para conectar en cascada módulos de contador entre sí, en forma síncrona?

7-14 MÁQUINA DE ESTADOS

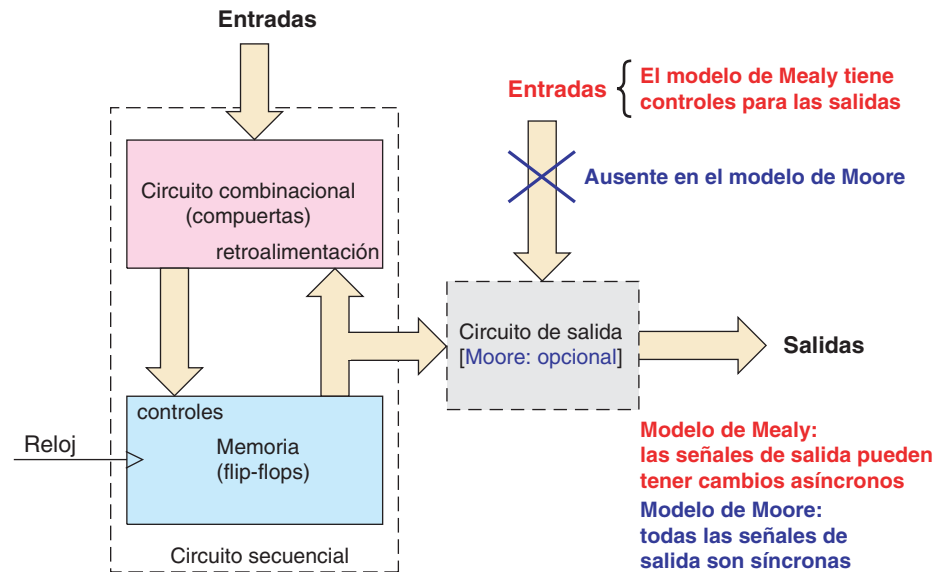
El término **máquina de estados** se refiere a un circuito que progresa en secuencia a través de un conjunto de estados predeterminados, controlado por un reloj y por otras señales de entrada. Por lo tanto, los circuitos tipo contador que hemos estudiado hasta ahora en el capítulo 7 son máquinas de estado. En general, utilizamos el término *contador* para los circuitos secuenciales que tienen una secuencia de conteo numérica regular. Pueden contar hacia arriba o hacia abajo, pueden tener módulos completos de 2^N o pueden tener un módulo $<2^N$, o pueden reciclarse o detenerse en forma automática en cierto estado predeterminado. Como su nombre lo indica, un contador se utiliza para contar cosas. Las cosas que se cuentan en realidad se conocen como pulsos de reloj, pero éstos pueden representar muchos tipos de eventos. Los pulsos pueden ser los ciclos de una señal para la división de frecuencia o pueden ser segundos, minutos y horas de un día para un reloj digital. Pueden indicar que un elemento se ha desplazado por el transportador en una fábrica o que un automóvil ha pasado por cierto punto en la autopista.

El término *máquina de estados* se utiliza con más frecuencia para describir otros tipos de circuitos secuenciales. Pueden tener un patrón de conteo irregular tal como nuestro circuito de control de motores a pasos de la sección 7-10. El objetivo para ese diseño fue controlar un motor a pasos, de manera que pudiera girar en pasos angulares precisos. El circuito de control tenía que producir la secuencia específica requerida de estados para ese movimiento, en lugar de contar en forma numérica. También existen muchas aplicaciones en las que no nos importa el valor binario específico para cada estado, ya que utilizaremos la lógica de decodificación apropiada para identificar estados específicos de interés y para generar las señales de salida deseadas. La distinción general entre los dos términos es que un *contador* se utiliza, por lo general, para contar eventos, mientras que una *máquina de estados* se utiliza por lo común para controlar eventos. El término descriptivo correcto depende de la manera en que deseamos utilizar el circuito secuencial.

El diagrama de bloques que se muestra en la figura 7-55 puede representar una máquina de estados o un contador. En la sección 7-10 vimos que el proceso de diseño clásico de un circuito secuencial era averiguar cuántos flip-flops se requerían y después determinar el circuito combinacional necesario para producir la secuencia

FIGURA 7-55

Diagrama de bloques para contadores y máquinas de estados.



deseada. La salida producida por un contador o una máquina de estados puede venir en forma directa de las salidas de los flip-flops o tal vez se requiera de algún circuito de compuertas, como se indica en el diagrama de bloques. Las dos variaciones se describen ya sea como un **modelo de Mealy** para un circuito secuencial o como un **modelo de Moore**. En el modelo de Mealy las señales de salida también se controlan mediante señales de entrada adicionales, mientras que el modelo de Moore no tiene controles externos para las señales de salida generadas. La salida de Moore es función sólo del estado actual del flip-flop. Un ejemplo de diseño tipo Moore sería el circuito MOD-5 decodificado de la sección 7-13. Por otro lado, el diseño del contador BCD en la misma sección sería un diseño tipo Mealy, debido a la entrada externa (*habilita*) que controla la salida de decodificación del estado terminal (*tc*). Una consecuencia considerable de esta sutil variación del diseño es que las salidas de un circuito tipo Moore serán por completo sincrónicas con el reloj del circuito, mientras que las salidas producidas por un circuito tipo Mealy pueden cambiar en forma asíncrona. En nuestro diseño MOD-10, la entrada de habilitación no está sincronizada con el reloj del sistema.

Desde luego que los HDLs pueden hacer que las máquinas de estado sean fáciles e intuitivas de describir. Como un ejemplo simplificado en extremo con el que todos pueden relacionarse, la siguiente descripción de hardware trata con cuatro estados, a través de los cuales podría progresar una máquina lavadora ordinaria. Aunque una máquina lavadora real es más compleja que este ejemplo, servirá para demostrar las técnicas. Esta máquina lavadora está inactiva hasta que se oprime el botón de arranque, después se llena con agua hasta que la tina está llena y luego opera el agitador hasta que un temporizador expira; por último gira la tina hasta que el agua se avienta hacia fuera y después de eso regresa a la inactividad. El punto de este ejemplo se enfoca en el uso de un conjunto de estados con nombre para los cuales no hay valores binarios definidos. El nombre de la variable contador es *lavar*, la cual puede estar en cualquiera de los siguientes estados con nombre: *inactiva*, *llenar*, *agitar* o *girar*.

MÁQUINA DE ESTADOS SIMPLE EN AHDL

El código de AHDL en la figura 7-56 muestra la sintaxis para declarar un contador con estados con nombre en las líneas 6 y 7. El nombre de este contador es *ciclo*. La palabra clave **MACHINE** se utiliza en AHDL para definir a *ciclo* como una máquina de estado. El número de bits necesarios para que este contador produzca los estados con nombre se determinará mediante el compilador. Observe que en la línea 7

FIGURA 7-56 Ejemplo de una máquina de estados mediante el uso de AHDL.

```

1  SUBDESIGN fig7 56
2  ( reloj, arranque, llena, tiempoagotado, secar :INPUT;
3    valvula agua, modo ag, modo gi :OUTPUT;
4  )
5  VARIABLE
6  ciclo: MACHINE
7    WITH STATES (inactiva, llenar, agitar, girar);
8  BEGIN
9  ciclo.clk = reloj;
10
11  CASE ciclo IS
12    WHEN inactiva => IF arranque THEN ciclo = llenar;
13                      ELSE ciclo = inactiva;
14                      END IF;
15    WHEN llenar => IF llena THEN ciclo = agitar;
16                      ELSE ciclo = llenar;
17                      END IF;
18    WHEN agitar => IF tiempoagotado THEN ciclo = girar;
19                      ELSE ciclo = agitar;
20                      END IF;
21    WHEN girar => IF secar THEN ciclo = inactiva;
22                      ELSE ciclo = girar;
23                      END IF;
24    WHEN OTHERS => ciclo = inactiva;
25  END CASE;
26
27  TABLE
28    ciclo      => valvula agua, modo ag, modo gi;
29    inactiva   => GND, GND, GND;
30    llenar     => VCC, GND, GND;
31    agitar     => GND, VCC, GND;
32    girar      => GND, GND, VCC;
33  END TABLE;
34  END;

```

los estados tienen nombre, pero el valor binario para cada estado también se deja para que el compilador lo determine. El diseñador no tiene que preocuparse por este nivel de detalle. La estructura CASE en las líneas 11 a 25 y la lógica de decodificación que controla las salidas (líneas 27 a 33) hace referencia a los estados por su nombre. Esto hace que la descripción sea fácil de leer y permite al compilador más libertad para minimizar los circuitos. Si el diseño requiere que la máquina de estado también se conecte a un puerto de salida, entonces la línea 6 se puede cambiar por:

```
ciclo: MACHINE OF BITS (st [1..0])
```

y el puerto de salida *st[1..0]* puede agregarse a la sección SUBDESIGN. Una segunda opción de máquina de estado disponible es la habilidad para que el diseñador defina un valor binario para cada estado. En este ejemplo, esto puede lograrse si se cambia la línea 7 por:

```
WITH STATES (inactiva = B"00", llenar = B"01", agitar = B"11", girar = B"10");
```

MÁQUINA DE ESTADOS SIMPLE EN VHDL

El código de VHDL en la figura 7-57 muestra la sintaxis para declarar un contador con estados con nombre. En la línea 6 se declara un objeto llamado *maquina_estados*. Observe la palabra clave TYPE. A esto se le conoce como un **tipo enumerado** en VHDL, en el cual el diseñador lista por nombres simbólicos todos los posibles valores que una señal, variable o puerto que se declara de ese tipo tiene permitido tener. Observe también que en la línea 6 los estados tienen nombre, pero el valor binario para cada estado se deja para que el compilador lo determine. El diseñador no necesita preocuparse por este nivel de detalle. La estructura CASE en las líneas 12 a 29 y la lógica de decodificación que controla las salidas (líneas 31 a 36) hacen referencia a los estados por su nombre. Esto hace que la descripción sea fácil de leer y permite al compilador más libertad para minimizar los circuitos.

Al utilizar el simulador para verificar nuestros diseños de HDL se producen los resultados que muestra la figura 7-58. El simulador de Altera permite también simular nodos intermedios en nuestros módulos de diseño. La máquina de estado “oculta” llamada *ciclo* se ha incluido en la simulación para poder confirmar que opere en forma correcta. Observe que los resultados para *ciclo* se proporcionan dos veces, ya que se mostrará en forma distinta para los dos HDLS. En realidad el

```

1  ENTITY fig7_57 IS
2  PORT( reloj, arranque, llena, tiempoagotado, secar           :IN BIT;
3        valvula_agua, modo_ag, modo_gi                       :OUT BIT);
4  END fig7_57;
5  ARCHITECTURE vhd1 OF fig7_57 IS
6  TYPE maquina_estados IS (inactiva, llenar, agitar, girar);
7  BEGIN
8      PROCESS (reloj)
9      VARIABLE ciclo :maquina_estado;
10     BEGIN
11         IF (reloj'EVENT AND reloj = '1') THEN
12             CASE ciclo IS
13                 WHEN inactiva =>
14                     IF arranque = '1' THEN          ciclo := llenar;
15                     ELSE                                ciclo := inactiva;
16                     END IF;
17                 WHEN llenar =>
18                     IF llena = '1' THEN              ciclo := agitar;
19                     ELSE                                ciclo := llenar;
20                     END IF;
21                 WHEN agitar =>
22                     IF tiempoagotado = '1' THEN      ciclo := girar;
23                     ELSE                                ciclo := agitar;
24                     END IF;
25                 WHEN girar =>
26                     IF secar = '1' THEN              ciclo := inactiva;
27                     ELSE                                ciclo := girar;
28                     END IF;
29             END CASE;
30         END IF;
31         CASE ciclo IS
32             WHEN inactiva => valvula_agua <= '0'; modo_ag <= '0'; modo_gi <= '0';
33             WHEN llenar=> valvula_agua <= '1'; modo_ag <= '0'; modo_gi <= '0';
34             WHEN agitar=> valvula_agua <= '0'; modo_ag <= '1'; modo_gi <= '0';
35             WHEN girar => valvula_agua <= '0'; modo_ag <= '0'; modo_gi <= '1';
36         END CASE;
37     END PROCESS;
38 END vhd1;

```

FIGURA 7-57 Ejemplo de máquina de estados mediante el uso de VHDL.

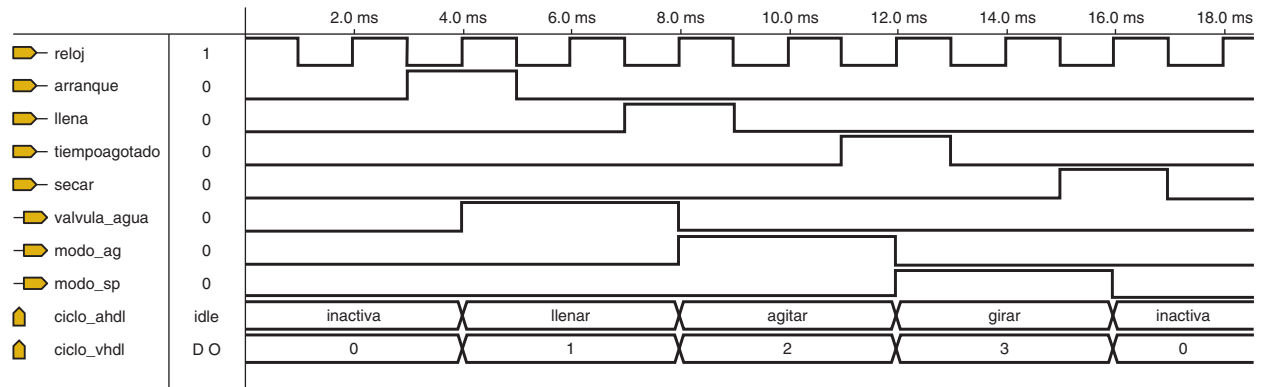


FIGURA 7-58 Simulación del ejemplo de diseño en HDL de la máquina lavadora para una máquina de estados.

simulador no puede mostrar las simulaciones para AHDL y VHDL al mismo tiempo. La información del segundo nodo oculto tan sólo se copió y se pegó para obtener aquí una figura compuesta. En AHDL se muestran los nombres de la máquina de estados, mientras que en VHDL se muestran en su lugar los valores asignados por el compilador para los nombres de los estados enumerados.

Máquina de estados del controlador de un semáforo

Vamos a estudiar el diseño de una máquina de estados de un circuito que es más complicado: un controlador de un semáforo. En la figura 7-59 se muestra el diagrama de bloques. Nuestro controlador está diseñado para controlar el flujo de tráfico en la intersección de un camino principal con un camino lateral menos fluido. El tráfico fluirá sin interrupción en el camino principal con una luz verde, hasta que se detecte un automóvil en el camino lateral (lo cual se indica mediante la entrada etiquetada como *auto*). Después de un lapso que se establece mediante la entrada binaria de cinco bits etiquetada como *tprinverde*, la luz del camino principal cambiará a ámbar. El *tprinverde* asegura que el camino principal reciba una luz verde por al menos ese lapso en

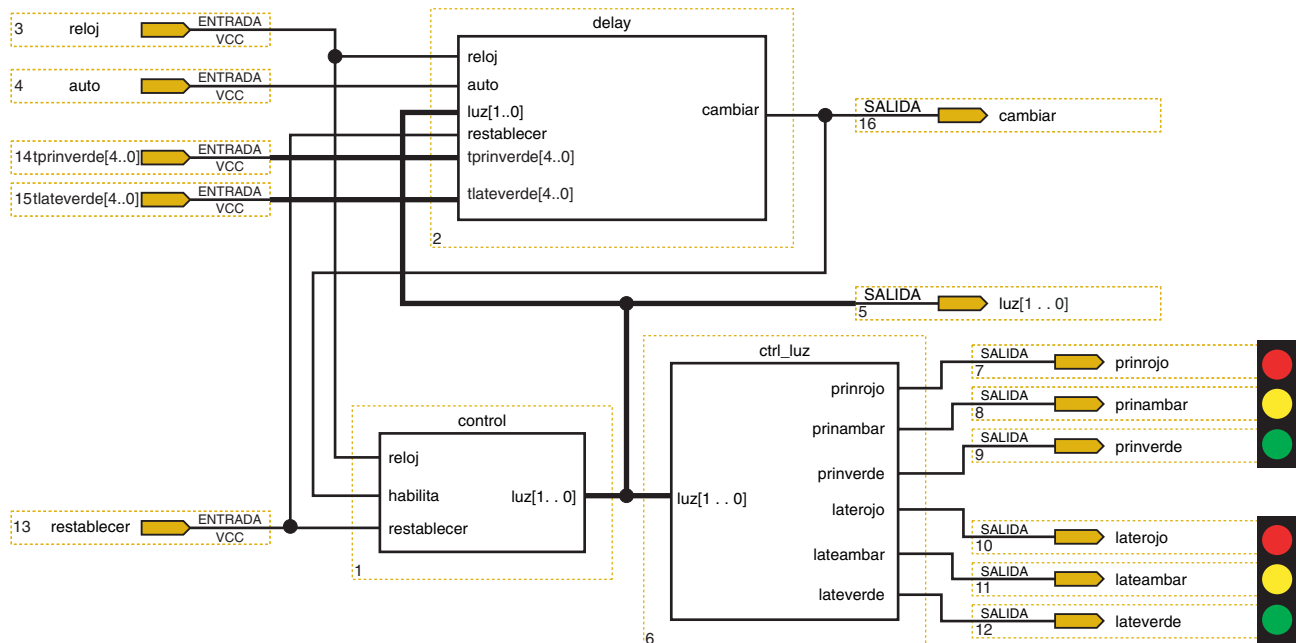


FIGURA 7-59 Controlador de semáforo.

cada ciclo de las luces. La luz ámbar durará una constante de tiempo, el cual se establece en el diseño de HDL y después realizará la transición a rojo. Cuando la luz del camino principal cambie a rojo, la luz del camino lateral cambiará a verde. La luz del camino lateral estará en verde durante un tiempo que se establece mediante la entrada binaria de cinco bits llamada *tlateverde*. De nuevo, la luz ámbar durará la misma constante de tiempo y después el camino lateral regresará a una luz roja y el camino principal tendrá luz verde de nuevo. El módulo generador de lapsos controlará los periodos de tiempo para cada una de las luces. Los lapsos actuales serán iguales al periodo del reloj del sistema multiplicado por el factor de lapso. El módulo de control determina el estado del controlador de tráfico. Hay cuatro combinaciones de luces: principal-verde/lateral-rojo, principal-ámbar/lateral-rojo, principal-rojo/lateral-verde y principal-rojo/lateral-ámbar; por lo tanto, el control requerirá cuatro estados. Los estados del semáforo se traducen en los patrones de encendido-apagado apropiados para cada uno de los seis pares de luces, mediante el módulo *ctrl_luz*. Las salidas etiquetadas como *cambiar* y *luz* se proporcionan para fines de diagnóstico. La pata de *Restablecer* (*Reset*) se utiliza para inicializar cada uno de los dos circuitos secuenciales.

CONTROLADOR DE SEMÁFORO EN AHDL

Los tres módulos de diseño para nuestro controlador de semáforo en AHDL se muestran juntos en la figura 7-60. En realidad son tres archivos de diseño separados que se interconectan con el diseño del diagrama de bloques que se muestra en la figura 7-59. El módulo generador de lapsos (líneas 1 a 23) es en esencia un contador descendente oculto (línea 20) llamado *mach*, el cual espera en cero cuando el camino principal tiene luz verde (*luz* = 0), hasta que se dispara mediante el sensor de autos (línea 13) para cargar el factor de lapso o retraso *tprinverde* - 1 en la línea 14. Como el contador se decrementa hasta cero, se resta uno a cada retraso para hacer que el módulo del contador de retraso sea igual al valor del retraso. Por ejemplo, si deseamos tener un factor de retraso de 25, el contador debe contar en forma descendente desde 24 hasta 0. La longitud actual de tiempo representada por los factores de retraso depende de la frecuencia del reloj. Con una frecuencia de reloj de 1-Hz el periodo sería de 1s, y los factores de retraso estarían en segundos. La línea 22 define una señal de salida llamada *cambio*, la cual detecta cuando *mach* es igual a uno. *Cambio* estará en ALTO para indicar que la condición de prueba es verdadera, lo cual a su vez habilitará la máquina de estados en el módulo de control para que avance a su siguiente estado (*luz* = 1) cuando se sincronice con el reloj, para indicar una luz ámbar en el camino principal. A medida que el contador de retraso *mach* cuente hacia abajo y llegue a cero, la instrucción CASE determinará que *luz* tiene un nuevo valor y se cargará el factor de retraso de tiempo fijo de 5 para una luz ámbar (en realidad se carga uno menos que 5, como dijimos antes) en *mach* (línea 16), en el siguiente pulso de reloj. El conteo descendente continuará a partir de este nuevo tiempo de retraso y *cambio* habilitará de nuevo el módulo de control para que avance a su siguiente estado (*luz* = 2) cuando *mach* sea igual a 1; el resultado será una luz verde para el camino lateral. Cuando *mach* llegue de nuevo a cero se cargará en el contador descendente (línea 17) el retraso de tiempo (*tlateverde* - 1) para una luz verde en el camino lateral. Cuando *cambio* se vuelva a activar, *luz* avanzará al estado 3 para una luz ámbar en el camino lateral. *Mach* se reciclará al valor 4 (5-1) en la línea 18 para el retraso de tiempo fijo para una luz ámbar. Cuando *cambio* se active esta vez, el módulo de control regresará al estado *luz* = 0 (luz verde en el camino principal). Cuando *mach* se decremente a su estado terminal (0) esta vez, las líneas 13 a 15 determinarán mediante el estado de la entrada del sensor *auto* si debe esperar a otro auto o si debe cargar el factor de retraso para una luz verde en el camino principal (*tprinverde* - 1), para empezar el ciclo de nuevo. El camino principal recibirá una luz verde durante al menos esta longitud de tiempo, aunque haya un flujo continuo de automóviles en el camino lateral. Es obvio que podríamos hacer mejoras a este diseño, pero sin duda eso lo complicaría aún más.


```

1  SUBDESIGN retraso
2  ( reloj, auto, luz[1..0], restablecer      :INPUT;
3    tprinverde[4..0], tlateverde[4..0]      :INPUT;
4    cambio                                   :OUTPUT; )
5  VARIABLE
6    mach[4..0]                               :DFF;
7  BEGIN
8    mach[ ].clk = reloj;                      -- con un reloj de 1 Hz, tiempos en segundos
9    mach[ ].crln = restablecer;
10   IF mach[ ] == 0 THEN
11     CASE luz[ ] IS                          -- comprueba el estado del controlador de luz
12       WHEN 0 =>
13         IF !auto THEN mach[ ].d = 0;        -- espera auto en camino lateral
14         ELSE mach[ ].d = tprinverde[ ] - 1; -- establece tiempo para luz verde principal
15         END IF;
16         WHEN 1 => mach[ ].d = 5 - 1;        -- establece tiempo para luz ambar principal
17         WHEN 2 => mach[ ].d = tlateverde[ ] - 1; -- establece tiempo para luz verde lateral
18         WHEN 3 => mach[ ].d = 5 - 1;        -- establece tiempo para luz ambar lateral
19     END CASE;
20     ELSE mach[ ].d = mach[ ].q - 1;          -- decrementa contador temporizador
21     END IF;
22     cambio = mach[ ] == 1;                  -- cambia luces en módulo de control
23   END;
24   -----
25  SUBDESIGN control
26  ( reloj, habilita, restablecer :INPUT;
27    luz[1..0]                   :OUTPUT; )
28  VARIABLE
29    semaforo: MACHINE OF BITS (luz[1..0]) -- necesita 4 estados para combinaciones de luces
30    WITH STATES (pvrđ = B"00", pamb = B"01", lvrđ = B"10", lamb = B"11");
31  BEGIN
32    semaforo.clk = reloj;
33    semaforo.restablecer = !restablecer; -- las máquinas tienen señal restablecer
34                                         -- asíncrona, activa en alto
35    CASE semaforo IS                    -- espera a que señal de habilitación cambie estados del semáforo
36      WHEN pvrđ => IF habilita THEN semaforo = pamb; ELSE semaforo = pvrđ; END IF;
37      WHEN pamb => IF habilita THEN semaforo = lvrđ; ELSE semaforo = pamb; END IF;
38      WHEN lvrđ => IF habilita THEN semaforo = lamb; ELSE semaforo = lvrđ; END IF;
39      WHEN lamb => IF habilita THEN semaforo = pvrđ; ELSE semaforo = lamb; END IF;
40    END CASE;
41  END;
42   -----
43  SUBDESIGN ctrl_luz
44  ( luz[1..0]                               :INPUT;
45    prinrojo, prinambar, prinverde          :OUTPUT;
46    laterojo, lateambar, lateverde          :OUTPUT; )
47  BEGIN
48    CASE luz[ ] IS                      -- determina cuales luces se van a encender
49      WHEN B"00" => prinverde = VCC; prinambar = GND; prinrojo = GND;
50                   lateverde = GND; lateambar = GND; laterojo = VCC;
51      WHEN B"01" => prinverde = GND; prinambar = VCC; prinrojo = GND;
52                   lateverde = GND; lateambar = GND; laterojo = VCC;
53      WHEN B"10" => prinverde = GND; prinambar = GND; prinrojo = VCC;
54                   lateverde = VCC; lateambar = GND; laterojo = GND;
55      WHEN B"11" => prinverde = GND; prinambar = GND; prinrojo = VCC;
56                   lateverde = GND; lateambar = VCC; laterojo = GND;
57    END CASE;
58  END;

```

FIGURA 7-60 Archivos de diseño de AHDL para el controlador de semáforo.

El módulo de control (líneas 25 a 40) contiene una máquina de estados llamada *semaforo*, la cual avanzará en secuencia a través de los cuatro estados para las combinaciones de semáforos. Los bits para la máquina de estados tienen nombre y están conectados como un puerto de salida para este módulo (líneas 27 y 29). Los cuatro estados para *semaforo* se llaman *pvr*, *pamb*, *svr* y *samb* en la línea 30. Cada estado representa cuál de los caminos (principal o lateral) va a recibir una luz verde o ámbar. El otro camino tendrá una luz roja. También se han especificado los valores para cada estado del módulo de control en la línea 30, de manera que podamos identificarlos como entradas para los otros dos módulos, *retraso* y *ctrl_luz*. La entrada *habilita* está conectada a la señal de salida *cambio*, producida por el módulo de retraso. Cuando se habilite, la máquina de estados *semaforo* avanzará al siguiente estado cuando se aplique un pulso de reloj, como se describe mediante la instrucción CASE y las instrucciones IF anidadas en las líneas 34 a 39. En caso contrario, *semaforo* permanecerá en el estado actual.

El módulo *ctrl_luz* (líneas 42 a 57) tiene como entrada *luz[1..0]*, la cual representa el estado de la máquina de estados *semaforo* del módulo de control, y producirá como salida las señales que enciendan las combinaciones apropiadas de luces verde, ámbar y roja para los caminos principal y lateral. En realidad, cada una de las salidas del módulo *ctrl_luz* estará conectada a circuitos controladores de lámparas para controlar los voltajes y las corrientes con valores más altos, necesarios para los focos reales en un semáforo. La instrucción CASE en las líneas 47 a 55 determina la combinación de luces del (camino principal, camino lateral) que se debe encender para cada estado del *semaforo*. La función del módulo *ctrl_luz* es muy parecida a la de un decodificador. En esencia, decodifica la combinación de cada estado de *luz* para encender una luz verde o ámbar para un camino, y una luz roja para el otro. Para cada estado de entrada se produce una combinación de salida única.

CONTROLADOR DE SEMÁFORO EN VHDL

En la figura 7-61 se muestra el diseño de VHDL para el controlador de semáforo. El nivel superior del diseño se describe en forma estructural, en las líneas 1 a 34. Hay que declarar tres módulos COMPONENT (líneas 10 a 24). En las líneas 26 a 33 aparecen las instrucciones PORT MAP, las cuales proporcionan las interconexiones de cableado entre cada módulo y el diseño de nivel superior.

