FUNDAMENTOS DE SISTEMAS DIGITALES

Novena Edición

THOMAS L. FLOYD

Traducción
Vuelapluma

Revisión Técnica
Eduardo Barrera López de Turiso
Departamento de Sistemas Electrónicos y de Control
Universidad Politécnica de Madrid

Madrid ● México ● Santa Fe de Bogotá ● Buenos Aires ● Caracas ● Lima
Montevideo ● San Juan ● San José ● Santiago ● São Paulo ● White Plains ●

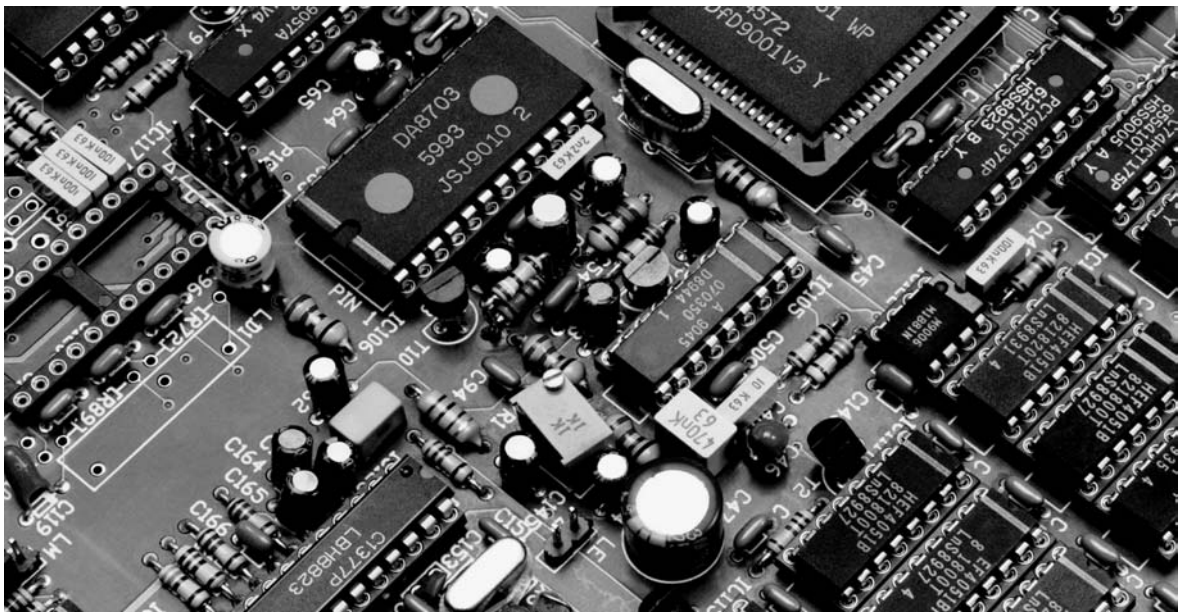
3

PUERTAS LÓGICAS

CONTENIDO DEL CAPÍTULO

OBJETIVOS DEL CAPÍTULO

- Describir el funcionamiento del inversor y de las puertas AND y OR.
- Describir el funcionamiento de las puertas NAND y NOR.
- Expresar las operaciones de las puertas NOT, AND, OR, NAND y NOR mediante el álgebra de Boole.
- Describir el funcionamiento de las puertas OR–exclusiva y NOR–exclusiva.
- Reconocer y utilizar los símbolos distintivos y los símbolos rectangulares de las puertas lógicas según el estándar ANSI/IEEE 91–1984.



- Elaborar los diagramas de tiempos que muestran las relaciones de tiempo de las entradas y las salidas de las diferentes puertas lógicas.
- Establecer las comparaciones básicas entre las principales tecnologías de circuitos integrados: TTL y CMOS.
- Explicar las diferencias entre las series de las familias TTL y CMOS.
- Definir, para las puertas lógicas, los siguientes parámetros: *tiempo de retardo de propagación, disipación de potencia, producto velocidad–potencia y fan–out*.
- Enumerar circuitos integrados de función fija que contengan varias puertas lógicas.
- Utilizar cada puerta lógica en aplicaciones sencillas.
- Localización de averías en las puertas lógicas debidas a circuitos abiertos o cortocircuitos, utilizando el pulsador y la sonda lógica o el osciloscopio.

PALABRAS CLAVE

- Inversor
- Tabla de verdad
- Diagrama de tiempos
- Álgebra booleana
- Complemento
- Puerta AND
- Habilitar
- Puerta OR
- Puerta NAND
- Puerta NOR
- Puerta OR–exclusiva
- Puerta NOR–exclusiva
- Matriz AND
- Fusible
- Antifusible
- EPROM
- EEPROM
- SRAM
- Dispositivo objetivo
- JTAG
- CMOS
- TTL
- Tiempo de retardo de propagación
- Fan–out
- Carga unidad

INTRODUCCIÓN

Este capítulo hace énfasis en el funcionamiento lógico, las aplicaciones y la localización de averías de las puertas lógicas. Se cubre la relación entre las formas de onda de entrada y de salida de una puerta utilizando los diagramas de tiempos.

Los símbolos lógicos que se usan para representar las puertas lógicas están de acuerdo con el estándar ANSI/IEEE 91–1984. Este estándar ha sido adoptado por la industria privada, y la industria militar lo utiliza para su documentación interna así como para sus publicaciones.

En este capítulo se aborda tanto la lógica programable como la lógica de función fija. Puesto que en todas las aplicaciones se usan los circuitos integrados (CI), generalmente, la función lógica de un dispositivo es más importante para el técnico que los detalles de operación del circuito en el nivel de componentes en el interior del CI. Por tanto, la cobertura detallada de los dispositivos en el nivel de componente puede tratarse como un tema opcional. Para aquéllos que lo necesiten y tengan tiempo, en el Capítulo 14 se cubren las tecnologías de los circuitos integrados digitales, haciéndose referencia a partes del mismo a lo largo del texto. *Sugerencia*: repase la Sección 1.3 antes de comenzar con este capítulo.

DISPOSITIVOS LÓGICOS DE FUNCIÓN FIJA

(SERIES CMOS Y TTL)

74XX00	74XX02	74XX04
74XX08	74XX10	74XX11
74XX20	74XX21	74XX27
74XX30	74XX32	74XX86
74XX266		

3.1 EL INVERSOR

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

Al finalizar esta sección, el lector deberá ser capaz de:

■ Identificar los indicadores de negación y polaridad. ■ Identificar un inversor tanto mediante su símbolo distintivo como por su símbolo rectangular. ■ Elaborar la tabla de verdad del inversor. ■ Describir el funcionamiento lógico de un inversor.

En la Figura 3.1 se muestran los símbolos lógicos estándar del ***inversor***. La parte (a) muestra los *símbolos distintivos*, y la (b) muestra los *símbolos rectangulares*. En este texto se usan los símbolos distintivos; sin embargo, los símbolos rectangulares suelen encontrarse en las documentaciones industriales, por lo que debería familiarizarse con ellos. Los símbolos lógicos cumplen el estándar **ANSI/IEEE** 91–1984.

(a) Símbolos distintivos con indicadores de negación.

(b) Símbolos rectangulares con indicadores de polaridad.

Figura 3.1 Símbolos lógicos estándar de la puerta inversora (Estándar ANSI/IEEE 91–1984).

Los indicadores de negación y de polaridad

El indicador de negación es un "círculo" (○) que indica **inversión** o *complementación*, cuando aparece en la entrada o en la salida de un elemento lógico, tal como muestra la Figura 3.1(a) para el inversor. Generalmente, las entradas se sitúan a la izquierda del símbolo lógico, y la salida a la derecha. Cuando en la entrada hay un círculo, quiere decir que el estado activo o *verdadero* de la entrada es 0, y se dice que la entrada es activa a nivel BAJO. Cuando el círculo se sitúa en la salida significa que el estado activo o verdadero de salida es 0, y se dice que la salida es activa a nivel BAJO. La ausencia de círculo en la entrada o en la salida significa que el estado activo o verdadero es 1 y, en este caso, se dice que la entrada o la salida es activa a nivel ALTO.

El indicador de polaridad o de nivel es un "triángulo" (◺) que indica inversión cuando aparece a la entrada o a la salida de un elemento lógico, como muestra la Figura 3.1(b). Cuando se presenta a la entrada, significa que un nivel BAJO es el estado de entrada activo o verdadero. Cuando se presenta a la salida, significa que un nivel BAJO es el estado de salida activo o verdadero.

Ambos indicadores (círculo y triángulo) pueden utilizarse tanto en los símbolos distintivos como en los rectangulares. La Figura 3.1(a) indica el principal uso de los símbolos del inversor en este texto. Observe que un cambio de posición del indicador de polaridad o de negación no implica un cambio en el modo de funcionamiento del inversor.

Tabla de verdad del inversor

Cuando se aplica un nivel ALTO a la entrada de un inversor, en su salida se presenta un nivel BAJO. Cuando se aplica un nivel BAJO a la entrada, en su salida se presenta un nivel ALTO. En la Tabla 3.1 se resume esta

operación. Esta tabla muestra la salida para cada posible entrada en términos de niveles y bits correspondientes. Una tabla tal como ésta se llama **tabla de verdad**.

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

Tabla 3.1 Tabla de verdad del inversor.

Funcionamiento del inversor

La Figura 3.2 muestra la salida de un inversor para un impulso de entrada, donde t_1 y t_2 indican los puntos que corresponden a los impulsos de entrada y salida.

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

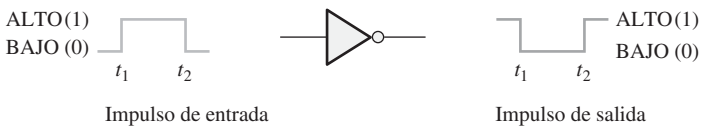


Figura 3.2 Funcionamiento del inversor con un impulso de entrada.

Diagramas de tiempos

▲ *Un diagrama de tiempos muestra cómo se relacionan dos o más señales en el tiempo.*

Recuerde del Capítulo 1 que un **diagrama de tiempos o cronograma** es básicamente *una gráfica que presenta de forma precisa las relaciones de dos o más formas de onda en función del tiempo*. Por ejemplo, la relación de tiempo del impulso de salida respecto al impulso de entrada de la Figura 3.2 puede representarse con un sencillo diagrama de tiempos, alineando los dos impulsos de modo que las ocurrencias de los flancos se presenten en los instantes de tiempo correctos. El flanco de subida del impulso de entrada y el flanco de bajada del impulso de salida se producen al mismo tiempo (idealmente). Igualmente, el flanco de bajada del impulso de entrada y el flanco de subida del impulso de salida se producen al mismo tiempo (idealmente). En la Figura 3.3 se muestra la relación de tiempos. Los diagramas de tiempos son muy útiles para ilustrar las relaciones de las señales digitales de impulsos múltiples.

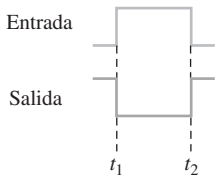


FIGURA 3.3 Diagrama de tiempos para el caso de la Figura 3.2.

Expresión lógica del inversor

En el **álgebra booleana**, que son las matemáticas de los circuitos lógicos y que se cubrirán en el Capítulo 4, una variable se designa mediante una letra. El **complemento** de una variable se designa mediante una barra

EJEMPLO 3.1

Al inversor de la Figura 3.4 se le aplica una señal. Determinar la forma de onda de salida correspondiente a la entrada y dibujar el diagrama de tiempos. De acuerdo con el emplazamiento del círculo ¿cuál es el estado activo de salida?

FIGURA 3.4

Solución La forma de onda de salida es exactamente la opuesta a la de entrada (es la entrada invertida), como se muestra en la Figura 3.5, que es el cronograma básico. El estado activo o verdadero de salida es 0.

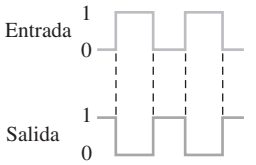


FIGURA 3.5

Problema relacionado* Si el inversor tiene el indicador negativo (círculo) en la entrada en lugar de en la salida, ¿cómo afecta esto al diagrama de tiempos?

* Las respuestas se encuentran al final del capítulo.

▲ *El álgebra booleana utiliza variables y operadores para describir un circuito lógico.*

encima de la letra. Una variable puede tomar uno de dos valores, 1 ó 0. Si una variable dada es 1, su complemento es 0, y viceversa.

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

$$X = \overline{A}$$

Esta expresión establece que la salida es el complemento de la entrada, de modo que si $A = 0$, entonces $X = 1$, y si $A = 1$, entonces $X = 0$. La Figura 3.6 ilustra esto. La variable complementada $\overline{A}$ se lee "A negada" o "A complementada".

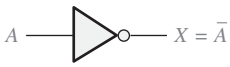


FIGURA 3.6 El inversor complementa una variable de entrada.

Aplicación

La Figura 3.7 muestra un circuito que genera el complemento a 1 de un número binario de 8 bits. Los bits del número binario se aplican a las entradas del inversor y el complemento a 1 se obtiene en las salidas.

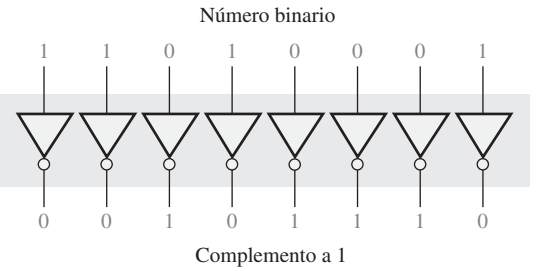


FIGURA 3.7 Ejemplo de un circuito que genera el complemento a 1 utilizando inversores.

REVISIÓN DE LA SECCIÓN 3.1

Las respuestas se encuentran al final del capítulo.

1. Cuando en la entrada de un inversor hay un 1, ¿cuál es la salida?
2. En la entrada de un inversor se requiere un impulso activo a nivel ALTO (el nivel ALTO es verdadero, y el nivel BAJO no).
   (a) Dibujar el símbolo lógico correspondiente, utilizando el indicador de negación y el símbolo distintivo para el inversor de esta aplicación.
   (b) Describir la salida cuando un impulso de subida se aplica a la entrada del inversor.

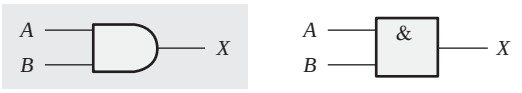
3.2 LA PUERTA AND

La puerta AND es una de las puertas básicas con la que se construyen todas las funciones lógicas. Una puerta AND puede tener dos o más entradas y realiza la operación que se conoce como multiplicación lógica.

Al finalizar este capítulo, el lector deberá ser capaz de:

■ Identificar una puerta AND mediante su símbolo distintivo y su símbolo rectangular. ■ Describir la operación lógica de una puerta AND. ■ Generar la tabla de verdad de una puerta AND con cualquier número de entradas. ■ Generar el cronograma de una puerta AND para cualquier forma de onda especificada en sus entradas. ■ Escribir la expresión lógica de una puerta AND con cualquier número de entradas. ■ Analizar ejemplos de aplicaciones de la puerta AND.

El término *puerta* se usa para describir un circuito que realiza una operación lógica básica. La puerta AND tiene dos o más entradas y una única salida, como indican los símbolos lógicos estándar mostrados en la Figura 3.8. En cada uno de los símbolos, las entradas se sitúan a la izquierda y la salida a la derecha. Se muestran puertas con dos entradas, pero una puerta AND puede tener cualquier número de entradas superior a éste. Aunque en los ejemplos se utilizan ambos tipos de símbolos, distintivos y rectangulares, en este libro, predominantemente, se emplea el símbolo distintivo de la Figura 3.8(a).



(a) Símbolo distintivo

(b) Símbolo rectangular, identificado mediante el carácter AND (&)

FIGURA 3.8 Símbolos lógicos estándar de la puerta AND con dos entradas (estándar ANSI/IEEE 91–1984).

NOTAS INFORMÁTICAS

Las puertas lógicas son los bloques de construcción de las computadoras. La mayor parte de las funciones en una computadora, con la excepción de ciertos tipos de memoria, se implementan mediante puertas lógicas utilizadas a muy gran escala. Por ejemplo, un microprocesador, que es la parte principal de una computadora, consta de cientos de miles de puertas lógicas.

Funcionamiento de la puerta AND

▲ *Una puerta AND puede tener más de dos entradas.*

La ***puerta AND*** genera una salida a nivel ALTO *sólo* cuando *todas* las entradas están a nivel ALTO. Cuando cualquiera de la entradas está a nivel BAJO, la salida se pone a nivel BAJO. Por tanto, el propósito básico de una puerta AND es determinar cuándo ciertas condiciones de entrada son simultáneamente verdaderas, como indican todas sus entradas estando a nivel ALTO, y producir una salida a nivel ALTO, para indicar que esas condiciones son verdaderas. Las entradas de la puerta AND de dos entradas de la Figura 3.8 se designan mediante *A* y *B*, y la salida con *X*, luego podemos establecer que el funcionamiento de la puerta es el siguiente:

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

La Figura 3.9 ilustra una puerta AND de 2 entradas en la que se indican las cuatro posibles combinaciones de entrada y el resultado correspondiente a cada una de ellas.

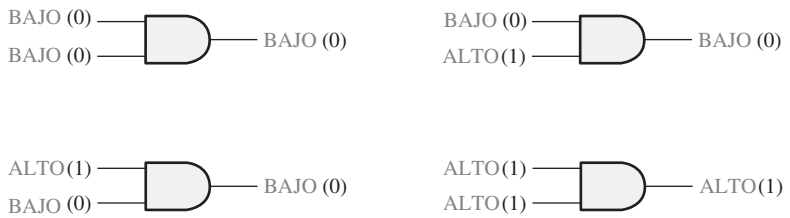


FIGURA 3.9 Todos los posibles niveles lógicos para una puerta AND de dos entradas.

Tabla de verdad de la puerta AND

▲ *En una puerta AND, si todas las entradas están a nivel ALTO, la salida es un nivel ALTO.*

La operación lógica de una puerta puede expresarse mediante una tabla de verdad, en la que se enumeran todas las combinaciones de entrada con las correspondientes salidas, como muestra la Tabla 3.2 para una puerta AND de dos entradas. La tabla de verdad puede ampliarse para cualquier número de entradas. Aunque los términos nivel ALTO y nivel BAJO dan un sentido "físico" a los estados de entrada y salida, la tabla de verdad se presenta con 1s y 0s, ya que un nivel ALTO es equivalente a un 1, y un nivel BAJO es equivalente a 0 en lógica positiva. Para cualquier puerta AND, independientemente del número de entradas, la salida es un nivel ALTO *sólo* cuando *todas* las entradas están a nivel ALTO.

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

Ecuación 3.1 $$N = 2^n$$

donde N es el número de posibles combinaciones de entrada y n es el número de variables de entrada:

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

1 = ALTO, 0 = BAJO

TABLA 3.2 Tabla de verdad de una puerta AND de dos entradas.

Para dos variables de entrada: $N = 2^2 = 4$ combinaciones
Para tres variables de entrada: $N = 2^3 = 8$ combinaciones
Para cuatro variables de entrada: $N = 2^4 = 16$ combinaciones

Utilizando la Ecuación 3.1 se puede determinar el número de combinaciones de bits de entrada para puertas con cualquier número de entradas.

EJEMPLO 3.2

(a) Desarrollar la tabla de verdad de una puerta AND de 3 entradas.
(b) Determinar el número total de posibles combinaciones de entrada para una puerta AND de 4 entradas.

Solución

(a) Para una puerta AND de 3 entradas existen ocho posibles combinaciones de entrada ($2^3 = 8$). Las entradas de la tabla de verdad (Tabla 3.3) muestran las ocho posibles combinaciones de tres bits. La salida es siempre 0, excepto cuando los tres bits de entrada son 1.

Entradas			Salidas
A	*B*	*C*	*X*
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

TABLA 3.3

(b) $N = 2^4 =$ **16**. Para una puerta AND de 4 entradas existen 16 posibles combinaciones de bits de entrada.

Problema relacionado Desarrollar la tabla de verdad para una puerta AND de 4 entradas.

Funcionamiento con trenes de impulsos

En la mayoría de las aplicaciones, las entradas a una puerta no son niveles estacionarios sino tensiones que cambian frecuentemente entre los niveles lógicos ALTO y BAJO. Ahora vamos a ver el funcionamiento de las puertas AND con entradas que son señales digitales, teniendo en mente que una puerta AND obedece a su tabla de verdad, independientemente de que sus entradas sean niveles constantes o señales que varíen de un nivel a otro.

Al examinar el funcionamiento de una puerta AND con trenes de impulsos, nos fijaremos en los niveles de entrada para determinar el nivel de salida en cualquier instante dado. En la Figura 3.10, ambas entradas *A* y *B* están a nivel ALTO (1) durante el intervalo de tiempo t_1, por lo que la salida *X* en este intervalo estará a nivel ALTO (1). Durante el intervalo t_2, la entrada *A* está a nivel BAJO (0) y la entrada *B* está a nivel ALTO (1), por lo que la salida se pondrá a nivel BAJO (0). De nuevo, durante el intervalo t_3, ambas entradas están a nivel ALTO (1) y, por tanto, la salida está a nivel ALTO (1). Durante el intervalo t_4, la entrada *A* está a nivel ALTO (1) y la *B* está a nivel BAJO (0), luego la salida está a nivel BAJO (0). Por último, durante el intervalo t_5, la entrada *A* está a nivel BAJO (0) y la entrada *B* está a nivel BAJO (0) y, por tanto, la salida está a nivel BAJO (0). Como ya sabe, un diagrama de las señales de entrada y de salida en función del tiempo se llama *diagrama de tiempos o cronograma*.

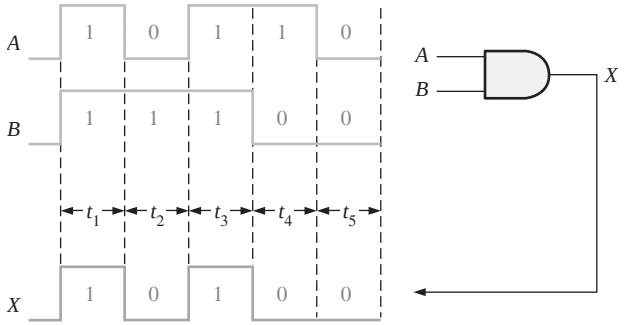
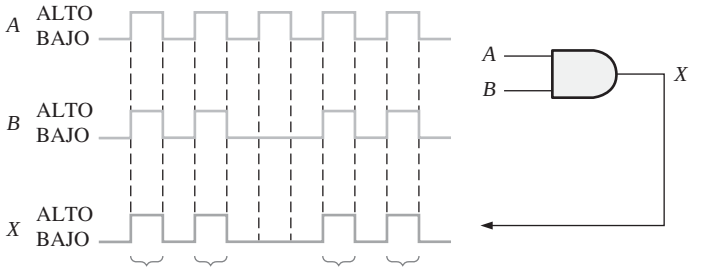


Figura 3.10 Ejemplo de funcionamiento de una puerta AND con trenes de impulsos, y cronograma que muestra las relaciones entre las entradas y la salida.

EJEMPLO 3.3

Si se aplican las formas de onda *A* y *B* de la Figura 3.11 a las entradas de una puerta AND, ¿cuál es la forma de onda resultante de salida?



A y *B* están a nivel ALTO durante estos cuatro intervalos de tiempo. Por tanto, *X* está a nivel ALTO.

FIGURA 3.11

Solución La forma de onda de salida X sólo está a nivel ALTO cuando A y B están a nivel ALTO, tal y como se muestra en el diagrama de tiempos de la Figura 3.11.

Problema relacionado Determinar la forma de onda de salida y dibujar el diagrama de tiempos si los impulsos segundo y cuarto de la señal A de la Figura 3.11 se reemplazan ambos por un nivel BAJO.

Es importante recordar que cuando se analiza el funcionamiento con trenes de impulsos de las puertas lógicas, hay que poner especial cuidado en las relaciones de tiempo de todas las entradas entre sí y con la salida.

EJEMPLO 3.4

Para las dos formas de onda de entrada, *A* y *B*, de la Figura 3.12, dibujar la onda de salida mostrando su relación con las entradas.

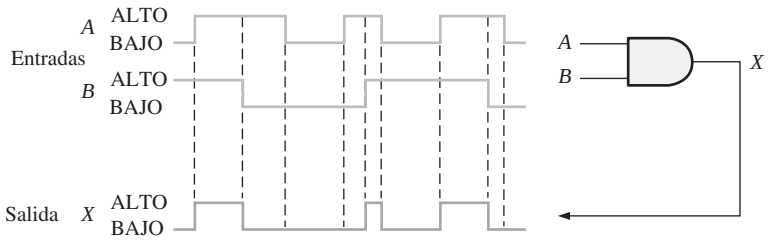


FIGURA 3.12

Solución La onda de salida está a nivel ALTO sólo cuando ambas entradas están a nivel ALTO, como se muestra en el diagrama de tiempos.

Problema relacionado Obtener la onda de salida si la entrada *B* de la puerta AND de la Figura 3.12 siempre está a nivel ALTO.

EJEMPLO 3.5

Para la puerta AND de 3 entradas de la Figura 3.13, determinar la forma de onda de salida con respecto a las entradas.

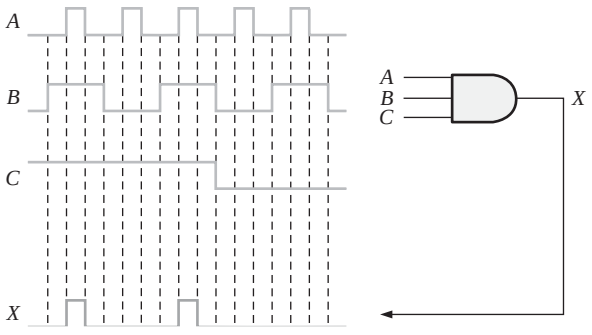


FIGURA 3.13

Solución	La onda de salida X de una puerta AND de 3 entradas está a nivel ALTO sólo cuando las tres entradas, *A*, *B* y *C* están a nivel ALTO.
Problema relacionado	¿Cuál es la onda de salida de la puerta AND de la Figura 3.13 si la entrada *C* está siempre a nivel ALTO?