En esencia, el módulo de retraso (líneas 36 a 66) es un contador descendente oculto (línea 59), el cual se crea con la variable entera *mach* que espera en cero cuando el camino principal tiene luz verde (*luz* = “00”), hasta que se dispara mediante el sensor de autos (línea 52) para cargar el factor de retraso *tprverde* – 1 en la línea 53. Como el contador se decrementa hasta llegar a cero, se resta uno a cada factor de retraso para hacer que el módulo del contador de retraso sea igual al valor del factor de retraso. Por ejemplo, si deseamos tener un factor de retraso de 25, el contador debe contar en forma descendente desde 24 hasta 0. La longitud actual de tiempo representada por los factores de retraso depende de la frecuencia del reloj. Con una frecuencia de reloj de 1-Hz el periodo sería de 1 s, y los factores de retraso estarían en segundos. Las líneas 62-64 definen una señal de salida llamada *cambio*, la cual detecta cuando *mach* es igual a uno. *Cambio* estará en ALTO para indicar que la condición de prueba es verdadera, lo cual a su vez habilitará la máquina de estado en el módulo de control para que avance a su siguiente estado (*luz* = “01”) cuando se sincronice con el reloj, para indicar una luz ámbar en el camino principal. Ahora cuando *mach* llegue a cero, la instrucción CASE determinará que *luz* tiene un nuevo valor y se cargará el factor de retraso de tiempo fijo de 5 para una luz ámbar (en realidad se carga uno menos, como dijimos antes) en *mach* (línea 55), en el siguiente pulso de reloj. El conteo descendente continuará a partir de este nuevo tiempo de retraso y *cambio* habilitará de nuevo el módulo de control para que avance a su siguiente estado (*luz* = “10”), con lo cual se obtendrá una luz verde para el camino lateral. Cuando *mach* llegue de nuevo a cero se cargará

```

1  ENTITY trafico IS
2  PORT (   reloj, auto, restablecer           :IN BIT;
3          tprinverde, rlateverde             :IN INTEGER RANGE 0 TO 31;
4          luz                                 :BUFFER INTEGER RANGE 0 TO 3;
5          cambio                             :BUFFER BIT;
6          prinrojo, prinambar, prinverde      :OUT BIT;
7          latrojo, latambar, latverde         :OUT BIT);
8  END trafico;
9  ARCHITECTURE nivelsuperior OF trafico IS
10 COMPONENT retraso
11     PORT (   reloj, auto, restablecer       :IN BIT;
12            luz                                 :IN INTEGER RANGE 0 TO 3;
13            tprinverde, tlateverde           :IN INTEGER RANGE 0 TO 31;
14            cambio                             :OUT BIT);
15 END COMPONENT;
16 COMPONENT control
17     PORT (   reloj, habilita, restablecer    :IN BIT;
18            luz                                 :OUT INTEGER RANGE 0 TO 3);
19 END COMPONENT;
20 COMPONENT ctrl_luz
21     PORT (   luz                             :IN INTEGER RANGE 0 TO 3;
22            prinrojo, prinambar, prinverde    :OUT BIT;
23            latrojo, latambar, latverde       :OUT BIT);
24 END COMPONENT;
25 BEGIN
26 modulo1: retraso PORT MAP ( reloj => reloj, auto => auto, restablecer => restablecer,
27                            luz => luz, tprinverde => tprinverde, tlateverde => tlateverde,
28                            cambio => cambio);
29 modulo2: control PORT MAP (reloj => reloj, habilita => cambio, restablecer => restablecer,
30                            luz => luz);
31 modulo3: ctrl_luz PORT MAP (luz => luz, prinrojo => prinrojo, prinambar => prinambar;
32                            prinverde => prinverde, latrojo => latrojo, latambar => latambar,
33                            latverde => latverde);
34 END nivelsuperior;
35 -----
36 ENTITY retraso IS
37 PORT   ( reloj, auto, restablecer           :IN BIT;
38         luz                                 :IN BIT_VECTOR (1 DOWNTO 0);
39         tprinverde, tlateverde             :IN INTEGER RANGE 0 TO 31;
40         cambio                             :OUT BIT);
41 END retraso;
42 ARCHITECTURE tiempo OF retraso IS
43 BEGIN
44     PROCESS (reloj, restablecer)
45     VARIABLE mach                          :INTEGER RANGE 0 TO 31;
46     BEGIN
47         IF restablecer = '0' THEN mach := 0;
48         ELSIF (reloj = '1' AND reloj'EVENT) THEN -- con reloj de 1 Hz, tiempo en segundos
49             IF mach = 0 THEN
50                 CASE luz IS
51                     WHEN "00"
52                         IF auto = '0' THEN mach := 0; -- espera auto en camino lateral
53                         ELSE mach:= tprinverde - 1; -- establece tiempo para luz verde principal
54                         END IF;
55                     WHEN "01" => mach := 5 - 1; -- establece tiempo para luz ambar principal
56                     WHEN "10" => mach := tlateverde - 1; -- establece tiempo para luz verde lateral
57                     WHEN "11" => mach := 5 - 1: -- establece tiempo para luz ambar lateral
58                     END CASE;
59                 ELSE mach := mach - 1; -- decrementa contador temporizador
60                 END IF;
61             END IF;

```

FIGURA 7-61 Diseño de VHDL para el controlador de semáforo.

```

62     IF mach = 1 THEN cambio <= '1';           -- cambia luces en control
63     ELSE cambio <= '0';
64     END IF;
65     END PROCESS;
66 END tiempo;
67 -----
68 ENTITY control IS
69 PORT (      reloj, habilita, restablecer  :IN BIT;
70         luz                                     :OUT BIT_VECTOR (1 DOWNTO 0) );
71 END control;
72 ARCHITECTURE a OF control IS
73 TYPE enumerado IS (pvrđ, pamb, lvrđ, lamb); -- necesita 4 estados para las
                                           combinaciones de luces
74 BEGIN
75     PROCESS (reloj, restablecer)
76     VARIABLE luces: enumerado;
77     BEGIN
78         IF restablecer = '0' THEN luces := pvrđ;
79         ELSIF (reloj = '1' AND reloj'EVENT) THEN
80             IF habilita = '1' THEN -- espera a habilita para cambiar estados de luces
81                 CASE luces IS
82                     WHEN pvrđ =>      luces := pamb;
83                     WHEN pamb  =>      luces := lvrđ;
84                     WHEN lvrđ  =>      luces := lamb;
85                     WHEN lamb  =>      luces := pvrđ;
86                 END CASE;
87             END IF;
88         END IF;
89         CASE luces IS -- patrones para los estados de las luces
90             WHEN pvrđ=> luz <= "00";
91             WHEN pamb=> luz <= "01";
92             WHEN lvrđ=> luz <= "10";
93             WHEN lamb=> luz <= "11";
94         END CASE;
95     END PROCESS;
96 END a;
97 -----
98 ENTITY ctrl_luz IS
99 PORT (      luz                                     :IN BIT_VECTOR (1 DOWNTO 0);
100         prinrojo, prinambar, prinverde             :OUT BIT;
101         latrojo, latambar, latverde                 :OUT BIT);
102 END ctrl_luz;
103 ARCHITECTURE patrones OF ctrl_luz IS
104 BEGIN
105     PROCESS (luz)
106     BEGIN
107         CASE luz IS -- el estado de control determina cuales luces encender/apagar
108             WHEN "00" => prinverde <= '1'; prinambar <= '0'; prinrojo <= '0';
109                         latverde <= '0'; latambar <= '0'; latrojo <= '1';
110             WHEN "01" => prinverde <= '0'; prinambar <= '1'; prinrojo <= '0';
111                         latverde <= '0'; latambar <= '0'; latrojo <= '1';
112             WHEN "10" => prinverde <= '0'; prinambar <= '0'; prinrojo <= '1';
113                         latverde <= '1'; latambar <= '0'; latrojo <= '0';
114             WHEN "11" => prinverde <= '0'; prinambar <= '0'; prinrojo <= '1';
115                         latverde <= '0'; latambar <= '1'; latrojo <= '0';
116         END CASE;
117     END PROCESS;
118 END patrones;

```

FIGURA 7-61 Continuación

en el contador descendente (línea 56) el retraso de tiempo (*tlateverde* – 1) para una luz verde en el camino lateral. Cuando *cambio* se vuelva a activar, *luz* avanzará al estado “11” para una luz ámbar en el camino lateral. *Mach* se reciclará al valor 4 (5-1) en la línea 57 para el retraso de tiempo fijo para una luz ámbar. Cuando *cam-bio* se active esta vez, el módulo de control regresará a *luz* = “00” (luz verde en el camino principal). Cuando *mach* se decremente a su estado terminal (0) esta vez, las líneas 52 a 54 determinarán mediante el estado de la entrada del sensor *auto* si debe esperar a otro auto o si debe cargar el factor de retraso para una luz verde en el camino principal (*tprinverde* – 1), para empezar el ciclo de nuevo. El camino principal recibirá una luz verde durante al menos esta longitud de tiempo, aún si hay un flujo continuo de automóviles en el camino lateral. Es obvio que podríamos hacer mejoras a este diseño, pero sin duda eso lo complicaría aún más.

El módulo de control (líneas 68 a 96) contiene una máquina de estado llamada *luces*, la cual avanzará en secuencia a través de cuatro estados enumerados para las combinaciones de luces del semáforo. Los cuatro estados enumerados para *luces* son *pvrld*, *pamb*, *lvrd* y *lamb* (líneas 73 y 76). Cada estado representa el camino (principal o lateral) que va a recibir una luz verde o ámbar. El otro camino tendrá una luz roja. La entrada *habilita* se conecta a la señal de salida *cambio* que produce el módulo de retraso. Cuando se habilite, la máquina de estado *luces* avanzará al siguiente estado cuando se sincronice por el reloj, como se describe en las instrucciones IF anidada y CASE de las líneas 79 a 88. En caso contrario, *luces* permanecerá en el estado actual. Los patrones de bits para el puerto de salida *luz* se han especificado para cada estado de *luces* mediante la instrucción CASE en las líneas 89-94, para que podamos identificarlas como entradas para los otros dos módulos: *retraso* y *ctrl_luz*.

El módulo *ctrl_luz* (líneas 98 a 118) tiene como entrada a *luz*, la cual representa el estado de la máquina de estados *luces* del módulo de control, y producirá como salida las señales que enciendan las combinaciones apropiadas de luces verde, ámbar y roja para los caminos principal y lateral. Cada salida del módulo *ctrl_luz* estará conectada a circuitos controladores de lámparas para suministrar los voltajes y corrientes necesarios para los focos en un semáforo real. La instrucción CASE en las líneas 107 a 116, la cual se invoca mediante el bloque PROCESS cuando cambia la entrada *luz*, determina cuál combinación de luces del camino principal/camino lateral deben encenderse para cada estado de *luces*. La función del módulo *ctrl_luz* es muy parecida a la de un decodificador. En esencia, decodifica cada combinación de estado de *luz* para encender una luz verde o ámbar para un camino y una luz roja para el otro. Para cada estado de entrada se produce una combinación de salida única.

En estos momentos tal vez usted se esté preguntando: ¿Por qué hay tantas formas de describir los circuitos lógicos? Si una es más sencilla que las demás, ¿por qué no sólo estudiar esa? Sin duda la respuesta es que cada nivel de abstracción ofrece ciertas ventajas en comparación con los demás, en ciertos casos. El método estructural proporciona el control más completo sobre las interconexiones. El uso de ecuaciones booleanas, tablas de verdad y tablas de estado PRESENTE / estado SIGUIENTE nos permite describir la manera en que fluyen los datos a través del circuito, mediante el uso de HDL. Por último, el método de comportamiento permite una descripción más abstracta de la operación del circuito, en términos de causa y efecto. En la práctica, cada archivo fuente puede tener porciones que se clasifiquen bajo cada uno de los niveles de abstracción. La acción de seleccionar el nivel correcto al escribir código no es una cuestión de bueno o malo, sino de estilo y preferencia.

También hay varias formas de lidiar con cualquier tarea, desde el punto de vista relacionado con la elección de las estructuras de control. ¿Debemos usar asignaciones de señales selectas o ecuaciones booleanas, IF/ELSE o CASE, procesos secuenciales o instrucciones concurrentes, macrofunciones o megafunciones? ¿O deberíamos escribir nuestro propio código? Las respuestas a esas preguntas definirán en última instancia la estrategia personal que usted utilice para resolver el problema. Sus preferencias y las ventajas que encuentre al utilizar un método en comparación con otro se establecerán con la práctica y la experiencia.

PREGUNTAS DE REPASO

1. ¿Cuál es la diferencia fundamental entre un contador y una máquina de estados?
2. ¿Cuál es la diferencia entre describir un contador y describir una máquina de estados en un HDL?
3. Si los estados binarios actuales para una máquina de estados no se definen en el código de HDL, ¿cómo se asignan?
4. ¿Cuál es la ventaja de utilizar la descripción de una máquina de estados?

RESUMEN PARTE 1

1. En los contadores asíncronos (de rizo) la señal de reloj se aplica al FF del LSB y todos los demás FFs se sincronizan mediante la salida del FF anterior.
2. El número MOD de un contador es el número de estados estables en su ciclo de conteo; también es la proporción máxima de división de frecuencia.
3. El número MOD normal (máximo) de un contador es 2^N . Una manera de modificar el número MOD de un contador es agregar circuitos que hagan que se recicle antes de llegar a su último conteo normal.
4. Los contadores pueden conectarse en cascada (encadenarse) para producir rangos de conteo mayores y proporciones de división de frecuencia más altas.
5. En un contador síncrono (en paralelo), todos los FFs se sincronizan en forma simultánea a partir de la señal de reloj de entrada.
6. La frecuencia de reloj máxima para un contador asíncrono ($f_{\text{máx}}$) se decrementa a medida que se incrementa el número de bits. En un contador síncrono $f_{\text{máx}}$ permanece igual, sin importar el número de bits.
7. Un contador de décadas es cualquier contador MOD-10. Un contador BCD es un contador de décadas que avanza en secuencia a través de los 10 códigos BCD (0-9).
8. Un contador preajustable puede cargarse con cualquier conteo inicial que se desee.
9. Un contador ascendente/descendente puede controlarse para contar hacia arriba o hacia abajo.
10. Las compuertas lógicas pueden utilizarse para decodificar (detectar) uno o todos los estados de un contador.
11. La secuencia de conteo para un contador síncrono puede determinarse con facilidad mediante el uso de una tabla de estado PRESENTE / estado SIGUIENTE, en la cual se listen todos los posibles estados, la información de control de entrada de los flip-flops y los estados SIGUIENTE resultantes.
12. Para implementar los contadores síncronos con secuencias de conteo arbitrarias se sigue un procedimiento de diseño estándar.
13. Los contadores pueden describirse de muchas formas distintas mediante el uso de HDL, incluyendo descripciones de cableado estructurales, tablas de estado PRESENTE estado SIGUIENTE y descripciones de comportamiento.
14. Todas las características disponibles en los diversos CIs estándar tipo contador de CI, tales como la carga o el borrado asíncronos o síncronos, la habilitación del conteo y la decodificación del conteo terminal, pueden describirse mediante HDL. Los contadores en HDL pueden modificarse con facilidad para números MOD más altos o cambios en los niveles activos para los controles.
15. Los sistemas digitales pueden subdividirse en módulos o bloques más pequeños, los cuales pueden interconectarse como un diseño jerárquico.
16. Las máquinas de estados pueden representarse en HDL mediante el uso de nombres descriptivos para cada estado, en vez de especificar una secuencia numérica de estados.

TÉRMINOS IMPORTANTES PARTE 1

carga en paralelo	contadores	modelo de Mealy
conexión en cascada	preajustables	modelo de Moore
contador ascendente	contadores síncronos	nivel de abstracción
contador asíncrono	(en paralelo)	de comportamiento
(de rizo)	decodificación	número MOD
contador autocorregible	deformaciones	tabla de estado
contador BCD	diseño de circuitos	PRESENTE, estado
contador de décadas	secuenciales	SIGUIENTE
contador descendente	diseño jerárquico	tabla de excitación
contadores ascendentes/	habilitación	de circuitos
descendentes	de conteo	tabla de excitación J-K
contadores de múltiples	MACHINE	tipo enumerado
etapas	máquina de estados	VARIABLE

PARTE 2

7-15 REGISTROS DE CIRCUITO INTEGRADO

Los diversos tipos de registros pueden clasificarse de acuerdo con la forma en que pueden introducirse datos en el registro para su almacenamiento y la forma en que se envían los datos de salida desde el registro. A continuación se muestra una lista de las diversas clasificaciones.

1. Entrada en paralelo/Salida en paralelo (PIPO).
2. Entrada en serie/Salida en serie (SISO).
3. Entrada en paralelo/Salida en serie (PISO).
4. Entrada en serie/Salida en paralelo (SIPO).

Cada uno de estos tipos y algunas variaciones de los mismos están disponibles en forma de CI para que un diseñador de lógica digital pueda encontrar con exactitud lo que se requiere para cierta aplicación. En las siguientes secciones examinaremos una forma de CI representativa de cada una de las categorías antes mencionadas.

7-16 ENTRADA EN PARALELO/SALIDA EN PARALELO: 74ALS174/74HC174

El registro tipo **entrada en paralelo/salida en paralelo** es un grupo de flip-flops que puede almacenar varios bits al mismo tiempo; en este tipo de registros todos los bits del valor binario almacenado están disponibles de manera directa. La figura 7-62(a) muestra el diagrama lógico para el 74ALS174 (también el 74HC174), un registro de seis bits que tiene las entradas en paralelo D_5 a D_0 y las salidas en paralelo Q_5 a Q_0 . Los datos en paralelo se cargan en el registro, en la PGT de la entrada de reloj CP . Puede utilizarse una entrada maestra MR (Master Reset) para restablecer a 0 en forma asíncrona todos los FFs del registro. La figura 7-62(b) muestra el símbolo lógico para el 74ALS174. Este símbolo se utiliza en los diagramas de circuitos para representar los circuitos de la figura 7-62(a).

Por lo general, el 74ALS174 se utiliza para la transferencia síncrona de datos en paralelo, en donde los niveles lógicos presentes en las entradas D se transfieren a las salidas Q correspondientes cuando se produce una PGT en la entrada de reloj CP . No obstante, este CI puede cablearse para la transferencia de datos en serie, como se mostrará en los siguientes ejemplos.

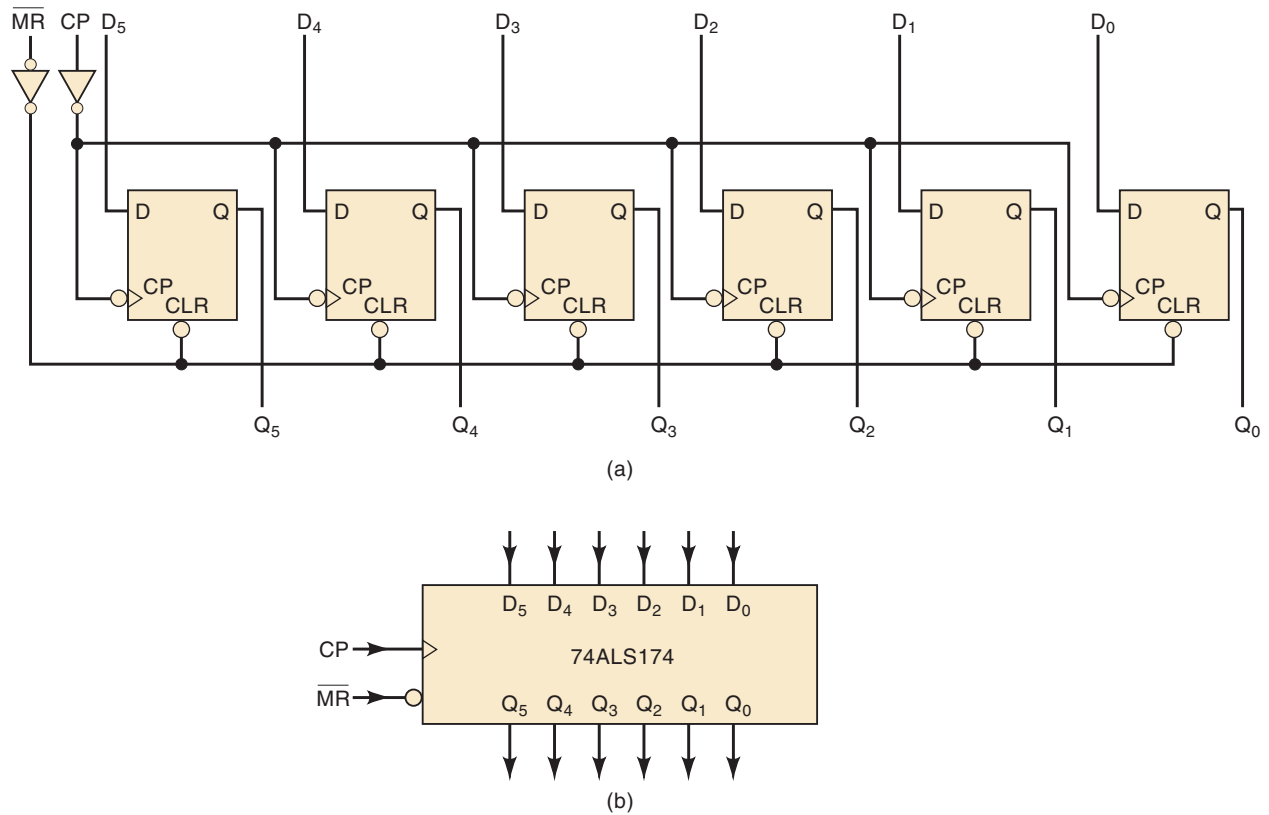


FIGURA 7-62 (a) Diagrama del circuito del 74ALS174; (b) símbolo lógico.

EJEMPLO 7-16

Muestre cómo conectar el 74ALS174 de manera que opere como un registro de desplazamiento en serie, en donde los datos se desplacen en cada PGT de CP como se indica a continuación: Entrada en serie $\rightarrow Q_5 \rightarrow Q_4 \rightarrow Q_3 \rightarrow Q_2 \rightarrow Q_1 \rightarrow Q_0$. En otras palabras, los datos en serie entrarán en D_5 y saldrán en Q_0 .

Solución

Si analizamos la figura 7-62(a) podremos ver que para conectar los seis FFs como un registro de desplazamiento en serie, tenemos que conectar la salida Q de uno a la entrada D del siguiente, de manera que los datos se transfieran de la forma requerida. La figura 7-63 muestra cómo se realiza esto. Observe que los datos se desplazan de izquierda a derecha; los datos de entrada se aplican en D_5 y los datos de salida aparecen en Q_0 .

EJEMPLO 7-17

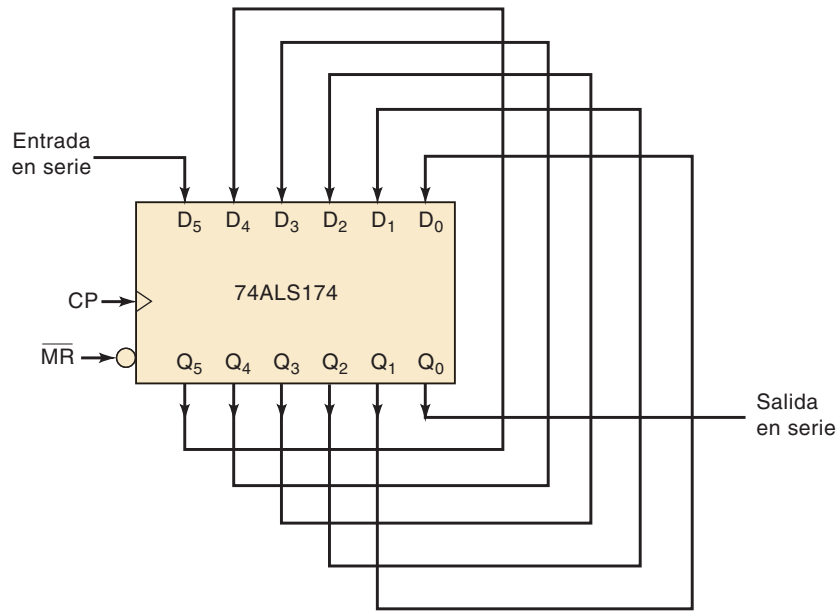
¿Cómo conectaría dos CIs 74ALS174 para operar como un registro de desplazamiento de 12 bits?

Solución

Se conecta un segundo CI 74ALS174 como registro de desplazamiento y se conecta Q_0 del primer CI a D_5 del segundo CI. Se conectan las entradas CP de ambos CIs, de manera que se sincronicen con la misma señal. Además se conectan las entradas MR juntas si se utiliza la señal de restablecer asíncrona (RESET).

FIGURA 7-63

Ejemplo 7-16: el 74ALS174 cableado como un registro de desplazamiento.



7-17 ENTRADA EN SERIE/SALIDA EN SERIE: 74ALS166/74HC166

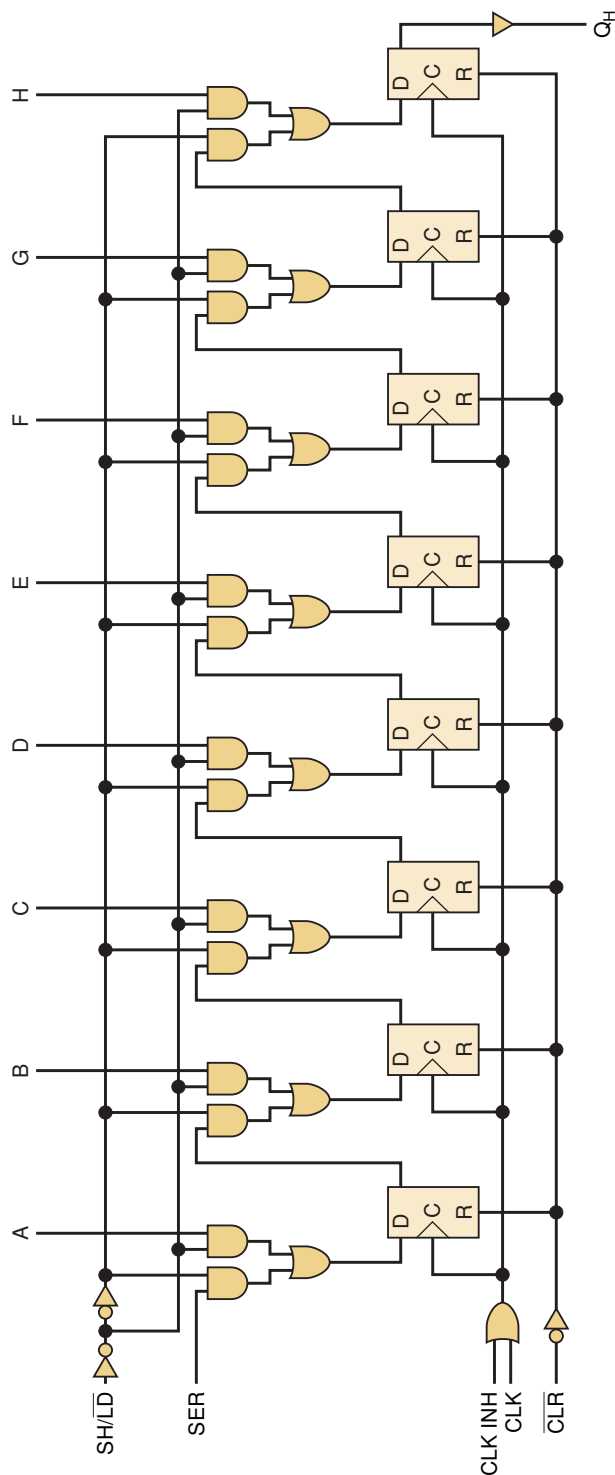
Un registro de desplazamiento del tipo **entrada en serie/salida en serie** cargará los datos un bit a la vez. Los datos se desplazarán bit por bit con cada pulso de reloj, a través del conjunto de flip-flops y hacia el otro extremo del registro. Con los pulsos de reloj continuos, los datos saldrán del registro uno por uno en el mismo orden en el que se cargaron. El 74HC166 (y también el 74ALS166) puede utilizarse como un registro de entrada en serie/salida en serie. La figura 7-64 muestra el diagrama lógico y el símbolo esquemático para el 74HC166. Es un registro de desplazamiento de ocho bits, del cual sólo se puede tener acceso al FF QH. Los datos en serie se introducen en *SER* y se almacenan en el FF QA. La salida en serie se obtiene en el otro extremo del registro de desplazamiento, en *QH*. Como podemos ver en la tabla de funciones de la figura 7-64(c) a este registro de desplazamiento también se le pueden cargar datos en paralelo. Si $SH/LD = 1$ la función del registro será desplazamiento en serie, mientras que un nivel BAJO indicará datos de carga en paralelo a través de las entradas *A – H*. Las funciones síncronas de desplazamiento en serie y carga en paralelo pueden inhibirse (deshabilitarse) mediante la aplicación de un nivel ALTO a la entrada de control *CLK INH*. El registro también tiene una entrada borrar asíncrona (*CLR*) activa en BAJO.

EJEMPLO 7-18

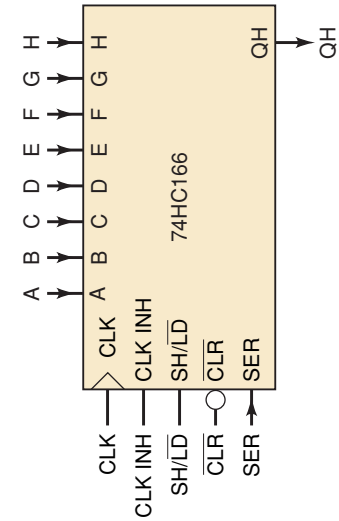
A menudo un registro de desplazamiento se utiliza como una forma de retrasar una señal digital, con base en cierto número de ciclos de reloj. La señal digital se aplica a la entrada en serie del registro de desplazamiento y se desplaza a través de éste mediante pulsos sucesivos de reloj hasta que llega al final del registro de desplazamiento, en donde aparece como la señal de salida. Este método para retrasar el efecto de una señal digital es común en el campo de las comunicaciones digitales. Por ejemplo, la señal digital podría ser la versión digitalizada de una señal de audio que debe retrasarse antes de transmitirla. Las formas de onda de entrada que se muestran en la figura 7-65 se aplican a un 74HC166. Determine la forma de onda de salida resultante.