Expresiones lógicas para la puerta AND

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

La **multiplicación booleana** sigue las mismas reglas básicas que gobiernan la multiplicación binaria, que hemos tratado en el Capítulo 2 y son las siguientes:

$$0 \cdot 0 = 0$$
$$0 \cdot 1 = 0$$
$$1 \cdot 0 = 0$$
$$1 \cdot 1 = 1$$

La multiplicación booleana es lo mismo que la función AND.

NOTAS INFORMÁTICAS

Las computadoras pueden utilizar todas las operaciones lógicas básicas cuando tienen que manipular de forma selectiva ciertos bits pertenecientes a uno o más bytes de datos. La manipulación selectiva de bits se lleva a cabo mediante una *máscara*. Por ejemplo, para borrar (poner a 0) los cuatro bits de la derecha de un byte de datos manteniendo los cuatro bits de la izquierda, aplique la operación AND al byte de datos y al valor 11110000. Observe que cualquier bit que se opere (AND) con cero dará 0 y cualquier byte al que se aplique la operación AND con el valor 1 quedará como está. Si se aplica la operación AND al valor 10101010 con la máscara 11110000, el resultado es 10100000.

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

$$X = AB$$

La Figura 3.14(a) muestra la puerta con las variables de entrada y de salida indicadas.

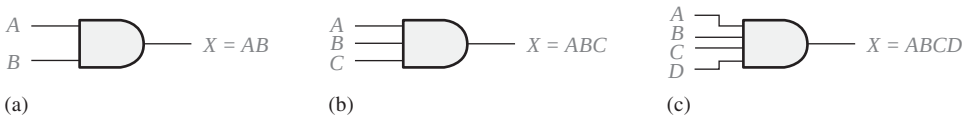


Figura 3.14 Expresión booleana para puertas AND con dos, tres y cuatro entradas.

▲ *Cuando las variables se escriben como en ABC, quiere decir que se aplica la operación AND.*

Para extender la expresión AND a más de dos variables de entrada, simplemente utilice una nueva letra para cada variable de entrada. Por ejemplo, la función de una puerta AND de 3 entradas, se puede expresar así: $X = ABC$, donde *A*, *B* y *C* son las variables de entrada. La expresión para una puerta AND de 4 entradas será $X = ABCD$, y así sucesivamente. Las partes (b) y (c) de la Figura 3.14 muestran, respectivamente, puertas AND de tres y cuatro variables de entrada.

Se puede evaluar el funcionamiento de una puerta AND utilizando las expresiones booleanas que facilitan la salida. Por ejemplo, cada variable de entrada puede ser 1 ó 0, luego para evaluar la puerta AND de dos entradas, se sustituyen dichos valores en la ecuación de salida, $X = AB$, como se muestra en la Tabla 3.4. Esta evaluación muestra que la salida X de una puerta AND es 1 (nivel ALTO) sólo cuando ambas entradas son 1 (nivel ALTO). Se puede hacer un análisis similar para cualquier número de variables de entrada.

A	*B*	*AB* = *X*
0	0	$0 \cdot 0 = 0$
0	1	$0 \cdot 1 = 0$
1	0	$1 \cdot 0 = 0$
1	1	$1 \cdot 1 = 1$

TABLA 3.4

Aplicaciones

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

Un ejemplo sencillo de este particular uso de la puerta AND se muestra en la Figura 3.15, donde la puerta AND controla el paso de una señal (*A*) a un contador digital. El propósito de este circuito es medir la frecuencia de la señal *A*. El impulso de habilitación tiene un ancho de exactamente 1 s. Cuando la entrada de habilitación está a nivel ALTO, la señal *A* pasa a través de la puerta hasta el contador, y cuando la entrada de habilitación está a nivel BAJO, se impide el paso de la señal a través de la puerta.

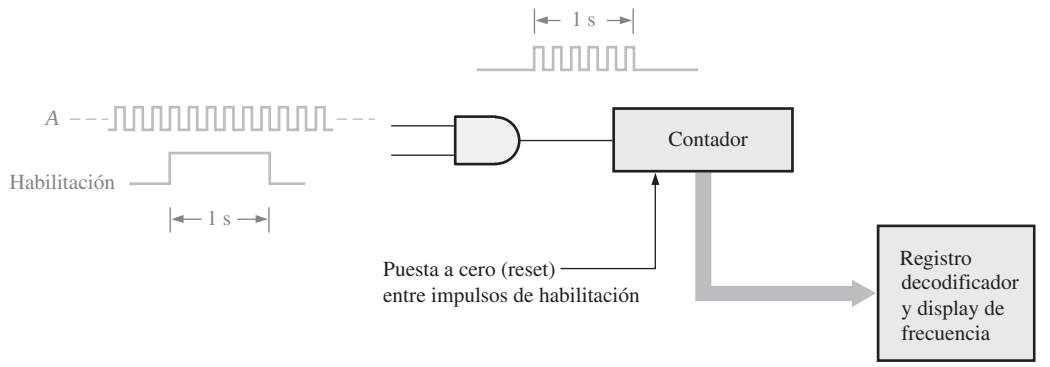


FIGURA 3.15 Una puerta AND que realiza la función de habilitación/inhibición para un contador de frecuencia.

Durante el intervalo de habilitación de 1 segundo, un cierto número de impulsos de la señal A pasan a través de la puerta AND hasta el contador. El número de impulsos que pasa durante 1 s es igual a la frecuencia de la señal. Por ejemplo, la Figura 3.15 muestra una señal en la que seis impulsos duran un segundo, lo que representa una frecuencia de 6 Hz. Si pasan 1.000 impulsos a través de la puerta en el intervalo de 1 s del impulso de habilitación, serán 1.000 impulsos/segundo, es decir una frecuencia de 1.000 Hz.

El contador cuenta el número de impulsos por segundo y genera una salida binaria que pasa al circuito decodificador y al display, en el que se genera una lectura de la frecuencia. El impulso de habilitación se repite en determinados instantes, en los que se lleva a cabo un nuevo recuento por si la frecuencia ha variado, y

el nuevo valor se presentará en el display. Entre los impulsos de habilitación, el contador se pone a cero para reinicializarse cada vez que se produce un impulso de habilitación. La frecuencia actual se almacena en un registro, de modo que el display no se vea afectado al poner a cero el contador.

Un sistema de alarma para el cinturón de seguridad. En la Figura 3.16, se usa una puerta AND en un sencillo sistema de alarma para el cinturón de seguridad del coche, el cual detecta cuándo el interruptor de arranque se ha activado y (*AND*) el cinturón de seguridad no está abrochado. Si el interruptor de arranque se ha activado, la entrada A de la puerta AND se pone a nivel ALTO. Si el cinturón de seguridad no está correctamente abrochado, la entrada *B* de la puerta AND se pone a nivel ALTO. También cuando el interruptor de arranque se activa, se inicializa un temporizador que pone a nivel ALTO la entrada *C* durante 30 s. Si estas tres condiciones se cumplen, es decir, si el interruptor de arranque está activado y (*AND*) el cinturón de seguridad está desabrochado y (*AND*) el temporizador está corriendo, la salida de la puerta AND se pone a nivel ALTO, y una alarma audible se activa para advertir al conductor.

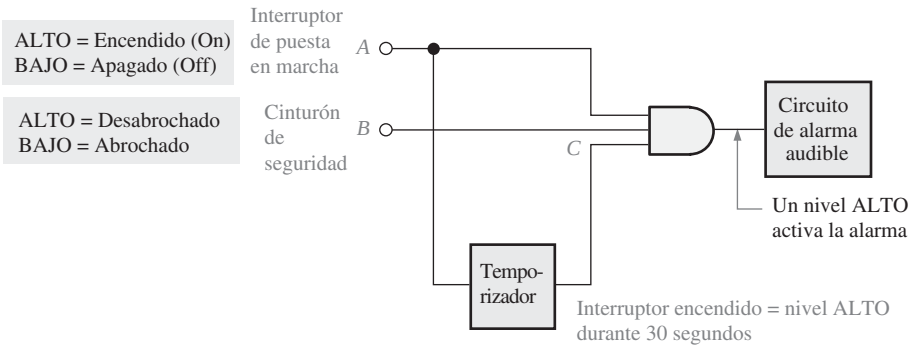


FIGURA 3.16 Un sencillo circuito de alarma para cinturón de seguridad utilizando una puerta AND.

REVISIÓN DE LA SECCIÓN 3.2

1. ¿Cuándo se pone a nivel ALTO la salida de una puerta AND?
2. ¿Cuándo se pone a nivel BAJO la salida de una puerta AND?
3. Describir la tabla de verdad para una puerta AND de cinco entradas.

3.3 LA PUERTA OR

La puerta OR es otra de las puertas básicas con las que se construyen todas las funciones lógicas. Una puerta OR puede tener dos o más entradas y realiza la operación que se conoce como suma lógica.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Identificar una puerta OR mediante su símbolo distintivo y su símbolo rectangular. ■ Describir la operación lógica de una puerta OR. ■ Generar la tabla de verdad de una puerta OR con cualquier número de entradas. ■ Generar el diagrama de tiempos de una puerta OR para cualquier forma de onda especificada en sus entradas. ■ Escribir la expresión lógica de una puerta OR con cualquier número de entradas. ■ Desarrollar ejemplos de aplicaciones de la puerta OR.

Una ***puerta OR*** tiene dos o más entradas y una salida, como indican los símbolos lógicos estándar de la Figura 3.17, en la que se muestran puertas OR con dos entradas. Una puerta OR puede tener cualquier número de entradas mayor o igual que dos. Aunque se presentan ambos tipos de símbolos, distintivo y rectangular, en este texto se utilizará el símbolo distintivo de la puerta OR.

(a) Símbolo distintivo. (b) Símbolo rectangular con el identificador OR (≥ 1).

FIGURA 3.17 Símbolos lógicos estándar de la puerta OR con dos entradas (Estándar ANSI/IEEE 91–1984).

Funcionamiento de la puerta OR

▲ *Una puerta OR puede tener más de dos entradas.*

Una puerta OR genera un nivel ALTO a la salida cuando *cualquiera* de sus entradas está a nivel ALTO. La salida se pone a nivel BAJO sólo cuando todas las entradas están a nivel BAJO. Por tanto, el propósito de una puerta OR es determinar cuándo una o más de sus entradas están a nivel ALTO y generar una salida a nivel ALTO que indique esta condición. Las entradas de la puerta OR de dos entradas de la Figura 3.17 están etiquetadas como *A* y *B*, y la salida como *X*. Podemos establecer el funcionamiento de la puerta como sigue:

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

El nivel ALTO es el nivel de salida activo o verdadero para la puerta OR. La Figura 3.18 ilustra la operación lógica para una puerta OR de dos entradas, indicando las cuatro posibles combinaciones de entrada.

FIGURA 3.18 Todos los posibles niveles lógicos para una puerta OR de 2 entradas.

Tabla de verdad de una puerta OR

▲ *En una puerta OR, si una entrada está a nivel ALTO la salida es un nivel ALTO.*

En la Tabla 3.5 se describe el funcionamiento lógico de una puerta OR de dos entradas. Esta tabla de verdad puede extenderse a cualquier número de entradas e, independientemente del número de entradas, la salida es un nivel ALTO cuando una o más entradas están a nivel ALTO.

Entradas		Salida
A	***B***	***X***
0	0	0
0	1	1
1	0	1
1	1	1
1 = ALTO, 0 = BAJO		

TABLA 3.5 Tabla de verdad para una puerta OR de dos entradas.

Funcionamiento con trenes de impulsos

Ahora vamos a ver el funcionamiento de una puerta OR con trenes de impulsos como entradas, teniendo en mente su modo de operación lógico. De nuevo, lo importante en el análisis del funcionamiento de la puerta con trenes de impulsos en las entradas es la relación de tiempos de todas las señales implicadas. Por ejemplo, en la Figura 3.19, las entradas *A* y *B* están a nivel ALTO (1) durante el intervalo t_1, haciendo que la salida *X* esté a nivel ALTO (1). Durante el intervalo t_2, la entrada *A* está a nivel BAJO (0) pero, puesto que la entrada *B* está a nivel ALTO (1), la salida estará a nivel ALTO (1). Durante el intervalo t_3, ambas entradas están a nivel BAJO (0), luego en este intervalo la salida estará a nivel BAJO (0). Durante el intervalo t_4, la salida está a nivel ALTO (1) ya que la entrada *A* está a nivel ALTO (1).

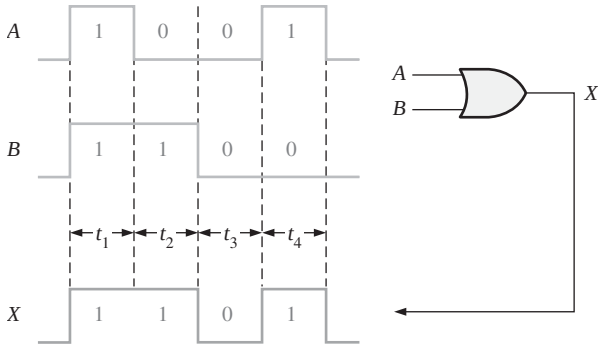
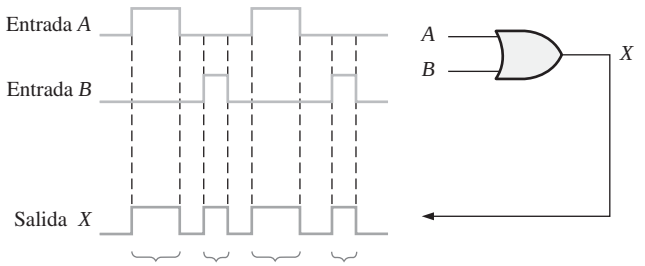


FIGURA 3.19 Ejemplo de funcionamiento de la puerta OR con trenes de impulsos junto con el cronograma que muestra la relación entre las entradas y la salida.

En este ejemplo, simplemente hemos aplicado la tabla de verdad de la puerta OR para cada uno de los intervalos de tiempo durante los cuales los niveles no cambiaban. Los ejemplos del 3.6 al 3.8 ilustran el funcionamiento de la puerta OR con diferentes señales de entrada.

EJEMPLO 3.6

Si se aplican las dos señales de entrada, *A* y *B*, de la Figura 3.20 a la puerta OR, ¿cuál es la señal de salida resultante?



Cuando cualquiera de las entradas o ambas están a nivel ALTO, la salida es un nivel ALTO.

FIGURA 3.20

Solución La señal de salida *X* de una puerta OR de dos entradas será un nivel ALTO cuando una o ambas entradas estén a nivel ALTO, tal como muestra el diagra-

ma de tiempos. En este caso, ambas entradas nunca están al tiempo a nivel ALTO.

Problema relacionado Determinar la señal de salida y dibujar el diagrama de tiempos si la entrada *A* se cambia de modo que está a nivel ALTO desde el inicio del primer impulso hasta el final del segundo impulso.

EJEMPLO 3.7

Para las dos ondas de entrada, *A* y *B*, de la Figura 3.21, dibujar la onda de salida indicando su relación respecto a las entradas.

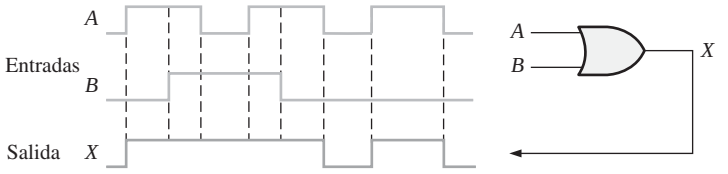


FIGURA 3.21

Solución Cuando una o ambas entradas están a nivel ALTO, la salida estará a nivel ALTO, como muestra la señal de salida *X* en el diagrama de tiempos.

Problema relacionado Determinar la señal de salida y dibujar el cronograma si el impulso central de la entrada *A* se sustituye por un nivel BAJO.

EJEMPLO 3.8

Para la puerta OR de 3 entradas de la Figura 3.22, determinar la señal de salida respecto de las entradas en función del tiempo.

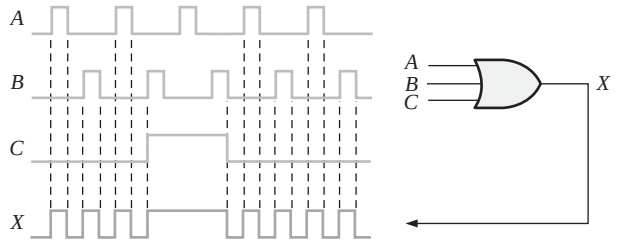


FIGURA 3.22

Solución La salida está a nivel ALTO cuando una o más entradas están a nivel ALTO, como muestra la señal de salida *X* en el diagrama de tiempos.

Problema relacionado Determinar la señal de salida y dibujar el diagrama de tiempos si la entrada *C* está siempre a nivel BAJO.

Expresiones lógicas de la puerta OR

▲ *Cuando las variables están separadas mediante el símbolo +, se aplica la operación OR.*

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

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

$$0 + 0 = 0$$
$$0 + 1 = 1$$
$$1 + 0 = 1$$
$$1 + 1 = 1$$

La suma booleana es lo mismo que la función OR.

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

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

$$X = A + B$$

La Figura 3.23(a) muestra el símbolo lógico de la puerta indicando las variables de entradas y de salida.

Para ampliar la expresión OR a más de dos variables de entrada, se usa una nueva letra para cada variable adicional. Por ejemplo, la función de una puerta OR de 3 entradas se puede expresar como $X = A + B + C$, y la expresión para una puerta OR de 4 entradas sería $X = A + B + C + D$, y así sucesivamente. Las ilustraciones (b) y (c) de la Figura 3.23 muestran, respectivamente, las variables de entrada de las puertas OR de tres y cuatro entradas.

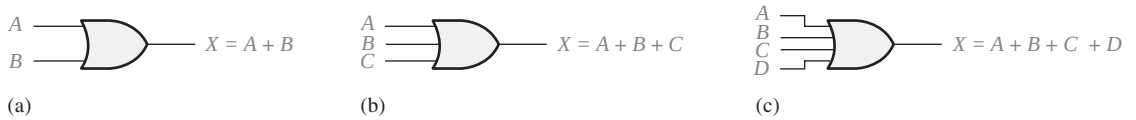


FIGURA 3.23 Expresiones booleanas de las puertas OR con dos, tres y cuatro entradas.

Como se muestra en la Tabla 3.6 para una puerta OR de dos entradas, el funcionamiento de la puerta OR se puede evaluar utilizando las expresiones booleanas para la salida X, sustituyendo todas las posibles combinaciones de 1 y 0 en las variables de entrada. Esta evaluación muestra que la salida X de una puerta OR es un 1 (nivel ALTO) cuando una o más de sus entradas son 1 (nivel ALTO). Un análisis similar se puede hacer para las puertas OR con cualquier número de variables de entrada.

A	*B*	*A + B = X*
0	0	0 + 0 = 0
0	1	0 + 1 = 1
1	0	1 + 0 = 1
1	1	1 + 1 = 1

TABLA 3.6

NOTAS INFORMÁTICAS

Otra operación de enmascaramiento que se usa en la programación de computadoras para hacer que determinados bits de un byte de datos sean igual a 1 (lo que se denomina activación) de forma selectiva, no afectando a otros bits, se lleva a cabo con la operación OR. Se utiliza una máscara que contenga un 1 en cualquier posición en la que un bit de datos desee ponerse a 1. Por ejemplo, si desea forzar que el bit más significativo de un byte de datos sea igual a 1, dejando los demás bits como están, puede aplicar la operación OR al byte de datos con la máscara 10000000.

Aplicación

En la Figura 3.24 se presenta parte de un sistema de alarma y detección de intrusos simplificado. Este sistema se podría utilizar en una habitación de una casa: una habitación con dos ventanas y una puerta. Los sensores son interruptores magnéticos que producen un nivel de salida ALTO cuando se abre la puerta (o las ventanas) y una salida a nivel BAJO cuando se cierra. Mientras que las ventanas y la puerta están aseguradas, los interruptores están cerrados y las tres entradas de la puerta OR están a nivel BAJO. Cuando se abre una de las ventanas o la puerta, en la entrada correspondiente de la puerta OR se genera un nivel ALTO y la salida de la puerta lógica se pone a nivel ALTO. Entonces se activa el circuito de alarma para advertir de la intrusión.

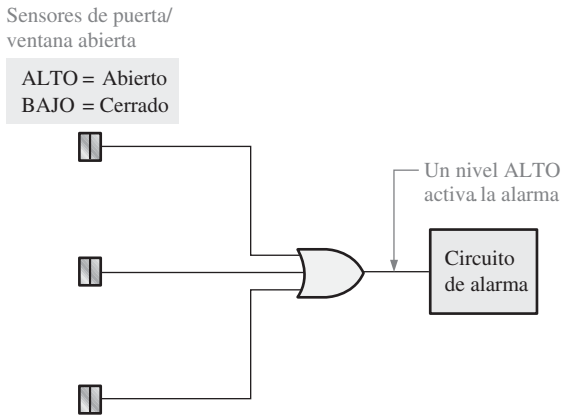


FIGURA 3.24 Un sistema de detección de intrusión simplificado que utiliza una puerta OR.

REVISIÓN DE LA SECCIÓN 3.3

1. ¿Cuándo se pone a nivel ALTO la salida de una puerta OR?
2. ¿Cuándo se pone a nivel BAJO la salida de una puerta OR?
3. Escribir la tabla de verdad para una puerta OR de tres entradas.

3.4 LA PUERTA NAND

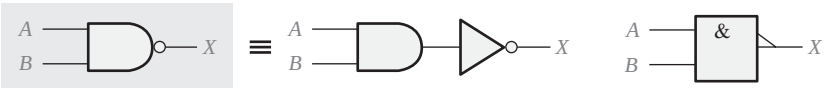
La puerta NAND es un elemento lógico popular, debido a que se puede utilizar como una puerta universal, es decir, las puertas NAND se pueden combinar para implementar las operaciones de las puertas AND, OR y del inversor. En el Capítulo 5, se examinará la propiedad universal de la puerta NAND.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Identificar una puerta NAND mediante su símbolo distintivo y su símbolo rectangular. ■ Describir la operación lógica de una puerta NAND. ■ Desarrollar la tabla de verdad de una puerta NAND con

cualquier número de entradas. ■ Generar el cronograma de una puerta NAND para cualquier forma de onda especificada en sus entradas. ■ Escribir la expresión lógica de una puerta NAND con cualquier número de entradas. ■ Describir la operación de la puerta NAND en términos de la puerta equivalente negativa–OR. ■ Desarrollar ejemplos de aplicaciones de la puerta NAND.

El término *NAND* es una contracción de NOT–AND, e implica una función AND con la salida complementada (negada). En la Figura 3.25(a) se muestra el símbolo lógico estándar para la puerta NAND de 2 entradas y su equivalente empleando los símbolos de la puerta AND seguida de un inversor, donde el símbolo ≡ significa "equivalente a". En la parte (b) se muestra el símbolo rectangular.



(a) Símbolo distintivo, puerta NAND de dos entradas y su equivalente NOT/AND.

(b) Símbolo rectangular, puerta NAND de dos entradas con indicador de polaridad.

FIGURA 3.25 Símbolos lógicos estándar de la puerta NAND (ANSI–/IEEE 91–1984).

Funcionamiento de la puerta NAND

La ***puerta NAND*** genera una salida a nivel BAJO sólo cuando todas las entradas están a nivel ALTO. Cuando cualquiera de las entradas está a nivel BAJO, la salida se pondrá a nivel ALTO. Para el caso concreto de la puerta NAND de dos entradas, como la mostrada en la Figura 3.25, con la designación *A* y *B* para las entradas y *X* para la salida, el modo de operación se puede establecer como sigue:

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

Observe que esta operación, en términos de nivel de salida, es la opuesta a la operación lógica AND. En una puerta NAND, el nivel BAJO (0) es el nivel activo o verdadero de salida, como indica el círculo de la salida. La Figura 3.26 ilustra la operación lógica de una puerta NAND de dos entradas, para las cuatro posibles combinaciones, y la Tabla 3.7 es la tabla de verdad, que resume la operación lógica de la puerta NAND de dos entradas.

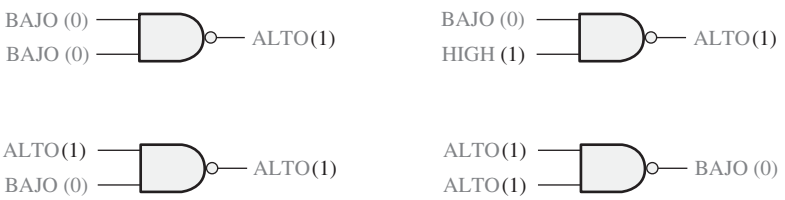


FIGURA 3.26 Funcionamiento de la puerta NAND de 2 entradas.

Funcionamiento con trenes de impulsos

Veamos ahora el funcionamiento de la puerta NAND con un tren de impulsos. Recuerde de su tabla de verdad que la salida se pone a nivel BAJO sólo cuando todas las entradas están a nivel ALTO.

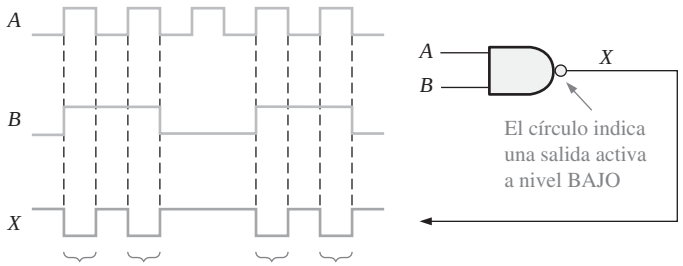
Operación equivalente negativa-OR de la puerta NAND. Es inherente al funcionamiento de la puerta NAND el hecho de que una o más entradas a nivel BAJO dan lugar a una salida a nivel ALTO. La Tabla 3.7 indica que

Entradas		Salida
A	*B*	*X*
0	0	1
0	1	1
1	0	1
1	1	0
1 = ALTO, 0 = BAJO		

TABLA 3.7 Tabla de verdad de la puerta NAND de 2 entradas.

EJEMPLO 3.9

Si a las entradas de una puerta NAND se aplican las formas de onda *A* y *B* de la Figura 3.27, determinar la forma de onda resultante de salida.



A y *B* están a nivel ALTO durante estos cuatro intervalos de tiempo. Por tanto, *X* es un nivel BAJO.

FIGURA 3.27

Solución La señal de salida *X* está a nivel BAJO sólo durante cuatro intervalos de tiempo, cuando ambas entradas, *A* y *B*, están a nivel ALTO, como muestra el diagrama de tiempos.