Solución

QH empieza en nivel BAJO, ya que al principio todos los flip-flops se borran mediante el nivel BAJO que se aplica a la entrada *CLR* asíncrona, en el comienzo del diagrama de sincronización.



(a)



(b)

ENTRADAS		SALIDAS	
		INTERNAS	Q _H
Q _A	Q _B	Q _A	Q _B
L	L	L	L
H	L	Q _{A0}	Q _{B0}
H	L	a	b
H	L	H	Q _{An}
H	L	L	Q _{An}
H	L	Q _{A0}	Q _{B0}
H	L	Q _{A0}	Q _{B0}

(c)

FIGURA 7-64 (a) Diagrama del circuito del 74HC166; (b) símbolo lógico; (c) tabla de funciones.

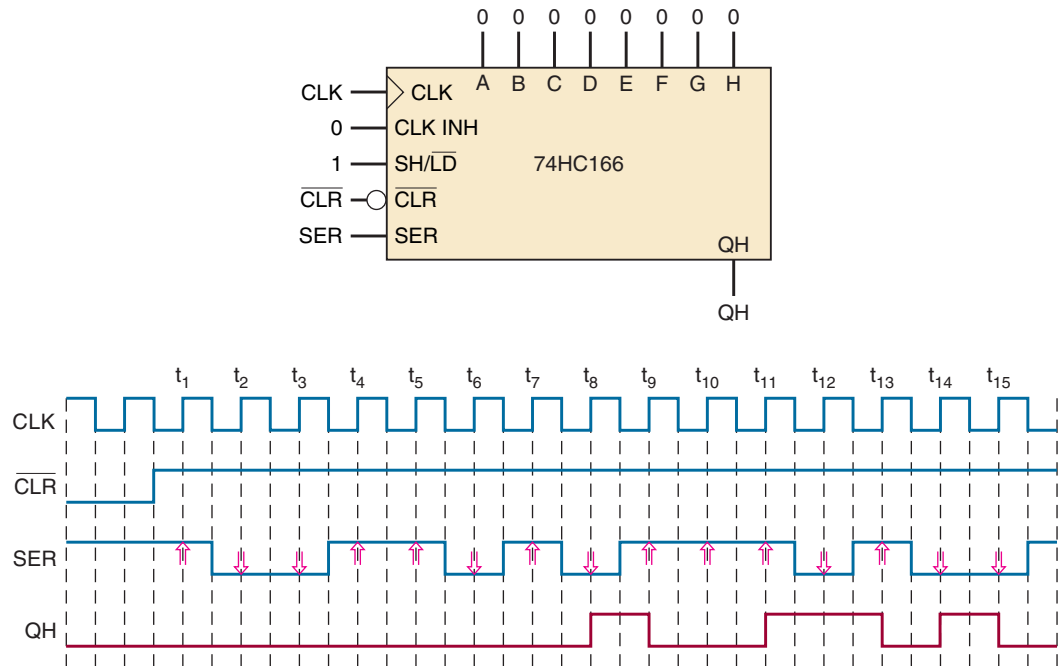


FIGURA 7-65 Ejemplo 7-18.

En t_1 el registro de desplazamiento recibirá como entrada el bit que se está aplicando a *SER* en esos momentos. Éste se almacenará en QA. En t_2 el primer bit se desplazará a QB y se almacenará el segundo bit de *SER* en QA. En t_3 el primer bit se desplazará ahora a QC y el tercer bit de *SER* se almacenará en QA. Por último, el primer bit de entrada de datos aparecerá en la salida QH, en t_8 . Cada bit de entrada sucesivo en *SER* llegará a QH, retrasado por ocho ciclos de reloj.

7-18 ENTRADA EN PARALELO/SALIDA EN SERIE: 74ALS165/74HC165

La figura 7-66(a) muestra el símbolo lógico para el 74HC165. Este CI es un registro de ocho bits, tipo **entrada en paralelo/salida en serie**. En realidad tiene una entrada de datos en serie a través de D_S y una entrada de datos síncrona en paralelo a través de $P_0 - P_7$. El registro contiene ocho FFS (Q_0 a Q_7) conectados en forma interna como un registro de desplazamiento, pero las únicas salidas accesibles de los FFS son Q_7 y $\bar{Q}_7$. CP es la entrada de reloj que se utiliza para la operación de desplazamiento. La entrada de inhibición de reloj $CP\ INH$ se utiliza para inhibir el efecto de la entrada CP . La entrada de desplazamiento/carga, SH/LD , controla qué operación se está llevando a cabo: desplazamiento o carga en paralelo. La tabla de funciones en la figura 7-66(b) muestra cómo las diversas combinaciones de entrada determinan cuál operación se está realizando, en caso de haberla. La carga en paralelo es asíncrona y el desplazamiento en serie es síncrono. La función de desplazamiento en serie siempre será síncrona, ya que el reloj necesita asegurar que los datos de entrada se desplacen sólo un bit a la vez, con cada flanco de reloj apropiado.

EJEMPLO 7-19

Examine la tabla de funciones del 74HC165 y determine: (a) las condiciones necesarias para cargar el registro con datos en paralelo; (b) las condiciones necesarias para la operación de desplazamiento.

FIGURA 7-66

(a) Símbolo lógico para el registro 74HC165 tipo entrada en paralelo/salida en serie; (b) tabla de funciones.

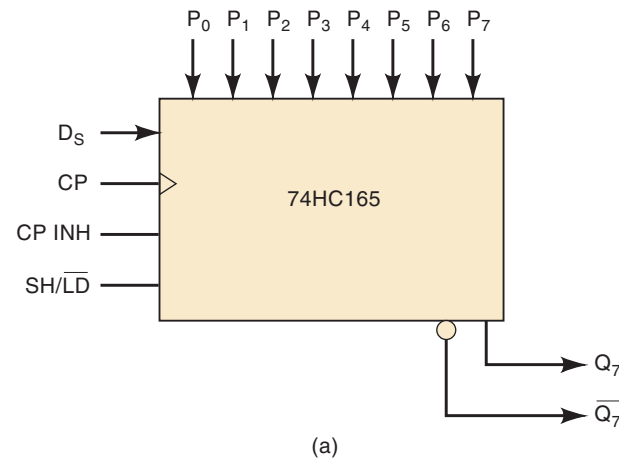


Tabla de funciones

Entradas			Operación	H = nivel alto L = nivel bajo X = no importa ⌫ = PGT
SH/LD	CP	CP INH		
L	X	X	Carga en paralelo	
H	H	X	Sin cambio	
H	X	H	Sin cambio	
H	⌫	L	Desplazamiento	
H	L	⌫	Desplazamiento	

(b)

Solución

- (a) La primera entrada en la tabla muestra que la entrada *SH/LD* tiene que estar en BAJO para la operación de carga en paralelo. Cuando esta entrada está en BAJO, los datos presentes en las entradas *P* se cargan en forma *asíncrona* en los FFs del registro, sin importar las entradas *CP* y *CP INH*. Desde luego que sólo las salidas del último FF están disponibles desde el exterior.
- (b) La operación de desplazamiento no puede llevarse a cabo sino hasta que la entrada *SH/LD* esté en ALTO y se produzca una PGT en *CP* mientras que *CP INH* esté en BAJO [vea la cuarta entrada de la tabla en la figura 7-66(b)]. Un nivel ALTO en *CP INH* inhibirá el efecto de cualquier pulso de reloj. Observe que pueden invertirse los papeles de las entradas *CP* y *CP INH* según lo indica la última entrada en la tabla, ya que dentro del CI se aplica una operación OR entre estas dos señales.

EJEMPLO 7-20

Determine la señal de entrada en Q_7 si conectamos un 74HC165 con $D_S = 0$ y $CPINH = 0$; después aplique las formas de onda de entrada que se muestran en la figura 7-67. $P_0 - P_7$ representan los datos paralelos en $P_0P_1P_2P_3P_4P_5P_6P_7$.

Solución

Hemos dibujado el diagrama de sincronización para los ocho FFs, de manera que se pueda rastrear su contenido en el tiempo, aún y cuando sólo Q_7 estará accesible. La carga en paralelo es síncrona y ocurrirá tan pronto como *SH/LD* cambie a BAJO. Después de que *SH/LD* regrese al nivel ALTO, los datos almacenados en el registro se desplazarán un FF a la derecha (hacia Q_7) con cada PGT en *CP*.

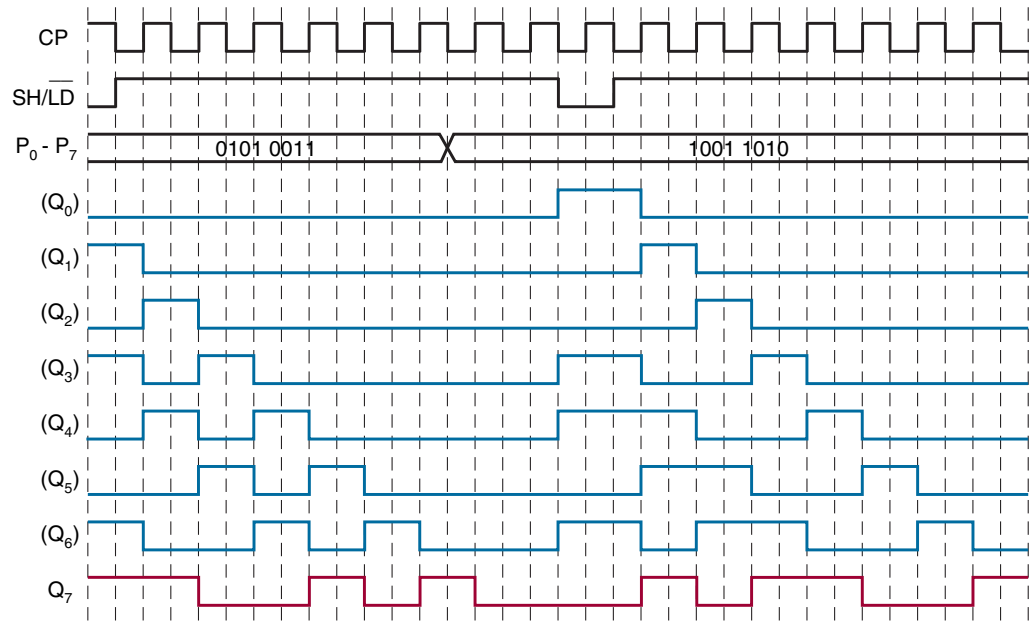


FIGURA 7-67 Ejemplo 7-20.

7-19 ENTRADA EN SERIE/SALIDA EN PARALELO: 74ALS164/74HC164

La figura 7-68(a) muestra el diagrama lógico para el 74ALS164. Es un registro de desplazamiento de ocho bits tipo **entrada en serie/salida en paralelo**, en el cual la salida de cada FF es accesible desde el exterior. En vez de una sola entrada en serie, una compuerta AND combina las entradas A y B para producir la entrada serial para el flip-flop Q_0 .

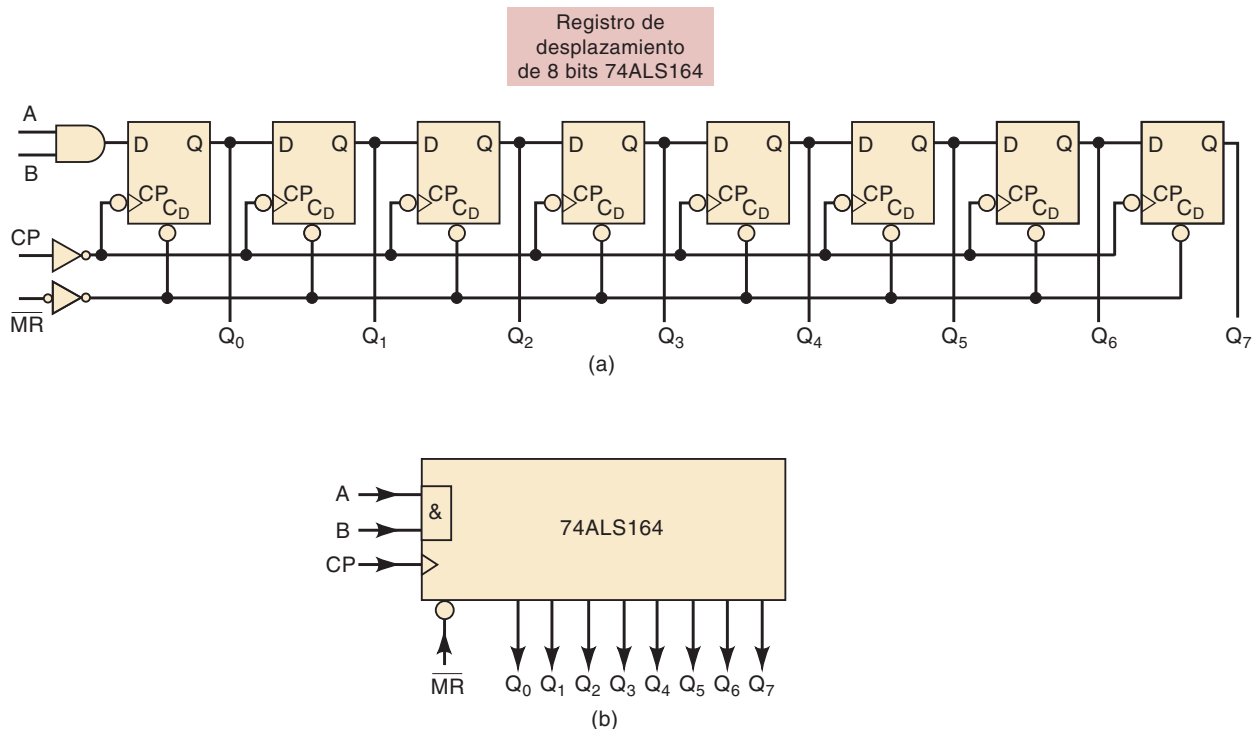


FIGURA 7-68 (a) Diagrama lógico para el 74ALS164; (b) símbolo lógico.

La operación de desplazamiento ocurre en las PGTs de la entrada de reloj CP . La entrada $\overline{MR}$ proporciona el restablecimiento asíncrono de todos los FFs, para que queden en nivel BAJO.

La figura 7-68(b) muestra el símbolo lógico para el 74ALS164. Observe que se utiliza el símbolo & dentro del bloque para indicar que se aplica una operación AND entre las entradas A y B dentro del CI, y que el resultado se aplica a la entrada D de Q_0 .

EJEMPLO 7-21

Suponga que el contenido inicial del registro 74ALS164 en la figura 7-69(a) es 00000000. Determine la secuencia de estados a medida que se aplican pulsos de reloj.

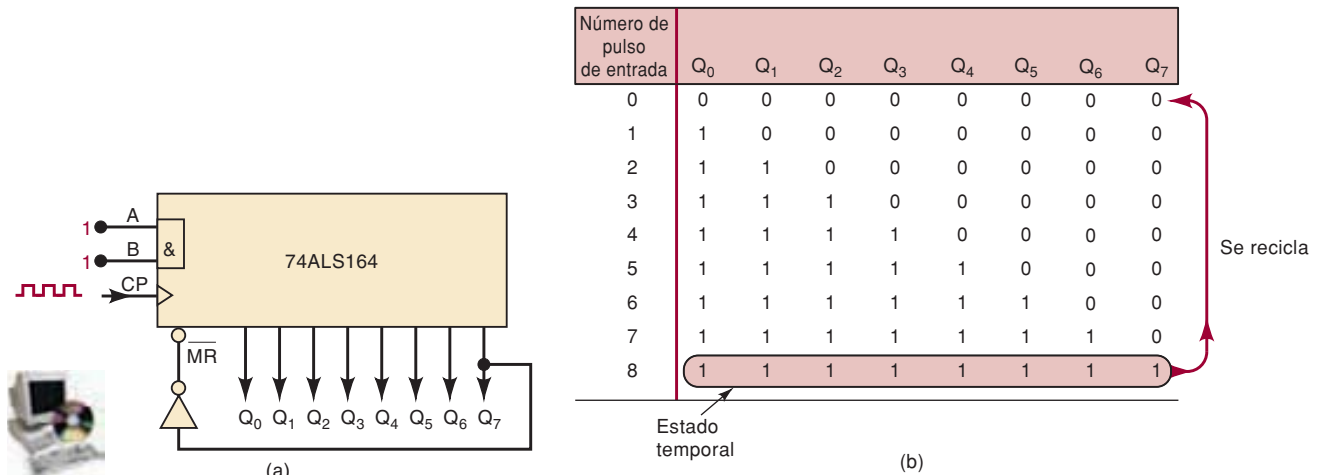


FIGURA 7-69 Ejemplo 7-21.

Solución

La figura 7-69(b) muestra la secuencia correcta. Cuando $A = B = 1$ la entrada en serie es 1, por lo que se desplazarán 1s en el registro durante cada PGT de CP . Debido a que al principio Q_7 está en 0, la entrada $\overline{MR}$ está inactiva.

En el octavo pulso, el registro trata de cambiar al estado 11111111 cuando el 1 de Q_6 se desplaza hacia Q_7 . Este estado ocurre sólo en forma momentánea debido a que $Q_7 = 1$ produce un nivel BAJO en $\overline{MR}$, con lo cual se restablece de inmediato el registro de vuelta a 00000000. Después se repite la secuencia en los siguientes ocho pulsos de reloj.

A continuación se muestra una lista de algunos otros CIs de registro que son variaciones de los que ya presentamos antes:

- 74194/ALS194/HC194. Este CI es un *registro de desplazamiento universal bidireccional de cuatro bits*, el cual puede realizar las operaciones de desplazamiento a la izquierda, desplazamiento a la derecha, entrada en paralelo y salida en paralelo. Estas operaciones se seleccionan mediante un código de selección de modo de dos bits, el cual se aplica en forma de entradas al dispositivo. (El problema 7-71 le dará la oportunidad de averiguar más acerca de este versátil CI).
- 74373/ALS373/HC373/HCT373. Este CI es un registro de entrada en paralelo/salida en paralelo de ocho bits (octal), el cual contiene ocho circuitos de retención (*latches*) tipo *D* con salidas de *tercer estado* (triestado). Una salida de tercer estado es un tipo especial de salida que permite enlazar salidas de dispositivos en forma segura. En el siguiente capítulo veremos las características de los dispositivos de tercer estado, tales como el 74373.
- 74374/ALS374/HC374/HCT374. Este CI es un registro de entrada en paralelo/salida en paralelo de ocho bits (octal), el cual contiene ocho flip-flops *D* disparados por flanco, con salidas triestado.

Los registros de CI que presentamos aquí son representativos de los diversos tipos que están disponibles en forma comercial. Aunque hay muchas variaciones de estos registros básicos, en este momento usted podrá entender con facilidad la mayoría de ellos, mediante la consulta de las hojas técnicas proporcionadas por los fabricantes.

En los problemas al final del capítulo y en el material que cubriremos en capítulos subsiguientes presentaremos varias aplicaciones de registros.

PREGUNTAS DE REPASO

1. ¿En qué tipo de registro se puede cargar un número binario completo y después desplazarlo a la salida un bit a la vez?
2. *Verdadero o falso:* un registro de entrada en serie/salida en paralelo puede mostrar todos sus bits al mismo tiempo.
3. ¿En qué tipo de registro se pueden introducir datos sólo un bit a la vez y en el que todos los bits de datos pueden estar disponibles como salida?
4. ¿En qué tipo de registro podemos almacenar datos un bit a la vez y tener acceso a sólo un bit de salida a la vez?
5. ¿Qué diferencia hay entre la entrada de datos en paralelo del 74165 y del 74174?
6. ¿Cómo funciona la entrada *CP INH* del 74ALS165?

7-20 CONTADORES DE REGISTRO DE DESPLAZAMIENTO

En la sección 5-18 vimos cómo conectar FFs en un arreglo de registro de desplazamiento para transferir datos de izquierda a derecha o viceversa, un bit a la vez (en serie). Los contadores de registro de desplazamiento utilizan *retroalimentación*, lo cual significa que la salida del último FF en el registro se conecta de regreso al primer FF.

Contador de anillo

El contador de registro de desplazamiento más sencillo es un **registro de desplazamiento circular**, éste es conectado de tal forma que el último FF desplaza su valor hacia el primer FF. La figura 7-70 muestra este arreglo, en el cual se utilizan FFs tipo D (también pueden utilizarse flip-flops J-K). Los FFs se conectan de tal forma que la información se desplaza de izquierda a derecha y de regreso desde Q_0 hasta Q_3 . En la mayoría de los casos sólo hay un 1 en el registro, el cual se pone a circular alrededor del registro mientras se aplican pulsos de reloj. Por esta razón se le conoce como **registro de anillo**.

Las formas de onda, la tabla de secuencias y el diagrama de estados de la figura 7-70 muestran los diversos estados de los FFs a medida que se aplican pulsos; aquí se supone un estado inicial de $Q_3 = 1$ y $Q_2 = Q_1 = Q_0 = 0$. Después del primer pulso el 1 se desplaza de Q_3 a Q_2 , de forma tal que el contador se encuentra en el estado 0100. El segundo pulso produce el estado 0010 y el tercer pulso produce el estado 0001. En el *cuarto* pulso de reloj, el 1 de Q_0 se transfiere a Q_3 para producir el estado 1000 que es, desde luego, el estado inicial. Los siguientes pulsos hacen que se repita la secuencia.

Este contador funciona como un contador MOD-4 debido a que tiene *cuatro* estados distintos antes de que se repita la secuencia. Aunque este circuito no progresa por la secuencia de conteo binaria normal, de todas formas se considera un contador ya que cada conteo corresponde a un conjunto único de estados de los FFs. Observe que cada una de las formas de onda de salida de los FFs tiene una frecuencia igual a un cuarto de la frecuencia de reloj, debido a que éste es un contador de anillo MOD-4.

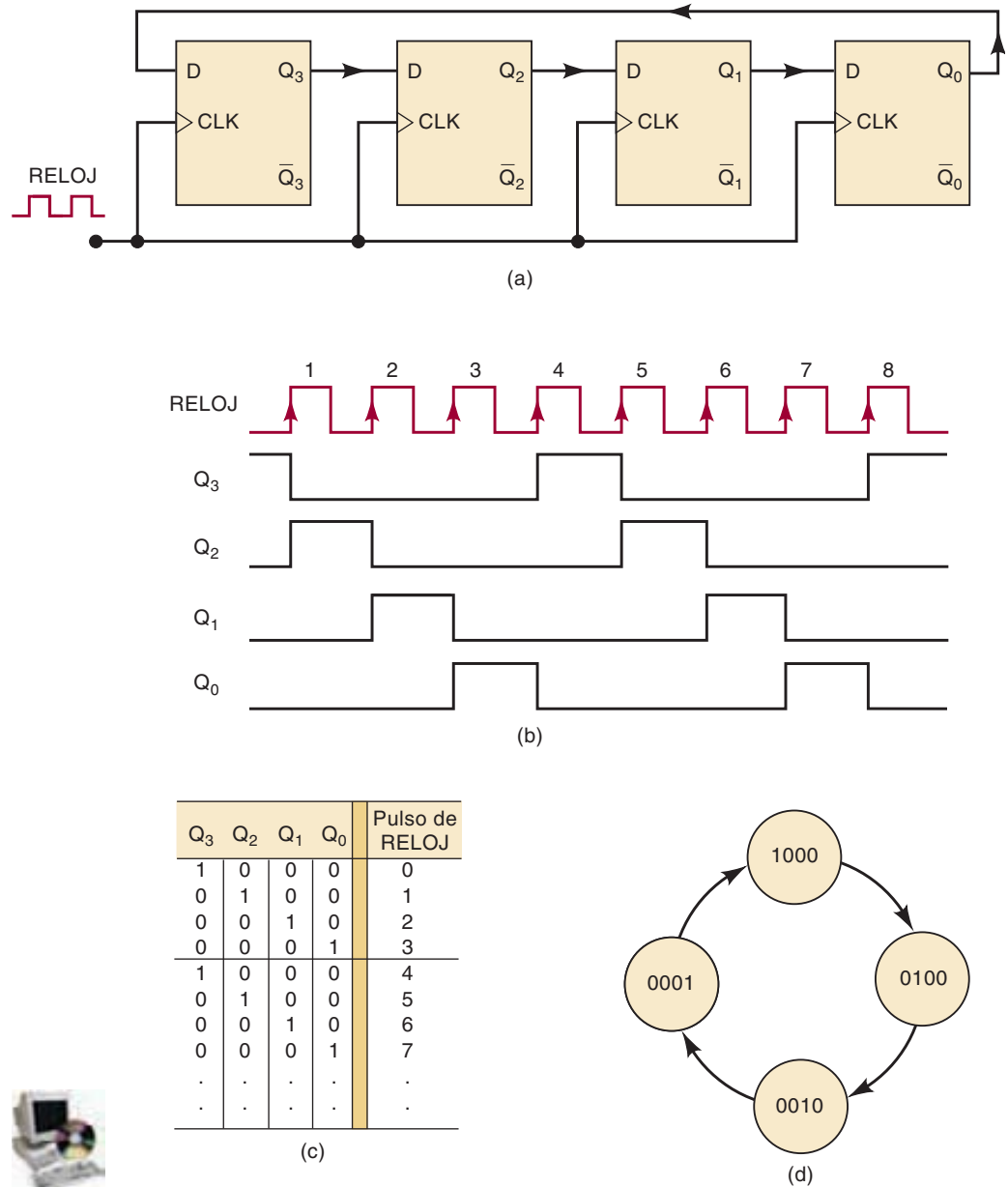


FIGURA 7-70 (a) Contador de anillo de cuatro bits; (b) formas de onda; (c) tabla de secuencia; (d) diagrama de estado.

Pueden construirse contadores de anillo para cualquier número MOD que se desee; un contador de anillo MOD- N utiliza N flip-flops conectados en el arreglo que aparece en la figura 7-70.

En general, un contador de anillo requiere más FFs que un contador binario para el mismo número MOD; por ejemplo, un contador de anillo MOD-8 requiere ocho FFs, mientras que un contador binario MOD-8 sólo requiere tres.

A pesar del hecho de que es menos eficiente en el uso de FFs, un contador de anillo sigue siendo útil debido a que puede decodificarse sin el uso de compuertas de decodificación. La señal de decodificación para cada estado se obtiene en la salida de su FF correspondiente. Compare las formas de onda de los FFs del contador de anillo con las formas de onda de decodificación en la figura 7-20. En algunos casos, un contador de anillo podría ser mejor elección que un contador binario con sus compuertas de decodificación asociadas. En especial esto es útil en aplicaciones en donde se utiliza el contador para controlar la secuencia de operaciones de un sistema.

Inicio de un contador de anillo

Para operar en forma apropiada, un contador de anillo debe comenzar con sólo un FF en el estado 1 y todos los demás en el estado 0. Como los estados iniciales de los FFs serán impredecibles al momento de encenderlos, el contador debe prestablecerse en el estado inicial requerido antes de aplicar pulsos de reloj. Una manera de hacer esto es aplicar un pulso momentáneo a la entrada $\overline{PRE}$ asíncrona de uno de los FFs (por ejemplo, Q_3 en la figura 7-70) y a la entrada $\overline{CLR}$ de todos los demás FFs. La figura 7-71 muestra otro método. Al momento del encendido, el capacitor se cargará lentamente hacia $+V_{CC}$. La salida del INVERSOR disparador de Schmitt 1 permanecerá en ALTO y la salida del INVERSOR 2 permanecerá en BAJO hasta que el voltaje del capacitor se exceda del voltaje de umbral con tendencia positiva (V_{T+}) de la entrada del INVERSOR 1 (un valor aproximado de 1.7 V). Con esto se retendrá en el estado BAJO las entradas $\overline{PRE}$ de Q_3 y $\overline{CLR}$ de Q_2 , Q_1 y Q_0 el tiempo suficiente durante el encendido como para asegurar que el contador empiece en 1000.

Contador Johnson

El contador de anillo básico puede modificarse un poco para producir otro tipo de contador de registro de desplazamiento, el cual tendrá propiedades un poco distintas. El **contador Johnson** o de **anillo torcido** se construye en forma idéntica a un contador de anillo normal, sólo que la salida *invertida* del último FF se conecta a la entrada del primer FF. En la figura 7-72 se muestra un contador Johnson de tres bits. Observe que la salida $\overline{Q}_0$ se conecta de regreso a la entrada D de Q_2 , lo cual significa que el *inverso* del nivel almacenado en Q_0 se transferirá a Q_2 en el pulso de reloj.

Es fácil analizar la operación del contador Johnson si tenemos en cuenta que en cada transición positiva del pulso de reloj el nivel en Q_2 se desplaza hacia Q_1 , el nivel en Q_1 se desplaza hacia Q_0 y el *inverso* del nivel en Q_0 se desplaza hacia Q_2 . Si utilizamos estas ideas y suponemos que todos los FFs están al principio en 0, se pueden generar las formas de onda, la tabla de secuencias y el diagrama de estados de la figura 7-72.

El análisis de las formas de onda y la tabla de secuencias nos revela los siguientes puntos importantes:

1. Este contador tiene seis estados distintos (000, 100, 110, 111, 011 y 001) antes de repetir la secuencia. Por ende, es un contador Johnson MOD-6. Observe que no cuenta con una secuencia binaria normal.
2. La forma de onda de cada FF es una onda cuadrada (ciclo de trabajo del 50 por ciento) que tiene una frecuencia igual a un sexto de la frecuencia del reloj. Además, la forma de onda de cada FF se desplaza por un periodo de reloj, con respecto a las demás.