Problema relacionado Determinar la onda de salida y dibujar el diagrama de tiempos si se invierte la entrada *B*.

la salida *X* está a nivel ALTO (1) cuando cualquiera de las entradas, *A* y *B*, están a nivel BAJO (0). Desde este punto de vista, la puerta NAND se puede usar para realizar la operación OR que requiere una o más entradas a nivel BAJO, para generar una salida a nivel ALTO. Este modo de operación se denomina **negativa–OR**. En este contexto, el término *negativa* significa que las entradas se definen para que su estado activo o verdadero sea un nivel BAJO.

En una puerta NAND de dos entradas que funciona como una puerta negativa–OR, la salida *X* es un nivel ALTO si cualquiera de las entradas, *A* o *B*, es un nivel BAJO, o si ambas entradas, *A* y *B*, están a nivel BAJO.

Cuando la puerta NAND se emplea con uno o más niveles bajos en sus entradas en lugar de con niveles altos, se trata como una puerta negativa–OR y se representa con el símbolo lógico estándar de la Figura 3.29.

EJEMPLO 3.10

Obtener la forma de onda de salida para la puerta NAND de tres entradas de la Figura 3.28, donde además se presenta el diagrama de tiempos de las entradas.

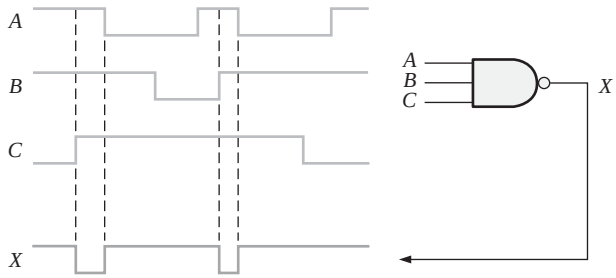


FIGURA 3.28

Solución La señal de salida *X* está a nivel BAJO sólo cuando las tres entradas están a nivel ALTO, tal como se muestra en el diagrama de tiempos.

Problema relacionado Determinar la forma de onda de salida y dibujar el diagrama de tiempos si se invierte la señal de entrada *A*.

Aunque los dos símbolos de esta figura representan la misma puerta física, sirven para definir el papel o el modo de operación en una aplicación particular, como se ilustra en los Ejemplos 3.11 hasta 3.13.

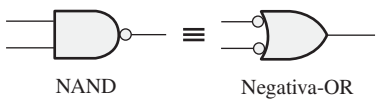


FIGURA 3.29 Símbolos estándar para representar las dos operaciones equivalentes de la puerta NAND.

EJEMPLO 3.11

Una planta de fabricación utiliza dos tanques para almacenar un determinado líquido químico que se requiere en un proceso de fabricación. Cada tanque dispone de un sensor que detecta cuándo el nivel del líquido cae al 25% del total. Los sensores generan una tensión de 5 V cuando los tanques están llenos por encima del 25%. Cuando el volumen de líquido en el tanque cae por debajo del 25%, el sensor genera un nivel de 0 V.

En el panel indicador se requiere un diodo emisor de luz (LED, *Light–Emitting Diode*) verde que indique que el nivel de ambos tanques está por encima del 25%. Como se indica, se puede utilizar una puerta NAND para implementar esta función.

Solución La Figura 3.30 muestra una puerta NAND de dos entradas conectadas a los sensores de nivel del tanque y la salida conectada al panel indicador. La operación puede definirse como sigue: si el nivel del tanque *A* y el nivel del tanque *B* están por encima de un cuarto del total, el LED se enciende.

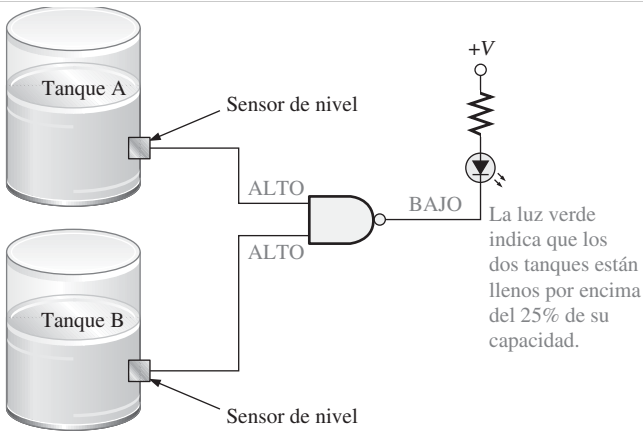


FIGURA 3.30

Mientras que la salida de *ambos* sensores esté a nivel ALTO (5 V), quiere decir que el nivel de los tanques está por encima del 25% del volumen total, y la salida de la puerta NAND está a nivel BAJO (0 V). El circuito del LED verde se activa para una tensión a nivel BAJO.

Problema relacionado ¿Cómo se puede modificar el circuito de la Figura 3.30 para controlar el nivel de tres tanques en lugar de dos?

EJEMPLO 3.12

El supervisor del proceso de fabricación descrito en el Ejemplo 3.11 ha decidido que sería preferible disponer de un LED rojo encendido cuando al menos el nivel de líquido de uno de los tanques estuviera por debajo del 25%, y que el LED verde se encendiera cuando el nivel en ambos tanques estuviera por encima de dicho límite. Veamos cómo se puede implementar este requisito.

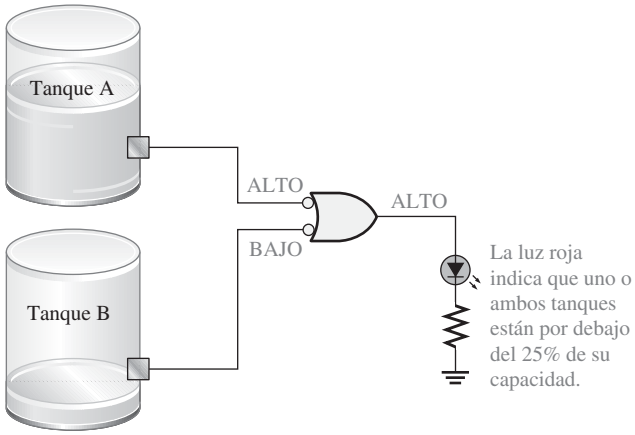


FIGURA 3.31

Solución La Figura 3.31 muestra que la puerta NAND está funcionando como una puerta negativa–OR, para detectar la ocurrencia de que al menos una de las entradas esté a nivel BAJO. Un sensor genera un nivel BAJO si el volumen en su tanque es del 25% o menor. Cuando esto ocurre, la salida de la puerta pasa a nivel ALTO, por lo que el LED rojo del panel indicador se enciende para un nivel de tensión ALTO. La operación puede definirse del siguiente modo: si el tanque *A* o el tanque *B*, *o* ambos están por debajo del 25% del total, entonces el LED se enciende.

Observe que en este ejemplo y en el Ejemplo 3.11, se usa la misma puerta NAND de dos entradas, pero en el esquema se ha empleado un símbolo de puerta diferente para ilustrar la diferencia funcional de las puertas NAND y negativa–OR.

Problema relacionado ¿Cómo se puede modificar el circuito de la Figura 3.31 para controlar cuatro tanques en lugar de dos?

EJEMPLO 3.13

Para la puerta NAND de cuatro entradas de la Figura 3.32, que opera como una puerta negativa–OR, determinar la salida con respecto a sus entradas.

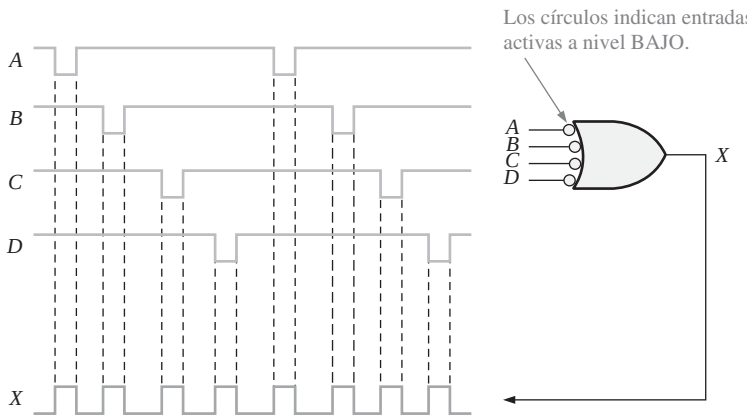


FIGURA 3.32

Solución La forma de onda de salida *X* estará a nivel ALTO siempre que una entrada esté a nivel BAJO, como se indica en el diagrama de tiempos.

Problema relacionado Determinar la forma de onda de salida, si se invierte la señal de entrada *A* antes de aplicarla a la puerta.

Expresiones lógicas para la puerta NAND

La expresión booleana para la puerta NAND de dos entradas es:

$$X = \overline{AB}$$

▲ *Una barra encima de una o varias variables indica una operación de inversión.*

Esta expresión quiere decir que las dos variables de entrada, A y B, se multiplican (AND) primero y luego se complementan, tal y como indica la barra sobre la expresión lógica correspondiente a AND. Ésta es una descripción lógica en forma de ecuación de la operación de una puerta NAND con dos entradas. Si evalúa esta expresión para todos los posibles valores de las dos variables de entrada, el resultado que se obtiene es el indicado en la Tabla 3.8.

A	B	$\overline{AB} = X$
0	0	$\overline{0 \cdot 0} = \overline{0} = 1$
0	1	$\overline{0 \cdot 1} = \overline{0} = 1$
1	0	$\overline{1 \cdot 0} = \overline{0} = 1$
1	1	$\overline{1 \cdot 1} = \overline{1} = 0$

TABLA 3.8

Una vez determinada la expresión lógica para una función lógica dada, esta función se puede evaluar para todos los valores posibles de las variables. La evaluación facilita de forma exacta la salida del circuito lógico para cada una de las condiciones de entrada y, por tanto, le proporciona una descripción completa de la operación lógica del circuito. La expresión para NAND se puede extender a más de dos variables de entrada, incluyendo letras adicionales para representar las otras variables.

REVISIÓN DE LA SECCIÓN 3.4

1. ¿Cuándo está a nivel BAJO la salida de una puerta NAND?
2. ¿Cuándo está a nivel ALTO la salida de una puerta NAND?
3. Describir las diferencias funcionales entre una puerta NAND y una puerta negativa–OR. ¿Tienen ambas la misma tabla de verdad?
4. Escribir la expresión de salida para una puerta NAND con tres entradas, A, B y C.

3.5 LA PUERTA NOR

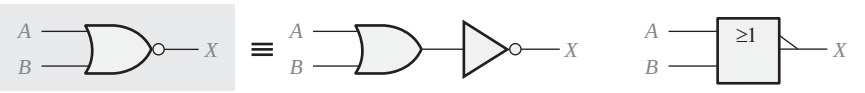
La puerta NOR, al igual que la puerta NAND, es un útil elemento lógico porque también se puede emplear como una puerta universal; es decir, las puertas NOR se pueden usar en combinación para implementar las operaciones AND, OR y del inversor. En el Capítulo 5 se examinará la propiedad universal de la puerta NOR.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Identificar una puerta NOR mediante su símbolo distintivo y su símbolo rectangular. ■ Describir el funcionamiento de una puerta NOR. ■ Desarrollar la tabla de verdad de una puerta NOR con cualquier número de entradas. ■ Generar el diagrama de tiempos de una puerta NOR para cualquier forma de onda especificada en sus entradas. ■ Escribir la expresión lógica de una puerta NOR con cualquier número de entradas. ■ Describir el funcionamiento de la puerta NOR en términos de su equivalente negativa–AND. ■ Desarrollar ejemplos de aplicaciones de la puerta NOR.

▲ *NOR es igual que OR excepto porque la salida está invertida.*

El término *NOR* es una contracción de NOT–OR e implica una función OR con la salida invertida (complementada). En la Figura 3.33(a) se muestra el símbolo lógico estándar para la puerta NOR de 2 entradas y su equivalente empleando los símbolos de la puerta OR seguida de un inversor. En la parte (b) se muestra el símbolo rectangular.



(a) Símbolo distintivo, puerta NOR de 2 entradas y su equivalente NOT/OR

(b) Símbolo rectangular, puerta NOR de 2 entradas con indicador de polaridad

FIGURA 3.33 Símbolo lógico estándar para la puerta NOR (ANSI/IEEE Std. 91–1984)

Funcionamiento de la puerta NOR

La ***puerta NOR*** genera una salida a nivel BAJO cuando *cualquiera* de sus entradas está a nivel ALTO. Sólo cuando todas sus entradas estén a nivel BAJO, la salida se pondrá a nivel ALTO. Para el caso concreto de la puerta NOR de dos entradas, que se muestra en la Figura 3.33, con la designación *A* y *B* para las entradas y *X* para la salida, el modo de operación se puede establecer como sigue:

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

Esta operación genera un nivel de salida opuesto al que genera la puerta OR. En una puerta NOR, el nivel BAJO es el nivel activo o verdadero de salida, como indica el círculo de la salida. La Figura 3.34 ilustra el funcionamiento de una puerta NOR de dos entradas, para las cuatro posibles combinaciones de entrada, y la Tabla 3.9 es la tabla de verdad para la puerta NOR de dos entradas.

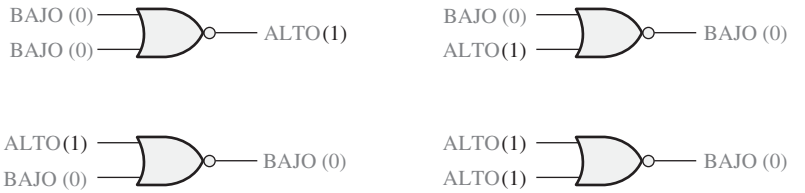


FIGURA 3.34 Funcionamiento de la puerta NOR de 2 entradas.

Entradas		Salida
A	***B***	***X***
0	0	1
0	1	0
1	0	0
1	1	0
1 = ALTO, 0 = BAJO		

TABLA 3.9 Tabla de verdad de una puerta NOR de 2 entradas.

Funcionamiento con trenes de impulsos

Los dos ejemplos siguientes ilustran la operación lógica de la puerta NOR con entradas que son trenes de impulsos. De nuevo, como en los demás tipos de puertas, simplemente deberemos aplicar la tabla de verdad para determinar la forma de onda de salida de acuerdo con el diagrama de tiempos de las entradas.

EJEMPLO 3.14

Si se aplican a la puerta NOR las dos señales mostradas en la Figura 3.35, ¿cómo es la señal de salida que se obtiene?

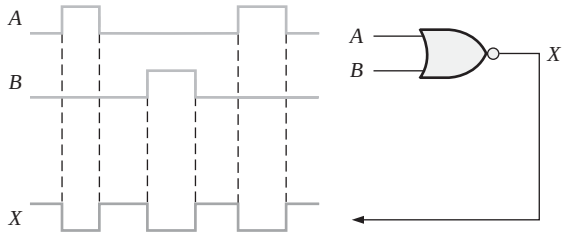


FIGURA 3.35

Solución

Cuando cualquier entrada de la puerta NOR está a nivel ALTO, la salida es un nivel BAJO, como se ve en la forma de onda de salida *X* del diagrama de tiempos.

Problema relacionado Invertir la entrada *B* y determinar la forma de onda de salida para esas entradas.

EJEMPLO 3.15

Determinar la señal de salida para la puerta NOR de tres entradas de la Figura 3.36, teniendo en cuenta el diagrama de tiempos correspondiente a las entradas.

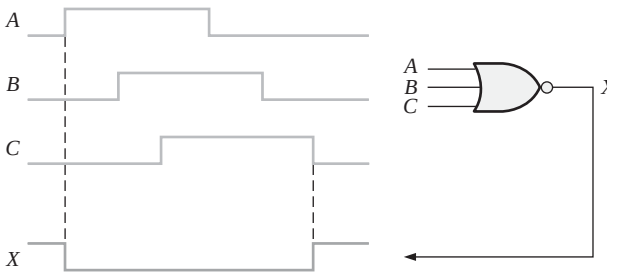


FIGURA 3.36

Solución La salida *X* se pone a nivel BAJO cuando cualquier entrada está a nivel ALTO, como muestra la señal de salida *X* en el diagrama de tiempos.

Problema relacionado Con las entradas *B* y *C* invertidas, determinar la salida y dibujar el diagrama de tiempos.

Operación equivalente negativa-AND de la puerta NOR. La puerta NOR, al igual que la puerta NAND, posee otro aspecto de su funcionamiento que es inherente a su función lógica. La Tabla 3.9 muestra que se genera un nivel ALTO en la salida sólo si todas las entradas están a nivel BAJO. Desde este punto de vista, la puerta NOR se puede utilizar como una operación AND cuyas entradas están a nivel BAJO y generan una salida a nivel ALTO. Este modo de operación se denomina **negativa–AND**. En este contexto, el término *negativa* significa que las entradas están definidas para que su estado activo o verdadero sea un nivel BAJO.

En una puerta NOR de dos entradas que funciona como una puerta negativa–AND, la salida *X* es un nivel ALTO cuando ambas entradas, *A* y *B*, están a nivel BAJO.

Cuando se quiere que la puerta NOR presente todas sus entradas a nivel BAJO en lugar de presentar uno o más niveles altos, se aplica la puerta negativa–AND y se representa mediante el símbolo estándar de la Figura 3.37. Es importante recordar que los dos símbolos de la Figura 3.37 representan la misma puerta física, y sólo sirven para distinguir entre los dos modos de operación lógicos. Los tres ejemplos siguientes ilustran esto.

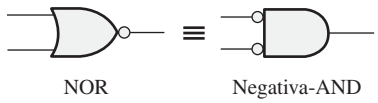


FIGURA 3.37 Símbolos estándar que representan dos operaciones equivalentes de la puerta NOR.

EJEMPLO 3.16

Se necesita un dispositivo para indicar cuándo se producen simultáneamente dos niveles de entrada bajos que dan lugar a un nivel de salida ALTO. Especificar el dispositivo.

Solución Para generar una salida a nivel ALTO cuando ambas entradas están a nivel BAJO se requiere una puerta negativa–AND, como se muestra en la Figura 3.38.

FIGURA 3.38

Problema relacionado Se necesita un dispositivo para indicar una salida a nivel BAJO, cuando uno o dos niveles altos se aplican a sus entradas. Especificar el dispositivo.

EJEMPLO 3.17

Como parte del sistema de monitorización funcional de un avión, se requiere un circuito para indicar el estado del tren de aterrizaje antes de tomar tierra. Se enciende un LED verde si los tres mecanismos de aterrizaje están correctamente extendidos cuando el interruptor para "bajar el tren de aterrizaje" se ha activado. Un LED rojo se enciende si cualquiera de los mecanismos falla al extenderse antes de aterrizar. Cuando uno de los mecanismos se extiende, el sensor correspondiente genera una tensión a nivel BAJO. Cuando uno de los mecanismos del tren de aterrizaje se retrae, su sensor genera una tensión a nivel ALTO. Implementar un circuito que cumpla estos requisitos.

Solución

La alimentación se aplica al circuito sólo cuando el interruptor para "bajar el tren de aterrizaje" se activa. Se debe utilizar una puerta NOR para cada uno de los dos requisitos, tal y como se muestra en la Figura 3.39. Una puerta NOR funciona como una puerta negativa–AND para detectar un nivel BAJO procedente de cada uno de los sensores de los mecanismos de aterrizaje. Cuando las tres entradas de la puerta están a nivel BAJO, los tres mecanismos (ruedas) están correctamente extendidos y dan lugar a una salida a nivel ALTO en la puerta negativa–AND, lo que hace que se encienda el LED verde. La otra puerta NOR funciona como tal para detectar si una o más de las ruedas de aterrizaje permanecen retraídas cuando se activa el interruptor que baja el tren de aterrizaje. Cuando una o más de las ruedas de aterrizaje permanecen retraídas; la salida a nivel ALTO procedente del sensor se detecta mediante la puerta NOR, que da lugar a una salida a nivel BAJO que enciende el LED rojo de aviso.

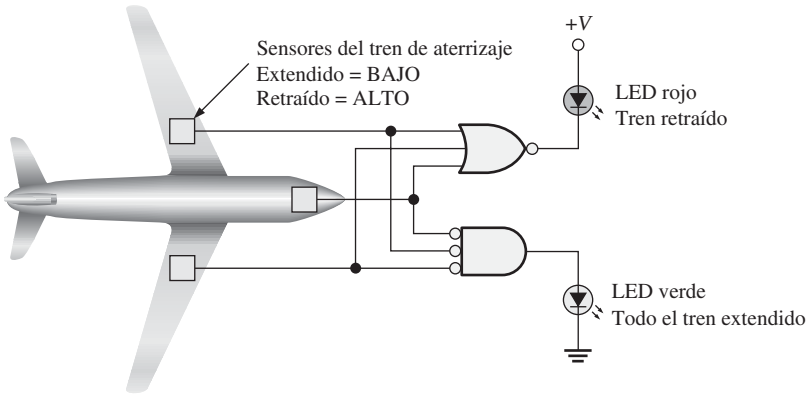


FIGURA 3.39

Problema relacionado

¿Qué tipo de puerta se debería usar para detectar si las tres ruedas del tren de aterrizaje están escondidas después de despegar? Suponer que se precisa un nivel de salida BAJO para activar un LED.

CONSEJOS PRÁCTICOS

Cuando se excita una carga tal como un LED con una puerta lógica, debería consultarse la hoja de características del fabricante para conocer los parámetros máximos de excitación (corriente de salida). Un circuito integrado de puerta lógica normal puede no ser capaz de manipular la corriente necesaria para determinadas cargas, como por ejemplo algunos LED. Hay disponibles puertas lógicas con salida tipo buffer, tal como una salida en colector abierto o en drenador abierto en muchos tipos de configuraciones de circuitos integrados. La capacidad de corriente de salida de las puertas lógicas en CI típicas están limitadas en el rango de los μA o de los mA. Por ejemplo, las puertas estándar TTL pueden manipular corrientes de salida de hasta 16 mA. La mayor parte de los LED precisan corrientes en el rango comprendido entre los 10 mA y 50 mA.

EJEMPLO 3.18

Para una puerta NOR de cuatro entradas que opera como una puerta negativa–AND como la de la Figura 3.40, determinar la salida en función de las entradas.

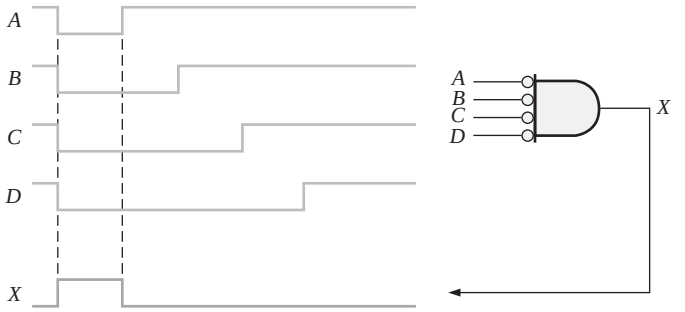


FIGURA 3.40

Solución En cualquier instante en el que todas las entradas estén a nivel BAJO, la salida estará a nivel ALTO, siendo la forma de onda de salida *X* la indicada en el cronograma.

Problema relacionado Determinar la salida para la entrada *D* invertida y dibujar el diagrama de tiempos.

Expresiones lógicas para la puerta NOR

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

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

Esta ecuación indica que las dos variables de entrada primero se suman (operación OR) y luego se complementan, tal y como indica la barra sobre la expresión lógica OR. Evaluando esta expresión, se obtienen los resultados mostrados en la Tabla 3.10. La expresión NOR puede extenderse a más de dos variables de entrada, incluyendo letras adicionales para representar otras variables.

A	B	$\overline{A + B} = X$
0	0	$\overline{0 + 0} = \overline{0} = 1$
0	1	$\overline{0 + 1} = \overline{1} = 0$
1	0	$\overline{1 + 0} = \overline{1} = 0$
1	1	$\overline{1 + 1} = \overline{1} = 0$

TABLA 3.10

REVISIÓN DE LA SECCIÓN 3.5

1. ¿Cuándo está a nivel ALTO la salida de una puerta NOR?
2. ¿Cuándo está a nivel BAJO la salida de una puerta NOR?

3. Describir la diferencia funcional entre una puerta NOR y una puerta negativa–AND. ¿Tienen ambas la misma tabla de verdad?
4. Escribir las expresiones de salida para una puerta NOR de tres entradas, siendo las variables de entrada *A*, *B* y *C*.

3.6 PUERTAS OR–EXCLUSIVA Y NOR–EXCLUSIVA

Las puertas OR–exclusiva y NOR–exclusiva se forman mediante la combinación de otras puertas que ya hemos tratado, como se verá en el Capítulo 5. Sin embargo, debido a su importancia fundamental en muchas aplicaciones, estas puertas se tratan como elementos lógicos básicos con su propio símbolo exclusivo.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Identificar las puertas OR–exclusiva y NOR–exclusiva mediante su símbolo distintivo y su símbolo rectangular. ■ Describir la operación lógica de las puertas OR–exclusiva y NOR–exclusiva. ■ Desarrollar las tablas de verdad de las puertas OR–exclusiva y NOR–exclusiva. ■ Generar el diagrama de tiempos de las puertas OR–exclusiva y NOR–exclusiva para cualquier forma de onda especificada en sus entradas. ■ Desarrollar ejemplos de aplicaciones de las puertas NOR–exclusiva y OR–exclusiva.

NOTAS INFORMÁTICAS

Varias puertas OR–exclusivas conectadas para formar un circuito sumador permiten que una computadora realice las operaciones de suma, sustracción, multiplicación y división en su unidad aritmético lógica (ALU). Una puerta OR–exclusiva se implementa con los circuitos lógicos básicos de las puertas AND, OR y NOT.

La puerta OR–exclusiva

▲ *En una puerta OR–exclusiva, entradas opuestas proporcionan una salida a nivel ALTO.*

En la Figura 3.41 se muestran los símbolos estándar para la puerta OR–exclusiva (XOR). La puerta XOR tiene sólo dos entradas.

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

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

(a) Símbolo distintivo (b) Símbolo rectangular con la puerta XOR

FIGURA 3.41 Símbolos lógicos estándar de la puerta OR–exclusiva.

En la Figura 3.42 se ilustran las cuatro posibles combinaciones de entrada y las salidas resultantes para la puerta XOR. El nivel ALTO es el nivel activo o verdadero de salida y sólo se produce cuando las entradas

están a niveles opuestos. La operación lógica de la puerta XOR se resume en la tabla de verdad mostrada en la Tabla 3.11.