El número MOD de un contador Johnson siempre será igual al *doble* del número de FFs. Por ejemplo, si conectamos cinco FFs en el arreglo de la figura 7-72 el

FIGURA 7-71 Circuito para asegurar que el contador de anillo de la figura 7-70 inicie en el estado 1000 al momento del encendido.

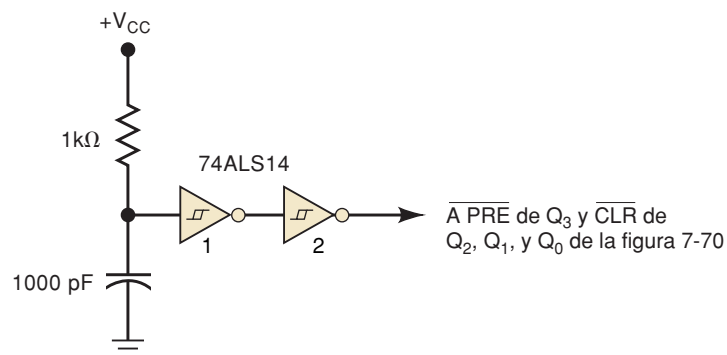
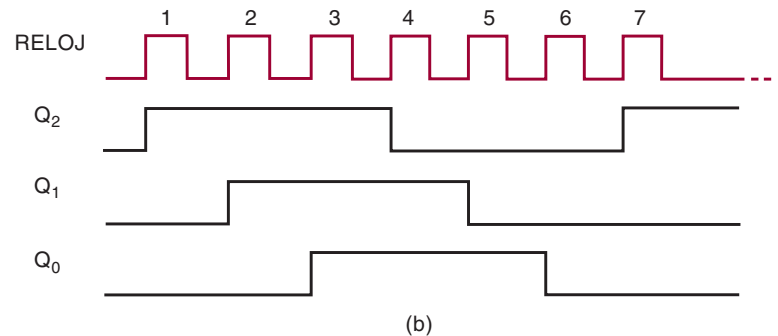
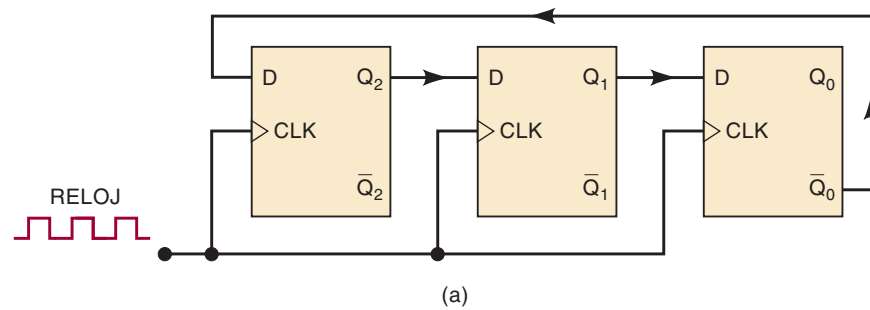
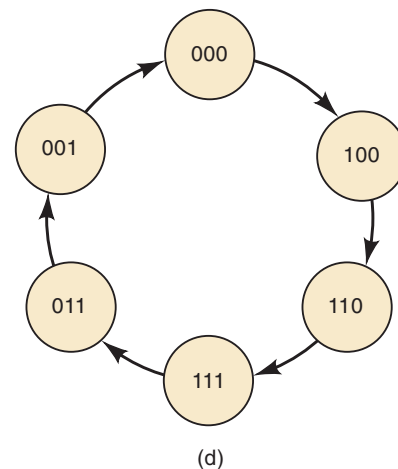


FIGURA 7-72

(a) Contador Johnson MOD-6; (b) forma de onda; (c) tabla de secuencias; (d) diagrama de estados.



Q ₂	Q ₁	Q ₀	Pulso de RELOJ
0	0	0	0
1	0	0	1
1	1	0	2
1	1	1	3
0	1	1	4
0	0	1	5
0	0	0	6
1	0	0	7
1	1	0	8
.	.	.	.
.	.	.	.
.	.	.	.



resultado será un contador Johnson MOD-10, en donde la forma de onda de salida de cada FF es una onda cuadrada a un décimo de la frecuencia del reloj. Por ende, es posible construir un contador MOD- N (en donde N es un número par) mediante la conexión de $N/2$ flip-flops en un arreglo de contadores Johnson.

Decodificación de un contador Johnson

Para un número MOD dado, un contador Johnson sólo requiere la mitad del número de FFs que requiere un contador de anillo. No obstante, un contador Johnson requiere compuertas de decodificación, mientras que un contador de anillo no. Al igual que el contador binario, el de Johnson utiliza una compuerta lógica para decodificar en cada conteo, pero cada compuerta sólo requiere dos entradas, sin importar el número de FFs en el contador. La figura 7-73 muestra las compuertas de decodificación para los seis estados del contador Johnson de la figura 7-72.

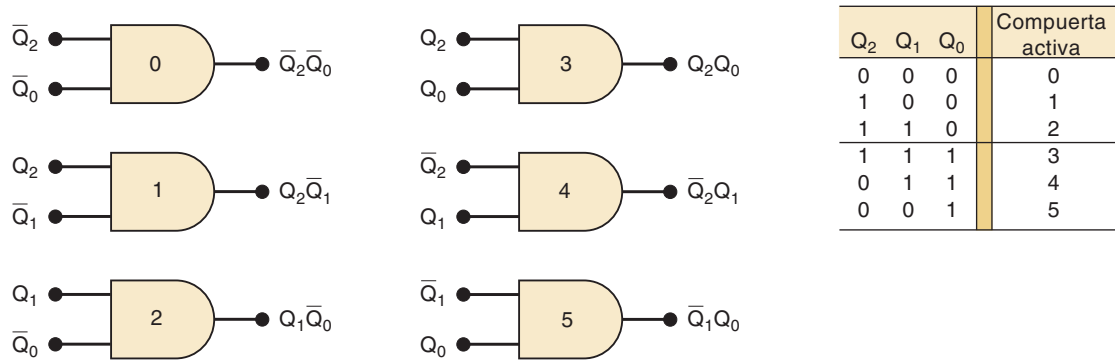


FIGURA 7-73 Lógica de decodificación para un contador Johnson MOD-6.

Observe que cada compuerta de decodificación sólo tiene dos entradas aún y cuando hay tres FFs en el contador, porque para cada conteo dos de los tres FFs se encuentran en una combinación única de estados. Por ejemplo, la combinación $Q_2 = Q_0 = 0$ sólo ocurre una vez en la secuencia de conteo, en la cuenta de 0. Por lo tanto, puede utilizarse la compuerta AND 0 con las entradas $\bar{Q}_2$ y $\bar{Q}_0$ para decodificar en este conteo. Esta misma característica la comparten todos los demás estados en la secuencia, como podrá verificar el lector. De hecho, para un contador Johnson de cualquier tamaño las compuertas de decodificación sólo tendrán dos entradas.

Los contadores Johnson representan un punto intermedio entre los contadores de anillo y los contadores binarios. Un contador Johnson requiere menos FFs que un contador de anillo, pero, por lo general, requiere más que un contador binario; tiene más circuitos de decodificación que un contador de anillo pero menos que un contador binario. Por ende, algunas veces representa una buena opción para ciertas aplicaciones.

Contadores de registro de desplazamiento de CI

Hay muy pocos contadores de anillo o Johnson disponibles como CIs, debido a que es muy fácil tomar un CI de registro de desplazamiento y cablearlo como un contador de anillo o como un Johnson. Algunos de los CIs de contador Johnson CMOS son el 74HC4017 y el 74HC4022, éstos incluyen circuitos de decodificación completos.

PREGUNTAS DE REPASO

1. ¿Cuál contador de registro de desplazamiento requiere la mayor cantidad de FFs para un número MOD dado?
2. ¿Cuál contador de registro de desplazamiento requiere la mayor cantidad de circuitos de decodificación?
3. ¿Cómo puede convertirse un contador de anillo en un contador Johnson?
4. Verdadero o falso:
 - (a) Las salidas de un contador de anillo son siempre ondas cuadradas.
 - (b) Los circuitos de decodificación para un contador Johnson son más simples que los de un contador binario.
 - (c) Los contadores de anillo y Johnson son contadores síncronos.
5. ¿Cuántos FFs se necesitan en un contador de anillo MOD-16? ¿Cuántos se necesitan en un contador Johnson MOD-16?

7-21 DIAGNÓSTICO DE FALLAS

Los flip-flops, los contadores y los registros son los principales componentes en los **sistemas lógicos secuenciales**. Debido a sus dispositivos de almacenamiento, un sistema lógico secuencial tiene la característica de que sus salidas y la secuencia de las operaciones dependen tanto de las entradas presentes como de las que se produjeron antes. Aún y cuando los sistemas lógicos secuenciales son, por lo general, más complejos que los combinacionales, los procedimientos esenciales para el diagnóstico de fallas se aplican con la misma eficacia en ambos tipos de sistemas. Los sistemas secuenciales sufren de los mismos tipos de fallas (circuitos abiertos, cortos, fallas internas en los CIs, etcétera) que los combinacionales.

Muchos de los mismos pasos que se utilizan para aislar fallas en un sistema combinacional pueden aplicarse a los sistemas secuenciales. Una de las técnicas de diagnóstico de fallas más efectiva comienza desde que el técnico de diagnóstico de fallas observa la operación del sistema y, mediante el razonamiento analítico, determina las posibles causas de la falla en el sistema. Después el técnico utiliza los instrumentos de prueba disponibles para aislar la falla exacta. Los siguientes ejemplos demostrarán los tipos de razonamiento analítico que deben ser el paso inicial al diagnosticar fallas en los sistemas secuenciales. Después de estudiar estos ejemplos, usted deberá estar preparado para enfrentar los problemas de diagnóstico de fallas al final del capítulo.

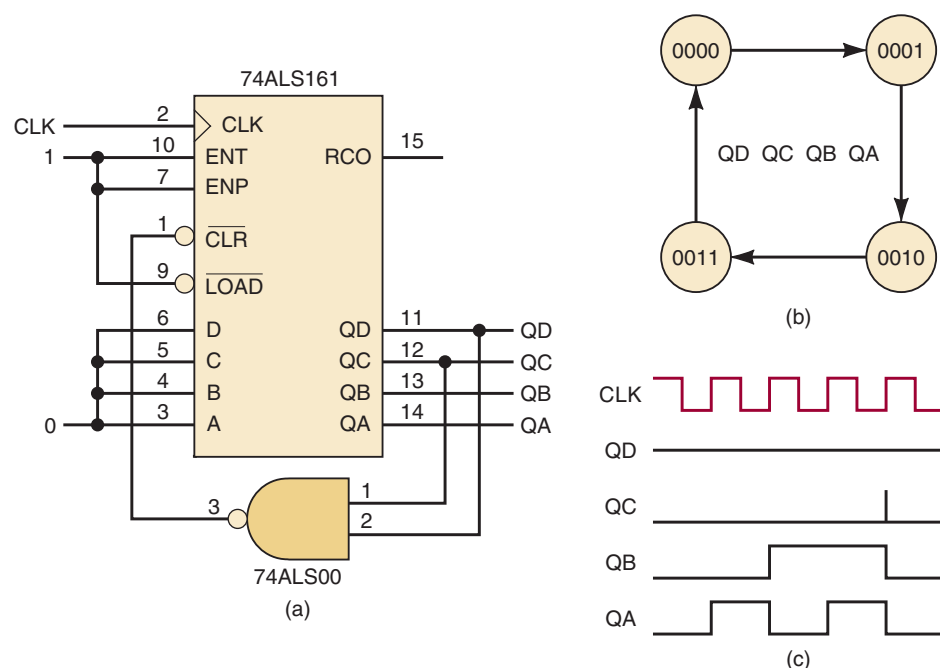
EJEMPLO 7-22

La figura 7-74(a) muestra un 74ALS161 cableado como un contador MOD-12, pero produce la secuencia de conteo que se muestra en la figura 7-74(b). Determine la causa del comportamiento incorrecto del circuito.

Solución

Las salidas QB y QA parecen operar en forma correcta, pero QC y QD permanecen en BAJO. Nuestra primera opción para la falla es que QC está en corto con tierra, pero al comprobar con un óhmetro no se confirma esto. El 74ALS161 podría tener una falla interna que evite que cuente más allá de 0011. Tratamos de extraer el chip NAND 7400 de su zócalo y ponemos en corto la terminal CLR con un nivel ALTO.

FIGURA 7-74 Ejemplo 7-22.



Ahora el contador cuenta en una secuencia MOD-16 regular, por lo que al menos las salidas del contador parecen estar bien. A continuación decidimos analizar la terminal CLR volviendo a conectar la compuerta NAND. Al utilizar una sonda lógica con su “captura de pulsos” encendida, descubrimos que la terminal CLR está recibiendo pulsos. Al conectar un osciloscopio a las salidas vemos que el contador produce las formas de onda que se muestran en la figura 7-74(c). Se observa una deformación en QC cuando el contador debería pasar al estado 0100. Eso indica que 0100 es un estado transitorio, cuando en realidad el estado transitorio debería ser 1100. Ahora sospechamos de la conexión entre QD y la compuerta NAND, por lo que utilizamos la sonda lógica para revisar la terminal 2. En esta terminal no hay ninguna señal lógica, lo cual nos lleva a la conclusión de que la falla es debido a un circuito abierto entre la salida QD y la terminal 2 en la compuerta NAND. La entrada de la compuerta NAND está flotando en ALTO, lo cual provoca que el circuito detecte el estado 0100 en vez del 1100, como debería ser.

EJEMPLO 7-23

Un técnico recibe una “orden de reparación” para un tablero de circuitos, en la cual se indica que el divisor de frecuencia variable opera “algunas veces”. Suena como uno de los temidos problemas en los que la falla es intermitente; a menudo uno de los problemas más difíciles de resolver. Su primer pensamiento es regresar el tablero con la nota “¡Utilícelo sólo cuando opere en forma correcta!”, pero decide investigar más, ya que se siente capaz de enfrentar un buen reto. En la figura 7-75 se muestra el diagrama esquemático para el bloque del circuito. El factor de división deseado se aplica a la entrada $f[7..0]$ en binario. El contador de ocho bits cuenta en

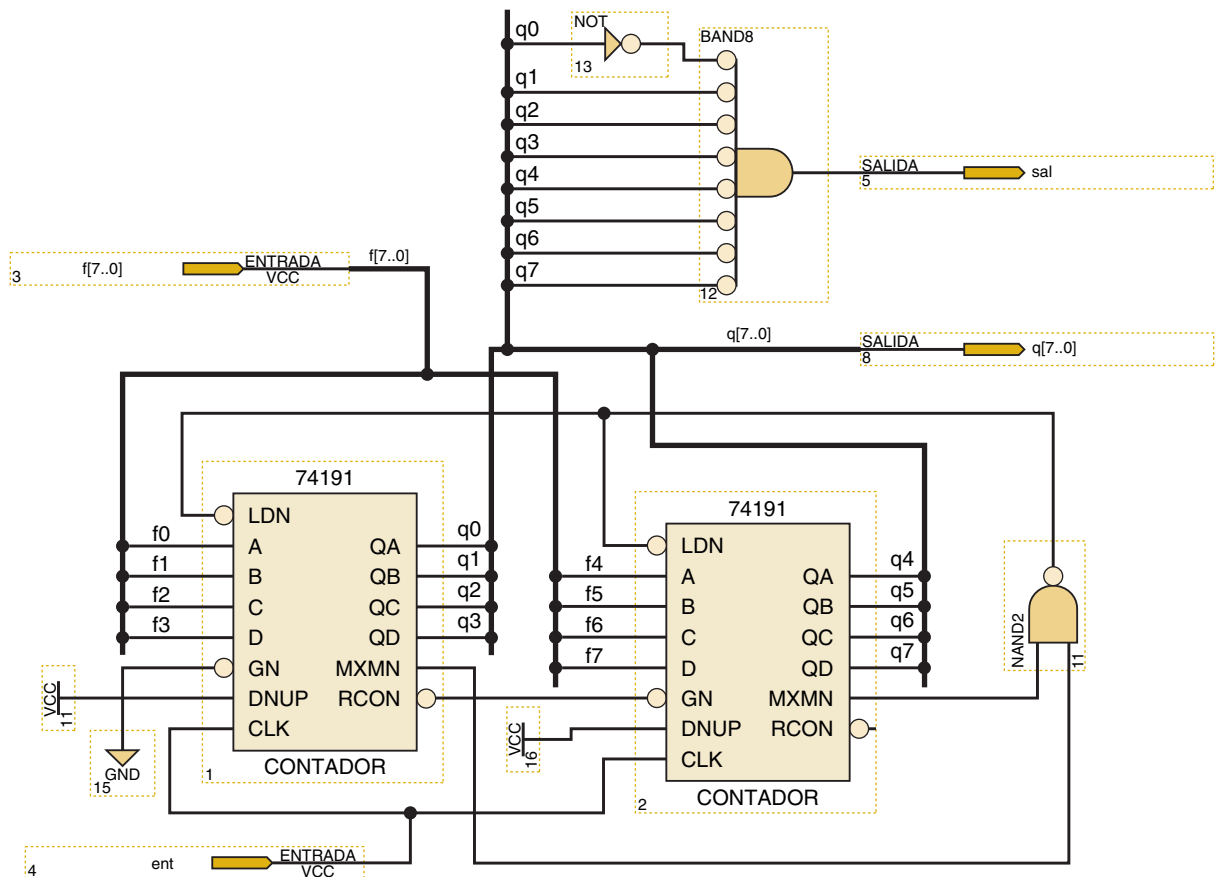


FIGURA 7-75 Ejemplo 7-23.

forma descendente a partir de este número hasta que llega a cero y después realiza una carga asíncrona de $f[]$ otra vez, convirtiendo al cero en un estado transitorio. El módulo resultante será igual al valor en $f[]$. Para obtener la señal de frecuencia de salida se decodifica el estado 00000001, con lo cual la frecuencia de sal se hace igual a la frecuencia de ent , dividida entre el valor binario $f[]$. En la aplicación, la frecuencia de ent es 100 kHz. Si se cambia $f[]$ se producirá una nueva frecuencia de salida.

Solución

El técnico decide que necesita obtener algunos resultados de prueba para analizarlos. Elige algunos factores de división sencillos para aplicarlos a f y registra los resultados que se muestran en la tabla 7-9.

TABLA 7-9

$f[]$ (decimal)	$f[]$ (binario)	f_{sal} medida	¿Correcto?
255	11111111	398.4 Hz	
240	11110000	416.7 Hz	✓
200	11001000	500.0 Hz	✓
100	01100100	1041.7 Hz	
50	00110010	2000.0 Hz	✓
25	00011001	4000.0 Hz	✓
15	00001111	9090.9 Hz	

El técnico observa que el circuito produce resultados correctos en algunos casos de prueba, pero produce resultados incorrectos en otros. El problema no parece ser intermitente después de todo, sino que parece depender del valor para f . El técnico decide calcular la relación entre las frecuencias de entrada y de salida para las tres pruebas que fallaron y obtiene lo siguiente:

$$100 \text{ kHz} / 398.4 \text{ Hz} = 251$$

$$100 \text{ kHz} / 1041.7 \text{ Hz} = 96$$

$$100 \text{ kHz} / 9090.9 \text{ Hz} = 11$$

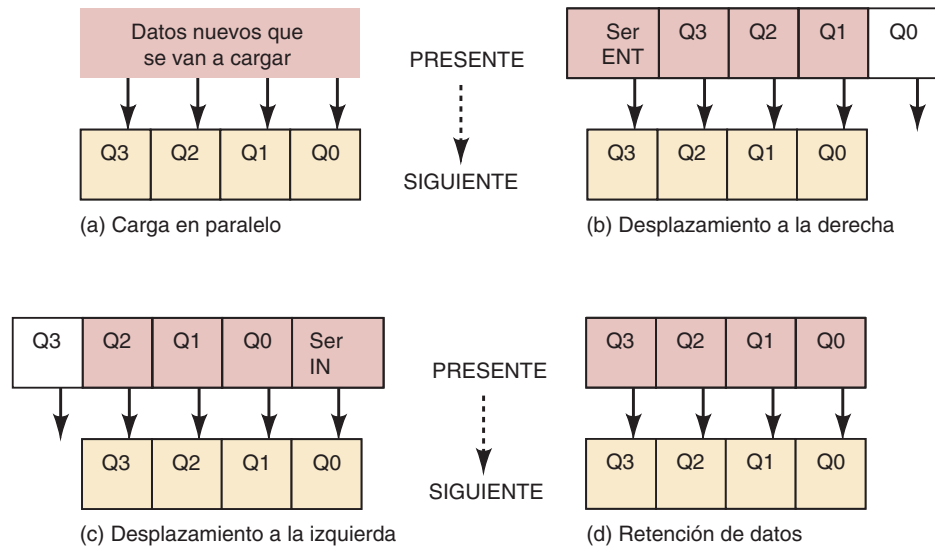
Cada falla parece un factor de división que es cuatro unidades menor que el valor que se aplicó a la entrada. Después de analizar otra vez la representación binaria para f , el técnico observa que todas las fallas ocurrieron cuando $f_2 = 1$. Sin duda, el peso para ese bit es cuatro. ¡Eureka! Ese bit no parece estar funcionando; es hora de hacer una prueba con una sonda lógica en la terminal f_2 . Asimismo, la sonda lógica indica que la terminal está en BAJO sin importar el valor para f_2 .

7-22 REGISTROS EN HDL

En las secciones 7-15 a 7-19 describimos con detalle las diversas opciones de la transferencia de datos en serie y en paralelo dentro de los registros; además describimos algunos ejemplos de CIs que realizan estas operaciones. La belleza de utilizar HDL para describir un registro está en el hecho de que a un circuito se le puede dar cualquiera de las opciones y se le pueden asignar tantos bits como sean necesarios con sólo cambiar unas cuantas palabras.

FIGURA 7-76

Transferencias de datos que se realizan en registros de desplazamiento:
 (a) carga en paralelo;
 (b) desplazamiento a la derecha;
 (c) desplazamiento a la izquierda;
 (d) retención de datos.



Las técnicas de HDL utilizan arreglos de bits para describir los datos de un registro y transferir esos datos en formato en paralelo o en serie. Para comprender cómo se desplazan los datos en HDL considere los diagramas de la figura 7-76, en donde se muestran cuatro flip-flops que realizan operaciones de transferencia de carga en paralelo, desplazamiento a la derecha, desplazamiento a la izquierda y retención de datos. Para todos estos diagramas los bits se transfieren en forma síncrona, lo cual significa que todos se desplazan en forma simultánea en un solo flanco de reloj. En la figura 7-76(a) los datos que se van a cargar en paralelo en el registro se presentan en las entradas *D*, y en el siguiente pulso de reloj se transferirán a las salidas *q*. Desplazar los datos a la derecha significa que cada bit se transfiere hacia la posición del bit que está a su derecha inmediata, mientras que se transfiere un nuevo bit de entrada en el extremo izquierdo y se pierde el último bit en el extremo derecho. Esta situación se ilustra en la figura 7-76(b). Observe que el conjunto de datos que queremos en el estado SIGUIENTE está compuesto de la nueva entrada en serie y tres de los cuatro bits en el arreglo del estado PRESENTE. Estos datos sólo necesitan desplazarse y sobrescribir los cuatro bits de datos del registro. La misma operación se produce en la figura 7-76(c), pero los datos se desplazan a la izquierda. La clave para desplazar el contenido del registro a la derecha o a la izquierda es agrupar los tres bits de datos apropiados del estado PRESENTE en orden correcto con el bit de entrada en serie, de manera que estos cuatro bits puedan cargarse en paralelo en el registro. Puede utilizarse la **concatenación** (agrupamiento en una secuencia específica) del conjunto deseado de bits de datos para describir el movimiento de datos necesario para el desplazamiento en serie en cualquier dirección. A la última posibilidad se le conoce como el modo de retención de datos y se muestra en la figura 7-76(d). Ésta puede parecer innecesaria, ya que los registros (flip-flops) retienen los datos por su propia naturaleza. No obstante, debemos tener en cuenta lo que debe hacerse a un registro para que pueda retener su valor cuando recibe pulsos de reloj. Las salidas *Q* deben enlazarse con las entradas *D* para cada flip-flop, de manera que los datos anteriores se recarguen en cada pulso de reloj. Ahora analicemos algunos ejemplos de circuitos de registro de desplazamiento en HDL.

REGISTRO SISO EN AHDL

La figura 7-77 lista un registro de cuatro bits, tipo entrada en serie/salida en serie (SISO), en AHDL. En la línea 7 se crea una instancia de un arreglo de cuatro flip-flops *D* y se obtiene la salida en serie a partir del último FF *q0* (línea 10). Si el control *desplaza* está en ALTO, *serial_ent* se desplazará hacia el registro y los demás

```

1 SUBDESIGN fig7_77
2 (
3     clk, desplaza, serial_ent      :INPUT;
4     serial_sal                     :OUTPUT;
5 )
6 VARIABLE
7     q[3..0]                       :DFF;
8 BEGIN
9     q[ ].clk = clk;                -- envía último bit del registro como salida
10    serial_sal = q0.q;
11    IF (desplaza == VCC) THEN
12        q[3..0].d = (serial_ent, q[3..1].q); -- concatena para el desplazamiento
13    ELSE
14        q[3..0].d = (q[3..0].q);          -- retiene los datos
15    END IF;
16 END;

```

FIGURA 7-77 Registro de entrada en serie/salida en serie mediante el uso de AHDL.

bits se moverán a la derecha (líneas 11 a 15). Al concatenar *serial_ent* con los bits de salida de los FFs *q3*, *q2* y *q1* en ese orden se crea el patrón de bits apropiado de entrada de datos de desplazamiento a la derecha (línea 12). Si el control *desplaza* está en BAJO, el registro retendrá los datos actuales (línea 14). En la figura 7-78 se muestran los resultados de la simulación.

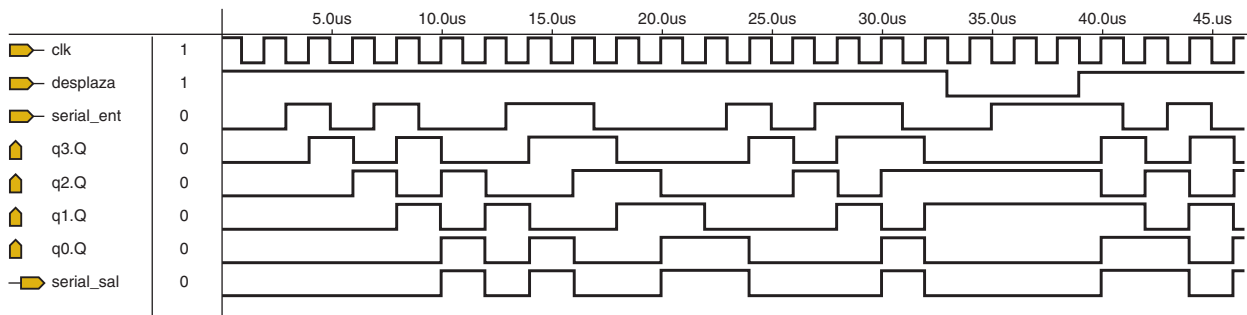


FIGURA 7-78 Simulación de un registro SISO.

REGISTRO SISO EN VHDL

La figura 7-79 lista un registro tipo entrada en serie/salida en serie (SISO) de cuatro bits en VHDL. En la línea 8 se crea un registro con la declaración de la variable *q* y se obtiene la salida en serie mediante el último bit del registro o *q(0)* (línea 10). Si el control *desplaza* está en ALTO, *serial_ent* se desplazará hacia el registro y los otros bits se moverán a la derecha (líneas 12 a 14). Al concatenar *serial_ent* y los bits del registro *q(3)*, *q(2)* y *q(1)* en ese orden se crea el patrón apropiado de bits de entrada de datos de desplazamiento a la derecha (línea 13). Si el control de desplazamiento está en BAJO, VHDL supondrá que la variable se queda igual y, por lo tanto, retendrá los datos actuales. La figura 7-78 muestra los resultados de la simulación.

```

1  ENTITY fig7_79  IS
2  PORT ( clk, desplaza, serial_ent          :IN BIT;
3         serial_sal          :OUT BIT      );
4  END fig7_79;
5  ARCHITECTURE vhdl OF fig7_79 IS
6  BEGIN
7  PROCESS (clk)
8      VARIABLE      q          :BIT_VECTOR (3 DOWNT0 0);
9      BEGIN
10     serial_sal <= q(0);          -- envía último bit del registro como salida
11     IF (clk'EVENT AND clk = '1') THEN
12         IF (desplaza = '1') THEN
13             q := (serial_ent & q(3 DOWNT0 1)); -- concatena para el desplazamiento
14         END IF;                  -- en caso contrario, retiene los datos
15     END IF;
16 END PROCESS;
17 END vhdl;

```

FIGURA 7-79 Registro de entrada en serie/salida en serie mediante el uso de VHDL.