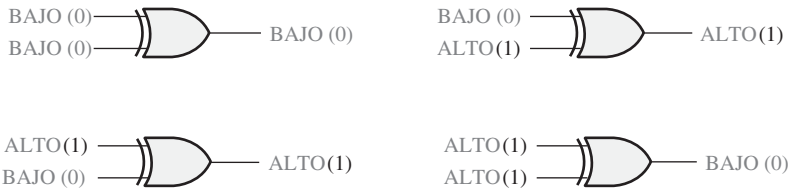


FIGURA 3.42 Todos los niveles lógicos posibles para una puerta OR–exclusiva.

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

TABLA 3.11 Tabla de verdad de la puerta OR–exclusiva

EJEMPLO 3.19

Un cierto sistema está formado por dos circuitos idénticos que funcionan en paralelo. Mientras que ambos funcionan correctamente, las salidas de los dos circuitos son iguales. Si uno de los circuitos falla, las salidas serán niveles opuestos en ese instante. Establecer un método para detectar que se ha producido un fallo en uno de los circuitos.

Solución

Las salidas de los circuitos se conectan a las entradas de una puerta XOR, tal y como muestra la Figura 3.43. Un fallo en cualquiera de los circuitos hace que las entradas de la puerta XOR tengan niveles opuestos. Esta condición da lugar a nivel ALTO en la salida de la puerta XOR, que indica que uno de los circuitos ha fallado.

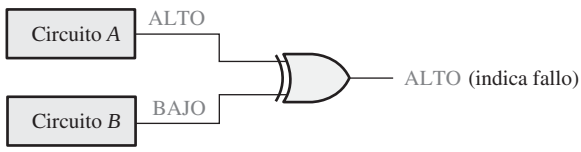


FIGURA 3.43

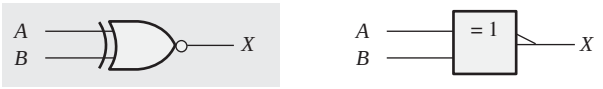
Problema relacionado

¿Detectará siempre la puerta OR–exclusiva fallos simultáneos en los dos circuitos de la Figura 3.43? Si la respuesta es no, ¿bajo qué condiciones los detectará?

La puerta NOR-exclusiva

En la Figura 3.44 se presentan los símbolos estándar de la **puerta NOR–exclusiva** (XNOR). Al igual que la puerta XOR, la puerta XNOR sólo tiene dos entradas. El círculo en la salida del símbolo de la puerta XNOR indica que su salida es la opuesta a la de la puerta XOR. Cuando dos niveles lógicos de entrada son opuestos, la salida de la puerta NOR–exclusiva es un nivel BAJO. La operación se puede expresar del siguiente modo (*A* y *B* son las entradas, y *X* es la salida).

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



(a) Símbolo distintivo (b) Símbolo rectangular

FIGURA 3.44 Símbolos lógicos estándar para la puerta NOR–exclusiva.

En la Figura 3.45 se muestran las cuatro posibles combinaciones de entrada y las salidas resultantes para la puerta XNOR. La operación lógica se resume en la Tabla 3.12. Observe que la salida es un nivel ALTO cuando las dos entradas están al mismo nivel.

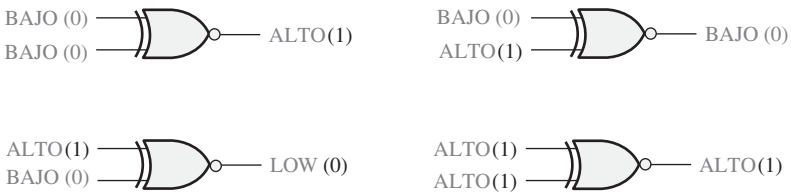


FIGURA 3.45 Todos los niveles lógicos posibles para una puerta NOR–exclusiva.

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

Tabla 3.12 Tabla de verdad de la puerta NOR–exclusiva.

Funcionamiento con trenes de impulsos

Al igual que hemos hecho con las puertas anteriores, vamos a examinar el funcionamiento de las puertas XOR y XNOR para trenes de impulsos en sus entradas. Como antes, aplicamos la tabla de verdad para cada intervalo de tiempo de los trenes de impulsos, como se muestra en la Figura 3.46 para una puerta XOR. Puede comprobar que las señales de entrada *A* y *B* tienen niveles opuestos en los intervalos t_2 y t_4. Por tanto, la sali-

da X se pone a nivel ALTO durante estos dos intervalos. En los intervalos t_1 y t_3, ambas entradas están al mismo nivel, BAJO o ALTO, y la salida se pone a nivel BAJO para estos impulsos, como muestra el diagrama de tiempos.

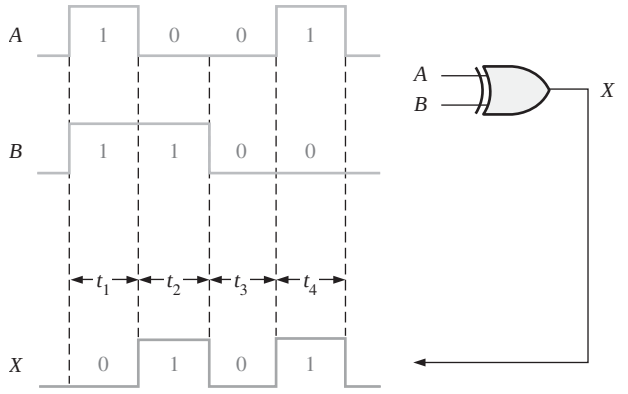


FIGURA 3.46 Ejemplo de funcionamiento de la puerta OR–exclusiva con trenes de impulsos.

EJEMPLO 3.20

Determinar las formas de onda de salida para la puerta XOR y la puerta XNOR, dadas las formas de onda de entrada A y B de la Figura 3.47.

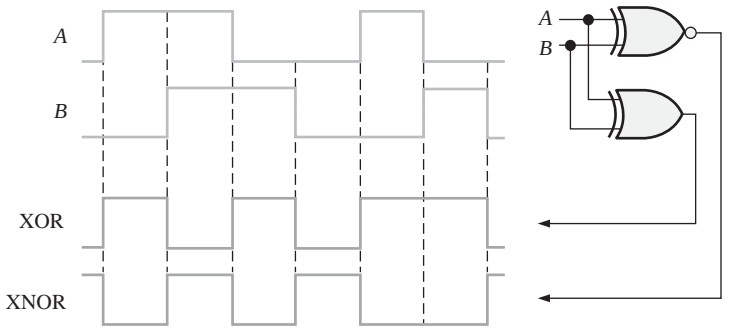


FIGURA 3.47

Solución Las formas de onda de salida son las que se indican en la Figura 3.47. Observe que la salida XOR está a nivel ALTO sólo cuando las entradas están a niveles opuestos. La salida XNOR está a nivel ALTO sólo cuando ambas entradas son iguales.

Problema relacionado Determinar la forma de onda de salida si las dos entradas, A y B, se invierten.

Aplicación

La puerta OR–exclusiva se puede utilizar como sumador de dos bits. Recuerde del Capítulo 2 que las reglas básicas de la suma binaria son: 0 + 0 = 0, 0 + 1 = 1, 1 + 0 = 1 y 1 +1 = 10. Un examen de la tabla de verdad

de la puerta XOR le demostrará que su salida es la suma binaria de los dos bits de entrada. En el caso en que ambas entradas sean 1, la salida de la suma es 0 y se pierde el acarreo de 1. En el Capítulo 6, verá cómo se combinan puertas XOR para implementar circuitos sumadores completos. La Figura 3.48 presenta la puerta XOR utilizada como sumador básico.

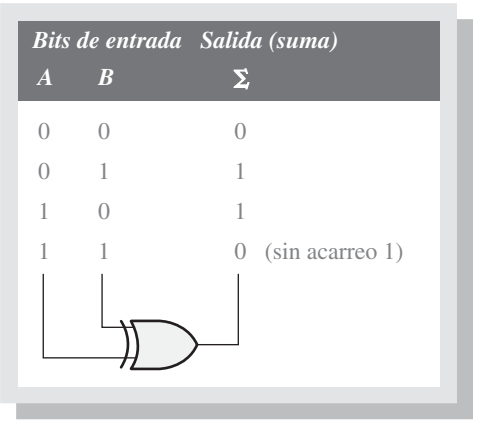


FIGURA 3.48 Puerta XOR utilizada como sumador de dos bits.

REVISIÓN DE LA SECCIÓN 3.6

1. ¿Cuándo está a nivel ALTO la salida de una puerta XOR?
2. ¿Cuándo está a nivel ALTO la salida de una puerta XNOR?
3. ¿Cómo se puede utilizar una puerta XOR para detectar que dos bits son diferentes?

3.7 LÓGICA PROGRAMABLE

En el Capítulo 1 se ha presentado la lógica programable. En esta sección, se aborda el concepto básico de matriz AND programable, que constituye el pilar de la mayor parte de la lógica programable y se cubren las principales tecnologías de proceso. Un dispositivo lógico programable (PLD, *Programmable Logic Device*) es aquel que inicialmente no tiene una función lógica fija pero que puede programarse para implementar cualquier diseño lógico. Como ya hemos visto, los dos tipos de PLD son el SPLD y el CPLD. Además del PLD, la otra categoría importante de dispositivo lógico programable es la FPGA. Con el fin de simplificar, haremos referencia a todos estos dispositivos como dispositivos PLD. También se abordan algunos conceptos importantes sobre programación.

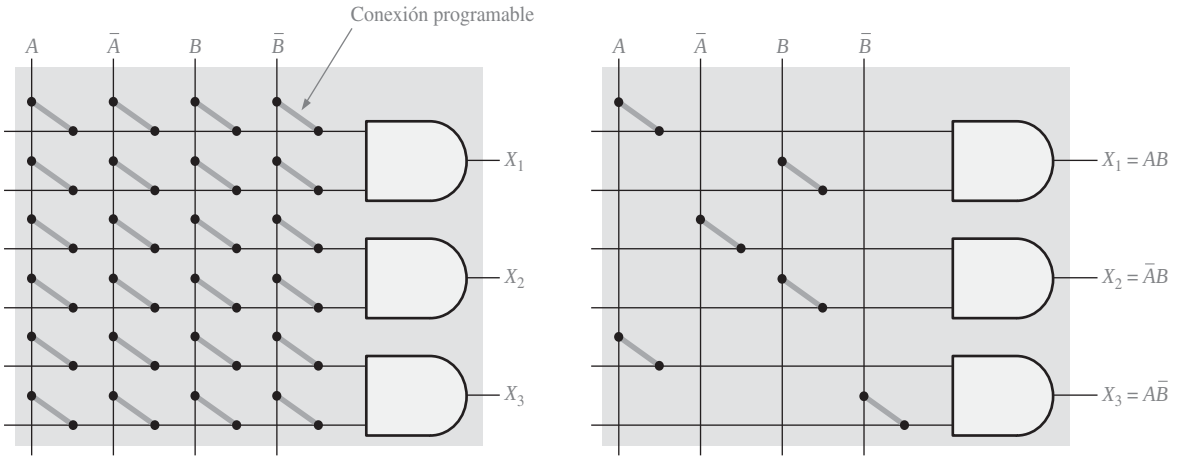
Al finalizar esta sección, el lector deberá ser capaz de:

■ Describir el concepto de matriz AND programable. ■ Explicar distintas tecnologías de procesos. ■ Describir la introducción de datos en forma de texto y mediante gráficos como los dos métodos de diseño lógico programable. ■ Describir métodos para descargar un diseño en un dispositivo lógico programable. ■ Explicar la programación dentro del sistema.

Concepto básico de la matriz AND

La mayor parte de los tipos de dispositivo PLD utilizan alguna forma de ***matriz AND***. Esta matriz está formada, básicamente, por puertas AND y una matriz de interconexiones con conexiones programables en cada

punto de intersección, como se muestra en la Figura 3.49(a). El propósito de las conexiones programables es establecer o interrumpir una conexión entre una línea de fila y una línea de columna de la matriz de interconexión. Para cada entrada de una puerta AND, sólo se deja intacta una conexión programable con el fin de conectar la variable deseada a la entrada de la puerta. La Figura 3.49(b) muestra una matriz después de haber sido programada.



(a) No programada (b) Programada

FIGURA 3.49 Concepto básico de una matriz AND programable.

EJEMPLO 3.21

Indicar cómo debe programarse la matriz AND de la Figura 3.49(a) para obtener las siguientes salidas: $X_1 = A\bar{B}$, $X_2 = \bar{A}B$ y $X_3 = \bar{A}\bar{B}$

Solución *Véase* la Figura 3.50.

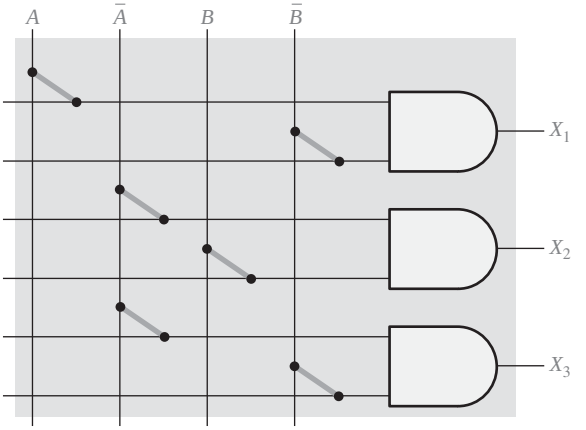


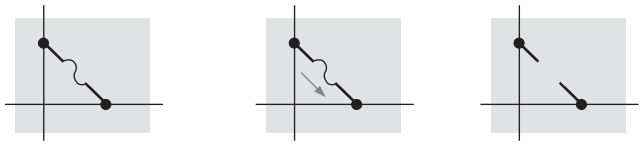
FIGURA 3.50

Problema relacionado ¿Cuántas filas, columnas y entradas de puerta AND son necesarias para tres variables de entrada en una matriz de puertas AND de 3 entradas?

Tecnologías de proceso basadas en conexiones programables

En los PLD se emplean varias tecnologías de proceso diferentes basadas en conexiones programables.

Tecnología basada en fusible. Ésta fue la tecnología original basada en conexiones programables, y todavía se emplea en algunos SPLD. El ***fusible*** es una conexión de metal que conecta una fila y una columna en la matriz de interconexión. Antes de realizar la programación, en cada intersección existe una conexión mediante fusible. Para programar un dispositivo, los fusibles seleccionados se abren haciendo pasar a su través una corriente que permita "fundir" el fusible y romper la conexión. Los fusibles que permanecen intactos proporcionan una conexión entre las filas y las columnas. En la Figura 3.51 se muestra una conexión mediante fusible. Los dispositivos lógicos programables que usan esta tecnología sólo pueden programarse una vez (**OTP,** *One–Time Programmable*).



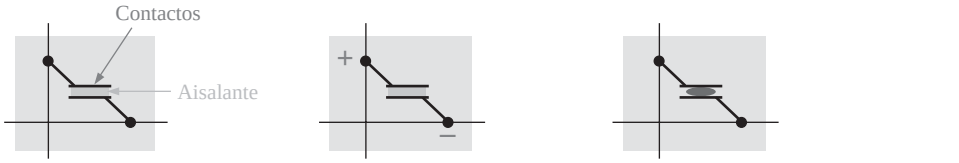
(a) Fusible intacto antes de la programación

(b) Corriente de programación

(c) Fusible abierto después de la programación

FIGURA 3.51 Conexión programable mediante fusible.

Tecnología basada en antifusible. Una conexión programable mediante **antifusible** es lo opuesto a una conexión mediante fusible. En lugar de interrumpir la conexión, durante la programación se establece una conexión. Inicialmente, un antifusible es un circuito abierto mientras que un fusible se comporta como un cortocircuito. Antes de llevar a cabo la programación, no existe ninguna conexión entre las filas y las columnas de la matriz de interconexión. Un antifusible está formado, básicamente, por dos conductores separados por un aislante. Para programar un dispositivo utilizando esta técnica, una herramienta de programación aplica una tensión adecuada a los antifusibles seleccionados para romper el aislamiento entre los dos conductores, de modo que el aislante pase a ser una conexión de baja resistencia. En la Figura 3.52 se muestra una conexión mediante antifusible. Los dispositivos antifusible también son dispositivos que sólo pueden programarse una vez (OTP).



(a) Antifusible abierto antes de la programación

(b) La tensión de programación rompe la capa de aislante para crear el contacto

(c) Antifusible cortocircuitado de forma efectiva después de la programación

FIGURA 3.52 Conexión programable mediante antifusible.

Tecnología basada en EPROM. En determinados dispositivos lógicos programables, las conexiones programables son similares a las celdas de memoria de las ***EPROM*** (*Electrically Programmable Read–Only Memory*, memoria de sólo lectura eléctricamente programable). Este tipo de PLD se programa empleando una herramienta especial conocida como programador de dispositivos. El dispositivo se inserta en el programador, el cual está conectado a una computadora que ejecuta el software de programación. La mayoría de los PLD basados en EPROM son dispositivos programables una sola vez (OTP). Sin embargo, aquellos con encapsulado de ventana pueden borrarse utilizando luz ultravioleta (UV) y volverse a programar utilizando una herramienta estándar de programación de dispositivos PLD. La tecnología EPROM emplea un tipo especial de transistor MOS, conocido como transistor de puerta flotante, como conexión programable. El dispositivo de puerta flotante usa un proceso denominado tunelización de Fowler–Nordheim para colocar electrones en la estructura de puerta flotante.

En una matriz AND programable, el transistor de puerta flotante actúa como un interruptor que permite conectar una fila bien a un nivel ALTO o a un nivel BAJO, dependiendo de la variable de entrada. Para las variables de entrada que no se utilizan, el transistor se programa de manera que esté permanentemente *desactivado* (abierto). La Figura 3.53 muestra una puerta AND en una matriz simple. La variable *A* controla el estado del transistor de la primera columna y la variable *B* controla el transistor de la tercera columna. Cuando un transistor está desactivado (no conduce), es decir, se comporta como un interruptor abierto, la línea de entrada a la puerta AND está a un nivel $+V$ (ALTO). Cuando un transistor está activado (*on*), es decir, se comporta como un interruptor cerrado, la línea de entrada se conecta a tierra (nivel BAJO). Si la variable *A* o *B* está a 0 (nivel BAJO), el transistor está activado manteniendo la línea de entrada a la puerta AND a nivel BAJO. Si la variable *A* o *B* está a 1 (nivel ALTO), el transistor no está *activado* (*off*) manteniendo la línea de entrada a la puerta AND a nivel ALTO.

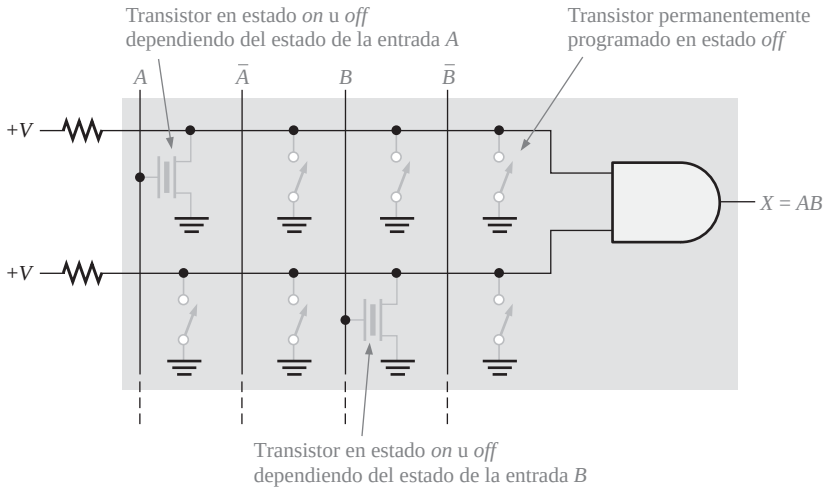


FIGURA 3.53 Una matriz AND con tecnología EPROM. Por simplicidad, sólo se muestra una puerta en la matriz.

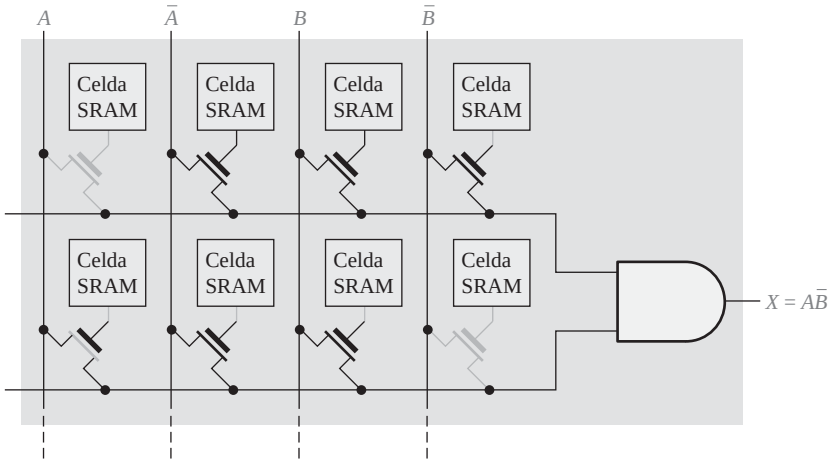
Tecnología basada en EEPROM. La tecnología EEPROM, (*Electrically Erasable Programmable Read–Only Memory*) es similar a la EPROM porque también utiliza un tipo de transistor de puerta flotante en celdas E^2CMOS. La diferencia es que las EEPROM se pueden borrar y volver a programar eléctricamente sin tener que usar luz UV o herramientas especiales. Un dispositivo E^2CMOS puede programarse después de haberse montado en una tarjeta de circuito impreso y muchos pueden reprogramarse mientras que operan dentro de un sistema. Esto se denomina programación dentro del sistema (**ISP**, *In–System Programming*). La Figura 3.53 también puede verse como un ejemplo para representar una matriz AND con tecnología EEPROM.

NOTAS INFORMÁTICAS

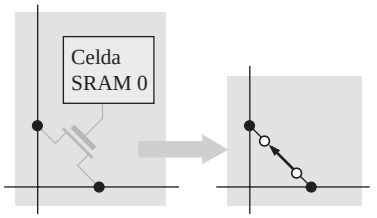
La mayoría de los diseños en el nivel de sistema incorporan diversos dispositivos como memorias RAM, ROM, controladores y procesadores que se interconectan mediante una gran cantidad de dispositivos lógicos de propósito general que, a menudo, se dice que son el "pegamento" lógico. Los PLD están reemplazando a muchos de los dispositivos "pegamento" SSI y MSI. El uso de los PLD proporciona una reducción en el número de encapsulados, por ejemplo, en los sistemas de memoria de las computadoras, los PLD pueden emplearse para la decodificación de direcciones de memoria y para generar señales de escritura en memoria, así como otras funciones.

Tecnología basada en SRAM. Muchas FPGA y algunos CPLD utilizan una tecnología de proceso similar a la que emplean las memorias ***SRAM*** (*Static Random Access Memory*, memoria estática de acceso aleatorio). El concepto fundamental de las matrices lógicas programables basadas en SRAM se ilustra en la Figura 3.54(a). Se emplea una celda de memoria tipo SRAM para activar o desactivar un transistor con el fin de conectar o desconectar las filas y columnas. Por ejemplo, cuando la celda de memoria contiene un 1 (en gris), el transistor se *activa* (*on)* y conecta las líneas de fila y columna asociadas, como se muestra en la parte (b) de la figura. Si la celda de memoria contiene un 0 (en negro), el transistor se *desactiva* (*off*) y no se establece ninguna conexión entre las líneas, como se muestra en la parte (c).

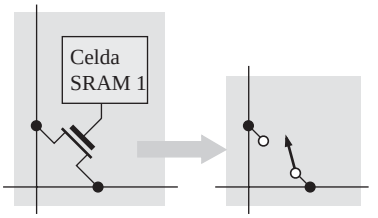
La tecnología basada en SRAM es diferente de las otras tecnologías de proceso vistas hasta ahora en el sentido de que es una tecnología volátil. Esto quiere decir que una celda SRAM no conserva los datos cuando se desconecta la alimentación. Los datos de la programación deben cargarse en una memoria no volátil y, cuando se conecta la alimentación, los datos almacenados en memoria reprograman el PLD basado en SRAM.



(a) Matriz programable basada en SRAM



(b) Transistor *on*



(c) Transistor *off*

FIGURA 3.54 Concepto básico de una matriz AND con tecnología basada en SRAM.

Las tecnologías de proceso basadas en fusible, antifusible, EPROM y EEPROM son no volátiles, por lo que conservan su programación cuando se desconecta la alimentación. Un fusible queda permanentemente abierto, un antifusible queda permanentemente cerrado y los transistores de puerta flotante empleados en las matrices basadas en EPROM y EEPROM pueden conservar indefinidamente sus estados *on* u *off*.

Programación de dispositivos

En el Capítulo 1 se ha presentado el concepto general de programación y también hemos visto cómo pueden realizarse interconexiones en una matriz sencilla abriendo y cerrando las conexiones programables. Los SPLD, los CPLD y las FPGA se programan prácticamente de la misma forma. Los dispositivos que emplean una tecnología de proceso OTP, como las basadas en fusibles, antifusibles o EPROM, deben programarse empleando una herramienta hardware especial denominada *programador*. El programador se conecta a una computadora mediante un cable de interfaz estándar, como se muestra en la Figura 3.55. El software de desarrollo se instala en la computadora y el dispositivo se inserta en el zócalo del programador. La mayoría de los programadores disponen de adaptadores, como el que se muestra en la figura, que permiten insertar diferentes tipos de encapsulados.

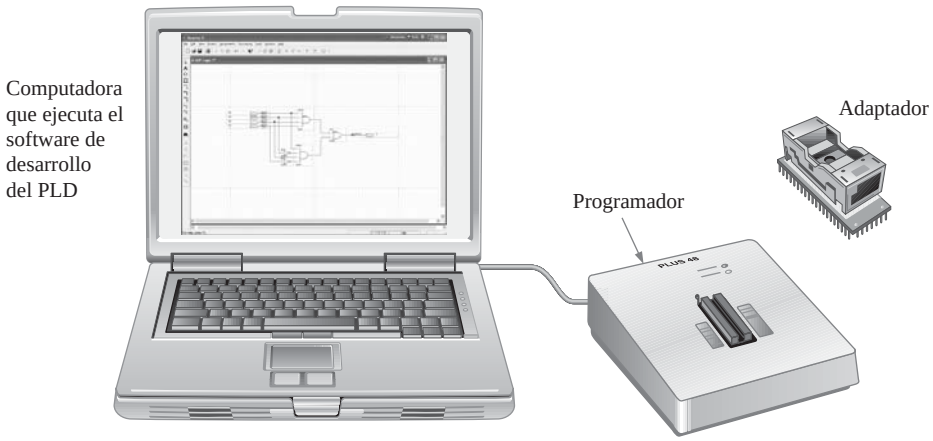


FIGURA 3.55 Configuración para programar un PLD con una herramienta de programación (programador).