REGISTRO PISO EN AHDL

La figura 7-80 lista un registro tipo entrada en paralelo/salida en serie (PISO) de cuatro bits. En la línea 8 se crea el registro llamado *q* mediante el uso de cuatro FFs D y en la línea 11 se describe la salida en serie de *q0*. El registro cuenta con controles separados de carga (*carga*) en paralelo y desplazamiento (*desplaza*) en serie. En las líneas 12 a 15 se definen las funciones del registro. Si *carga* está en ALTO, se cargará en forma síncrona la entrada externa *datos[3..0]*. *Carga* tiene prioridad y debe estar en BAJO para desplazar en serie el contenido del registro en cada PGT de *clk*, cuando *desplaza* está en ALTO. En la línea 13 se crea el patrón para desplazar datos a la derecha mediante la concatenación. Observe que un nivel BAJO constante será la entrada de datos en serie para una operación de desplazamiento. Si ni *carga* ni *desplaza* están en ALTO, el registro retendrá el valor de datos actual (línea 14). La figura 7-81 muestra los resultados de la simulación.

```

1  SUBDESIGN fig7_80
2  (
3      clk, desplaza, carga      :INPUT;
4      datos[3..0]              :INPUT;
5      serial_sal                :OUTPUT;
6  )
7  VARIABLE
8      q[3..0]                  :DFF;
9  BEGIN
10     q[ ].clk = clk;
11     serial_sal = q0.q;          -- envía último bit del registro como salida
12     IF (carga == VCC) THEN q[3..0].d = datos[3..0]; -- carga en paralelo
13     ELSIF (desplaza == VCC) THEN q[3..0].d = (GND, q[3..1].q); -- desplaza
14     ELSE q[3..0].d = q[3..0].q; -- retiene
15     END IF;
16 END;

```

FIGURA 7-80 Registro de entrada en paralelo/salida en serie mediante el uso de AHDL.

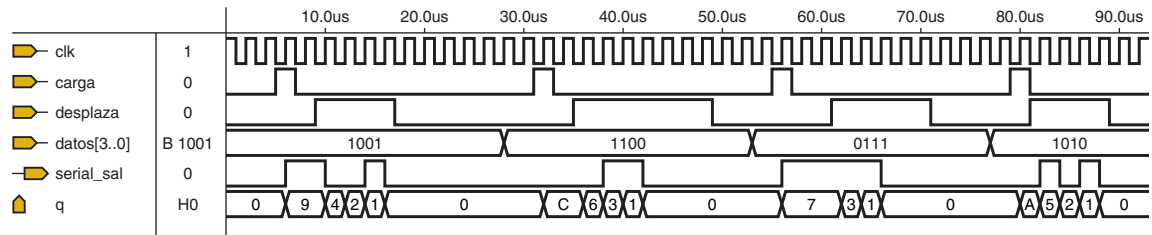


FIGURA 7-81 Simulación de un registro PISO.

REGISTRO PISO EN VHDL

La figura 7-82 lista un registro tipo entrada en paralelo/salida en serie (PISO) de cuatro bits. En la línea 11 se crea el registro con la declaración de la variable *q* y en la línea 13 se describe la salida en serie de *q(0)*. El registro tiene controles separados de carga (*carga*) y desplazamiento en serie (*desplaza*). En las líneas 14 a 18 se definen las funciones del registro. Si *carga* está en ALTO, la entrada externa *datos* se cargará en forma síncrona. *Carga* tiene prioridad y debe estar en BAJO para desplazar en serie el contenido del registro en cada PGT de *clk* cuando *cambia* está en ALTO. En la línea 16 se crea el patrón para desplazar datos a la derecha mediante la concatenación. Observe que la entrada de datos en serie será un nivel BAJO constante para una operación de desplazamiento. Si ni *carga* ni *desplaza* están en ALTO, el registro retendrá el valor actual de datos mediante la operación implícita de VHDL. La figura 7-81 muestra los resultados de la simulación.

```

1  ENTITY fig7_82 IS
2  PORT (
3      clk, desplaza, carga          :IN BIT;
4      datos                        :IN BIT_VECTOR (3 DOWNTO 0);
5      serial_sal                   :OUT BIT
6  );
7  END fig7_82;
8  ARCHITECTURE vhd1 OF fig7_82 IS
9  BEGIN
10     PROCESS (clk)
11         VARIABLE q                :BIT_VECTOR (3 DOWNTO 0);
12     BEGIN
13         serial_sal <= q(0);        -- envía último bit del registro como salida
14         IF (clk'EVENT AND clk = '1') THEN
15             IF (carga = '1') THEN q := datos; -- carga en paralelo
16             ELSIF (desplaza = '1') THEN q:= ('0' & q[3 DOWNTO 1]); -- desplaza
17             ENDIF;                  -- en caso contrario retiene
18         END IF;
19     END PROCESS;
20 END vhd1;
```

FIGURA 7-82 Registro de entrada en paralelo/salida en serie mediante el uso de VHDL.

EJEMPLO 7-24

Suponga que deseamos diseñar un registro de desplazamiento de cuatro bits universal mediante HDL, el cual tenga cuatro modos síncronos de operación: Retener datos, Desplazar a la izquierda, Desplazar a la derecha y Carga en paralelo. Dos bits de entrada seleccionarán la operación que deba realizarse en cada flanco de subida del reloj. Para implementar un registro de desplazamiento podemos utilizar código

estructural para describir una cadena de flip-flops. Para hacer que el registro de desplazamiento sea eficiente y nos permita desplazar a la izquierda o a la derecha, o realizar la carga en paralelo, este archivo se haría bastante largo y por ende difícil de leer y comprender utilizando los métodos estructurales. Un enfoque mucho más conveniente es utilizar los métodos más abstractos e intuitivos disponibles en HDL para describir el circuito en forma concisa. Para ello debemos desarrollar una estrategia para crear la acción de desplazamiento. El concepto es muy similar al que se presenta en el ejemplo 7-16, en donde se cableó un chip de registro de flip-flop D (74174) para formar un registro de desplazamiento. En vez de pensar en el registro de desplazamiento como una cadena en serie de flip-flops, debemos considerarlo como un registro en paralelo cuyo contenido se va a transferir en paralelo a un conjunto de bits que está desplazado por una posición de un bit. La figura 7-76 demuestra el concepto de cada transferencia necesaria en este diseño.

Solución

Un primer paso muy razonable es definir una entrada de dos bits llamada *modo*, con la cual podemos especificar el modo 0, 1, 2 o 3. El siguiente reto es decidir cómo elegir entre las cuatro operaciones mediante el uso de HDL. Aquí pueden funcionar varios métodos. Se eligió la estructura CASE debido a que nos permite elegir un conjunto distinto de instrucciones en HDL para cada uno de todos los posibles valores de modo. No hay prioridad asociada con la comprobación de la configuración de los modos existentes o de los intervalos traslapados de números de modo, por lo que no necesitamos las ventajas de la instrucción IF/ELSE. En las figuras 7-83 y 7-84 se muestran las soluciones en HDL. En cada una de las soluciones se definen las mismas entradas y salidas: un reloj, cuatro bits de datos de carga en paralelo, un bit individual para la entrada en serie al registro, dos bits para la selección del modo y cuatro bits de salida.

```

1  SUBDESIGN fig7 83
2  (
3      reloj          :INPUT;
4      dent[3..0]     :INPUT;      datos en paralelo de entrada
5      ser ent        :INPUT;      datos en serie de entrada, de izquierda o derecha
6      modo [1..0]    :INPUT;      Selección de MODO: 0 = retención,
                                   1 = derecha, 2=izquierda, 3=carga
7      q[3..0]        :OUTPUT;
8  )
9  VARIABLE
10     ff[3..0] :DFF;              define el conjunto de registros
11 BEGIN
12     ff[].clk = reloj;           reloj síncrono
13     CASE modo[] IS
14         WHEN 0 => ff[].d        = ff[].q;          retiene desplazamiento
15         WHEN 1 => ff[2..0].d     = ff[3..1].q;      desplazamiento a la derecha
16                 ff[3].d         = ser ent;         nuevos datos provenientes
                                                         de la izquierda
17         WHEN 2 => ff[3..1].d     = ff[2..0].q;      desplazamiento a la izquierda
18                 ff[0].d         = ser ent;         nuevos datos provenientes
                                                         de la derecha
19         WHEN 3 => ff[].d         = dent[];          carga en paralelo
20     END CASE;
21     q[] = ff[];                actualiza las salidas
22 END;
```

FIGURA 7-83 Registro de desplazamiento universal en AHDL.

```

1  ENTITY fig7_84 IS
2  PORT (
3      reloj      :IN BIT;
4      dent       :IN BIT_VECTOR (3 DOWNTO 0);      -- datos en paralelo de entrada
5      ser_ent    :IN BIT;                          -- datos en serie en Izq o Der
6      modo       :IN INTEGER RANGE 0 TO 3;          -- 0=retención 1=der 2=izq 3=carga
7      q          :OUT BIT_VECTOR (3 DOWNTO 0));
8  END fig7_84;
9  ARCHITECTURE a OF fig7_84 IS
10     BEGIN
11         PROCESS (reloj)                            -- responde al reloj
12             VARIABLE ff :BIT_VECTOR (3 DOWNTO 0);
13             BEGIN
14                 IF (reloj'EVENT AND reloj = '1') THEN
15                     CASE modo IS
16                         WHEN 0 => ff := ff;          -- retiene los datos
17                         WHEN 1 => ff(2 DOWNTO 0) := ff(3 DOWNTO 1); -- desplaza a la derecha
18                                 ff(3) := ser_ent;
19                         WHEN 2 => ff(3 DOWNTO 1) := ff(2 DOWNTO 0); -- desplaza a la izquierda
20                                 f(0) := ser_ent;
21                         WHEN 3 => ff := dent;         -- carga en paralelo
22                     END CASE;
23                 END IF;
24                 q <= ff;                             -- actualiza las salidas
25             END PROCESS;
26     END a;

```

FIGURA 7-84 Registro de desplazamiento universal mediante el uso de VHDL.

SOLUCIÓN EN AHDL

La solución en AHDL de la figura 7-83 utiliza un registro de flip-flops D, los cuales se declaran mediante el nombre *ff* en la línea 10 para representar el estado actual del registro. Debido a que todos los flip-flops necesitan recibir pulsos de reloj al mismo tiempo (en forma síncrona), todas las entradas de reloj se asignan a *reloj* en la línea 12. La instrucción CASE selecciona una configuración de transferencia distinta para cada valor de las entradas *modo*. El modo 0 (retener datos) utiliza una transferencia en paralelo directa del estado actual a las mismas posiciones de bit en las entradas *D*, para producir el estado SIGUIENTE idéntico. El modo 1 (desplazar a la derecha), que se describe en las líneas 15 y 16, transfiere los bits 3, 2 y 1 a las posiciones de bit 2, 1 y 0, en forma respectiva, y carga el bit 3 de la entrada serial. El modo 2 (desplazar a la izquierda) realiza una operación similar en el sentido opuesto (vea las líneas 17 y 18). El modo 3 (carga en paralelo) transfiere el valor en las entradas de datos en paralelo para que se conviertan en el estado SIGUIENTE del registro. El código crea el circuito que selecciona una de estas operaciones lógicas en el registro actual y los datos apropiados se transfieren a las terminales de salida en el siguiente pulso de reloj. Este código se puede reducir si combinamos las líneas 15 y 16 en una sola instrucción que concatene *ser_ent* con los tres bits de datos y los agrupe como un conjunto de cuatro bits. La instrucción que puede sustituir las líneas 15 y 16 es:

```
WHEN 1 => ff[ ].d = (ser ent, ff[3..1].q);
```

Las líneas 17 y 18 también pueden sustituirse por:

```
WHEN 2 => ff[ ].d = (ff[2..0].q, ser ent);
```


SOLUCIÓN EN VHDL

La solución en VHDL de la figura 7-84 define una variable interna por el nombre *ff* en la línea 12, la cual representa el estado actual del registro. Debido a que todas las operaciones de transferencia necesitan llevarse a cabo en respuesta a un flanco de subida del reloj, se utiliza un bloque **PROCESS** en el cual se especifica *reloj* en la lista de sensibilidad. La instrucción **CASE** selecciona una configuración de transferencia distinta para cada valor de las entradas *modo*. El modo 0 (retener datos) utiliza una transferencia en paralelo directa del estado actual a las mismas posiciones de bit para producir el estado **SIGUIENTE** idéntico. El modo 1 (desplazar a la derecha) transfiere los bits 3, 2 y 1 a las posiciones de bit 2, 1 y 0, en forma respectiva (línea 17) y carga el bit 3 de la entrada en serie (línea 18). El modo 2 (desplazar a la izquierda) realiza una operación similar en el sentido opuesto. El modo 3 (carga en paralelo) transfiere el valor en las entradas de datos en paralelo al estado **SIGUIENTE** del registro. Después de elegir una de estas operaciones en el registro actual, los datos se transfieren a las terminales de salida en la línea 24. Este código puede reducirse si combinamos las líneas 17 y 18 en una sola instrucción que concatene *ser_ent* con los tres bits de datos y los agrupe como un conjunto de cuatro bits. La instrucción que puede sustituir las líneas 17 y 18 es:

```
WHEN 1 => ff := ser_ent & ff(3 DOWNT0 1);
```

Las líneas 19 y 20 también pueden sustituirse por:

```
WHEN 2 => ff := ff(2 DOWNT0 0) & ser_ent;
```

PREGUNTAS DE REPASO

1. Escriba una expresión en HDL para implementar un desplazamiento a la izquierda de un arreglo de ocho bits llamado *reg[7..0]* con la entrada en serie *dat*.
2. ¿Por qué es necesario recargar los datos actuales durante el modo de retención de datos en un registro de desplazamiento?

7-23 CONTADORES DE ANILLO EN HDL

En la sección 7-20 utilizamos un registro de desplazamiento para crear un contador que circula un solo nivel lógico activo a través de todos sus flip-flops. A este tipo de registro se le conoce como contador de anillo. Una característica de los contadores de anillo es que el módulo es igual al número de flip-flops en el registro y, por lo tanto, siempre hay muchos estados sin usar e inválidos. Ya hemos visto las formas de describir a los contadores mediante el uso de la instrucción **CASE** para especificar las transiciones del estado **PRESENTE** y del estado **SIGUIENTE**. En esos ejemplos nos hicimos cargo de los estados inválidos al incluirlos bajo “otros” (**OTHERS**). Este método también funciona para los contadores de anillo. No obstante, en esta sección veremos una manera más intuitiva para describir los contadores de desplazamiento.

Estos métodos utilizan las mismas técnicas descritas en la sección 7-22 para que el registro se desplace una posición en cada pulso de reloj. La principal característica de este código es el método para completar el “anillo” al dividir la línea *ser_ent* del registro de desplazamiento. Con un poco de planeación también deberíamos ser capaces de asegurar que el contador llegue en algún momento a la secuencia deseada, sin importar en qué estado se encuentre al principio. Para este ejemplo recreamos la operación del contador de anillo cuyo diagrama de estado se muestra en la figura 7-70(d). Para poder hacer que este contador inicie en forma automática sin utilizar entradas asíncronas, controlamos la línea *ser_ent* del registro de despla-

miento mediante el uso de una instrucción IF/ELSE. Cada vez que detectamos que los tres bits superiores están en BAJO, suponemos que el bit de menor orden está en ALTO y en el siguiente ciclo de reloj queremos desplazar un nivel ALTO hacia *ser_ent*. Para todos los demás estados (válidos e inválidos) desplazamos un nivel BAJO. Sin importar el estado con el que se inicialice el contador, en un momento dado se llenará con ceros; en ese momento nuestra lógica desplazará un nivel ALTO para comenzar la secuencia de anillo.

CONTADOR DE ANILLO EN AHDL

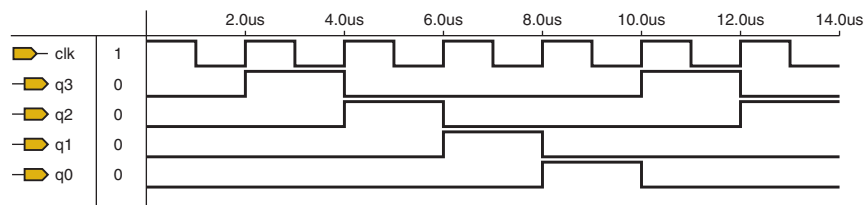
Para estos momentos, el código de AHDL que se muestra en la figura 7-85 le deberá ser familiar. Las líneas 11 y 12 controlan la entrada en serie mediante el uso de la estrategia que acabamos de describir. Observe el uso del operador de doble signo de igualdad (==) en la línea 11. Este operador evalúa si las expresiones en cada lado son iguales o no. Recuerde que el operador de signo de igualdad (=) asigna (es decir, conecta) un objeto a otro. La línea 14 implementa la acción de desplazamiento a la derecha que describimos en la sección anterior. La figura 7-86 muestra los resultados de la simulación.

```

1  SUBDESIGN fig7 85
2  (
3      clk          :INPUT;
4      q[3..0]      :OUTPUT;
5  )
6  VARIABLE
7      ff[3..0]     :DFF;
8      ser ent      :NODE;
9  BEGIN
10     ff[].clk = clk;
11     IF ff[3..1].q == B"000" THEN ser ent = VCC;      inicio automático
12     ELSE ser ent = GND;
13     END IF;
14     ff[3..0].d = (ser ent, ff[3..1].q);              desplaza a la derecha
15     q[] = ff[];
16 END;
```

FIGURA 7-85 Contador de anillo de cuatro bits en AHDL.

FIGURA 7-86 Simulación del contador de anillo en HDL.



CONTADOR DE ANILLO EN VHDL

Para estos momentos, el código de VHDL que se muestra en la figura 7-87 deberá serle familiar. Las líneas 12 y 13 controlan la entrada en serie mediante el uso de la estrategia que acabamos de describir. La línea 16 implementa la acción de desplazamiento a la derecha que describimos en la sección anterior. Los resultados de la simulación se muestran en la figura 7-86.

```

1  ENTITY fig7_87 IS
2  PORT (    clk          :IN BIT;
3          q             :OUT BIT_VECTOR (3 DOWNTO 0));
4  END fig7_87;
5
6  ARCHITECTURE vhd1 OF fig7_87 IS
7  SIGNAL ser_ent        :BIT;
8  BEGIN
9  PROCESS (clk)
10     VARIABLE ff        :BIT_VECTOR (3 DOWNTO 0);
11     BEGIN
12         IF (ff (3 DOWNTO 1) = "000") THEN ser_ent <= '1'; -- inicio automático
13         ELSE ser_ent <= '0';
14         END IF;
15         IF (clk'EVENT AND clk = '1') THEN
16             ff(3 DOWNTO 0) := (ser_ent & ff(3 DOWNTO 1)); -- desplaza a la derecha
17         END IF;
18         q <= ff;
19     END PROCESS;
20 END vhd1;

```

FIGURA 7-87 Contador de anillo de cuatro bits en VHDL.

PREGUNTAS DE REPASO

1. ¿Qué significa que un contador de anillo inicie en forma automática?
2. ¿Qué líneas de la figura 7-85 aseguran que el contador de anillo inicie en forma automática?
3. ¿Qué líneas de la figura 7-87 aseguran que el contador de anillo inicie en forma automática?

7-24 MONOESTABLES EN HDL

Otro circuito importante que hemos estudiado es el circuito monoestable (*one-shot*). Podemos aplicar el concepto de un contador para implementar un **monoestable digital** mediante el uso de HDL. Si recuerda del capítulo 5, los monoestables son dispositivos que producen un pulso de una anchura predefinida cada vez que se activa la entrada de disparo. Un monoestable *no redispensible* ignora la señal de disparo siempre y cuando la salida de pulso siga estando activa. Un monoestable *redispensible* inicia un pulso en respuesta a una señal de disparo y limpia el temporizador de pulso interno cada vez que ocurre un flanco de disparo subsiguiente, antes de completar el pulso. El primer ejemplo que investigaremos es un monoestable digital no redispensible, que se dispara por un nivel ALTO. Los monoestables que estudiamos en el capítulo 5 utilizaban una resistencia y un capacitor como el mecanismo de sincronización de pulso interno. Para poder crear un monoestable mediante las técnicas de HDL, utilizamos un contador de cuatro bits para determinar la anchura del pulso. Las entradas son una señal de reloj, disparo, borrar y el valor de la anchura del pulso. La única salida es la de pulso, *Q*. La idea es bastante simple. Cada vez que se detecte un disparo hay que hacer que el pulso cambie a ALTO y se cargue un contador descendente con un número proveniente de las entradas de anchura de pulso. Entre mayor sea el número, más tiempo tardará el conteo hasta cero. La ventaja de este circuito es que la anchura del pulso puede ajustarse con facilidad si se cambia el valor que se cargó en el contador. A medida que vaya leyendo las siguientes secciones, considere la siguiente pregunta: “¿Qué hace que este circuito sea no redispensible y qué hace que se dispare por nivel?”

MONOESTABLES EN AHDL

La figura 7-88 muestra la descripción en AHDL de un monoestable no redisparable, sensible al nivel. En la línea 8 se crea un registro de cuatro flip-flops y sirve como el contador para realizar el conteo descendente durante el pulso. La señal *reloj* se conecta en paralelo a todos los flip-flops en la línea 10. La función de borrado (reset) se implementa mediante la conexión de la línea de control *reset* en forma directa a la entrada *clrn* asíncrona de cada flip-flop en la línea 11. Después de estas asignaciones la primera condición que se prueba es el disparo. Si se activa (ALTO) en cualquier momento mientras que el valor del conteo sea 0 (es decir, que termine el pulso anterior), entonces se cargará el valor de retraso en el contador. En la línea 14 se prueba para ver si el pulso ya terminó; para ello se comprueba si el contador llegó hasta cero o no. Si llegó a cero, entonces el contador no debe reciclarse sino permanecer en cero. Si el conteo no está en cero, entonces debe estar contando, por lo que la línea 15 prepara los flip-flops para que se decrementen en el siguiente pulso de reloj. Por último, la línea 17 genera el pulso de salida. Esta expresión booleana puede definirse de la siguiente manera: “Hacer que el pulso (*Q*) sea ALTO cuando el *conteo* sea cualquier número distinto de cero.”

FIGURA 7-88

Monoestable no redisparable en AHDL.

```

1  SUBDESIGN fig7 88
2  (
3      reloj, disparo, reset      : INPUT;
4      retraso[3..0]            : INPUT;
5      q                         : OUTPUT;
6  )
7  VARIABLE
8      conteo[3..0]             : DFF;
9  BEGIN
10     conteo[].clk = reloj;
11     conteo[].clrn = reset;
12     IF disparo & conteo[].q == b"0000" THEN
13         conteo[].d = retraso[];
14     ELSIF conteo[].q == B"0000" THEN conteo[].d = B"0000";
15     ELSE conteo[].d = conteo[].q  1;
16     END IF;
17     q = conteo[].q != B"0000";      crea el pulso de salida
18 END;
```

MONOESTABLES SIMPLES EN VHDL

La figura 7-89 muestra la descripción en VHDL de un circuito digital monoestable no redisparable, sensible al nivel. Las entradas y las salidas se muestran en las líneas 3 a 5, como se describió antes. En la descripción de la arquitectura se utiliza un bloque **PROCESS** (línea 11) para responder a cualquiera de las dos entradas: reloj o *reset*. Dentro de este bloque **PROCESS** se utiliza una variable para representar el valor en el contador. La entrada que debería tener precedencia sobre las demás es la señal *reset*. Ésta se evalúa primero (línea 14) y si está activa, *conteo* se borra de inmediato. Si *reset* no está activa se evalúa la línea 15 y se busca un flanco de subida en *reloj*. La línea 16 comprueba el disparo. Si se activa en cualquier momento mientras que el valor del conteo sea 0 (es decir, si terminó el pulso anterior), entonces se carga el valor de la anchura en el contador. En la línea 18 se evalúa si el pulso terminó mediante una comprobación para ver si el contador llegó a cero. Si es así, entonces el contador no deberá reciclarse sino permanecer en cero.

```

1  ENTITY fig7 89 IS
2  PORT (
3      reloj, disparo, reset      :IN BIT;
4      retraso                    :IN INTEGER RANGE 0 TO 15;
5      q                          :OUT BIT
6  );
7  END fig 7 89;
8
9  ARCHITECTURE vhd1 OF fig7 89 IS
10 BEGIN
11     PROCESS (reloj, reset)
12     VARIABLE conteo            : INTEGER RANGE 0 TO 15;
13     BEGIN
14         IF restablecer = '0' THEN conteo := 0;
15         ELSIF (reloj EVENT AND reloj = '1' ) THEN
16             IF disparo = '1' AND conteo = 0 THEN
17                 conteo := retraso;                carga el contador
18             ELSIF conteo = 0 THEN conteo := 0;
19             ELSE conteo := conteo  1;
20             END IF;
21         END IF;
22         IF conteo /= 0 THEN q <= '1';
23         ELSE q <= '0';
24         END IF;
25     END PROCESS;
26 END vhd1;

```

FIGURA 7-89 Monoestable no redispensible en VHDL.

Si *conteo* no está en cero significa que debe estar contando, por lo que la línea 19 prepara los flip-flops para que se decrementen en el siguiente pulso de reloj. Por último, las líneas 22 y 23 generan el pulso de salida. Esta expresión booleana puede definirse de la siguiente manera: “Hacer que el pulso (*q*) sea ALTO cuando el conteo sea cualquier número distinto de cero”.

Ahora que revisamos el código que describe a este monoestable, vamos a evaluar su rendimiento. La conversión de un circuito analógico tradicional en uno digital, por lo general, ofrece ciertas ventajas y ciertas desventajas. En un CI monoestable estándar, el pulso de salida empieza justo después del disparo. Para el monoestable digital que describimos aquí, el pulso de salida comienza en el siguiente flanco de reloj y dura todo el tiempo que el contador sea mayor que cero. Esta situación se muestra en la figura 7-90 durante los primeros ms de la simulación. Observe que el disparo cambia a nivel alto casi 0.5 ms después de que responde la salida *q*. Si ocurre otro evento de disparo mientras el contador está contando hacia abajo (como el que ocurre justo antes de los 3 ms), se ignora. Ésta es la característica no redispensible.

Otro punto a recalcar para este monoestable digital es que el pulso de disparo debe ser lo bastante largo como para verse como un nivel ALTO en el flanco de subida del reloj. Casi en la marca de los 4.5 ms se produce un pulso en la entrada de disparo, pero cambia a BAJO antes del flanco de subida del reloj. Este circuito *no* responde a este evento de entrada. Justo después de los 5 ms la entrada de disparo cambia a ALTO y permanece ahí. El pulso dura exactamente 6 ms, pero debido a que la entrada de disparo permanece en ALTO, responde con otro pulso de salida un ciclo de reloj después. La razón de esta situación es que este circuito se dispara por nivel en vez de dispararse por flanco, como la mayoría de los CIs monoestables convencionales.

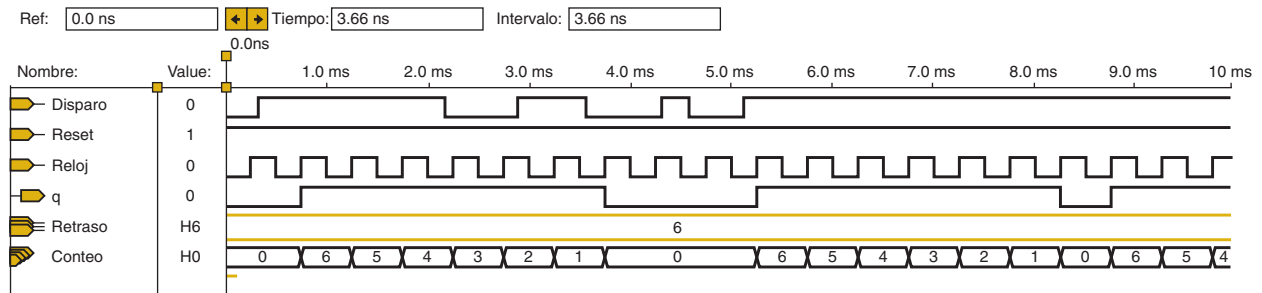


FIGURA 7-90 Simulación de los monoestables no redisparables.

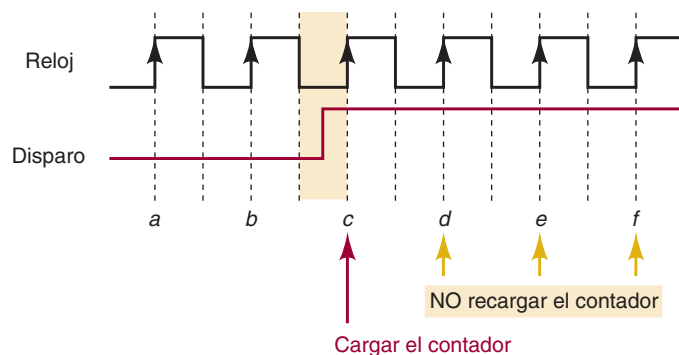
Monoestables redisparables activados por flanco en HDL

Muchas aplicaciones de los monoestables requieren que el circuito responda a un flanco en vez de a un nivel. ¿Cómo podemos utilizar código de HDL para hacer que el circuito responda una vez a cada transición positiva en su entrada de disparo? La técnica que describiremos aquí se conoce como *captura de flancos* y ha sido una herramienta muy útil en la programación de microcontroladores durante años. Como veremos, también es muy útil para describir el disparo por flanco para un circuito digital mediante el uso de HDL. En esta sección ilustraremos un ejemplo de un monoestable redisparable y también demostraremos la captura de flancos, que puede ser útil en muchas otras situaciones.