Los dispositivos lógicos programables basados en EEPROM y SRAM son reprogramables y pueden reconfigurarse múltiples veces. Aunque para este tipo de dispositivo puede utilizarse un programador de dispositivos, generalmente, se programan inicialmente sobre una tarjeta de desarrollo de PLD, como se muestra en la Figura 3.56. Puede desarrollarse un diseño lógico utilizando esta técnica porque cualquier cambio necesario que surja durante el proceso de diseño podrá implementarse fácilmente simplemente reprogramando el PLD. Un PLD en el que se puede descargar un diseño lógico software se denomina ***dispositivo objetivo***. Además del dispositivo objetivo, normalmente, las tarjetas de desarrollo proporcionan circuitería adicional y conectores para poder establecer la interfaz con la computadora y otros circuitos periféricos. Además, la tarjeta de desarrollo dispone de puntos de prueba y de dispositivos de presentación que permiten observar el funcionamiento del dispositivo programado

Introducción del diseño. Como hemos visto en el Capítulo 1, la introducción del diseño consiste en programar el diseño lógico empleando el software de desarrollo. Las dos formas principales para introducir un diseño son mediante una interfaz de texto o mediante una interfaz gráfica (esquemáticos), por lo que habitual-

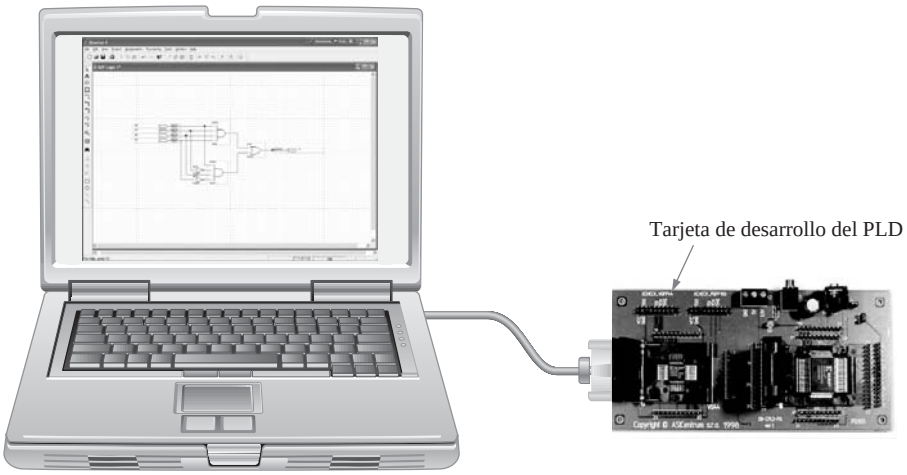


FIGURA 3.56 Configuración para programar dispositivos lógicos reprogramables.

mente los fabricantes de dispositivos lógicos programables proporcionan paquetes de software para sus dispositivos que permiten utilizar ambos métodos para la introducción de datos.

En la mayoría de los software de desarrollo, independientemente del fabricante, la **interfaz de texto** permite emplear dos o más lenguajes de desarrollo hardware (**HDL**, *Hardware Development Language*). Por ejemplo, todos los paquetes de software soportan los lenguajes estándar HDL del IEEE, VHDL y Verilog. Algunos paquetes de software también permiten el uso de ciertos lenguajes propietarios como ABEL, CUPL y AHDL.

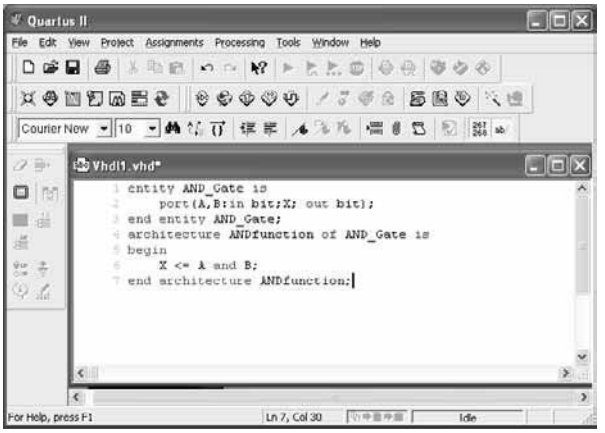
Con la **interfaz gráfica (esquemáticos),** los símbolos lógicos como por ejemplo, de las puertas AND y OR se colocan en la pantalla y se interconectan para formar el circuito deseado. Con este método, se emplean los símbolos lógicos familiares, aunque lo que hace realmente el software es convertir cada símbolo y las interconexiones en un archivo de texto que la computadora puede utilizar; el usuario no ve este proceso. En la Figura 3.57 se muestra un ejemplo de ambas interfaces. Por regla general, la interfaz gráfica se utiliza para los circuitos lógicos menos complejos y la interfaz de texto, aunque puede utilizarse para lógica sencilla, se usa para implementaciones más complejas.

Programación dentro del sistema (ISP)

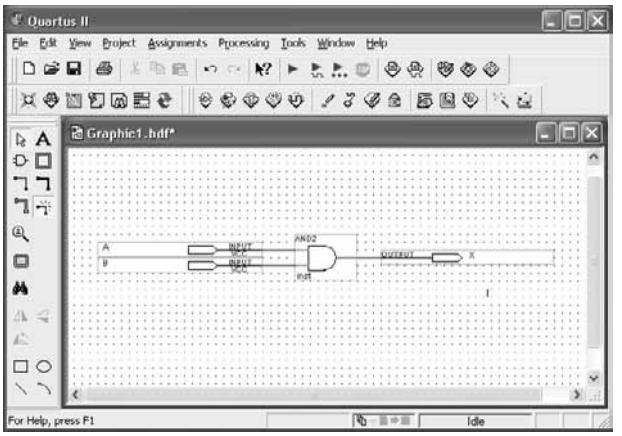
Ciertos CPLD y FPGA pueden programarse después de haberse incluido en la placa de circuito impreso (PCB) de un sistema. Después de haber desarrollado y probado por completo un diseño lógico en una tarjeta de desarrollo, se puede programar entonces un dispositivo "en blanco" que ya haya sido soldado sobre la placa del sistema en que tendrá que operar. También, si es necesario incorporar un cambio en el diseño, el dispositivo ya montado en la placa puede reconfigurarse para incluir las modificaciones al diseño.

En el caso de un diseño ya en producción, la programación de un dispositivo montado en la tarjeta del sistema minimiza la manipulación y elimina la necesidad de mantener stocks de dispositivos preprogramados. Esto también descarta la posibilidad de que se incluyan componentes erróneos en un producto. Los dispositivos no programados (en blanco) pueden mantenerse en el almacén y programarse a medida que se necesiten. Esto minimiza el capital que necesita un negocio para inventarios y mejora la calidad de los productos.

JTAG. El estándar establecido por el grupo *Joint Test Action Group* es el nombre normalmente utilizado para el estándar IEEE 1149.1. El estándar ***JTAG*** fue desarrollado para proporcionar un método sencillo, denominado análisis de contorno (*boundary scan*), para probar la funcionalidad de los dispositivos programables, así como para probar circuitos impresos con el fin de detectar conexiones malas (pines cortocircuitados, pines



(a) Entrada de texto VHDL



(b) Entrada gráfica equivalente (esquemático)

FIGURA 3.57 Ejemplos de introducción del diseño de una puerta AND.

abiertos, pistas cortadas, etc.). Más recientemente, JTAG se ha utilizado como una forma adecuada de configurar los dispositivos programables dentro del sistema. A medida que la demanda de productos actualizables sobre el terreno aumente, el uso de JTAG como una técnica apropiada de reprogramación de dispositivos CPLD y FPGA continuará creciendo.

Los dispositivos compatibles con JTAG disponen de hardware interno dedicado que interpreta las instrucciones y datos proporcionados por cuatro señales dedicadas. El estándar JTAG define estas señales como TDI (*Test Data In*), TDO (*Test Data Out*), TMS (*Test Mode Select*) y TCK (*Test Clock*). El hardware dedicado JTAG interpreta las instrucciones y los datos de las señales TDI y TMS, y envía los datos en la señal TDO. La señal TCK se utiliza para sincronizar el proceso. En la Figura 3.58 se muestra una tarjeta de circuito impreso compatible con JTAG.

Procesador integrado. Otra técnica para la programación dentro del sistema es el uso de una memoria y un microprocesador integrado. El procesador se integra dentro del sistema junto con el CPLD o la FPGA y el resto de la circuitería, y tiene como objetivo establecer la configuración dentro del sistema del dispositivo programable.

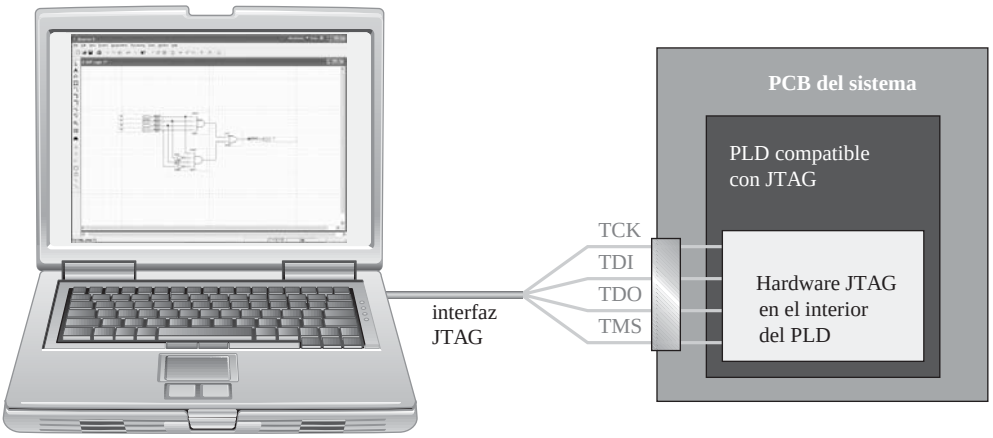


FIGURA 3.58 Ilustración simplificada de una programación dentro del sistema a través de una interfaz JTAG.

Como ya ha aprendido, los dispositivos basados en SRAM son volátiles y pierden los datos programados cuando se desconecta la alimentación. Por tanto, es necesario almacenar los datos de la programación en una PROM (*Programmable Read–Only Memory*, memoria programable de sólo lectura), que es no volátil. Cuando se conecta la alimentación, el procesador integrado toma el control para transferir los datos almacenados desde la PROM al CPLD o la FPGA.

Además, el procesador integrado en ocasiones se emplea para reconfigurar un dispositivo programable mientras que el sistema está en funcionamiento. En este caso, los cambios de diseño se realizan por software y los nuevos datos se cargan en un PROM sin perturbar el funcionamiento del sistema. El procesador controla la transferencia de los datos al dispositivo "sobre la marcha" durante un instante apropiado. En la Figura 3.59 se muestra un simple diagrama de bloques de un sistema lógico programable con procesador integrado.

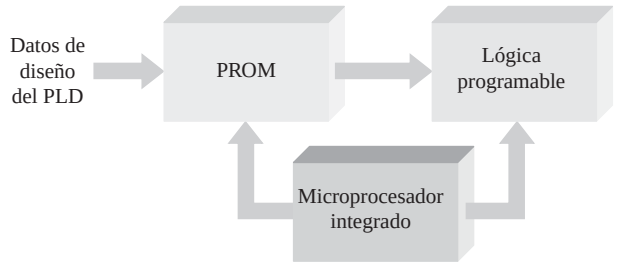


FIGURA 3.59 Diagrama de bloques simplificado de un PLD con una memoria y un procesador integrado.

REVISIÓN DE LA SECCIÓN 3.7

1. Enumere cinco tecnologías de proceso utilizadas para las conexiones programables en los dispositivos lógicos programables.
2. ¿Qué significa el término *volátil* en relación con los PLD y qué tecnologías de proceso son volátiles?
3. ¿Cuáles son los dos métodos disponibles para la introducción del diseño en la programación de los PLD y las FPGA?
4. Defina JTAG.

3.8 LÓGICA DE FUNCIÓN FIJA

Existen dos tecnologías de circuitos integrados digitales que se usan para implementar las puertas lógicas básicas: CMOS y TTL. Las operaciones lógicas NOT, AND, OR, NAND, NOR y OR–exclusiva son las mismas, independientemente de la tecnología de circuitos integrados que se utilice; es decir, una puerta AND tiene la misma función lógica se implemente con la tecnología CMOS o TTL.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Identificar las series más comunes CMOS y TTL. ■ Comparar las tecnologías CMOS y TTL en términos de tipos de dispositivos y parámetros de funcionamiento. ■ Definir el *retardo de propagación*. ■ Definir la *disipación de potencia*. ■ Definir el *fan–out*. ■ Definir el producto *velocidad–potencia*. ■ Interpretar la información básica de una hoja de características.

El término ***CMOS*** corresponde a *Complementary Metal–Oxide Semiconductor* (semiconductor metal–óxido complementario) y se implementa con un tipo de transistor de efecto de campo. ***TTL*** (*Transistor-Transistor Logic*, lógica transistor–transistor) y se implementa mediante transistores bipolares. Tenga en cuenta que CMOS y TTL sólo difieren en el tipo de componentes de circuito y los valores de los parámetros, y no en las operaciones lógicas básicas. Una puerta AND CMOS realiza la misma operación lógica que una puerta AND TTL. Esto también es cierto para todas las operaciones lógicas básicas restantes. La diferencia entre CMOS y TTL se encuentra en las características de funcionamiento, tal como la velocidad de conmutación (retardo de propagación), la disipación de potencia, la inmunidad al ruido y otros parámetros.

CMOS

Existe poco desacuerdo sobre cuál es la tecnología de circuitos, CMOS o TTL, más ampliamente utilizada. Parece que CMOS está comenzando a ser la tecnología dominante y podría reemplazar a la tecnología TTL en los CI de pequeña y mediana escala. Aunque TTL ha dominado durante muchos años, principalmente debido a sus altas velocidades de conmutación y a una enorme variedad de tipos de dispositivos, la tecnología CMOS siempre ha tenido la ventaja de ofrecer una mucho menor disipación de potencia, aunque dicho parámetro depende de la frecuencia. Las velocidades de conmutación de CMOS han mejorado extremadamente y ahora pueden competir con TTL, a la vez que la baja disipación de potencia y otros factores deseables se han mantenido a medida que la tecnología avanzaba.

Series CMOS. Las categorías de CMOS en términos de tensión de alimentación continua son la serie CMOS de 5 V, la serie CMOS de 3,3 V, la serie CMOS de 2,5 V y la serie CMOS de 1,8 V. Las series CMOS de más baja tensión son el resultado de un desarrollo más reciente y de un esfuerzo por reducir la disipación de potencia. Puesto que la disipación de potencia es proporcional al cuadrado de la tensión, una reducción de 5 V a 3,3 V, por ejemplo, disminuye la potencia en un 34%, sin que el resto de los factores varíen.

Dentro de cada categoría según la tensión de alimentación, hay disponibles varias series de puertas lógicas CMOS. Estas series pertenecientes a la familia CMOS difieren en sus características de funcionamiento y se designan mediante los prefijos 74 ó 54, seguidos de una letra o letras que indican la serie y, a continuación, un número que indica el tipo de dispositivo lógico. El prefijo 74 indica que se trata de un dispositivo comercial de propósito general, y el prefijo 54 indica que es un dispositivo militar para aplicaciones en entornos más exigentes. En este libro sólo haremos referencia a los dispositivos que llevan el prefijo 74. La serie básica CMOS de 5 V y sus denominaciones son las siguientes:

- 74HC y 74HCT. CMOS de alta velocidad (la "T" indica compatibilidad TTL)
- 74AC y 74ACT. CMOS avanzada
- 74AHC y 74AHCT. CMOS de alta velocidad avanzada

La serie básica CMOS de 3,3 V y sus denominaciones son las siguientes:

- 74LV. CMOS de baja tensión
- 74LVC. CMOS de baja tensión.
- 74ALVC. CMOS de baja tensión avanzada.

Además de la serie 74, todavía existe la serie 4000, que es una tecnología CMOS más antigua y de baja velocidad, aunque su uso está limitado. Además de las series CMOS "puras" existen series que combinan ambas tecnologías, CMOS y TTL, que se denominan BiCMOS. La serie básica BiCMOS y sus denominaciones son las siguientes:

- 74BCT. BiCMOS
- 74ABT. BiCMOS avanzada.
- 74LVT. BiCMOS de baja tensión.
- 74ALB. BiCMOS de baja tensión.

TTL

La tecnología TTL ha sido y es todavía una tecnología de circuitos integrados digitales muy popular. Una ventaja de esta tecnología es que no es sensible a las descargas electrostáticas como lo es la tecnología CMOS y, por tanto, es más práctica en la realización de experimentos de laboratorio y la elaboración de prototipos, ya que no es necesario preocuparse por los problemas de manipulación.

Series TTL. Al igual que con la tecnología CMOS, hay disponibles varias series de puertas lógicas TTL, las cuales operan todas ellas con 5 V de alimentación de continua. Estas series pertenecientes a la familia TTL difieren en sus características de funcionamiento y se denominan mediante los prefijos 74 ó 54 seguidos por una letra o letras que indican la serie y un número que indica el tipo de dispositivo lógico de la serie. Un circuito integrado TTL puede distinguirse de un circuito integrado CMOS por las letras que siguen a los prefijos 74 y 54.

Las series básicas TTL y sus denominaciones son las siguientes:

- 74: TTL estándar (sin letra).
- 74S: TTL Schottky.
- 74AS: TTL Schottky avanzada.
- 74LS: TTL Schottky de baja potencia.
- 74ALS: TTL Schottky de baja potencia avanzada.
- 74F: TTL rápida.

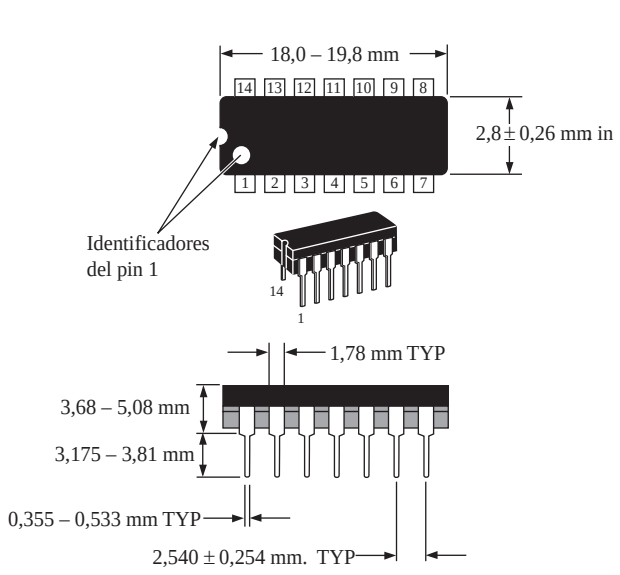
Tipos de puertas lógicas de función fija

Todas las operaciones lógicas básicas: NOT, AND, OR, NAND, NOR, OR–exclusiva (XOR) y NOR–exclusiva (XNOR) están disponibles en las tecnologías CMOS y TTL. También están disponibles puertas con salida búfer para excitar cargas que requieran altas corrientes. Los tipos de configuraciones de puerta normalmente disponibles en los circuitos integrados se identifican mediante los dos o tres dígitos finales de la designación de la serie. Por ejemplo, 74LS04 es un circuito integrado inversor séxtuple **Schottky** de baja potencia. Algunas de las configuraciones de puertas lógicas habituales y sus dígitos de identificación estándar son los siguientes:

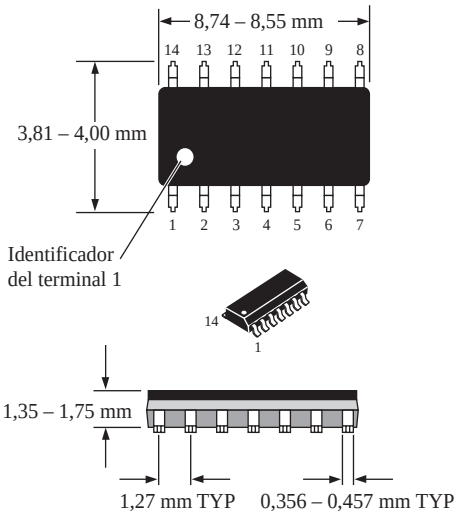
- Cuádruple NAND de dos entradas: **00**

- Cuádruple NOR de dos entradas: **02**
- Inversor séxtuple: **04**
- Cuádruple AND de dos entradas: **08**
- Triple NAND de tres entradas: **10**
- Triple AND de tres entradas: **11**
- Doble NAND de cuatro entradas: **20**
- Doble AND de dos entradas: **21**
- Triple NOR de tres entradas: **27**
- NAND de ocho entradas: **30**
- Cuádruple OR de dos entradas: **32**
- Cuádruple XOR: **86**
- Cuádruple XNOR: **266**

Encapsulados de circuitos integrados. Todos los CMOS de la serie 74 son compatibles en su patillaje con respecto a los mismos tipos de dispositivos en tecnología TTL. Esto quiere decir que un circuito integrado digital CMOS, como el 74AHC00 (cuádruple NAND de 2 entradas), que contiene cuatro puertas NAND de 2 entradas en un único circuito integrado, tiene exactamente los mismos números de pin para cada entrada y salida que el correspondiente dispositivo TTL. En la Figura 3.60 se muestra los encapsulados típicos de circuitos integrados, el encapsulado DIP (*Dual In–line Package*) para montaje de inserción y el encapsulado SOIC (*Small–Outline Integrated Circuit*) para montaje superficial. En algunos casos, hay disponibles algunos otros tipos de encapsulados. Aunque no se muestra a escala, el encapsulado SOIC es significativamente más pequeño que el encapsulado DIP. Los diagramas de configuración de los pines para la mayor parte de los dispositivos lógicos enumerados anteriormente se muestran en la Figura 3.61.



(a) Encapsulado DIP de 14 pines para montaje de inserción



(b) Encapsulado SOIC de 14 pines para montaje superficial

FIGURA 3.60 Encapsulados típicos DIP y SOIC con sus dimensiones básicas y la numeración de los pines.

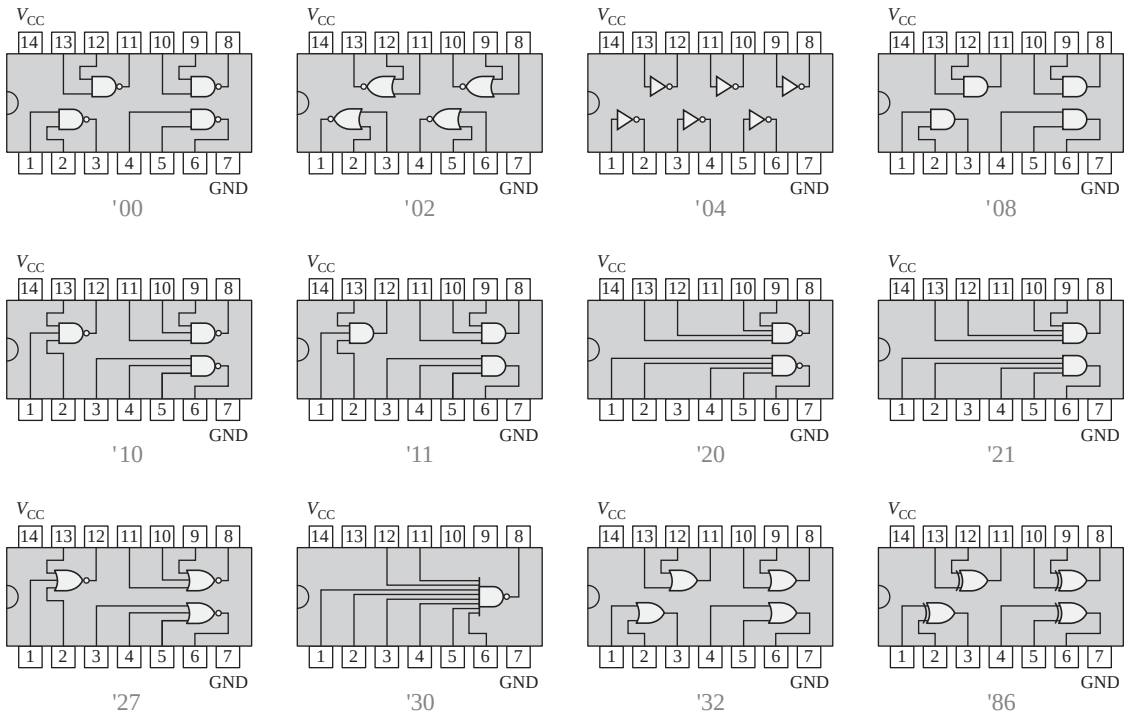


FIGURA 3.61 Diagramas de configuración de los pines para algunas de las configuraciones de puertas integradas de función fija más comunes.