La operación general de este monoestable redisparable requiere que responda a un flanco de subida de la entrada de disparo. Tan pronto como se detecta el flanco, debe empezar a sincronizar el pulso. En el monoestable digital esto significa que debe cargar el contador lo más pronto posible después del flanco de disparo y debe empezar a contar en forma descendente hasta cero. Si ocurre otro evento de disparo (flanco de subida) antes de que termine el pulso, el contador se recarga de inmediato y la sincronización del pulso empieza de nuevo a partir del principio, con lo cual se sostiene el pulso. Si se activa la señal de reset en cualquier punto se forzará al contador a regresar a cero y se terminará el pulso. La anchura mínima del pulso de salida es tan sólo el número que se aplica a la entrada, multiplicado por el periodo de reloj.

La figura 7-91 demuestra la estrategia de captura de flancos para un monoestable. En cada flanco activo del reloj hay dos partes importantes de información. La primera es el estado de la entrada *disparo ahora* y la segunda es el estado de la entrada *disparo* cuando ocurrió el último flanco activo de reloj. Empiece con el punto *a* en el diagrama de la figura 7-91 y determine estos dos valores, después avance al punto *b* y así en lo sucesivo. Cuando complete esta tarea llegará a la conclusión de que, en el punto *c* se ha obtenido un resultado único. El *disparo* está en ALTO ahora pero estaba en BAJO durante el último flanco activo del reloj. Éste es el punto en el que detectamos el evento de *disparo* por flanco.

FIGURA 7-91
Detección de flancos.



Para poder saber cuál era el *disparo* en el último flanco activo del reloj, el sistema debe recordar el último valor que *tenía el disparo en ese punto*. Para ello hay que almacenar el valor del bit de disparo en un flip-flop. Recuerde que vimos un concepto similar en el capítulo 5, cuando hablamos sobre el uso de un flip-flop para detectar una secuencia. El código para un monoestable se escribe de forma que el contador se cargue sólo después de detectar un flanco de subida en la entrada *disparo*.

MONOESTABLE REDISPARABLE EN AHDL, DISPARADO POR FLANCO

Las primeras cinco líneas de la figura 7-92 son idénticas al ejemplo anterior del monoestable no redispensible. En AHDL, la única forma de recordar un valor que se obtuvo en el pasado es almacenarlo en un flip-flop. En esta sección utilizaremos un flip-flop llamado *disp_era* (línea 9) para almacenar el valor que había en el disparo, durante el último flanco activo del reloj. Este flip-flop se conecta de manera que el disparo esté en su entrada *D* (línea 14) y el reloj esté conectado a su entrada *clk* (línea 13). La salida *Q* de *disp_era* recuerda el valor de *disparo* justo hasta el siguiente flanco de reloj. En este punto utilizamos la línea 16 para evaluar si se ha producido un flanco de disparo. Si *disparo* está en ALTO (ahora) pero estaba en BAJO (último ciclo de reloj), es tiempo de cargar el contador (línea 17). La línea 18 asegura que una vez que el *conteo* llegue a cero, permanecerá en cero hasta que se produzca un nuevo disparo. Si las decisiones permiten evaluar la línea 19, significa que hay un valor cargado en el contador y no es cero, por lo que necesita decrementarse. Por último, el pulso de salida se cambia a ALTO cada vez que hay un valor distinto de 000 en el contador, como vimos antes.

FIGURA 7-92

Descripción de un circuito monoestable redispensible en AHDL, con disparo por flanco.

```

1  SUBDESIGN fig7 92
2  (
3      reloj, disparo, restablecer  : INPUT;
4      delay[3..0]                  : INPUT;
5      q                             : OUTPUT;
6  )
7  VARIABLE
8      conteo[3..0]                  : DFF;
9      disp era                      : DFF;
10 BEGIN
11     conteo[].clk = reloj;
12     conteo[].clrn = restablecer;
13     disp era.clk = reloj;
14     disp era.d = disparo;
15
16     IF disparo & !disp era.q THEN
17         conteo[].d = retraso[];
18     ELSIF conteo[].q == B"0000" THEN conteo[].d = B"0000";
19     ELSE conteo[].d = conteo[].q  1;
20     END IF;
21     q = conteo[].q != B"0000";
22 END;
```


DESCRIPCIÓN VHDL DE MONOESTABLE REDISPARABLE DISPARADO POR FLANCO

La descripción ENTITY en la figura 7-93 es idéntica al ejemplo anterior del monoestable no redisparable. De hecho, las únicas diferencias entre este ejemplo y el que se muestra en la figura 7-89 están relacionadas con la lógica del proceso de decisión. Cuando queremos recordar un valor en VHDL, debemos almacenarlo en una VARIABLE. Recuerde que podemos considerar un bloque PROCESS como una descripción de lo que ocurre cada vez que una señal en la lista de sensibilidad cambia de estado. Una VARIABLE retiene el último valor que se le asigna entre los tiempos en los que se invoca el proceso. En este sentido actúa como un flip-flop. Para el monoestable necesitamos almacenar un valor que nos indique en qué estado se encontraba el disparo en el último flanco activo del reloj. La línea 11 declara un bit variable para servir a este fin. La primera decisión (línea 13) es la de predominancia que comprueba y responde a la entrada *reset*. Observe que éste es un control asíncrono, ya que se evalúa antes de que se detecte el flanco del reloj en la línea 14. Esta línea determina que se ha producido un flanco de subida del reloj, y después se evalúa la lógica principal de este proceso entre las líneas 15 y 20.

Cuando se produce un flanco de reloj, existe una de tres condiciones:

1. Se ha producido un flanco de disparo y debemos cargar el contador.
2. El contador está en cero y necesitamos mantenerlo en cero.
3. El contador no está en cero y necesitamos contar uno hacia abajo.

```

1  ENTITY fig7 93 IS
2  PORT (   reloj, disparo, reset   : IN BIT;
3          retraso                  : IN INTEGER RANGE 0 TO 15;
4          q                        : OUT BIT);
5  END fig7 93;
6
7  ARCHITECTURE vhd1 OF fig7 93 IS
8  BEGIN
9      PROCESS (reloj, reset)
10         VARIABLE conteo          : INTEGER RANGE 0 TO 15;
11         VARIABLE disp era        : BIT;
12     BEGIN
13         IF reset = '0' THEN conteo := 0;
14         ELSIF (reloj'EVENT AND reloj = '1' ) THEN
15             IF disparo = '1' AND disp era = '0' THEN
16                 conteo := retraso;           carga el contador
17                 disp era := '1';             "recuerda" el flanco detectado
18             ELSIF conteo = 0 THEN conteo := 0;   retiene en 0
19             ELSE conteo := conteo  1;          decrementa
20             END IF;
21             IF disparo = '0' THEN disp era := '0';
22             END IF;
23         END IF;
24         IF conteo /= 0 THEN q <= '1';
25         ELSE q <= '0';
26         END IF;
27     END PROCESS;
28 END vhd1;

```

FIGURA 7-93 Monoestable redisparable en VHDL, con disparo por flanco.

Recuerde que es muy importante considerar el orden en el que se hacen las preguntas y las asignaciones en las instrucciones PROCESS de VHDL, ya que la *secuencia* afecta la operación del circuito que estamos describiendo. El código que actualiza la variable *disp_era* debe ocurrir después de la evaluación de su condición anterior. Por esta razón, las condiciones necesarias para detectar un flanco de subida en *disparo* se evalúan en la línea 15. Si se produjo un flanco, entonces se carga el contador (línea 16) y la variable se actualiza (línea 17) para recordar esto para la próxima vez. Si no se produjo un flanco de disparo, el código se retiene en cero (línea 18) o cuenta hacia abajo (línea 19). La línea 21 se asegura de que tan pronto como la entrada de *disparo* cambie a BAJO, la variable *disp_era* recuerde esto mediante la operación reset. Por último, las líneas 24 a 25 se utilizan para crear el pulso de salida durante el tiempo en el que el contador no sea cero.

Las dos mejoras que se hicieron en este monoestable, en comparación con el otro ejemplo, son el disparo por flanco y la característica de ser redisparable. La figura 7-94 evalúa las nuevas características de rendimiento. Observe en el primer ms del diagrama de sincronización que se detecta un flanco de disparo, pero la respuesta no es inmediata. El pulso de salida cambia a nivel alto en el siguiente flanco del reloj. Ésta es una desventaja para el monoestable digital. La característica de redisparable se demuestra casi en la marca de los 2 ms. Observe que *disparo* cambia a nivel alto y en el siguiente flanco de reloj el *conteo* empieza de nuevo en 5, con lo que se sostiene el pulso de salida. Observe además que aún después de que se termina el pulso de salida *q* y el *disparo* se encuentra todavía en ALTO, el circuito monoestable no dispara otro pulso ya que no se dispara por nivel, sino por flanco. En la marca de los 6 ms se produce un pulso de disparo corto pero se ignora debido a que no permanece en ALTO sino hasta el siguiente ciclo de reloj. Por otro lado, un pulso de disparo aún más corto que se produce justo después de la marca de los 7 ms dispara el monoestable, ya que está presente durante el flanco de subida del reloj. El pulso de salida resultante dura exactamente cinco ciclos de reloj, ya que no se producen otros disparos durante este periodo.

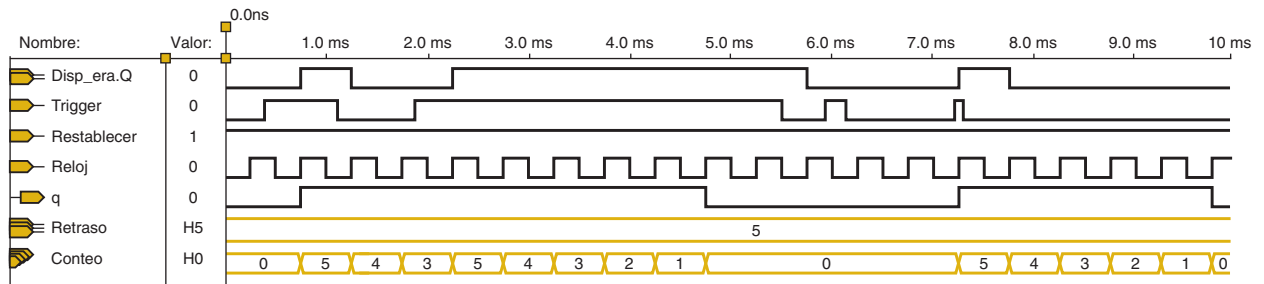


FIGURA 7-94 Simulación del monoestable redisparable, con disparo por flanco.

Para minimizar los efectos de la respuesta retrasada a los flancos de disparo y la posibilidad de dejar pasar los flancos de disparo que sean demasiado cortos, podemos mejorar este circuito de una forma muy sencilla. La frecuencia de reloj y el número de bits que se utilizan para cargar el valor de retraso pueden incrementarse para proporcionar el mismo rango de anchuras de pulso (con un control más preciso) al tiempo que se reduce la anchura mínima del pulso de disparo. Para poder curar este problema por completo, el monoestable debe responder en forma asíncrona a la entrada de disparo. Esto es posible tanto en AHDL como en VHDL, pero siempre se producirá un pulso que fluctúe en anchura de hasta un periodo de reloj.

PREGUNTAS DE REPASO

1. ¿Qué señal de entrada de control tiene la prioridad más alta para cada una de las descripciones de un circuito monoestable?
2. Nombre dos factores que determinen cuánto tiempo durará un pulso de un circuito monoestable digital.
3. Para los circuitos monoestables que se muestran en esta sección, ¿los contadores se cargan en forma síncrona o asíncrona?
4. ¿Cuál es la ventaja de cargar un contador en forma síncrona?
5. ¿Cuál es la ventaja de cargar el contador en forma asíncrona?
6. ¿Cuáles son las dos piezas de información necesarias para detectar un flanco?

RESUMEN PARTE 2

1. Hay numerosos registros de CI disponibles y pueden clasificarse dependiendo de si sus entradas son en paralelo (todos los bits se introducen en forma simultánea), en serie (un bit a la vez) o ambas. De igual forma, los registros pueden tener salidas en paralelo (todos los bits disponibles al mismo tiempo) o en serie (un bit disponible a la vez).
2. Un sistema lógico secuencial utiliza FFs, contadores y registros, además de compuertas lógicas. Sus salidas y la secuencia de las operaciones dependen de las entradas presentes y pasadas.
3. El diagnóstico de fallas en un sistema lógico secuencial comienza con la observación de la operación del sistema, seguida de un razonamiento analítico para determinar las posibles causas de cualquier falla y por último se realizan mediciones de prueba para aislar la verdadera falla.
4. Un contador de anillo es en realidad un registro de desplazamiento de N bits, que recircula un solo 1 en forma continua, actuando en consecuencia como un contador MOD- N . Un contador Johnson es un contador de anillo modificado que opera como un contador MOD- $2N$.
5. Para implementar los registros de desplazamiento con HDL se escriben descripciones personalizadas de su operación.
6. Es muy importante comprender el funcionamiento de los arreglos de bits/vectores de bits y su notación para describir las operaciones de los registros de desplazamiento.
7. Los contadores de registro de desplazamiento tales como los contadores Johnson y de anillo pueden implementarse con facilidad en HDL. Es fácil escribir las características de decodificación e inicio automático en la descripción.
8. Los monoestables digitales se implementan con un contador cargado con un valor de retraso cuando se detecta la entrada de disparo y cuenta en forma descendente hasta cero. Durante el tiempo del conteo descendente, el pulso de salida se mantiene en ALTO.
9. Mediante la colocación estratégica de las instrucciones de descripción de hardware, se puede hacer que los circuitos monoestables en HDL se disparen por flanco o por nivel, y que sean redisparesables o no redisparesables, ya que producen un pulso de salida que responde en forma síncrona o asíncrona al disparo.

TÉRMINOS IMPORTANTES PARTE 2

concatenación	entrada en paralelo/salida	registro de desplazamiento
contador de anillo	en serie	circulante
contador Johnson	entrada en serie/salida	sistema lógico
(contador de anillo	en paralelo	secuencial
torcido)	entrada en serie/salida	
entrada en paralelo/salida	en serie	
en paralelo	monoestable digital	

PROBLEMAS

PARTE 1

SECCIÓN 7-1

- B** 7-1.* Agregue otro flip-flop llamado *E* al contador de la figura 7-1. La señal de reloj es una onda cuadrada de 8-MHz.
- (a) ¿Cuál será la frecuencia en la salida *E*? ¿Cuál será el ciclo de trabajo de esta señal?
 - (b) Repita el inciso (a) si la señal de reloj tiene un ciclo de trabajo del 20 por ciento.
 - (c) ¿Cuál será la frecuencia en la salida *C*?
 - (d) ¿Cuál es el número MOD de este contador?
- B** 7-2. Dibuje un contador binario que convierta una señal de pulso de 64-kHz en una onda cuadrada de 1-kHz.
- B** 7-3.* Suponga que un contador binario de cinco bits comienza en el estado 00000. ¿Cuál será el conteo después de 144 pulsos de entrada?
- B** 7-4. A un contador de rizo de 10 bits se le aplica una señal de reloj de 256-kHz.
- (a) ¿Cuál es el número MOD de este contador?
 - (b) ¿Cuál será la frecuencia en la salida del MSB?
 - (c) ¿Cuál será el ciclo de trabajo de la señal del MSB?
 - (d) Suponga que el contador empieza en cero. ¿Cuál será el conteo en hexadecimal después de 1000 pulsos de entrada?

SECCIÓN 7-2

- 7-5.* Un contador de rizo de cuatro bits se controla mediante una señal de reloj de 20-MHz. Dibuje las formas de onda en la salida de cada FF si cada uno tiene un $t_{pd} = 20$ ns. Determine cuáles estados del contador (si es que los hay) no se producirán debido a los retrasos de propagación.
- 7-6. (a) ¿Cuál es la máxima frecuencia de reloj que puede utilizarse con el contador del problema 7-5?
- (b) ¿Cuál sería $f_{m\acute{a}x}$ si el contador se expandiera a seis bits?

SECCIONES 7-3 Y 7-4

- B** 7-7.* (a) Dibuje el diagrama del circuito para un contador síncrono MOD-32.
- (b) Determine $f_{m\acute{a}x}$ para este contador si cada FF tiene un $t_{pd} = 20$ ns y cada compuerta tiene un $t_{pd} = 10$ ns.
- B** 7-8. (a) Dibuje el diagrama del circuito para un contador síncrono MOD-64.
- (b) Determine $f_{m\acute{a}x}$ para este contador si cada FF tiene un $t_{pd} = 20$ ns y cada compuerta tiene un $t_{pd} = 10$ ns.
- B** 7-9.* Dibuje las formas de onda para todos los FFs en el contador de décadas de la figura 7-8(b), en respuesta a una frecuencia de reloj de 1-kHz. Muestre cualquier deformación que pudiera aparecer en cualquiera de las salidas de los FFs. Determine la frecuencia en la salida *D*.
- B** 7-10. Repita el problema 7-9 para el contador de la figura 7-8(a).
- B** 7-11.* Cambie las entradas para la compuerta NAND de la figura 7-9, de manera que el contador divida la frecuencia de entrada entre 50.
- B** 7-12. Dibuje un contador síncrono que produzca como salida una señal de 10-kHz cuando se aplica un reloj de 1-MHz.

* Encontrará las respuestas a los problemas marcados con asterisco al final del libro.

SECCIONES 7-5 Y 7-6

- B** 7-13.* Dibuje un contador descendente síncrono, MOD-32.
- B** 7-14. Dibuje un contador ascendente/descendente síncrono, MOD-16. La dirección de conteo se controla mediante *dir* (*dir* = 0 para contar hacia arriba).
- A, F** 7-15.* Determine la secuencia de conteo del contador ascendente/descendente en la figura 7-11, si la salida INVERSOR estuviera forzada en ALTO. Suponga que el contador empieza en 000.
- 7-16. Complete el diagrama de sincronización de la figura 7-95 para el contador preajustable de la figura 7-12. Observe que la condición inicial para el contador se proporciona en el diagrama de sincronización.

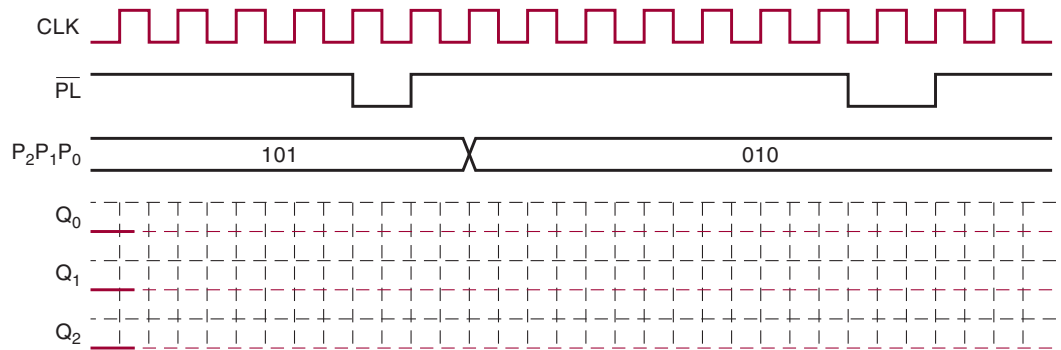


FIGURA 7-95 Diagrama de sincronización del problema 7-16.

SECCIÓN 7.7

- 7-17.* Complete el diagrama de sincronización de la figura 7-96 para un 74ALS161, al cual se le aplican las formas de onda de entrada que se indican. Suponga que el estado inicial es 0000.

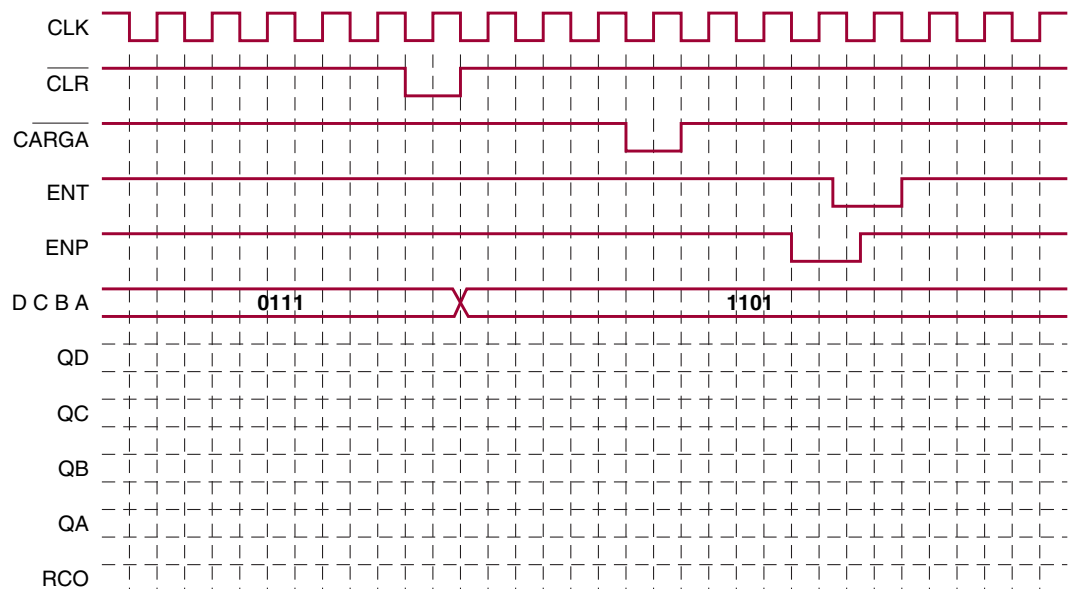


FIGURA 7-96 Diagrama de sincronización del problema 7-17.

- 7-18. Complete el diagrama de sincronización de la figura 7-97 para el 74ALS162, al cual se le aplican las formas de onda de entrada que se indican. Suponga que el estado inicial es 0000.

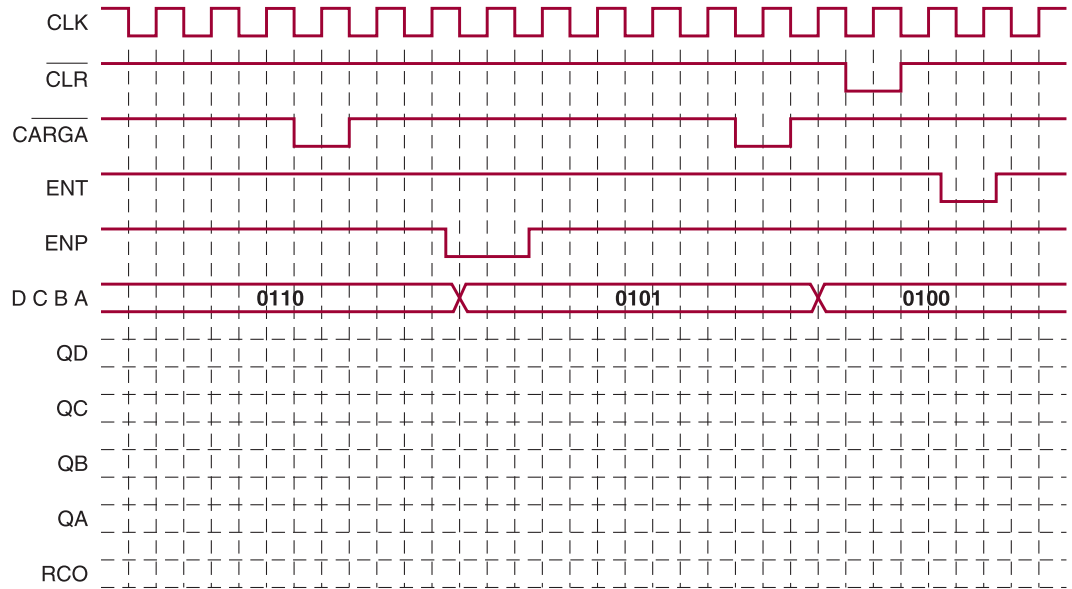


FIGURA 7-97 Diagrama de sincronización del problema 7-18.

- 7-19.* Complete el diagrama de sincronización de la figura 7-98 para un 74ALS190, al cual se le aplican las formas de onda de entrada que se indican. La entrada DCBA es 0101.

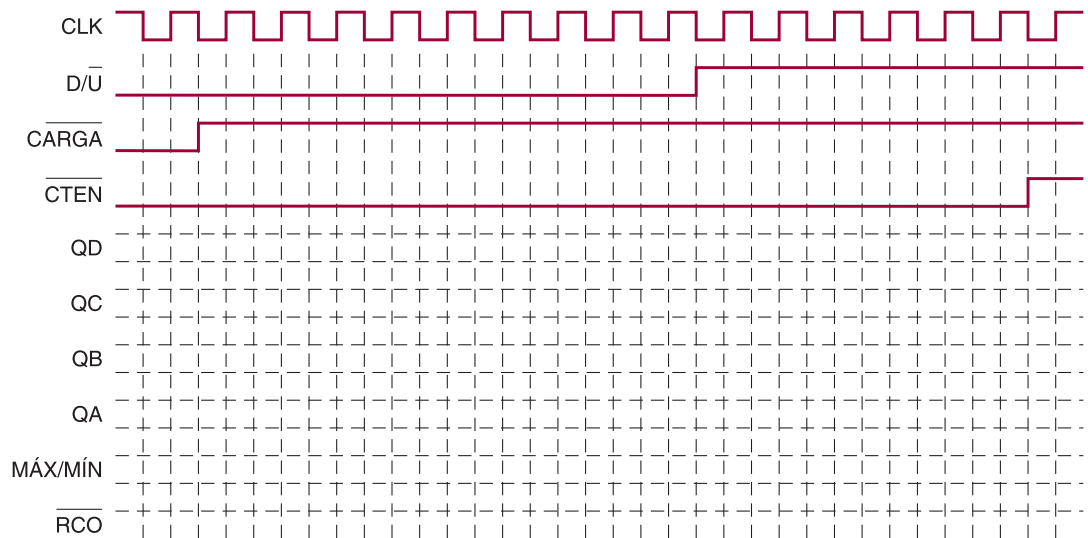


FIGURA 7-98 Diagrama de sincronización de los problemas 7-19 y 7-20.

- 7-20. Repita el problema 7-19 para un 74ALS191 y una entrada DCBA de 1100.

- B** 7-21.* Consulte el circuito del contador del CI en la figura 7-99(a):
- Dibuje el diagrama de transición de estados para las salidas QD QC QB QA del contador.
 - Determine el módulo del contador.

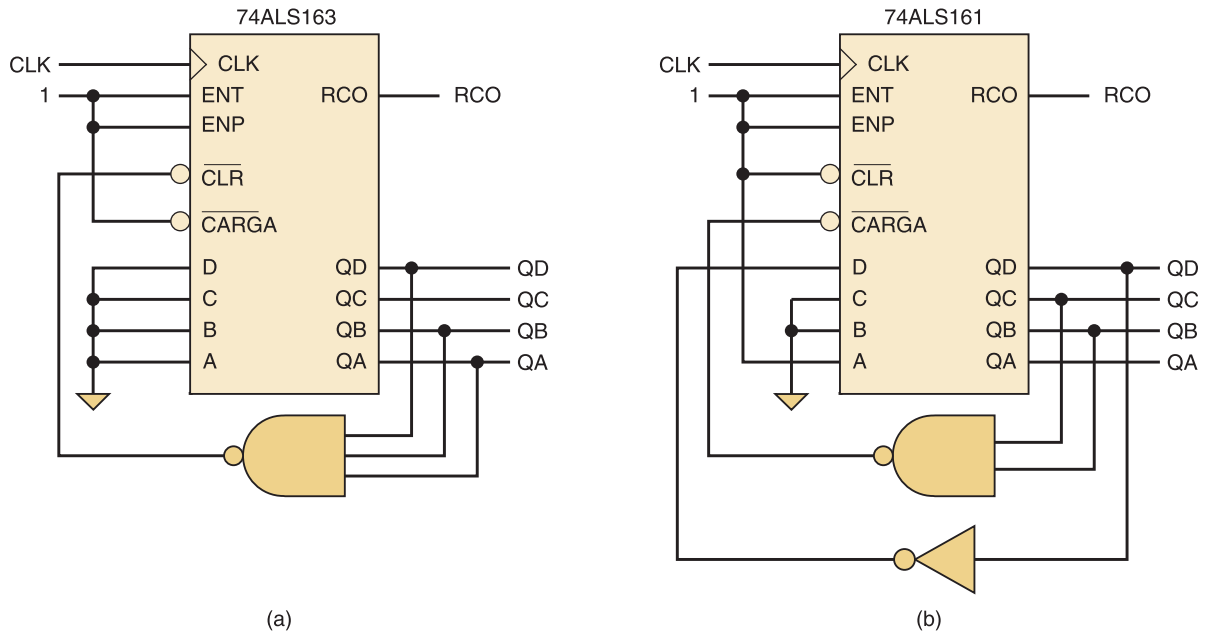


FIGURA 7-99 Problemas 7-21 y 7-22.

- (c) ¿Cuál es la relación entre la frecuencia de salida del MSB con la frecuencia de entrada de CLK ?
- (d) ¿Cuál es el ciclo de trabajo de la forma de onda de salida del MSB?
- B** 7-22. Repita el problema 7-21 para el circuito contador de CI en la figura 7-99(b).
- B** 7-23.* Consulte el circuito contador de CI en la figura 7-100(a).
- (a) Dibuje el diagrama de sincronización para las salidas QD QC QB QA .
- (b) ¿Cuál es el módulo del contador?
- (c) ¿Cuál es la secuencia de conteo? ¿Cuenta hacia arriba o hacia abajo?
- (d) ¿Podemos producir el mismo módulo con un 74HC190? ¿Podemos producir la misma secuencia de conteo con un 74HC190?

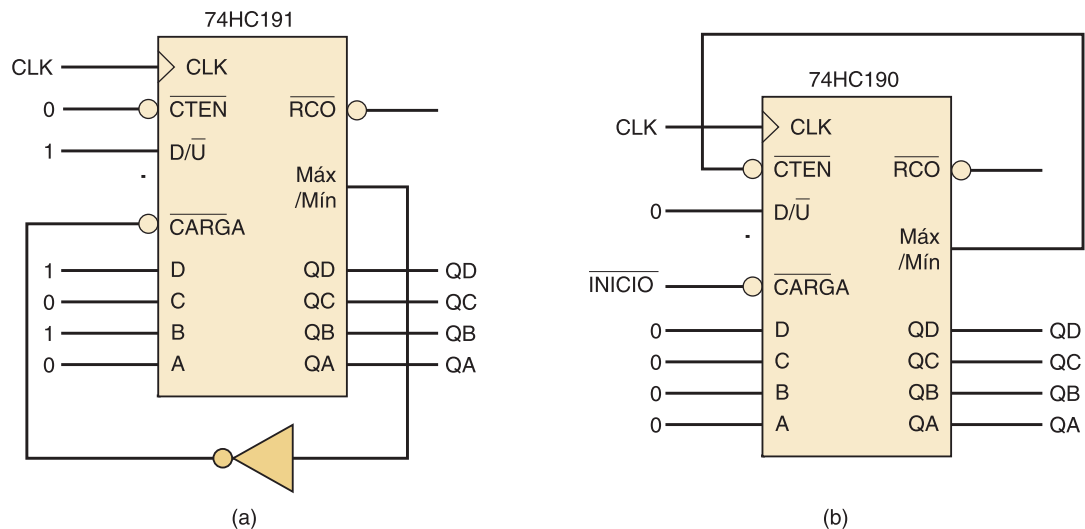


FIGURA 7-100 Problemas 7-23 y 7-24.

- 7-24. Consulte el circuito contador de CI en la figura 7-100(b):
- Describa la salida del contador en $QD\ QC\ QB\ QA$ si $\overline{INICIO}$ está en BAJO.
 - Describa la salida del contador en $QD\ QC\ QB\ QA$ si se aplica un pulso momentáneo a $\overline{INICIO}$ para que cambie a BAJO y después regresa al nivel ALTO.
 - ¿Cuál es el módulo del contador? ¿Es un contador reciclable?
- D** 7-25.* Dibuje un diagrama esquemático para crear un contador MOD-6 reciclable que utilice:
- el control borrar en un 74ALS160.
 - el control borrar en un 74ALS162.
- D** 7-26. Dibuje un diagrama esquemático para crear un contador MOD-6 reciclable que produzca la secuencia de conteo:
- 1, 2, 3, 4, 5, 6 y se repita con un 74ALS162.
 - 5, 4, 3, 2, 1, 0 y se repita con un 74ALS190.
 - 6, 5, 4, 3, 2, 1 y se repita con un 74ALS190.
- D** 7-27.* Diseñe un contador binario MOD-100; para ello utilice dos CIs 74HC161 o dos CIs 74HC163 y todas las compuertas necesarias. Los CIs contadores deberán conectarse en cascada en forma síncrona para producir la secuencia de conteo binaria del 0 al 99. El contador MOD-100 deberá tener dos entradas de control, una de habilitación de conteo activa en BAJO ($\overline{EN}$) y una borrar asíncrona activa en BAJO ($\overline{CLR}$). Etiquete a las salidas del contador como $Q0, Q1, Q2$, etc., con $Q0 = \text{LSB}$. ¿Cuál salida es el MSB?
- D** 7-28. Diseñe un contador BCD MOD-100; para ello utilice dos CIs 74HC160 o dos CIs 74HC162 y todas las compuertas necesarias. Los CIs contadores deberán conectarse en cascada en forma síncrona para producir la secuencia de conteo BCD del 0 al 99. El contador MOD-100 deberá tener dos entradas de control, una de habilitación de conteo activa en ALTO ($\overline{EN}$) y una de carga asíncrona activa en ALTO ($\overline{LD}$). Etiquete a las salidas del contador como $Q0, Q1, Q2$, etc., con $Q0 = \text{LSB}$. ¿Cuál conjunto de salidas representa el dígito de los 10s?
- B** 7-29.* Con una entrada de reloj de 6-MHz para un 74ALS163 que tenga las cuatro entradas de control en ALTO, determine la frecuencia de salida y el ciclo de trabajo para cada una de las cinco salidas (incluyendo RCO).
- B** 7-30. Con una entrada de reloj de 6-MHz para un 74ALS162 que tenga las cuatro entradas de control en ALTO, determine la frecuencia de salida y el ciclo de trabajo para cada una de las siguientes salidas: QA, QC, QD, RCO. ¿Qué hay de inusual acerca del patrón de la forma de onda que se produciría mediante la salida QB? Esta característica del patrón produce un ciclo de trabajo indefinido.
- B** 7-31.* La frecuencia de f_{ent} es de 6 MHz en la figura 7-101. Los dos CIs contadores se han conectado en cascada en forma asíncrona, de manera que la frecuencia de salida producida por el contador U1 es la frecuencia de entrada para el contador U2. Determine la frecuencia de salida para f_{sal1} y f_{sal2} .
- B** 7-32. La frecuencia de f_{ent} es de 1.5 MHz en la figura 7-102. Los dos CIs contadores se han conectado en cascada en forma asíncrona, de manera que la frecuencia de salida producida por el contador U1 es la frecuencia de entrada para el contador U2. Determine la frecuencia de salida para f_{sal1} y f_{sal2} .
- D** 7-33.* Diseñe un circuito divisor de frecuencia que produzca las siguientes tres frecuencias de señal de salida: 1.5 MHz, 150 kHz y 100 kHz. Utilice los chips tipo contador 74HC162 y 74HC163, junto con todas las compuertas necesarias. La frecuencia de entrada es de 12 MHz.
- D** 7-34. Diseñe un circuito divisor de frecuencia que produzca las siguientes tres frecuencias de señal de salida: 1 MHz, 800 kHz y 100 kHz. Utilice los CIs tipo contador 74HC160 y 74HC161, junto con todas las compuertas necesarias. La frecuencia de entrada es de 12 MHz.

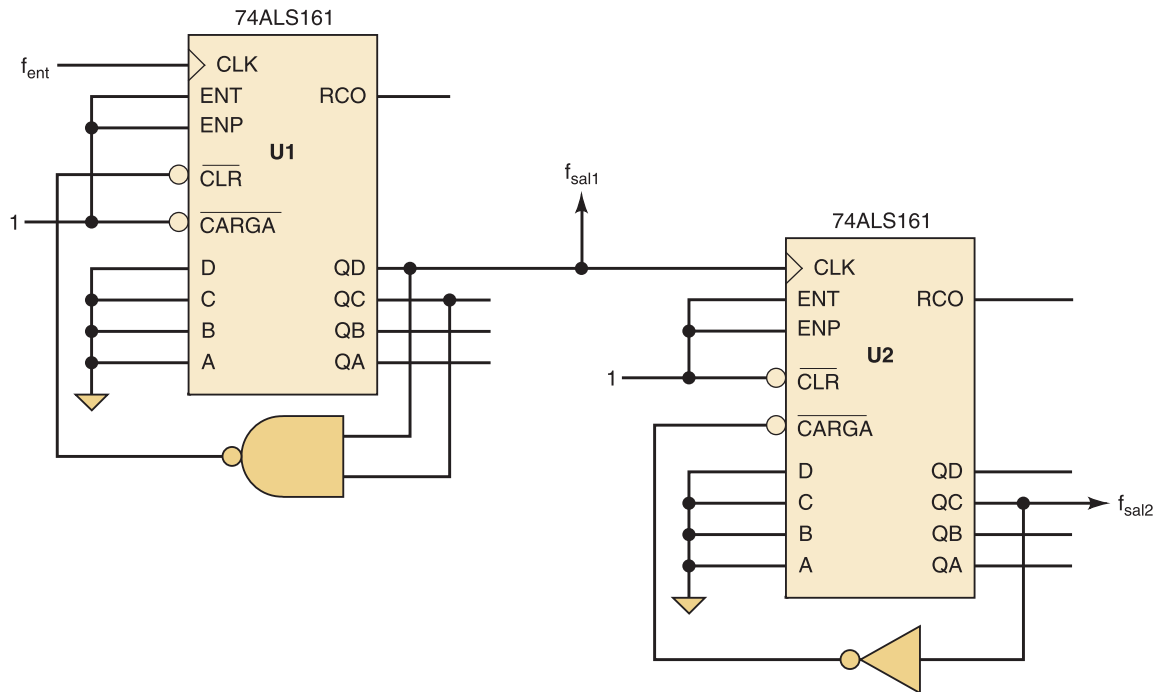


FIGURA 7-101 Problema 7-31.

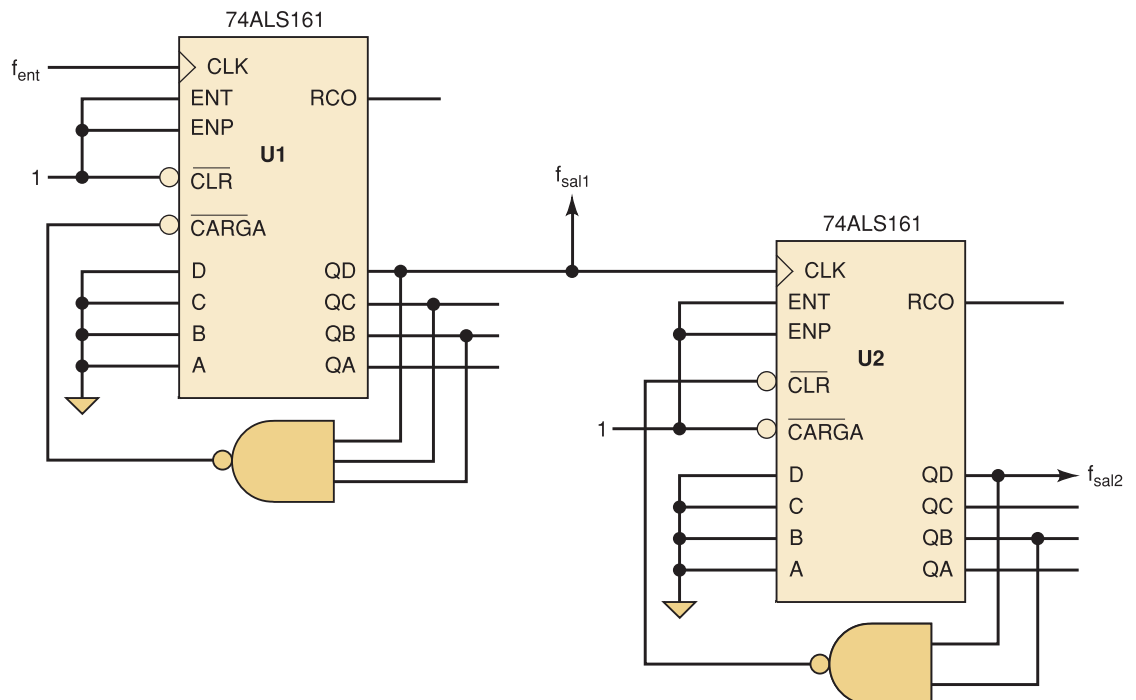


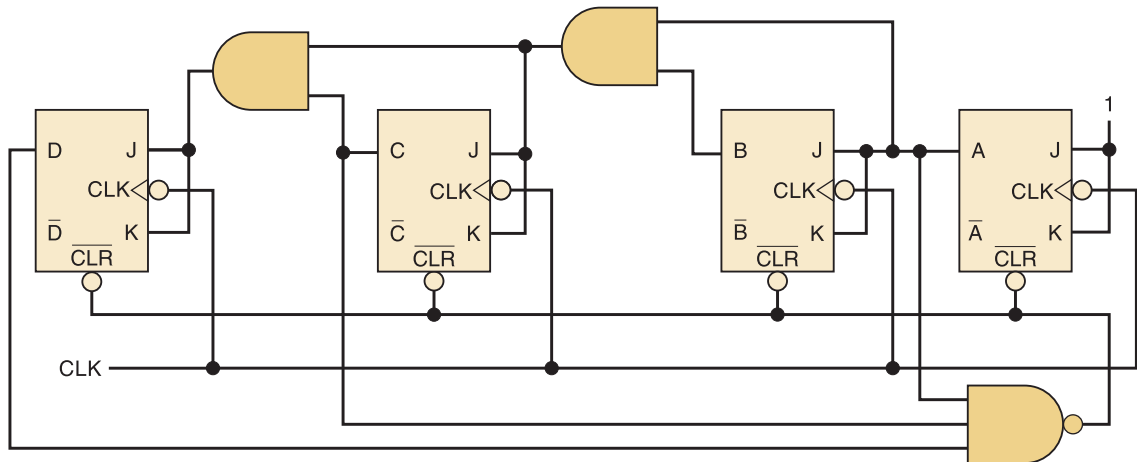
FIGURA 7-102 Problema 7-32.

SECCIÓN 7-8

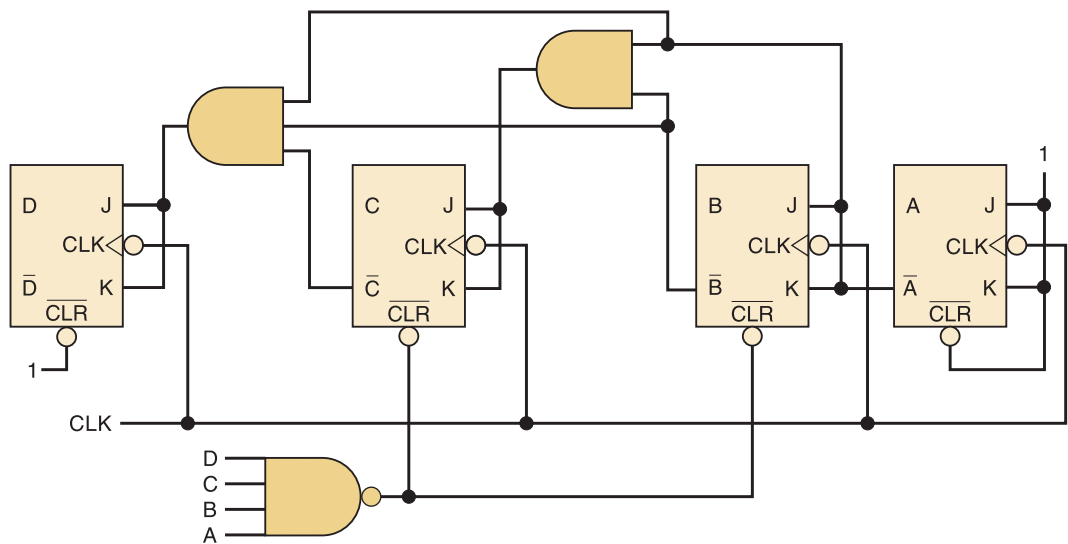
- B** 7-35.* Dibuje las compuertas necesarias para decodificar todos los estados de un contador MOD-16 que utilice salidas activas en BAJO.
- B** 7-36. Dibuje las compuertas AND necesarias para decodificar los 10 estados del contador BCD de la figura 7-8(b).

SECCIÓN 7-9

- A** 7-37.* Analice el contador asíncrono en la figura 7-103(a). Dibuje su diagrama de sincronización y determine el módulo del contador.
- A** 7-38. Repita el problema 7-37 para la figura 7-103(b).



(a)



(b)

FIGURA 7-103 Problemas 7-37 y 7-38.

- A** 7-39.* Analice el contador síncrono de la figura 7-104(a). Dibuje su diagrama de sincronización y determine el módulo del contador.
- A** 7-40. Repita el problema 7-39 para la figura 7-104(b).
- A** 7-41.* Analice el contador síncrono de la figura 7-105(a). F es una entrada de control. Dibuje su diagrama de transición de estados y determine el módulo del contador.
- A** 7-42. Analice el contador síncrono de la figura 7-105(b). Dibuje su diagrama de transición de estados completo y determine el módulo del contador. ¿Es el contador autocorregible?

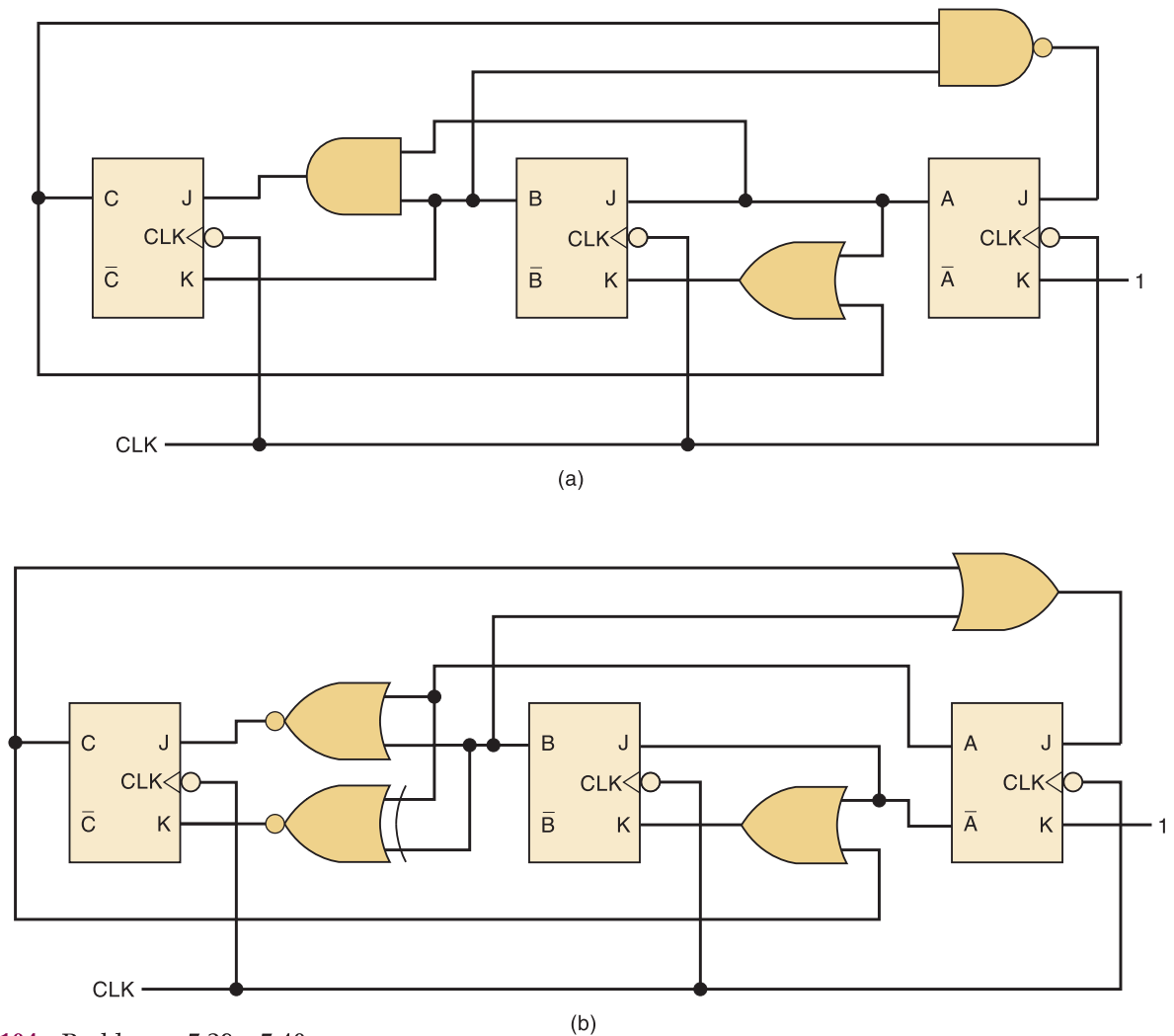


FIGURA 7-104 Problemas 7-39 y 7-40.

SECCIÓN 7-10

- D** 7-43.* Diseñe un contador síncrono; para ello utilice FFs J-K. La secuencia del contador deberá ser: 000, 010, 101, 110 y se debe repetir. Los estados indeseables (no usados) 001, 011, 100 y 111 deberán siempre cambiar a 000 en el siguiente pulso de reloj.
- (b) Rediseñe el contador de la parte (a) de manera que no tenga ningún requerimiento sobre los estados no utilizados; esto es, que sus SIGUIENTES estados puedan ser “no importa”. Compare este diseño con el del inciso (a).
- D** 7-44. Diseñe un contador descendente MOD-5, síncrono y reciclable, que produzca la secuencia 100, 011, 010, 001, 000 y se repita. Utilice flip-flops J-K.
- (a) Haga que los estados no usados cambien a 000 en el siguiente pulso de reloj.
- (b) Utilice SIGUIENTES estados tipo “no importa” para los estados no usados. ¿Es este diseño autocorregible?
- D** 7-45.* Diseñe un contador descendente BCD, síncrono y reciclable; para ello utilice FFs J-K con los estados SIGUIENTES tipo “no importa”.
- D** 7-46. Diseñe un contador ascendente/descendente MOD-7, síncrono y reciclable; para ello utilice FFs J-K. Use los estados del 000 al 110 en el contador. Controle la dirección de conteo mediante la entrada D ($D = 0$ para contar hacia arriba y $D = 1$ para contar hacia abajo).

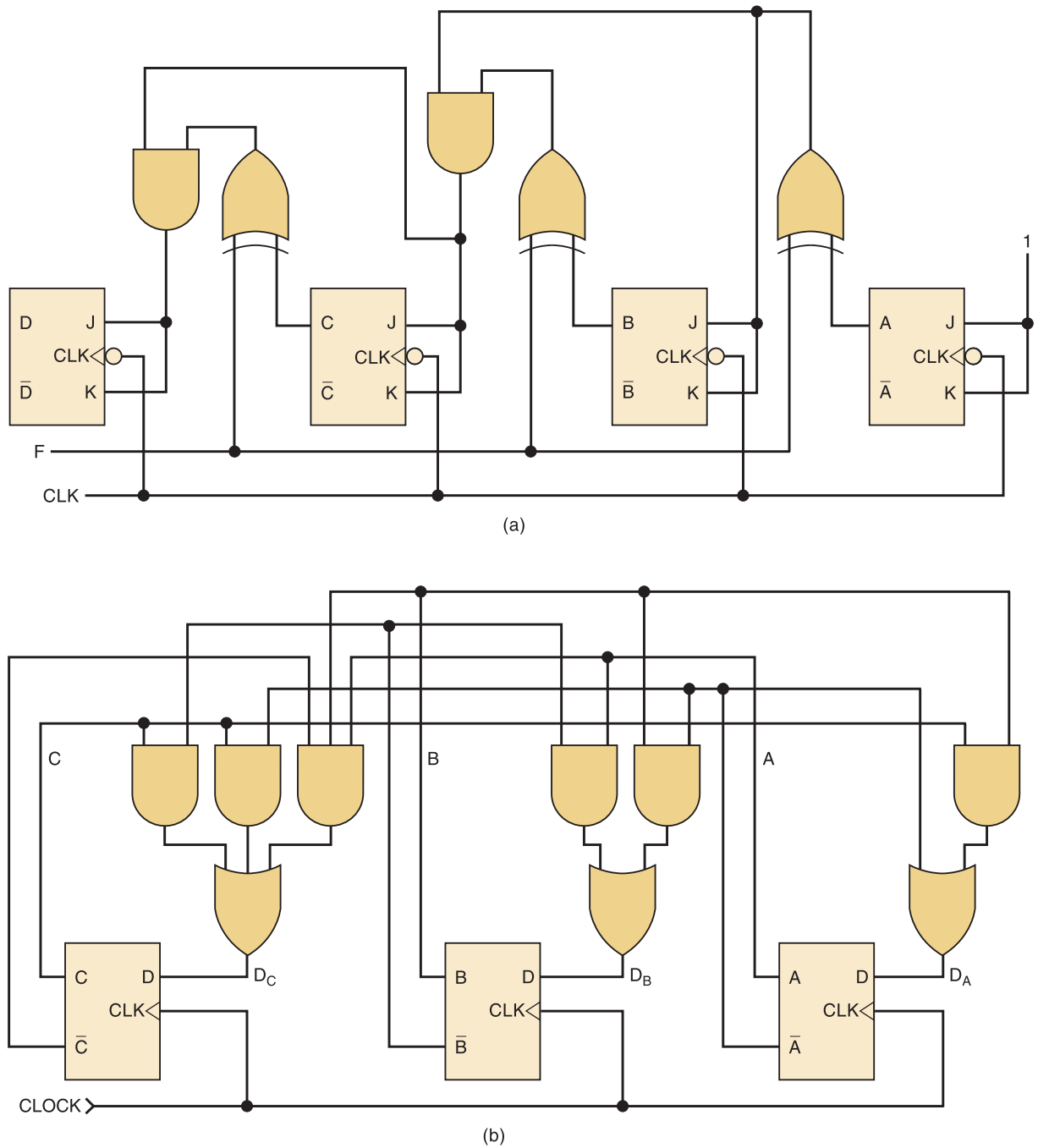


FIGURA 7-105 Problemas 7-41 y 7-42.

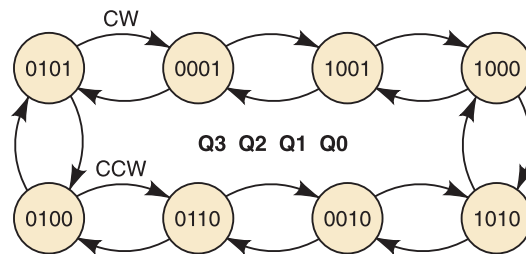
- D** 7-47.* Diseñe un contador descendente binario MOD-8, síncrono y reciclable; para ello utilice flip-flops D.
- D** 7-48. Diseñe un contador MOD-12, síncrono y reciclable; para ello utilice FFs D. Use los estados 0000 al 1011 en el contador.

SECCIONES 7-11 Y 7-12

- H, D** 7-49.* Diseñe un contador ascendente MOD-13 reciclable; para ello utilice un HDL. La secuencia de conteo deberá ser del 0000 al 1100. Simule el contador.

- H, D** 7-50. Diseñe un contador descendente MOD-25 reciclable; para ello utilice un HDL. La secuencia de conteo deberá ser del 11000 al 00000. Simule el contador.
- H, D** 7-51.* Diseñe un contador de código Gray MOD-16 reciclable; para ello utilice un HDL. El contador deberá tener una señal de habilitación activa en ALTO (*cnt*). Simule el contador.
- H, D** 7-52. Diseñe un controlador de medio paso bidireccional para un motor a pasos; para ello utilice un HDL. La entrada de control de dirección (*dir*) producirá un patrón en sentido a favor de las manecillas del reloj (CW) cuando esté en ALTO, o un patrón en sentido en contra de las manecillas del reloj cuando esté en BAJO. En la figura 7-106 se muestra la secuencia. Simule el circuito secuencial.

FIGURA 7-106 Problema 7-52.



- H, D** 7-53.* Diseñe un circuito divisor de frecuencia para producir como salida una señal de 100-kHz; para ello utilice un HDL. La frecuencia de entrada es de 5 MHz. Simule el contador.
- H, D** 7-54. Diseñe un circuito divisor de frecuencia que produzca como salida una de dos señales de frecuencia especificadas; para ello utilice un HDL. La frecuencia de salida se selecciona mediante la entrada de control f_{selec} . El divisor producirá como salida una frecuencia de 5 kHz cuando $f_{selec} = 0$ o de 12 kHz cuando $f_{selec} = 1$. La frecuencia de entrada es de 60 kHz. Simule el contador.
- H, B** 7-55.* Expanda el contador en HDL con todas sus características de la sección 7-12 para que sea un contador MOD-256. Simule el contador.
- H, B** 7-56. Expanda el contador en HDL con todas sus características de la sección 7-12 para que sea un contador MOD-1024. Simule el contador.
- H, B** 7-57.* Diseñe un contador descendente MOD-16 reciclable; para ello utilice un HDL. El contador deberá tener los siguientes controles (de menor a mayor prioridad): habilitación de conteo activa en BAJO (*en*), borrar síncrona activa en ALTO (*clr*) y carga síncrona activa en BAJO (*ld*). Decodifique el conteo terminal cuando se habilite mediante *en*. Simule el contador.
- H, D** 7-58. Diseñe un contador ascendente/descendente MOD-10 reciclable; para ello utilice un HDL. El contador deberá contar hacia arriba cuando *arriba* = 1 y deberá contar hacia abajo cuando *arriba* = 0. También deberá tener los siguientes controles (de menor a mayor prioridad): habilitación de conteo activa en ALTO (*habilita*), carga síncrona activa en ALTO (*carga*) y borrar asíncrona activa en BAJO (*borrar*). Decodifique el conteo terminal cuando se habilite mediante *habilita*. Simule el contador.