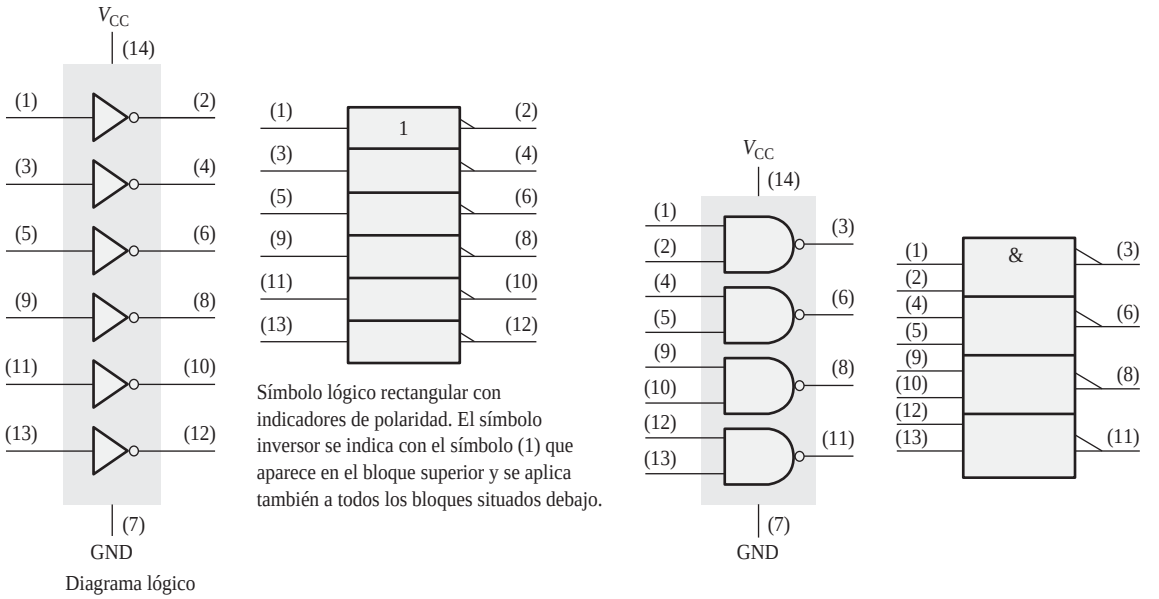
Lógica monopuerta. Hay disponible una selección limitada de puertas CMOS en encapsulados monopuerta. Con una única puerta por chip, esta serie se vende en pequeños encapsulados de cinco pines que sirven para llevar a cabo las modificaciones de último momento para incluir determinados circuitos lógicos en zonas de alta densidad, cuando el espacio disponible es limitado.

Símbolos lógicos. Los símbolos lógicos para los circuitos integrados de función fija utilizan los símbolos de puerta estándares y presentan el número de puertas en el encapsulado del circuito integrado y los números de pines asociados a cada puerta, así como los números de pines para V_{CC} y masa. En la Figura 3.62 se muestra un ejemplo para el inversor séxtuple y para la puerta cuádruple NAND de 2 entradas. Se muestran el diagrama lógico y el símbolo rectangular. Independientemente de la familia lógica, todos los dispositivos con el mismo sufijo tienen pines compatibles; es decir, tienen la misma disposición de números de pines. Por ejemplo, los dispositivos 7400, 74S00, 74LS00, 74ALS00, 74F00, 74HC00 y 74AHC00 son todos ellos circuitos integrados cuádruples NAND de 2 entradas compatibles en cuanto a sus pines.

Características y parámetros de funcionamiento

▲ *La lógica de alta velocidad presenta un tiempo de retardo de propagación corto.*

Existen varios puntos que definen el funcionamiento de un circuito lógico. Las características de funcionamiento son: la velocidad de conmutación medida en términos del retardo de propagación, la disipación de potencia, el fan–out o capacidad de excitación, el producto velocidad–potencia, la tensión de alimentación continua y los niveles lógicos de entrada/salida.



(a) Inversor séxtuple

(b) Cuádruple NAND de dos entradas

FIGURA 3.62 Símbolos lógicos para el inversor séxtuple (sufijo 04) y la puerta NAND cuádruple de 2 entradas (sufijo 00). El símbolo se aplica al mismo dispositivo en cualquier serie CMOS o TTL.

Tiempo de retardo de propagación. Este parámetro limita la frecuencia o velocidad de conmutación a la que un circuito lógico puede operar. Cuando se aplican a los circuitos lógicos, los términos *baja velocidad* y *alta velocidad* hacen referencia al retardo de propagación. Cuanto menor sea el tiempo de propagación, mayor será la velocidad del circuito y mayor será la frecuencia a la que puede operar.

El **tiempo de retardo de propagación**, t_P, de una puerta lógica es el intervalo de tiempo entre la aplicación de un impulso de entrada y la aparición del impulso de salida resultante. Existen dos medidas diferentes del tiempo de retardo de propagación asociado con una puerta lógica, que se aplican a todos los tipos de puertas básicas:

- t_{PHL}: es el tiempo entre un punto de referencia especificado en el impulso de entrada y el correspondiente punto de referencia en el impulso de salida, cuando la salida cambia del nivel ALTO (H) al nivel BAJO (L).
- t_{PLH}: es el tiempo entre un punto de referencia especificado en el impulso de entrada y el correspondiente punto de referencia en el impulso de salida, cuando la salida cambia del nivel BAJO (L) al nivel ALTO (H).

EJEMPLO 3.22

Determinar los tiempos de propagación en el inversor de la Figura 3.63(a).

Solución Los tiempos de propagación, t_{PHL} y t_{PLH}, se indican en la parte (b) de la figura. En este caso, los retrasos se miden entre los puntos de pendiente del 50% en los correspondientes flancos de los impulsos de entrada y salida. Los valores de t_{PHL} y t_{PLH} no necesariamente son iguales pero, en la mayoría de los casos, sí lo son.

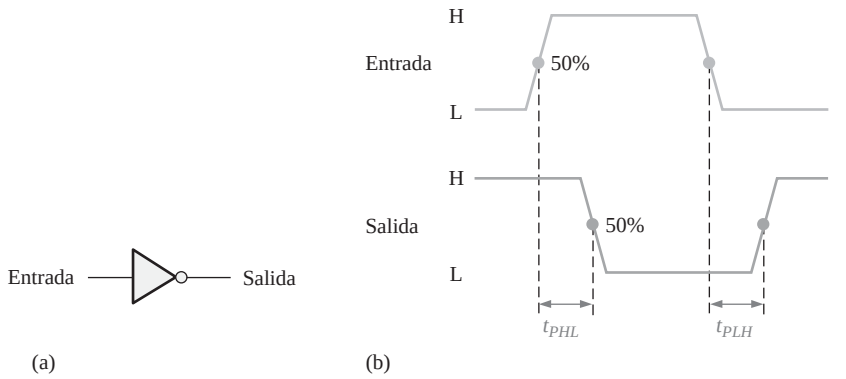


FIGURA 3.63

Problema relacionado Una puerta lógica tiene una valor máximo especificado de 10 ns para t_{PHL} y t_{PLH}. Para otros tipos de puertas este valor es de 4 ns. ¿Cuál es la puerta que puede trabajar a la frecuencia mayor?

En las puertas TTL de la serie estándar, el retardo de propagación típico es de 11 ns y para las puertas de la serie F es de 3,3 ns. Para las puertas CMOS de la serie HCT, el retardo de propagación es 7 ns, para la serie AC es de 5 ns y para la serie ALVC es de 3 ns. Todos los valores especificados dependen de determinadas condiciones de operación, tal y como se establece en las hojas de características.

Tensión de alimentación continua (V_{CC}). La tensión de alimentación continua típica para CMOS puede ser 5 V; 3,3 V, 2,5 V o 1,8 V, dependiendo de la categoría. Una ventaja de CMOS es que las tensiones de alimentación pueden variar en un rango más amplio que los dispositivos TTL. Los CMOS de 5 V pueden tolerar variaciones de alimentación desde los 2 V a 6 V y aún así funcionarán adecuadamente, aunque el retardo de propagación y la disipación de potencia se vean significativamente afectadas. Los dispositivos CMOS de 3,3 V pueden operar con tensiones de alimentación desde 2 V hasta 3,6 V. La tensión de alimentación continua típica para dispositivos TTL es 5,0 V con un mínimo de 4,5 V y un máximo de 5,5 V.

Disipación de potencia. La **disipación de potencia**, P_D, de una puerta lógica es el producto de la tensión de alimentación continua y de la corriente media de alimentación. Normalmente, la corriente de alimentación cuando la salida de la puerta está a nivel BAJO es mayor que cuando la salida de la puerta está a nivel ALTO. Generalmente, las hojas de características del fabricante especifican la corriente de alimentación para el estado de salida BAJO como I_{CCL} y para el estado ALTO como I_{CCH}. La corriente media de alimentación se determina en función de un ciclo de trabajo del 50% (nivel de salida BAJO la mitad del tiempo y la otra mitad nivel de salida ALTO), por tanto la disipación de potencia media de una puerta lógica es

Ecuación 3.2

$$P_D = V_{CC}\left(\frac{I_{CCH} + I_{CCL}}{2}\right)$$

Las puertas de la serie CMOS tienen disipaciones de potencia muy bajas en comparación con las series TTL. Sin embargo, la disipación de potencia en los dispositivos CMOS depende de la frecuencia de funcionamiento. Para una frecuencia cero, la potencia está normalmente en el rango de los microvatios por puerta, y en la frecuencia máxima de funcionamiento puede estar en el rango de los milivatios; por tanto, algunas veces la potencia se especifica para una frecuencia determinada. Por ejemplo, la serie HC tiene una potencia de 2,75 μW/puerta para una frecuencia igual a 0 Hz y de 600 μW/puerta para 1 MHz.

La disipación de potencia para los dispositivos TTL es independiente de la frecuencia. Por ejemplo, la serie ALS disipa 1,4 mW/puerta, independientemente de la frecuencia y la serie F disipa 6 mW/puerta.

Niveles lógicos de entrada y salida. V_{IL} es la tensión del nivel de entrada BAJO para una puerta lógica y V_{IH} es la tensión de entrada del nivel ALTO. Los dispositivos CMOS de 5 V aceptan una tensión máxima de 1,5 V para V_{IL} y una tensión mínima de 3,5 V para V_{IH}. Los dispositivos TTL aceptan una tensión máxima de 0,8 V para V_{IL} y una tensión mínima de 2 V para V_{IH}.

V_{OL} es la tensión de salida para el nivel BAJO y V_{OH} es la tensión de salida para el nivel ALTO. Para los dispositivos CMOS de 5 V, el valor máximo de V_{OL} es de 0,33 V y el valor mínimo para V_{OH} es de 4,4 V. Para los dispositivos TTL, el valor máximo V_{OL} es de 0,4 V y el mínimo V_{OH} es de 2,4 V. Todos los valores dependen de las condiciones de operación, tal y como se especifica en la hoja de características.

Producto velocidad-potencia (SPP). El parámetro **SPP** (*Speed–Power Product*) puede utilizarse como una medida del funcionamiento de un circuito lógico que tiene en cuenta el retardo de propagación y la disipación de potencia. Es especialmente útil para comparar las distintas series de puertas lógicas de las familias CMOS o TTL o para comparar una puerta CMOS con una puerta TTL.

El producto SPP de un circuito lógico es igual al producto del retardo de propagación por la disipación de potencia, y se expresa en julios (J), que es una unidad de energía. La fórmula es

Ecuación 3.3 $$SPP = t_p P_D$$

EJEMPLO 3.23

Una determinada puerta tiene un retardo de propagación de 5 ns, $I_{CCH} = 1$ mA e $I_{CCL} = 2{,}5$ mA, con una tensión de alimentación continua de 5 V. Determinar el producto velocidad–potencia.

Solución

$$P_D = V_{CC}\left(\frac{I_{CCH} + I_{CCL}}{2}\right) = 5\,\text{V}\left(\frac{1\,\text{mA} + 2{,}5\,\text{mA}}{2}\right) = 5\,\text{V}(1{,}75\,\text{mA}) = 8{,}75\,\text{W}$$

$$SPP = (5\,\text{ns})(8{,}75\,\text{mW}) = \mathbf{43{,}75\ pJ}$$

Problema relacionado Si el retardo de propagación de una puerta es 15 ns y su *SPP* es igual a 150 pJ, ¿cuál es su disipación de potencia media?

Fan-out y carga. El **fan–out** de una puerta lógica es el número máximo de entradas de la familia de circuitos integrados de la misma serie que la puerta puede excitar, manteniendo a la vez los niveles de salida dentro de los límites especificados. El fan–out es un parámetro importante sólo en la tecnología TTL. Dado que con los circuitos CMOS se asocian impedancias muy altas, el fan–out es muy alto, aunque depende de la frecuencia debido a los efectos capacitivos.

El fan–out se especifica en términos de ***cargas unidad***. Una carga unidad para una puerta lógica es igual a una entrada de un circuito similar. Por ejemplo, una carga unidad para una puerta NAND 74LS00 es igual a *una* entrada a una puerta lógica en la serie 74LS (no necesariamente una puerta NAND). Puesto que la corriente para una entrada a nivel BAJO (I_{IL}) de una puerta 74LS00 es de 0,4 mA y la corriente que una salida a nivel BAJO (I_{IL}) puede aceptar es de 8,0 mA, el número de cargas unidad que una puerta 74LS00 puede excitar en el estado BAJO es

$$\text{Cargas unidad} = \frac{I_{OL}}{I_{IL}} = \frac{8{,}0\,\text{mA}}{0{,}4\,\text{mA}} = 20$$

La Figura 3.64 muestra puertas lógicas LS que excitan una serie de puertas con la misma tecnología de circuitos, donde el número de puertas depende de la tecnología de circuitos particular. Por ejemplo, como hemos visto, el número máximo de entradas de puerta (cargas unidad) que una puerta TTL de la serie 74LS puede excitar es 20.

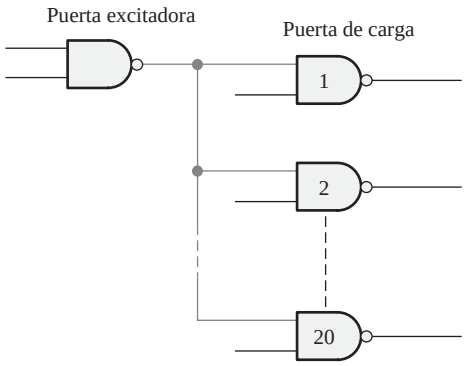


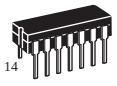
FIGURA 3.64 La salida de la puerta NAND LS TTL admite como carga máxima la entrada de 20 puertas LS TTL.

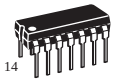
QUAD 2-INPUT NAND GATE

• ESD > 3500 Volts

SN54/74LS00

QUAD 2-INPUT NAND GATE
LOW POWER SCHOTTKY

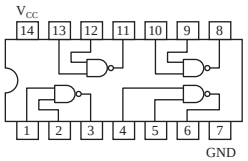

J SUFFIX
CERAMIC
CASE 632-08


N SUFFIX
PLASTIC
CASE 646-06


D SUFFIX
SOIC
CASE 751A-02

ORDERING INFORMATION

SN54LSXXJ	Ceramic
SN74LSXXN	Plastic
SN74LSXXD	SOIC



SN54/74LS00

DC CHARACTERISTICS OVER OPERATING TEMPERATURE RANGE (unless otherwise specified)

Symbol	Parameter		Limits Min	Typ	Max	Unit	Test Conditions
V_{IH}	Input HIGH Voltage		2.0			V	Guaranteed Input HIGH Voltage for All Inputs
V_{IL}	Input LOW Voltage	54			0.7	V	Guaranteed Input LOW Voltage for All Inputs
		74			0.8		
V_{IK}	Input Clamp Diode Voltage			–0.65	–1.5	V	V_{CC} = MIN, I_{IN} = –18 mA
V_{OH}	Ouput HIGH Voltage	54	2.5	3.5		V	V_{CC} = MIN, I_{OH} = MAX, V_{IN} = V_{IH} or V_{IL} per Truth Table
		74	2.7	3.5		V	
V_{OL}	Ouput LOW Voltage	54, 74		0.25	0.4	V	I_{OL} = 4.0 mA; V_{CC} = V_{CC} MIN, V_{IN} = V_{IL} or V_{IH} per Truth Table
		74		0.35	0.5	V	I_{OL} = 8.0 mA
I_{IH}	Input HIGH Current				20	μA	V_{CC} = MAX, V_{IN} = 2.7 V
					0.1	mA	V_{CC} = MAX, V_{IN} = 7.0 V
I_{IL}	Input LOW Current				–0.4	mA	V_{CC} = MAX, I_N = 0.4 V
I_{OS}	Short Circuit Current (Note 1)		–20		–100	mA	V_{CC} = MAX
I_{CC}	Power Supply Current Total, Output HIGH				1.6	mA	V_{CC} = MAX
	Total, Output LOW				4.4		

NOTE 1: Not more than one output should be shorted at a time, nor for more than 1 second.

AC CHARACTERISTICS (T_A = 25 C)

Symbol	Parameter	Limits Min	Typ	Max	Unit	Test Conditions
t_{PLH}	Turn-Off Delay, Input to Output		9.0	15	ns	V_{CC} = 5.0 V, C_L = 15 pF
t_{PHL}	Turn-On Delay, Input to Output		10	15	ns	

GUARANTEED OPERATING RANGES

Symbol	Parameter		Min	Typ	Max	Unit
V_{CC}	Supply Voltage	54	4.5	5.0	5.5	V
		74	4.75	5.0	5.25	
T_A	Operating Ambient Temperature Range	54	–55	25	125	C
		74	0	25	70	
I_{OH}	Output Current — High	54, 74			–0.4	mA
I_{OL}	Output Current — Low	54			4.0	mA
		74			8.0	

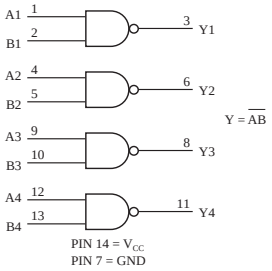
FIGURA 3.65 Parte de una hoja de características de un 74LS00.

Quad 2-Input NAND Gate **High-Performance Silicon–Gate CMOS**

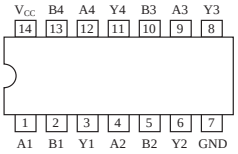
The MC54/74HC00A is identical in pinout to the LS00. The device inputs are compatible with Standard CMOS outputs; with pullup resistors, they are compatible with LSTTL outputs.

- Output Drive Capability: 10 LSTTLLoads
- Outputs Directly Interface to CMOS, NMOS andTTL
- OperatingVoltage Range: 2 to 6 V
- Low Input Current: 1μA
- High Noise Immunity Characteristic of CMOS Devices
- In ComplianceWith the JEDEC Standard No. 7A Requirements
- Chip Complexity: 32 FETs or 8 Equivalent Gates

LOGIC DIAGRAM

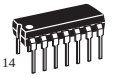


Pinout: 14–Load Packages (Top View)



MC54/74HC00A

J SUFFIX
CERAMIC PACKAGE
CASE 632-08



N SUFFIX
PLASTIC PACKAGE
CASE 646-06

D SUFFIX
SOIC PACKAGE
CASE 751A-03

DT SUFFIX
TSSOP PACKAGE
CASE 948G-01

ORDERING INFORMATION

MC54HCXXAJ	Ceramic
MC74HCXXAN	Plastic
MC74HCXXAD	SOIC
MC74HCXXADT	TSSOP

FUNCTION TABLE

Inputs		Output
A	B	Y
L	L	H
L	H	H
H	L	H
H	H	L

MAXIMUM RATINGS*

Symbol	Parameter	Value	Unit
V_{CC}	DC Supply Voltage (Referenced to GND)	- 0.5 to + 7.0	V
V_{in}	DC Input Voltage (Referenced to GND)	−0.5 to V_{CC} + 0.5	V
V_{out}	DC Output Voltage (Referenced to GND)	−0.5 to V_{CC} + 0.5	V
I_{in}	DC Input Current, per Pin	± 20	mA
I_{out}	DC Output Current, per Pin	± 25	mA
I_{CC}	DC Supply Current, V_{CC} and GND Pins	± 50	mA
P_D	Power Dissipation in Still Air, Plastic or Ceramic DIP	750	mW
	SOIC Package	500	
	TSSOP Package	450	
T_{stg}	Storage Temperature	− 65 to + 150	°C
T_L	Lead Temperature, 1 mm from Case for 10 Seconds		°C
	Plastic DIP, SOIC or TSSOP Package	260	
	Ceramic DIP	300	

* Maximum Ratings are those values beyond which damage to the device may occur. Functional operation should be restricted to the Recommended Operating Conditions.
Derating — Plastic DIP: – 10 mW/°C from 65° to 125° C
Ceramic DIP: – 10 mW/°C from 100° to 125° C
SOIC Package: – 7 mW/°C from 65° to 125° C
TSSOP Package: – 6.1 mW/°C from 65° to 125° C

RECOMMENDED OPERATING CONDITIONS

Symbol	Parameter		Min	Max	Unit
V_{CC}	DC Supply Voltage (Referenced to GND)		2.0	6.0	V
V_{in}, V_{out}	DC Input Voltage, Output Voltage (Referenced to GND)		0	V_{CC}	V
T_A	Operating Temperature, All Package Types		− 55	+125	°C
t_r, t_f	Input Rise and Fall Time	V_{CC} = 2.0 V	0	1000	ns
		V_{CC} = 4.5 V	0	500	
		V_{CC} = 6.0 V	0	400	

DC CHARACTERISTICS (Voltages Referenced to GND) MC54/74HC00A

Symbol	Parameter	Condition	V_{CC} V	Guaranteed Limit			Unit
				–55 to 25°C	≤85°C	≤125°C	
V_{IH}	Minimum High-Level Input Voltage	V_{out} = 0.1V or V_{CC} − 0.1V $\|I_{out}\| \le 20\mu A$	2.0	1.50	1.50	1.50	V
			3.0	2.10	2.10	2.10	
			4.5	3.15	3.15	3.15	
			6.0	4.20	4.20	4.20	
V_{IL}	Maximum Low-Level Input Voltage	V_{out} = 0.1V or V_{CC} − 0.1V $\|I_{out}\| \le 20\mu A$	2.0	0.50	0.50	0.50	V
			3.0	0.90	0.90	0.90	
			4.5	1.35	1.35	1.35	
			6.0	1.80	1.80	1.80	
V_{OH}	Minimum High-Level Output Voltage	V_{in} = V_{IH} or V_{IL} $\|I_{out}\| \le 20\mu A$	2.0	1.9	1.9	1.9	V
			4.5	4.4	4.4	4.4	
			6.0	5.9	5.9	5.9	
		V_{in} = V_{IH} or V_{IL} $\|I_{out}\| \le 2.4$mA	3.0	2.48	2.34	2.20	
		$\|I_{out}\| \le 4.0$mA	4.5	3.98	3.84	3.70	
		$\|I_{out}\| \le 5.2$mA	6.0	5.48	5.34	5.20	
V_{OL}	Maximum Low-Level Output Voltage	V_{in} = V_{IH} or V_{IL} $\|I_{out}\| \le 20\mu A$	2.0	0.1	0.1	0.1	V
			4.5	0.1	0.1	0.1	
			6.0	0.1	0.1	0.1	
		V_{in} = V_{IH} or V_{IL} $\|I_{out}\| \le 2.4$mA	3.0	0.26	0.33	0.40	
		$\|I_{out}\| \le 4.0$mA	4.5	0.26	0.33	0.40	
		$\|I_{out}\| \le 5.2$mA	6.0	0.26	0.33	0.40	
I_{in}	Maximum Input Leakage Current	V_{in} = V_{CC} or GND	6.0	±0.1	±1.0	±1.0	μA
I_{CC}	Maximum Quiescent Supply Current (per Package)	V_{in} = V_{CC} or GND I_{out} = 0μA	6.0	1.0	10	40	μA

AC CHARACTERISTICS (C_L = 50 pF, Input t_r = t_f = 6 ns)

Symbol	Parameter	V_{CC} V	Guaranteed Limit			Unit
			–55 to 25°C	≤85°C	≤125°C	
t_{PLH}, t_{PHL}	Maximum Propagation Delay, Input A or B to Output Y	2.0	75	95	110	ns
		3.0	30	40	55	
		4.5	15	19	22	
		6.0	13	16	19	
t_{TLH}, t_{THL}	Maximum Output Transition Time, Any Output	2.0	75	95	110	ns
		3.0	27	32	36	
		4.5	15	19	22	
		6.0	13	16	19	
C_{in}	Maximum Input Capacitance		10	10	10	pF

		Typical @ 25°C, V_{CC} = 5.0 V, V_{EE} = 0 V	
C_{PD}	Power Dissipation Capacitance (Per Buffer)	22	pF

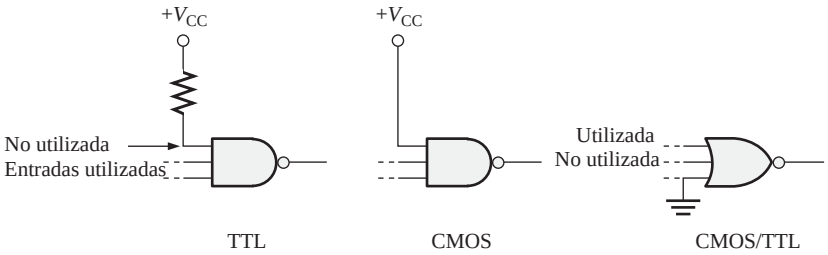
FIGURA 3.66 Parte de una hoja de características de un 74HC00A.

Hojas de características

Una hoja de características típica consta de una página de información que muestra, entre otras cosas, el diagrama lógico y los encapsulados, las condiciones de operación recomendadas, las características eléctricas y las características de conmutación. En las Figuras 3.65 y 3.66 se muestran, respectivamente, parte de las hojas de características para un 74LS00 y un 74HC00A. La longitud de las hojas de características varía y unas aportan mucha más información que otras.

CONSEJOS PRÁCTICOS

Las entradas de puerta no utilizadas para TTL y CMOS deberían conectarse al nivel lógico apropiado (ALTO o BAJO). Para las puertas AND y NAND, es recomendable que las entradas no utilizadas se conecten a V_{CC} (a través de una resistencia de 1,0 kΩ para TTL), y para las puertas OR y NOR, las puertas no utilizadas deberían conectarse a tierra.



REVISIÓN DE LA SECCIÓN 3.8

1. Enumere los dos tipos de tecnologías de circuitos integrados más ampliamente utilizadas.
2. Identifique las siguientes designaciones lógicas de circuitos integrados:

(a) LS **(b)** ALS **(c)** F **(d)** HC **(e)** AC **(f)** HCT **(g)** LV

3. Identifique los siguientes dispositivos de acuerdo a su función lógica:

(a) 74LS04 **(b)** 74HC00 **(c)** 74LV08 **(d)** 74ALS10

(e) 7432 **(f)** 74ACT11 **(g)** 74AHC02

4. Generalmente, ¿qué tecnología de circuitos integrados tienen la disipación de potencia más baja?
5. ¿Qué quiere decir el término *inversor séxtuple*? ¿Y el término *cuádruple NAND de 2 entradas*?
6. Se aplica un pulso positivo a una entrada inversora. El tiempo desde el flanco de subida de la entrada hasta el flanco de subida de la salida es 10 ns. El tiempo desde el flanco de bajada de la entrada hasta el flanco de bajada de la salida es de 8 ns. ¿Cuáles son los valores de t_{PLH} y t_{PHL}?
7. Una determinada puerta tiene un retardo de propagación de 6 ns y una disipación de potencia de 3 mW. Determinar el producto velocidad–potencia.
8. Defina I_{CCL} e I_{CCH}.
9. Defina V_{IL} y V_{IH}.
10. Defina V_{OL} y V_{OH}.

3.9 LOCALIZACIÓN DE AVERÍAS

La localización de averías es el proceso de reconocer, aislar y corregir un fallo en un sistema o circuito. Para poder localizar las averías de forma efectiva, debe entender cómo se supone que trabaja el circuito o sistema y debe estar en disposición de reconocer un funcionamiento incorrecto. Por ejemplo, para determinar si una puerta lógica tiene un fallo, debe saber cuál debe ser la salida para unas entradas dadas.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Comprobar la existencia de entradas y salidas abiertas internamente en las puertas lógicas de los CI. ■ Reconocer los efectos de una entrada o una salida del CI cortocircuitada. ■ Detectar en una tarjeta de circuito impreso la existencia de fallos externos. ■ Localizar las averías en un sencillo contador de frecuencia utilizando un osciloscopio.

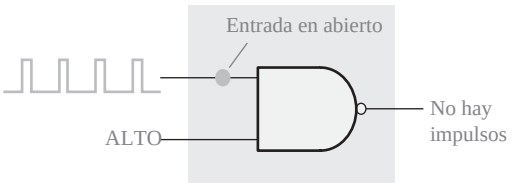
Fallos internos en las puertas lógicas de los CI

Los circuitos abiertos y los cortocircuitos son los fallos más comunes en las puertas internas del CI. Se pueden producir tanto en las entradas como en la salida de una puerta contenida en el encapsulado del CI. *Antes de intentar solucionar cualquier avería, compruebe que la alimentación continua y la masa son correctas.*

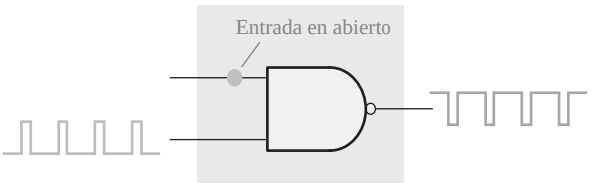
Efectos de una entrada que se encuentra en circuito abierto internamente. Un circuito abierto interno es el resultado de un componente en circuito abierto o de una ruptura en la conexión entre el chip y el pin del encapsulado. Una entrada en circuito abierto impide que una señal de impulsos en esta entrada dé lugar a una salida, como se muestra en la Figura 3.67(a) para la puerta NAND de 2 entradas. Una entrada TTL en abierto actúa como un nivel ALTO, por lo que los impulsos aplicados a la entrada que está en buen estado pasan a través de la puerta NAND hasta la salida, como se muestra en la Figura 3.67(b).

Condiciones para probar las puertas. Al probar una puerta NAND o una puerta AND, debe asegurarse siempre de que las entradas a las que no se aplican impulsos se encuentren a nivel ALTO, para activar la puerta. Cuando pruebe una puerta NOR o una puerta OR, debe asegurarse siempre de que las entradas a las que no se aplican impulsos se encuentran a nivel BAJO. Cuando se prueba una puerta XOR o XNOR, el nivel de la entrada a la que no se aplican impulsos no importa, ya que los impulsos aplicados en la otra entrada forzarán a que las entradas se encuentren, alternativamente, en el mismo nivel o en niveles opuestos.

Localización de fallo: entrada en circuito abierto. La localización de este tipo de fallo es, en la mayoría de los casos, muy fácil utilizando un osciloscopio y un generador de funciones, como se muestra en la Figura 3.68, para el caso de una puerta NAND de 2 entradas. Al medir las señales digitales con un osciloscopio, emplee siempre el acoplamiento en continua.

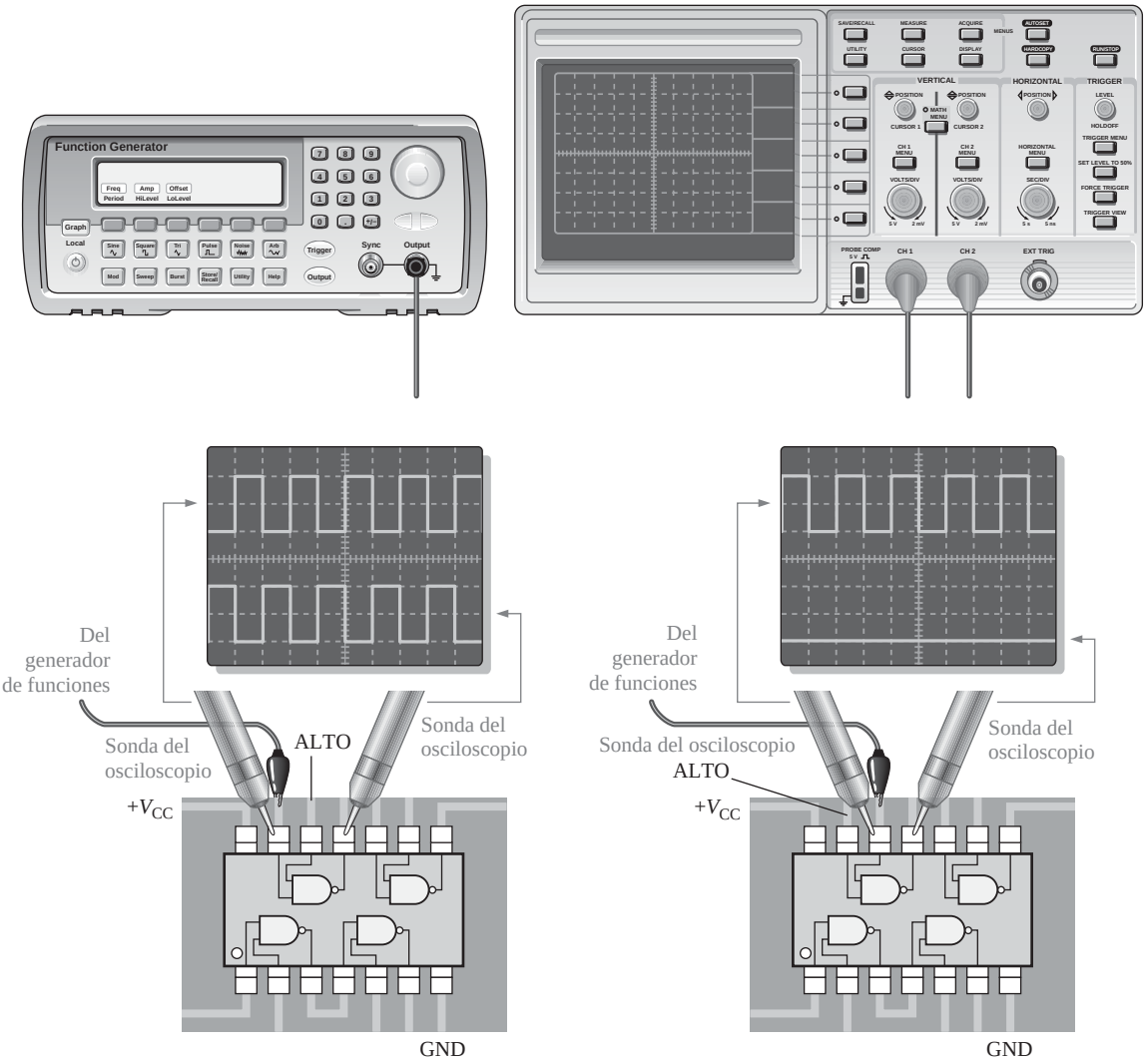


(a) Si se aplican impulsos en una entrada en abierto no se generan impulsos a la salida.



(b) Si se aplican impulsos en la entrada que está bien, se generarán impulsos a la salida en las puertas TTL NAND y AND, debido a que la entrada en abierto actúa como un nivel ALTO. Esto no se cumple para CMOS.

FIGURA 3.67 Efecto de una entrada en circuito abierto en una puerta NAND.



(a) El pin 13 de entrada y el pin 11 de salida están bien.

(b) El pin 12 de entrada está en abierto.

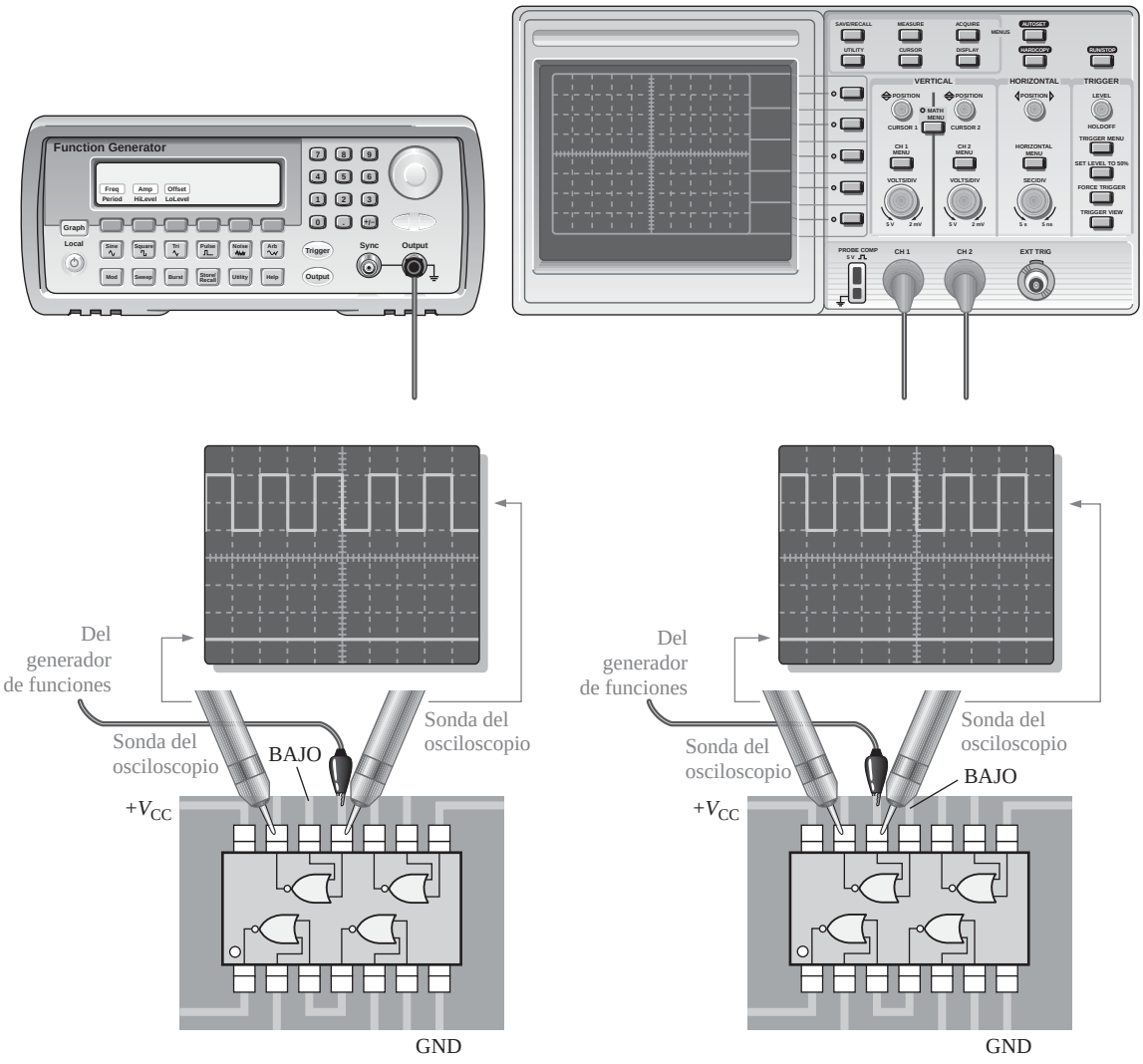
FIGURA 3.68 Localización de averías en una puerta NAND con una entrada en circuito abierto.

El primer paso en la localización de averías de un CI, cuando se sospecha que está fallando, es asegurarse de que la tensión de alimentación continua (V_{CC}) y la masa están conectadas a los pines apropiados del CI. Después, se aplican impulsos continuos a una de las entradas de la puerta, asegurándose de que las otras entradas están a nivel ALTO (en el caso de una puerta NAND). En la Figura 3.68(a), se comienza a aplicar los impulsos en el pin 13, ya que se ha determinado que es una de las entradas de la puerta de la que se sospecha el fallo. Si en la salida correspondiente (en este caso el pin 11) se detecta un tren de impulsos, entonces el pin 13 de entrada no está en abierto. Consecuentemente, esto prueba también que la salida no está en abierto. A continuación, se aplica otro tren de impulsos a otra entrada de la puerta (pin 12), asegurándose de que la otra entrada está a nivel ALTO. En la salida (en el pin 11) no se detecta un tren de impulsos y la salida está a nivel BAJO, lo que indica que la entrada del pin 12 está en abierto, como se muestra en la Figura 3.68(b). Observe que la entrada en la que no se aplican impulsos debe estar a nivel ALTO en el caso de una puerta NAND o

una puerta AND. Si se tratara de una puerta NOR, la entrada en la que no se aplican impulsos debería estar a nivel BAJO.

Efectos de una salida que se encuentra internamente en circuito abierto. Una salida que esté internamente en circuito abierto impide que una señal en cualquiera de las entradas llegue hasta la salida. Por tanto, independientemente de las condiciones de entrada, la salida no se ve afectada. El nivel en el pin de salida del CI dependerá de a qué esté conectada la salida externamente, por lo que podría estar a nivel ALTO, BAJO o flotante (no fijado a ninguna referencia). En cualquier caso, no habrá señal en el pin de salida.

Localización de fallo: salida en abierto. La Figura 3.69 ilustra la localización de una salida en abierto en una puerta NOR. En la parte (a) de la figura, se aplican impulsos a una de las entradas de las que se sospecha (en



(a) Se aplica un impulso de entrada en el pin 11. No hay impulsos en la salida

(b) Se aplica un impulso de entrada en el pin 12. No hay impulsos en la salida

FIGURA 3.69 Localización de una salida en circuito abierto en una puerta NOR.

este caso, el pin 11), y la salida (pin 13) no presenta ningún impulso. En la parte (b) de la figura, se aplican impulsos a la otra entrada (pin 12) y, de nuevo, no hay indicación de impulsos en la salida. Bajo la condición de que en la entrada en la que no se aplican impulsos esté a nivel BAJO, esta prueba demuestra que la salida está internamente en circuito abierto.

Entrada o salida cortocircuitadas. Aunque no es un fallo común como un abierto, se puede producir un cortocircuito interno a la tensión de alimentación, a masa, a otra entrada o a una salida. Cuando una entrada o una salida se cortocircuita a la alimentación, permanecerá en estado ALTO. Si una entrada o una salida se cortocircuita a masa, permanecerá a nivel BAJO (0 V). Si dos entradas o una entrada y una salida se cortocircuitan entre sí, entonces estarán siempre al mismo nivel.

Abiertos y cortos externos

Muchos de los fallos que afectan a los CI digitales se deben a fallos externos a los mismos. Se incluyen en este tipo de fallos las soldaduras incorrectas, salpicaduras de estaño, cortes de conductores, tarjetas de circuito

EJEMPLO 3.24

Se quiere probar un 74LS10, una triple puerta NAND de 3 entradas, que es uno de los muchos CI que contiene una tarjeta de circuito impreso. Ha probado los pines 1 y 2, y ambos están a nivel ALTO. Después, aplica un tren de impulsos al pin 13, y coloca la sonda del osciloscopio, en primer lugar, en el pin 12 y luego en la pista del circuito impreso, como se indica en la Figura 3.70. Basándose en la observación de la pantalla del osciloscopio, ¿cuál es, probablemente, el principal problema?

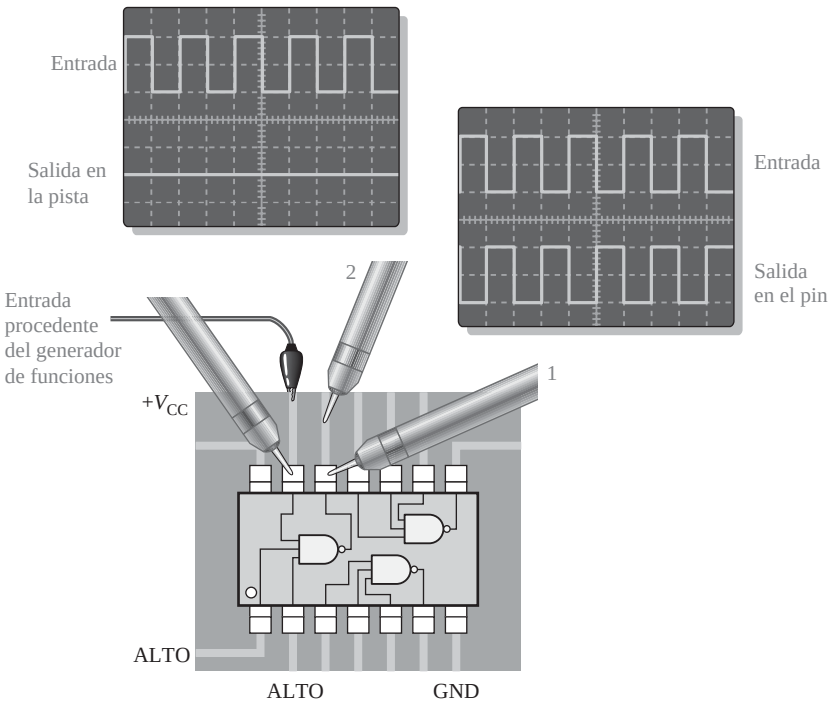


FIGURA 3.70

Solución La forma de onda cuando se coloca la sonda en la posición 1 muestra que hay actividad de impulsos en la salida de la puerta en el pin 12 pero, sin embargo, no detecta actividad en la pista de la tarjeta de circuito impreso, como indica la sonda en la posición 2. La puerta está trabajando correctamente, pero la señal no pasa del pin 12 del CI a la pista del circuito impreso.
Lo más probable es que haya una mala soldadura entre el pin 12 del CI y la tarjeta de circuito impreso, lo que da lugar a un circuito abierto. Por tanto, debería volver a soldar este punto y probar de nuevo.

Problema relacionado Si no se detectan impulsos en ningún punto de la Figura 3.70, ¿qué fallo(s) indica esto?

impreso grabadas incorrectamente, y rupturas o interrupciones en las conexiones o en las interconexiones del circuito impreso. Estas condiciones de circuitos abiertos o cortocircuitos tienen los mismos efectos sobre las puertas lógicas que los fallos internos y, básicamente, se localizan mediante los mismos métodos. Cuando se sospecha que algo está fallando, lo primero que debe hacer un técnico es una inspección visual del circuito.

Casi siempre deberá localizar fallos en circuitos integrados que están montados en tarjetas de circuito impreso o en prototipos ensamblados, y están interconectados con otros circuitos integrados. Según vaya avanzando a través de este libro, aprenderá cómo se usan diferentes tipos de CI digitales conjuntamente para realizar funciones de sistemas. Sin embargo, en este momento, vamos a concentrarnos en las puertas individuales de los circuitos digitales. Esta limitación no nos impide abordar el concepto de sistema en un nivel básico y simplificado.

Para continuar con el concepto de sistemas, los Ejemplos 3.25 y 3.26 se ocupan de la localización de averías en el contador de frecuencia introducido en la Sección 3.2.

EJEMPLO 3.25

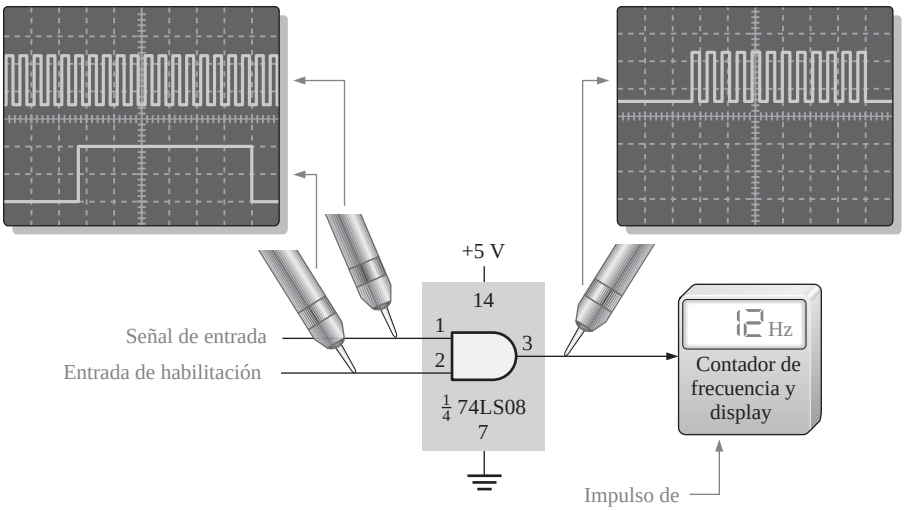
Después de intentar trabajar con el contador de frecuencia de la Figura 3.71, se encuentra con que constantemente da una lectura de todo 0s en el display, independientemente de la frecuencia de entrada. Determinar la causa de este mal funcionamiento. El impulso de habilitación tiene una anchura de 1 s.

La Figura 3.71(a) proporciona un ejemplo de cómo debería funcionar el contador de frecuencia con un tren de impulsos a 12 Hz aplicado en la entrada de la puerta AND. La parte (b) de la figura muestra que el display indica, incorrectamente, 0 Hz.

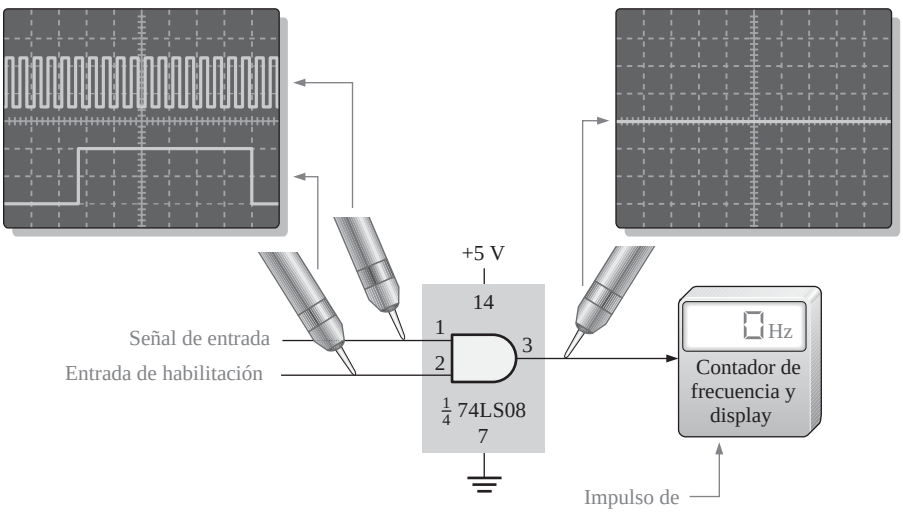
Solución Existen tres posibles causas:

1. Un nivel activo o verdadero constante en la entrada reset del contador hace que el contador esté siempre a cero.
2. No hay tren de impulsos en la entrada del contador, debido a que existe un circuito abierto o un cortocircuito en el contador. Este problema impediría que el contador avanzará después de haber sido puesto a cero.
3. No hay tren de impulsos a la entrada del contador porque hay un circuito abierto en la salida de la puerta AND o por la ausencia de señales de entrada, lo que de nuevo impide que el contador avance a partir de cero.

El primer paso consiste en asegurarse de que V_{CC} y masa están correctamente conectadas en todos los puntos; supongamos que están bien. A continuación, probamos aplicando impulsos en ambas entradas de la puerta AND. El osciloscopio indica que hay impulsos en ambas entradas. Se com-



(a) El contador está funcionando correctamente.



(b) El contador no está midiendo una frecuencia.

FIGURA 3.71

prueba la entrada reset del contador y muestra un nivel BAJO, que se sabe que corresponde a un nivel no verdadero y, por tanto, aquí no está el problema. La siguiente prueba se hace en el pin 3 del 74LS08, y muestra que no hay impulsos en la salida de la puerta AND, lo que indica que la salida de la puerta está internamente en circuito abierto. Debe reemplazar el CI 74LS08 y probar de nuevo el funcionamiento.

Problema relacionado Si el pin 2 de la puerta AND 74LS08 estuviese en circuito abierto, ¿qué indicación vería en el display?

EJEMPLO 3.26

El contador de frecuencia mostrado en la Figura 3.72 parece medir incorrectamente la frecuencia de las señales de entrada. Se determina que, cuando se aplica al pin 1 de la puerta AND una señal con una frecuencia conocida, la pantalla del osciloscopio indica una frecuencia más alta. Determinar dónde está el fallo. Las lecturas de la pantalla están expresadas en segundos/división.