SECCIÓN 7-13

- H** 7-59.* Cree un contador BCD MOD-1000; para ello conecte en cascada tres de los módulos de contador BCD en HDL (descritos en la sección 7-13). Simule el contador.
- H** 7-60. Cree un contador binario MOD-256; para ello conecte en cascada dos de los módulos de contador en HDL MOD-16, con todas sus características (descritas en la sección 7-12). Simule el contador.
- H, D** 7-61.* Diseñe un contador BCD MOD-50 síncrono; para ello conecte en cascada los diseños en HDL para un contador MOD-10 y otro contador MOD-5. El

contador MOD-50 deberá tener una señal de habilitación de conteo activa en ALTO (*habilita*) y una señal borrar síncrona activa en BAJO (*clm*). Asegúrese de incluir la detección del conteo terminal para el dígito de los 1s, para conectarlo en cascada con el dígito de los 10s. Simule el contador.

- H, D** 7-62. Diseñe un contador descendente BCD MOD-100 síncrono; para ello conecte en cascada dos módulos de contador descendente MOD-10 en HDL. El contador MOD-100 deberá tener una señal de carga en paralelo síncrona (*carga*). Simule el contador.

SECCIÓN 7-14

- H** 7-63.* Modifique la descripción de HDL de la figura 7-56 o de la figura 7-57 para agregar una secuencia de enjuagado después de lavar la ropa. La nueva secuencia de la máquina de estado deberá ser *inactiva* → *lavar_llena* → *lavar_agitar* → *lavar_girar* → *enjuagar_llena* → *enjuagar_agitar* → *enjuagar_girar* → *inactiva*. Utilice agua caliente para lavar y agua fría para enjuagar (agregue bits de salida para controlar dos válvulas de agua). Simule el diseño en HDL modificado.
- H** 7-64. Simule el diseño del controlador de semáforos en HDL que presentamos en la sección 7-14.

PARTE 2

SECCIONES 7-15 A 7-19

- B** 7-65.* Un conjunto de registros 74ALS174 se conecta como se muestra en la figura 7-107. ¿Qué tipo de transferencia de datos se realiza con cada registro? Determine la salida de cada registro cuando se aplica un pulso momentáneo en *MR* para que cambie a BAJO y después de cada uno de los pulsos de reloj indicados (CP#) en la tabla 7-10. ¿Cuántos pulsos de reloj deben aplicarse antes de que los datos que se introducen en *I5-I0* estén disponibles en *Z5-Z0*?

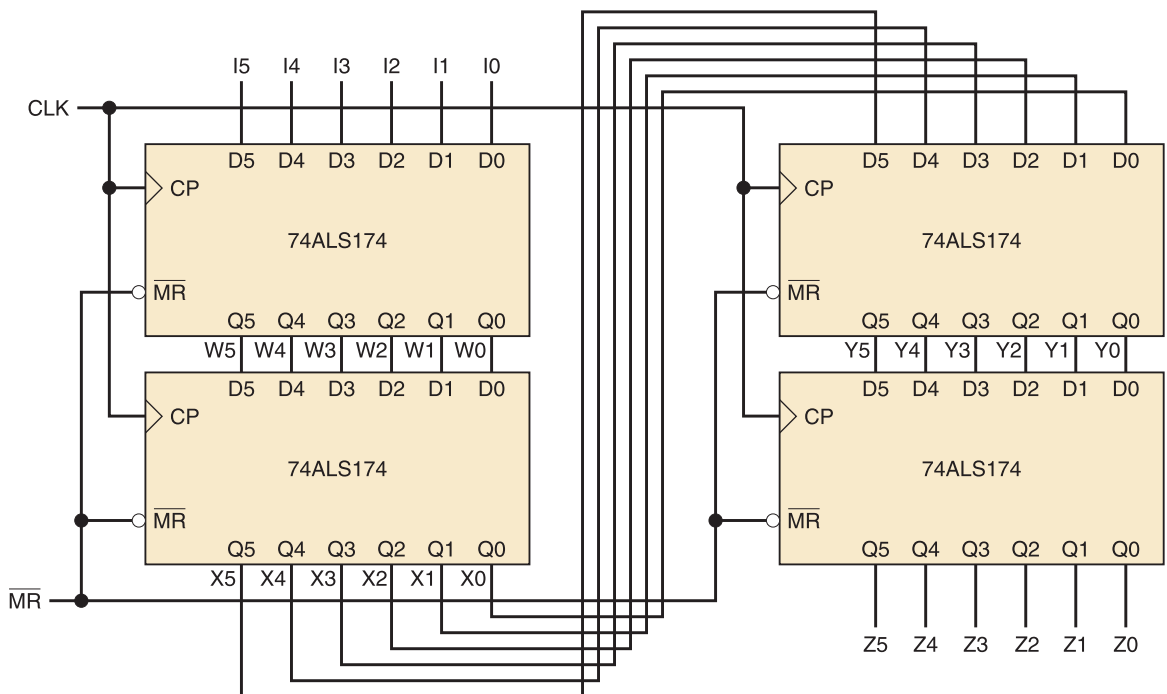


FIGURA 7-107 Problema 7-65.

TABLA 7-10

↑CLK	$\overline{\text{MR}}$	I5–I10	W5–W0	X5–X0	Y5–Y0	Z5–Z0
X	0	101010				
CP1	1	101010				
CP2	1	010101				
CP3	1	000111				
CP4	1	111000				
CP5	1	011011				
CP6	1	001101				
CP7	1	000000				
CP8	1	000000				

B 7-66. Complete el diagrama de sincronización de la figura 7-108 para un 74HC174. ¿Cómo muestra el diagrama de sincronización que la señal restablecer maestra es asíncrona?

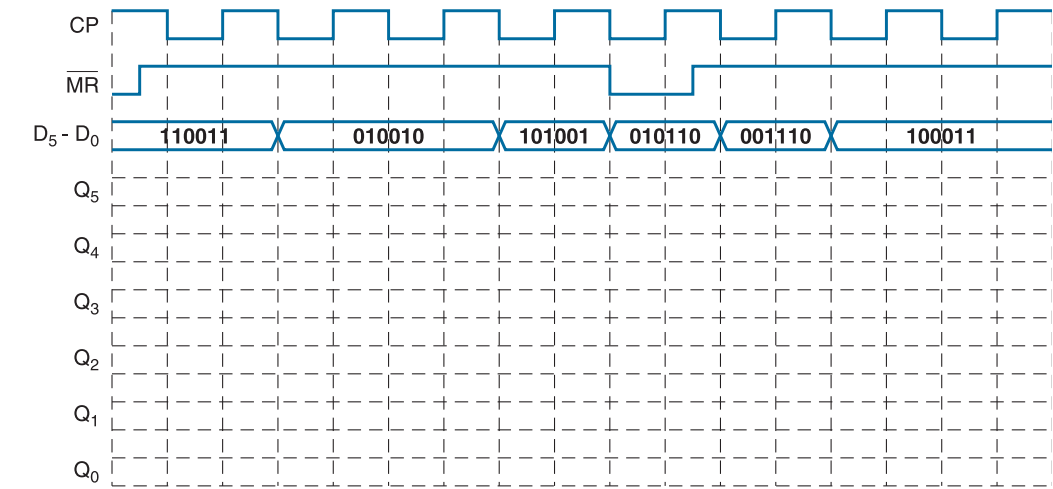


FIGURA 7-108 Problema 7-66.

B 7-67.* ¿Cuántos pulsos de reloj se necesitarán para cargar por completo ocho bits de datos en serie en un 74ALS166? ¿Cómo se relaciona esto con el número de flip-flops que contiene el registro?

B 7-68. Repita el ejemplo 7-18 para las formas de onda de entrada que se proporcionan en la figura 7-109.

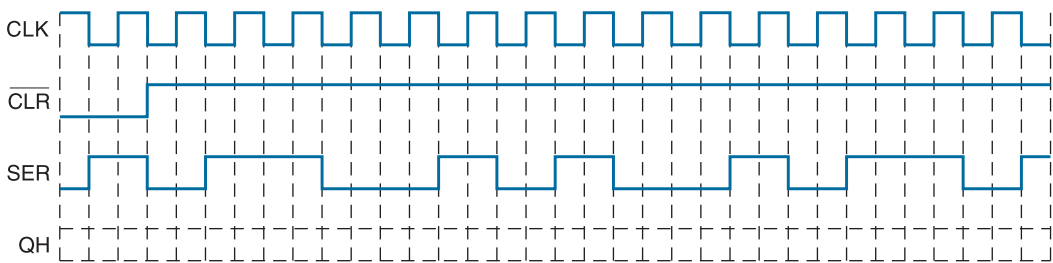


FIGURA 7-109 Problema 7-68.

7-69.* Repita el ejemplo 7-20 con $D_S = 1$ y las formas de onda de entrada que se proporcionan en la figura 7-110.

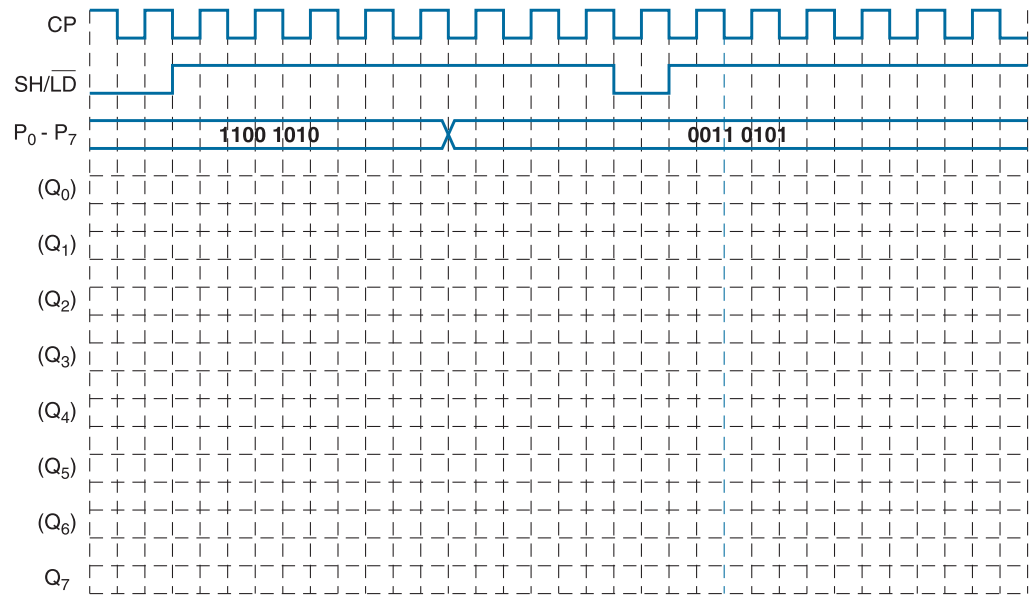
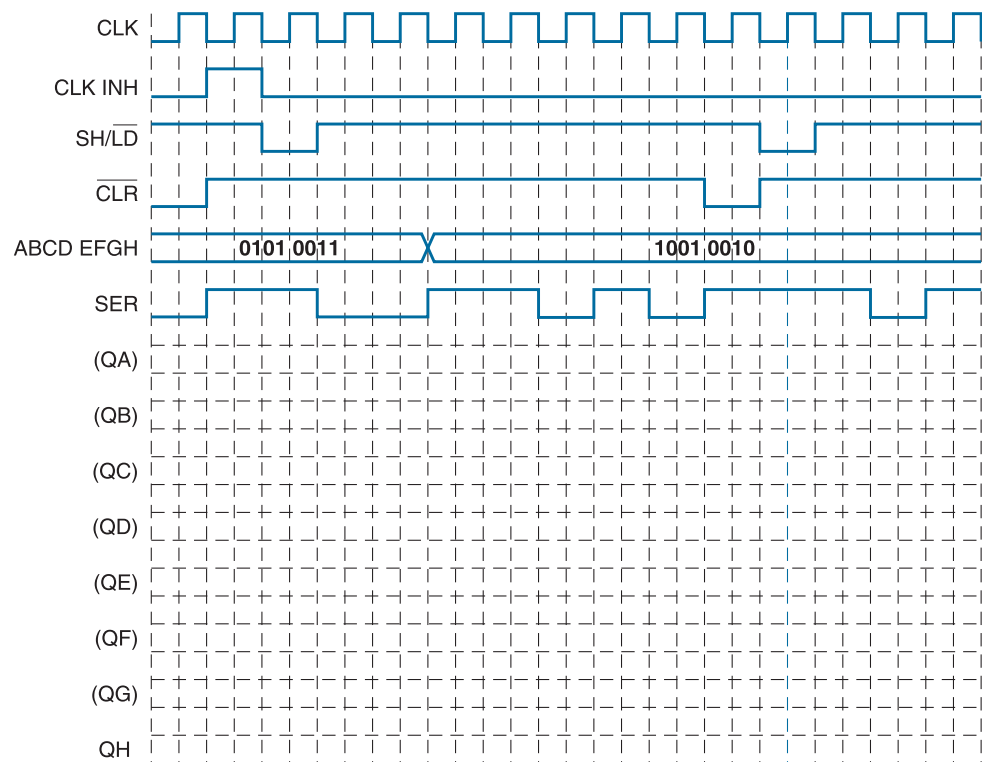


FIGURA 7-110 Problema 7-69.

7-70. Aplique las formas de onda de onda de entrada que se proporcionan en la figura 7-111 a un 74ALS166 y determine la salida que se produce.

FIGURA 7-111
Problema 7-70.



- B** 7-71.* Al examinar el diagrama esquemático para una pieza de equipo, es común que un técnico o un ingeniero se topen con un CI desconocido. En tales casos a menudo es necesario consultar la hoja técnica del fabricante para ver las especificaciones del dispositivo. Investigue la hoja técnica para el registro de desplazamiento universal bidireccional 74AS194, para responder a las siguientes preguntas:

- ¿Es la entrada $\overline{CLR}$ asíncrona o síncrona?
- Verdadero o falso*: cuando CLK está en BAJO, las entradas S_0 y S_1 no tienen efecto sobre el registro.
- Suponga las siguientes condiciones:

$$Q_A \ Q_B \ Q_C \ Q_D = 1 \ 0 \ 1 \ 1$$

$$A \ B \ C \ D = 0 \ 1 \ 1 \ 0$$

$$\overline{CLR} = 1$$

$$SR \ SER = 0$$

$$SL \ SER = 1$$

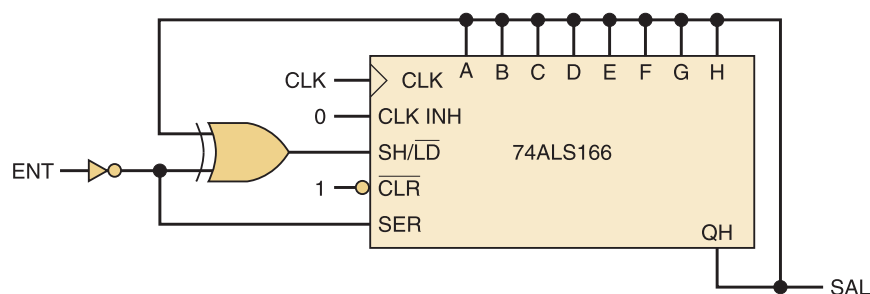
Si $S_0 = 0$ y $S_1 = 1$, determine las salidas del registro después de un pulso en CLK . Después de dos pulsos en CLK . Después de tres. Después de cuatro.

- Utilice las mismas condiciones excepto $S_0 = 1$ y $S_1 = 0$, y repita la parte (c).
- Repita la parte (c) con $S_0 = 1$ y $S_1 = 1$.
- Repita la parte (c) con $S_0 = 0$ y $S_1 = 0$.
- Use las mismas condiciones que en la parte (c), sólo que esta vez suponga que Q_A está conectada a $SL \ SER$. ¿Cuáles serán las salidas del registro después de cuatro pulsos en CLK ?

- A** 7-72. Consulte la figura 7-112 para responder a las siguientes preguntas:

- ¿Qué función del registro (carga o desplazamiento) se realizará en el siguiente ciclo de reloj si $ent = 1$ y $sal = 0$? ¿Qué valor de datos se introducirá cuando se aplique un pulso de reloj?
- ¿Qué función del registro (carga o desplazamiento) se realizará en el siguiente ciclo de reloj si $ent = 0$ y $sal = 1$? ¿Qué valor de datos se introducirá cuando se aplique un pulso de reloj?
- ¿Qué función del registro (carga o desplazamiento) se realizará en el siguiente ciclo de reloj si $ent = 0$ y $sal = 0$? ¿Qué valor de datos se introducirá cuando se aplique un pulso de reloj?
- ¿Qué función del registro (carga o desplazamiento) se realizará en el siguiente ciclo de reloj si $ent = 1$ y $sal = 1$? ¿Qué valor de datos se introducirá cuando se aplique un pulso de reloj?
- ¿Qué condición de entrada hará que en algún momento dado (después de varios pulsos de reloj) la salida cambie de estado?

FIGURA 7-112
Problema 7-72.



- (f) Para cambiar el nivel lógico de salida se requiere que la nueva condición de entrada dure por lo menos ¿cuántos pulsos de reloj?
- (g) Si la señal de entrada cambia de nivel y después regresa a su nivel lógico antes del número de pulsos de reloj especificados en la parte (f), ¿qué ocurre con la señal de salida?
- (h) Explique por qué este circuito puede usarse para eliminar los rebotes en los interruptores.

SECCIÓN 7-20

- B** 7-73.* Dibuje el diagrama para un contador de anillo MOD-5; para ello utilice flip-flops J-K. Asegúrese de que el contador inicie la secuencia de conteo apropiada cuando se encienda.
- 7-74. Agregue un flip-flop más para convertir el contador de anillo MOD-5 del problema 7-73 en un contador MOD-10. Determine la secuencia de estados para este contador. Éste es un ejemplo de un contador de décadas que no es un contador BCD. Dibuje el circuito de decodificación para este contador.
- B** 7-75.* Dibuje el diagrama para un contador Johnson MOD-10 mediante el uso de un 74HC164. Asegúrese de que el contador inicie la secuencia de conteo apropiada cuando se encienda. Determine la secuencia de conteo para este contador y dibuje el circuito de decodificación necesario para decodificar cada uno de los 10 estados. Éste es otro ejemplo de un contador de décadas que no es un contador BCD.
- 7-76. La entrada de reloj para el contador Johnson del problema 7-75 es de 10 Hz. ¿Cuál es la frecuencia y el ciclo de trabajo para cada una de las salidas del contador?

SECCIÓN 7-21

- F** 7-77.* El contador MOD-10 de la figura 7-8(b) produce la secuencia de conteo 0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111 y se repite. Identifique algunas posibles condiciones de falla que podrían producir este resultado.
- F** 7-78. El contador MOD-10 de la figura 7-8(b) produce la secuencia de conteo 0000, 0101, 0010, 0111, 1000, 1101, 1010, 1111 y se repite. Identifique algunas posibles condiciones de falla que podrían producir este resultado.

SECCIONES 7-22 Y 7-23

- H** 7-79.* Cree un registro de desplazamiento SISO de ocho bits; para ello utilice HDL. La entrada en serie se llama *ser* y la salida en serie se llama *qsal*. Una señal de habilitación activa en BAJO (*en*) controla el registro de desplazamiento. Simule el diseño.
- H** 7-80. Cree un registro de desplazamiento PIPO de ocho bits; para ello utilice HDL. Los datos de entrada están en *d[7..0]* y las salidas en *q[7..0]*. Una señal de habilitación activa en ALTO (*ld*) controla el registro de desplazamiento. Simule el diseño.
- H** 7-81.* Cree un registro de desplazamiento PISO de ocho bits; para ello utilice HDL. Los datos de entrada están en *d[7..0]* y la salida es *q0*. La función del registro de desplazamiento se controla mediante *sh_ld* (*sh_ld* = 0 para la carga síncrona en paralelo y *sh_ld* = 1 para el desplazamiento en serie). El registro también debe tener una señal borrar asíncrona activa en BAJO (*clrn*). Simule el diseño.
- H** 7-82. Cree un registro de desplazamiento SIPO de ocho bits; para ello utilice HDL. Los datos de entrada están en *ser_ent* y las salidas en *q[7..0]*. La función del registro de desplazamiento se habilita mediante una señal de control activa en ALTO llamada *desplaza*. El registro de desplazamiento también tiene una señal borrar síncrona activa en ALTO (*borrar*), de mayor prioridad. El registro de desplazamiento también tiene una señal borrar síncrona activa en ALTO (*borrar*) de mayor prioridad. Simule el diseño.
- H** 7-83.* Simule el diseño del registro de desplazamiento universal del ejemplo 7-24.

- H** 7-84. Cree un registro de desplazamiento universal de ocho bits; para ello conecte en cascada dos de los módulos del ejemplo 7-24. Simule el diseño.
- H, D** 7-85.* Diseñe un contador Johnson MOD-10 de inicio automático con una señal restablecer asíncrona (*reset*), activa en ALTO; para ello utilice un HDL. Simule el diseño.
- H, D** 7-86. Algunas veces una aplicación digital puede necesitar un contador de anillo que recircule un solo cero en vez de un solo uno. Así, el contador de anillo tendría una salida activa en BAJO, en vez de una salida activa en ALTO. Diseñe un contador de anillo MOD-8 de inicio automático con una salida activa en BAJO; para ello utilice un HDL. El contador de anillo también deberá tener una señal de control *retener* activa en ALTO para deshabilitar el conteo. Simule el diseño.

SECCIÓN 7-24

- H** 7-87.* Utilice el simulador de Altera para probar el ejemplo de diseño del circuito monoestable no redisparable sensible al nivel de la figura 7-88 (AHDL) o 7-89 (VHDL). Utilice un reloj de 1-kHz y cree un pulso de salida de 10 ms para la simulación. Verifique que:
 - (a) Se cree la anchura de pulso correcta cuando se dispare.
 - (b) La salida pueda terminarse antes de tiempo con la entrada reset.
 - (c) El diseño del circuito monoestable sea no redisparable y que no pueda dispararse otra vez sino hasta que se haya agotado el tiempo.
 - (d) La señal de disparo debe durar lo suficiente como para que el reloj la atrape.
 - (e) La anchura de pulso pueda cambiarse a un valor distinto.
- H** 7-88. Modifique el ejemplo de diseño del circuito monoestable no redisparable sensible al nivel de la figura 7-88 (AHDL) o de la figura 7-89 (VHDL), de manera que el circuito monoestable sea redisparable pero continúe siendo sensible al nivel. Simule el diseño.

PREGUNTA DE PRÁCTICA

- B** 7-89.* Para cada una de las siguientes aseveraciones, indique el (los) tipo(s) de contador(es) que se describen.
 - (a) Se aplica un pulso de reloj a cada FF al mismo tiempo.
 - (b) Cada FF divide la frecuencia en su entrada CLK entre 2.
 - (c) La secuencia de conteo es 11, 10, 101, 100, 011, 010, 001, 000.
 - (d) El contador tiene 10 estados distintos.
 - (e) El retraso total de conmutación es la suma de los retrasos individuales de cada FF.
 - (f) Este contador no requiere lógica de decodificación.
 - (g) El número MOD siempre es el doble del número de FFs.
 - (h) Este contador divide la frecuencia de entrada entre su número MOD.
 - (i) Este contador puede comenzar su secuencia de conteo en cualquier estado inicial deseado.
 - (j) Este contador puede contar en cualquier dirección.
 - (k) Este contador puede sufrir de deformaciones en la decodificación, debido a sus retrasos de propagación.
 - (l) Este contador sólo cuenta del 0 al 9.
 - (m) Este contador puede diseñarse para que cuente a través de secuencias arbitrarias mediante la determinación del circuito lógico necesario en las entradas de control síncronas de cada flip-flop.

RESPUESTAS A LAS PREGUNTAS DE REPASO DE LAS SECCIONES

PARTE 1

SECCIÓN 7-1

1. Falso 2. 0000 3. 128

SECCIÓN 7-2

1. Cada FF agrega su retraso de propagación al retraso total del contador, en respuesta a un pulso de reloj. 2. MOD-256

SECCIÓN 7-3

1. Puede operar a frecuencias de reloj más altas y tiene un circuito más complejo.
2. Seis FFs y cuatro compuertas AND 3. ABCDE

SECCIÓN 7-4

1. D, C y A 2. Verdadero, ya que un contador BCD tiene 10 estados distintos.
3. 5 kHz

SECCIÓN 7-5

1. En un contador ascendente, el conteo se incrementa en 1 con cada pulso de reloj; en un contador descendente, el conteo se decrementa en 1 con cada pulso. 2. Hay que cambiar las conexiones a las salidas invertidas respectivas, en vez de las Qs.

SECCIÓN 7-6

1. Puede prestablecerse a cualquier conteo inicial deseado. 2. El preajuste asíncrono es independiente de la entrada de reloj, mientras que el preajuste síncrono ocurre en el flanco activo de la señal de reloj.

SECCIÓN 7-7

1. $\overline{\text{CARGA}}$ es el control que habilita la carga en paralelo de las entradas de datos D C B A (A = LSB). 2. $\overline{\text{CLR}}$ es el control que habilita el restablecimiento del contador a 0000. 3. Verdadero 4. Todas las entradas de control ($\overline{\text{CLR}}$, $\overline{\text{CARGA}}$, ENT y ENP) en el 74162 deben estar en ALTO. 5. $\overline{\text{CARGA}} = 1$, $\overline{\text{CTEN}} = 0$ y $D/U = 1$ para contar hacia abajo. 6. 74HC163: de 0 a 65,535; 74ALS190: de 0 a 9999 o de 9999 a 0.

SECCIÓN 7-8

1. Sesenta y cuatro 2. Una compuerta NAND de seis entradas: A, B, C, D, E y F.

SECCIÓN 7-9

1. No tendríamos que tratar con los estados transitorios y las posibles deformaciones en las formas de onda de salida. 2. Tabla de estado PRESENTE y estado SIGUIENTE.
3. Las compuertas controlan la secuencia de conteo. 4. Todos los estados sin usar conducen de regreso a la secuencia de conteo del contador.

SECCIÓN 7-10

1. Consulte el texto. 2. Asocia todos los posibles estados PRESENTES con su estado SIGUIENTE deseado. 3. Muestra los niveles necesarios en la entrada asíncrona de cada flip-flop para producir las transiciones de estado del contador. 4. Verdadero

SECCIÓN 7-11

1. Tablas de estado PRESENTE / estado SIGUIENTE.
2. El estado SIGUIENTE deseado.
3. AHDL:
ff[].clk = !reloj
- VHDL:
IF (reloj = ?0? AND reloj?EVENT) THEN
4. Descripción del comportamiento.

SECCIÓN 7-12

1. La señal borrar asíncrona hace que el contador se borre de inmediato. La carga asíncrona se produce en el siguiente flanco activo del reloj.
2. AHDL: utilice el puerto .clrn en los FFs; VHDL: defina la función borrar antes de comprobar el flanco del reloj.
3. Por el orden de evaluación en una instrucción IF.

SECCIÓN 7-13

1. Ambos HDLs pueden utilizar un diagrama de bloques para conectar los módulos; VHDL también puede usar un archivo de texto que describa las conexiones entre los componentes.
2. Un bus es una colección de líneas de señal; se representa en forma gráfica mediante una línea gruesa.
3. Habilitación de conteo y decodificación del conteo terminal.

SECCIÓN 7-14

1. Un contador se utiliza, por lo general, para contar eventos, mientras que una máquina de estados se utiliza, por lo general, para controlar eventos.
2. Una máquina de estados puede describirse mediante el uso de símbolos para describir sus estados, en vez de los estados binarios actuales.
3. El compilador asigna los valores óptimos para minimizar el circuito.
4. La descripción es mucho más fácil de escribir y de comprender.

PARTE 2**SECCIÓN 7-19**

1. Entrada en paralelo/salida en serie.
2. Verdadero
3. Entrada en serie/salida en paralelo.
4. Entrada en serie/salida en serie.
5. El 74165 utiliza transferencia de datos asíncrona en paralelo; el 74174 utiliza transferencia de datos síncrona en paralelo.
6. Un nivel ALTO evita el desplazamiento en las entradas CP.

SECCIÓN 7-20

1. Contador de anillo.
2. Contador Johnson.
3. La salida invertida del último FF se conecta a la entrada del primer FF.
4. (a) Falso (b) Verdadero (c) Verdadero
5. Dieciséis; ocho

SECCIÓN 7-22

1. AHDL:
reg[].d = (reg[6..0], dat)
- VHDL:
reg := reg (6 DOWNT0 0) & dat
2. Porque el registro puede seguir recibiendo flancos de reloj durante la retención.

SECCIÓN 7-23

1. Puede comenzar en cualquier estado, pero en cierto momento llegará a la secuencia de anillo deseada.
2. Las líneas 1 y 12
3. Las líneas 12 y 13

SECCIÓN 7-24

1. La entrada restablecer.
2. La frecuencia de reloj y el valor de retraso que se carga en el contador.
3. En forma síncrona.
4. La anchura del pulso de salida es muy consistente.
5. El pulso de salida responde al flanco de disparo de inmediato.
6. El estado del disparo en el flanco de reloj actual y su estado en el flanco anterior.