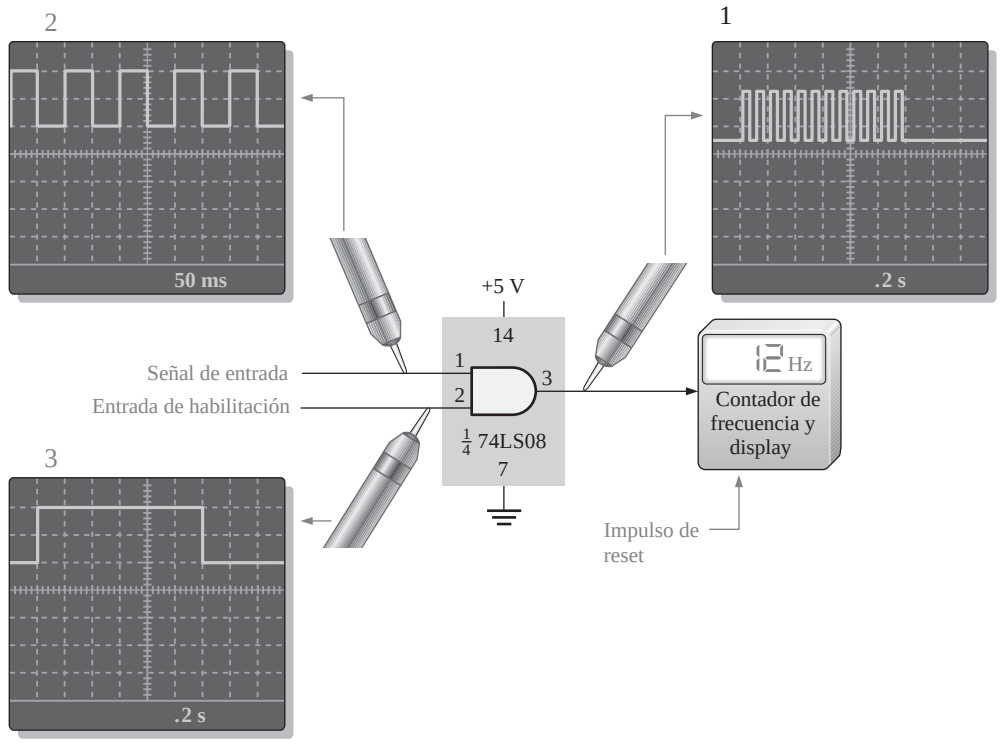


FIGURA 3.72

Solución

Recuerde de la Sección 3.2 que los impulsos de entrada pueden pasar a través de la puerta AND exactamente durante 1 s. El número de impulsos que se cuentan en 1 s es igual a la frecuencia en hertzios (ciclos por segundo). Por tanto, el intervalo de 1 s, que es generado por el impulso de habilitación aplicado en el pin 2 de la puerta AND, es crítico para obtener una medida de frecuencia precisa. Los impulsos de habilitación se generan internamente mediante un circuito oscilador de precisión. El ancho del impulso debe ser exactamente de 1 s y, en este caso, se produce cada 3 s para actualizar el contador. Justo después de cada impulso de habilitación, el contador se pone a cero, por lo que empieza a contar de nuevo.

Puesto que el contador cuenta más impulsos que los que debería, dando lugar a que se obtenga una lectura de frecuencia más alta, parece que el impulso de habilitación es el sospechoso principal. La medida exacta del intervalo de tiempo debe hacerse con el osciloscopio.

Se aplica al pin 1 de la puerta AND un tren de impulsos de entrada de exactamente 10 Hz e, incorrectamente, el display presenta 12 Hz. La primera medida en el osciloscopio, en la salida de la puerta AND, muestra que hay 12 impulsos por cada impulso de activación. La segunda medida del osciloscopio verifica que la frecuencia de entrada es exactamente 10 Hz (período = 100 ms). La tercera medida del osciloscopio determina el ancho del impulso de habilitación, que es 1,2 s, mayor por tanto que 1 s.
La conclusión es que el impulso de habilitación no está bien calibrado por alguna razón.

Problema relacionado ¿De qué sospecharía si la lectura indicara una frecuencia menor de lo que debe ser?

CONSEJOS PRÁCTICOS

La correcta puesta a tierra es muy importante cuando se prepara un circuito para tomar medidas o trabajar en él. La correcta puesta a tierra del osciloscopio protege al usuario frente a descargas eléctricas y la conexión a tierra del propio usuario protege los circuitos frente a posibles daños. Poner a tierra el osciloscopio quiere decir conectarlo a la toma de masa, introduciendo el conector de tres tomas en un enchufe con toma de masa. Como ya sabrá, ponerse a tierra el usuario significa que debe utilizarse una pulsera de conexión a masa, especialmente cuando se trabaja con circuitos CMOS.

Para obtener medidas precisas, también debe asegurarse de que la tierra del circuito que se está probando sea la misma que la del osciloscopio. Esto puede hacerse conectando el terminal de tierra de la sonda del osciloscopio a un punto de tierra conocido del circuito, como puede ser el chasis metálico o un punto de tierra en la tarjeta de circuito impreso. También se puede conectar la tierra del circuito al conector GND ubicado en el panel frontal del osciloscopio.

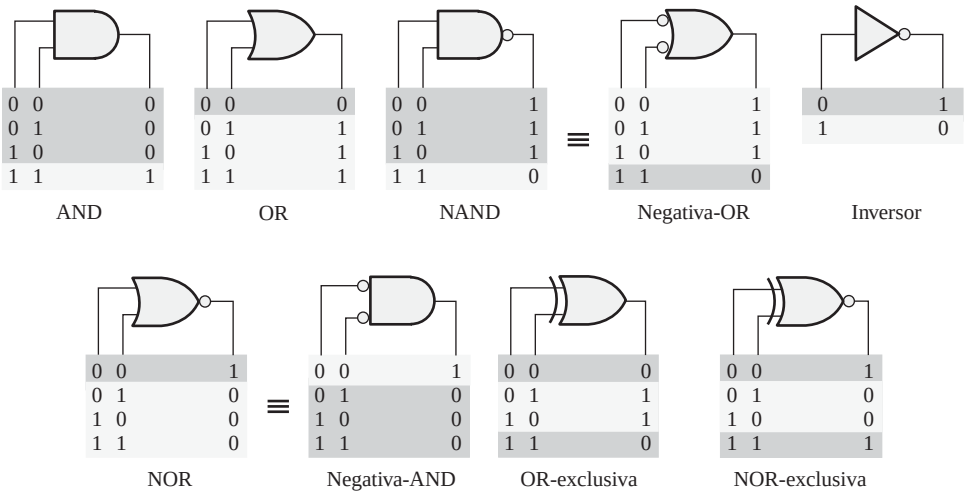
REVISIÓN DE LA SECCIÓN 3.9

1. ¿Cuáles son los tipos de fallos más comunes en los circuitos integrados?
2. Si se aplican dos señales de entrada diferentes a una puerta NAND TTL de 2 entradas y la señal de salida es como una de las entradas, pero invertida, ¿cuál es el problema más probable?
3. Cite dos características de los trenes de impulsos que puedan medirse con un osciloscopio.

RESUMEN

- La salida de un inversor es el complemento de la entrada.
- La salida de una puerta AND es un nivel ALTO sólo si todas las entradas están a nivel ALTO.
- La salida de una puerta OR es un nivel ALTO si cualquiera de las entradas está a nivel ALTO.
- La salida de una puerta NAND es un nivel BAJO sólo si todas las entradas están a nivel ALTO.
- La puerta NAND puede expresarse como una puerta negativa–OR, cuya salida es un nivel ALTO cuando cualquier entrada está a nivel BAJO.
- La salida de una puerta NOR es un nivel BAJO si cualquiera de las entradas está a nivel ALTO.
- La puerta NOR puede expresarse como una puerta negativa–AND, cuya salida es un nivel ALTO sólo si todas las entradas están a nivel BAJO.
- La salida de una puerta OR–exclusiva es un nivel ALTO cuando las entradas son distintas.
- La salida de una puerta NOR–exclusiva es un nivel BAJO cuando las entradas son distintas.

- Los símbolos distintivos y tablas de verdad para los distintos tipos de puertas lógicas (sólo para dos entradas) se muestran en la Figura 3.73.



Nota: Los estados activos se muestran en gris claro.

FIGURA 3.73

- La mayoría de los dispositivos lógicos programables (PLD) están basados en alguna forma de matriz AND.
- Las tecnologías basadas en conexiones programables son las tecnologías basadas en fusibles, antifusibles, EPROM, EEPROM y SRAM.
- Un PLD puede programarse utilizando una herramienta hardware denominada programador o montado en una tarjeta de circuito impreso de desarrollo.
- Los PLD tienen asociado un paquete de desarrollo software para la tarea de programación.
- Los dos métodos disponibles para introducir el diseño en un software de programación son la interfaz de texto (HDL) y la interfaz gráfica (esquemáticos)
- Los PLD de programación dentro del sistema (ISP) pueden programarse después de ser instalados en un sistema.
- JTAG (*Joint Test Action Group*) es una interfaz estándar (IEEE Std. 1149.1) utilizada para programar y probar los dispositivos PLD.
- La tecnología CMOS se basa en los transistores de efecto de campo MOS.
- La tecnología TTL se basa en transistores de unión bipolares.
- Por regla general, los dispositivos CMOS tienen el consumo de potencia más bajo que los TTL.
- La disipación media de potencia de una puerta lógica es:

$$P_D = V_{CC}\left(\frac{I_{CCH} + I_{CCL}}{2}\right)$$

- El producto velocidad–potencia de una puerta lógica es

$$SPP = t_p P_D$$

PALABRAS CLAVE

Las palabras clave y los términos que se han resaltado en negrita se encuentran en el glosario final del libro.

Álgebra booleana Las matemáticas de los circuitos lógicos.

Antifusible Tipo de conexión programable no volátil de un PLD que se puede dejar abierta o se puede cortocircuitar una sola vez según indique el programa.
Carga unidad Una medida del fan–out. Una entrada de puerta representa una carga unidad a la salida de la puerta, dentro de la misma familia de circuitos integrados.
CMOS *Complementary Metal–Oxide Semiconductor*, semiconductor complementario de metal–óxido; un tipo de circuito lógico integrado que se implementa con transistores de efecto de campo.
Complemento El inverso u opuesto de un número. Un nivel BAJO es el complemento de un nivel ALTO y 0 es el complemento de 1.
Diagrama de tiempos Diagrama de señales que muestra las relaciones temporales de todas las señales.
Dispositivo objetivo PLD montado en una herramienta de programación o tarjeta de desarrollo en el que se descarga un diseño lógico hardware.
EPROM Tipo de conexión programable no volátil de un PLD basado en celdas de memoria de sólo lectura eléctricamente programables y que se pueden activar y desactivar repetidamente por programación.
EEPROM Tipo de conexión programable no volátil de un PLD basado en celdas de memoria de sólo lectura eléctricamente programables y borrables, que se pueden activar y desactivar repetidamente por programación.
Fan–out Número de entradas de puertas equivalentes de la misma familia que puede excitar una puerta lógica.
Fusible Tipo de conexión programable no volátil de un PLD que se puede dejar cortocircuitada o se puede dejar en abierto una sola vez según indique el programa
Habilitar Activar o poner en modo operacional. Una entrada de un circuito lógico que activa su funcionamiento.
Inversor Circuito lógico que invierta o complementa su entrada.
Matriz AND Matriz de puertas AND que consta de una matriz de interconexiones programables.
Puerta AND Puerta lógica que produce una salida a nivel ALTO sólo cuando todas las entradas están a nivel ALTO.
Puerta NAND Puerta lógica que produce una salida a nivel BAJO sólo si todas las entradas están a nivel ALTO.
Puerta NOR Puerta lógica en la que la salida es un nivel BAJO cuando al menos una de las entradas está a nivel ALTO.
Puerta NOR–exclusiva (XNOR) Puerta lógica que produce una salida a nivel BAJO sólo cuando las dos entradas tienen niveles opuestos.
Puerta OR Puerta lógica que produce una salida a nivel ALTO cuando una o más entradas están a nivel ALTO.
Puerta OR–exclusiva (XOR) Puerta lógica que produce una salida a nivel ALTO sólo cuando las dos entradas tienen niveles opuestos.
SRAM Tipo de conexión programable volátil de PLD basada en celdas de memoria de acceso aleatorio y que se pueden activar u desactivar repetidamente mediante programación.
Tiempo de retardo de propagación Intervalo de tiempo que transcurre entre la transición de entrada y su correspondiente transición de salida en un circuito lógico.
Tabla de verdad Tabla que muestra las entradas y los correspondientes niveles de salida de un circuito lógico.
TTL *Transistor–Transistor Logic*, lógica transistor–transistor, un tipo de circuito integrado que utiliza transistores de unión bipolares.

AUTOTEST

Las respuestas se encuentran al final del capítulo

1. Cuando la entrada de un inversor es un nivel ALTO (1), la salida es:

(a) un nivel ALTO o 1 **(b)** un nivel BAJO o 1

(c) un nivel ALTO o 0 **(d)** un nivel BAJO o 0

2. Un inversor realiza la operación conocida como:

(a) complementación **(b)** afirmación

(c) inversión **(d)** las respuestas (a) y (c)

3. La salida de una puerta AND con entradas *A*, *B*, y *C* está a 1 (nivel ALTO) cuando

(a) A = 1, B = 1, C =1 **(b)** A = 1, B = 0, C =1 **(c)** A = 0, B = 0, C =0

4. La salida de una puerta OR con entradas *A*, *B*, y *C* está a 1 (nivel ALTO) cuando

(a) $A = 1, B = 1, C = 1$ **(b)** $A = 0, B = 0, C = 1$ **(c)** $A = 0, B = 0, C = 0$

(d) las respuestas (a), (b) y (c) **(e)** sólo las respuestas (a) y (b)

5. Se aplica un impulso a cada una de las entradas de una puerta NAND de dos entradas. Para $t = 0$ un impulso está a nivel ALTO y pasa a nivel BAJO en el instante $t = 1$ ms. El otro impulso está a nivel ALTO en $t = 0{,}8$ ms y pasa a nivel BAJO en $t = 3$ ms. El impulso de salida puede describirse como sigue:

(a) A nivel BAJO en $t = 0$, y pasará a nivel ALTO en $t = 3$ ms.

(b) A nivel BAJO en $t = 0{,}8$ ms, y pasará a nivel ALTO en $t = 3$ ms.

(c) A nivel BAJO en $t = 0{,}8$ ms, y pasará a nivel ALTO en $t = 1$ ms.

(d) A nivel ALTO en $t = 0{,}8$ ms, y pasará a nivel BAJO en $t = 1$ ms.

6. Se aplica un impulso a cada una de las entradas de una puerta NOR de dos entradas. Para $t = 0$ un impulso está a nivel ALTO y pasa a nivel BAJO en el instante $t = 1$ ms. El otro impulso está a nivel ALTO en $t = 0{,}8$ ms y pasa a nivel BAJO en $t = 3$ ms. El impulso de salida puede describirse como sigue:

(a) A nivel BAJO en $t = 0$, y pasará a nivel ALTO en $t = 3$ ms.

(b) A nivel BAJO en = 0,8 ms, y pasará a nivel ALTO en $t = 3$ ms.

(c) A nivel BAJO en $t = 0{,}8$ ms, y pasará a nivel ALTO en $t = 1$ ms.

(d) A nivel ALTO en $t = 0{,}8$ ms, y pasará a nivel BAJO en $t = 1$ ms.

7. Se aplica un impulso a cada una de las entradas de una puerta OR–exclusiva. Para $t = 0$ un impulso está a nivel ALTO y pasa a nivel BAJO en el instante $t = 1$ ms. El otro impulso está a nivel ALTO en $t = 0{,}8$ ms y pasa a nivel BAJO en $t = 3$ ms. El impulso de salida estará:

(a) A nivel BAJO en $t = 0$, y pasará a nivel BAJO en $t = 3$ ms.

(b) A nivel BAJO en $t = 0$ ms, y pasará a nivel BAJO en $t = 0{,}8$ ms.

(c) A nivel ALTO en $t = 1$ ms, y pasará a nivel ALTO en $t = 3$ ms.

(d) las respuestas (b) y (c)

8. Se aplica un impulso positivo a un inversor. El intervalo de tiempo entre el flanco anterior de la entrada y el flanco anterior de la salida es 7 ns. Este parámetro es

(a) el producto velocidad–potencia **(b)** el retardo de propagación t_{PHL}.

(c) el retardo de propagación t_{PLH}. **(d)** el ancho del impulso

9. El propósito de una conexión programable en una matriz AND es

(a) conectar una variable de entrada a una entrada de puerta

(b) conectar una fila a una columna de la matriz de interconexiones

(c) desconectar una fila de una columna de la matriz de interconexiones

(d) hacer todo lo anterior

10. El término OTP significa

(a) *Open Test Point* **(b)** *One–Time Programmable*

(c) *Output Test Program* **(d)** *Output Terminal Positive*

11. Los tipos de tecnologías de proceso basadas en conexiones programables son:

(a) antifusible **(b)** EEPROM **(c)** ROM

(d) las respuestas (a) y (b) **(e)** las respuestas (a) y (c)

12. Una tecnología de proceso volátil basada en conexiones programables es:

(a) fusible **(b)** EPROM

(c) SRAM **(d)** EEPROM

13. Dos métodos de introducir un diseño lógico utilizando un software de desarrollo de PLD son:

(a) interfaz de texto e interfaz numérica **(b)** interfaz de texto e interfaz gráfica

(c) interfaz gráfica y codificación **(d)** compilación y ordenación

14. JTAG es el acrónimo de

(a) *Joint Test Action Group* **(b)** *Java Top Array Grou*p

(c) *Joint Test Array Group* **(d)** *Joint Time Analysis Group*

15. En la programación dentro del sistema de un PLD normalmente se emplea

(a) un generador de señal de reloj integrado **(b)** un procesador integrado

(c) una PROM integrada **(d)** las respuestas (a) y (b)

(e) las respuestas (b) y (c)

16. Para medir el período de un tren de impulsos, se debe usar

(a) un multímetro digital **(b)** una sonda lógica

(c) un osciloscopio **(d)** un pulsador lógico

17. Una vez medido el período de un tren de impulsos, la frecuencia se calcula

(a) utilizando otro ajuste **(b)** midiendo el ciclo de trabajo

(c) hallando el recíproco del período **(d)** usando otro tipo de instrumento

PROBLEMAS

Las respuestas a los problemas impares se encuentran al final del libro.

SECCIÓN 3.1 El inversor

1. La señal de entrada mostrada en la Figura 3.74 se aplica a un inversor. Dibujar el diagrama de tiempos de la señal de salida respecto a su entrada.

FIGURA 3.74

2. En la Figura 3.75 se muestra una red de inversores en cascada. Si se aplica un nivel ALTO en el punto *A*, determinar los niveles lógicos de los puntos *B* hasta *F*.

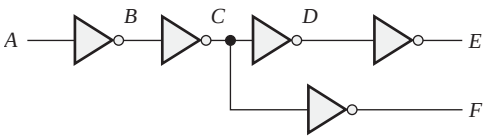


FIGURA 3.75

SECCIÓN 3.2 La puerta AND

3. Determinar la salida *X* para una puerta AND de dos entradas a la que se la aplican las señales de entrada mostradas en la Figura 3.76. Mostrar las relaciones de tiempo de la salida y las entradas mediante un cronograma.

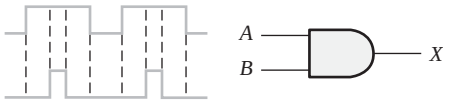


FIGURA 3.76

4. Repetir el problema 3 para las señales de la Figura 3.77.

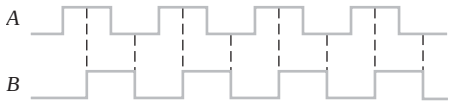


FIGURA 3.77

5. Las señales de entrada que se aplican a una puerta AND de tres entradas son las que se indican en la Figura 3.78. Determinar la señal de salida para las entradas dadas en función del tiempo, utilizando un diagrama de tiempos.

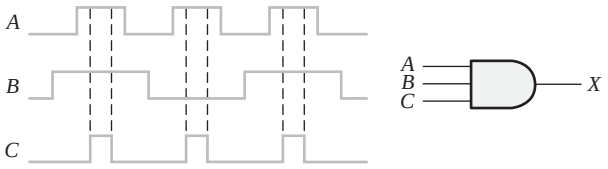


FIGURA 3.78

6. En la Figura 3.79 se indican las señales de entrada que se aplican a una puerta AND de cuatro entradas. Determinar la señal de salida para las entradas dadas en función del tiempo, mediante un cronograma.

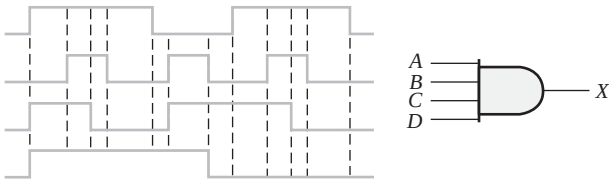


FIGURA 3.79

SECCIÓN 3.3 La puerta OR

7. Determinar la salida de una puerta OR de dos entradas cuando se aplican las señales de entrada dadas en la Figura 3.77 y dibujar el diagrama de tiempos.

8. Repetir el problema 5 para una puerta OR de 3 entradas.

9. Repetir el problema 6 para una puerta OR de 4 entradas.

10. Para las cinco señales de entrada de la Figura 3.80, determinar la salida en una puerta AND de 5 entradas y de una puerta OR de 5 entradas. Dibujar el diagrama de tiempos.

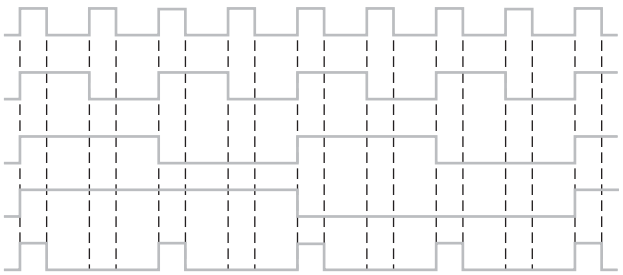


FIGURA 3.80

SECCIÓN 3.4 La puerta NAND

11. Para el conjunto de señales de entrada de la Figura 3.81, determinar la salida de la puerta mostrada y dibujar el diagrama de tiempos.

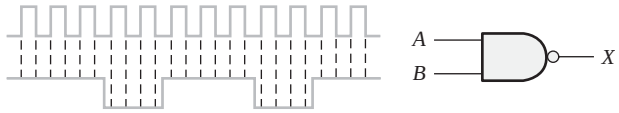


FIGURA 3.81

12. Determinar la salida de la puerta para la señales de entrada de la Figura 3.82 y dibujar el diagrama de tiempos.

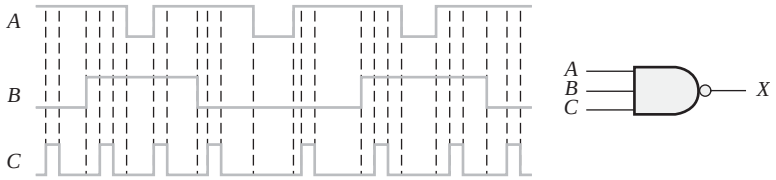


FIGURA 3.82

13. Determinar la señal de la salida correspondiente a la Figura 3.83.

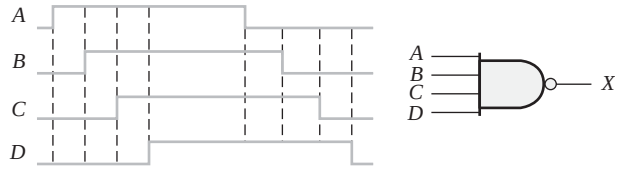


FIGURA 3.83

14. Como ya sabe, los dos símbolos lógicos representados en la Figura 3.84 representan operaciones equivalentes. La diferencia entre ellos es estrictamente de tipo funcional. Para el símbolo NAND, se requieren dos entradas a nivel ALTO para obtener una salida a nivel BAJO. Para el símbolo negativa–OR se requiere al menos una entrada a nivel BAJO para obtener una salida a nivel ALTO. Utilizando estos dos puntos de vista funcionales, demostrar que producirán la misma salida para las entradas dadas.

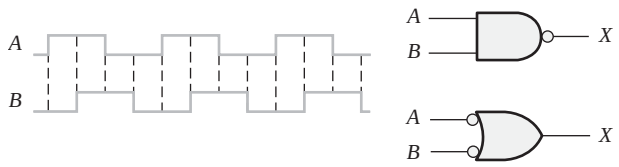


FIGURA 3.84

SECCIÓN 3.5 La puerta NOR

15. Repetir el problema 11 para una puerta NOR de 2 entradas.

16. Determinar la señal de salida para las entradas indicadas en la Figura 3.85, y dibujar el diagrama de tiempos.

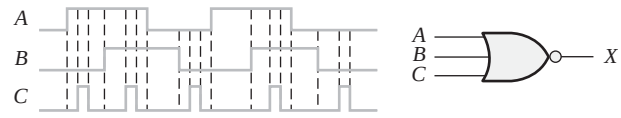


FIGURA 3.85

17. Repetir el problema 13 para una puerta NOR de 4 entradas.

18. Los símbolos de las puertas NAND y negativa–OR representan operaciones equivalentes, pero son funcionalmente diferentes. Para la puerta NOR, se necesita al menos una de las entradas a nivel ALTO para obtener un nivel BAJO de salida. Para la puerta negativa–AND, se necesita que las dos entradas estén a nivel BAJO para obtener un nivel de salida ALTO. Utilizando estos dos puntos de vista funcionales, demostrar que ambas puertas de la Figura 3.86 generarán la misma salida para las entradas dadas.

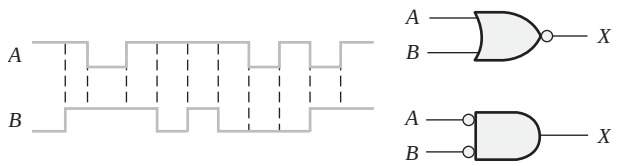


FIGURA 3.86

SECCIÓN 3.6 Puertas OR–exclusiva y NOR–exclusiva

19. ¿En qué difiere la operación lógica de la puerta OR–exclusiva de la puerta OR?

20. Repetir el problema 11 para una puerta OR–exclusiva.

21. Repetir el problema 11 para una puerta NOR–exclusiva.

22. Determinar la salida de una puerta OR–exclusiva para las entradas indicadas en la Figura 3.77, y dibujar el diagrama de tiempos.

SECCIÓN 3.7 Lógica programable

23. En la matriz AND programada mediante conexiones programables de la Figura 3.87, determinar las expresiones booleanas de salida.

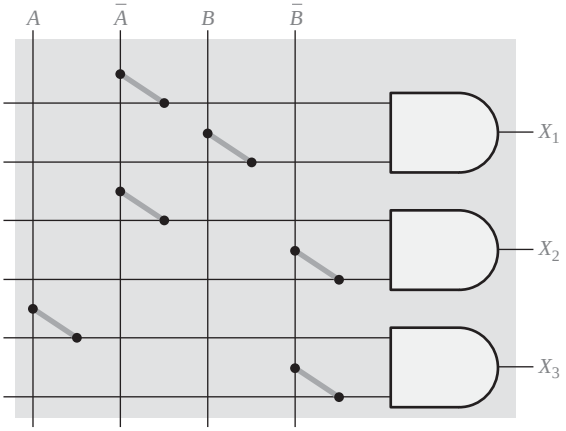


FIGURA 3.87

24. Determinar mediante los números de fila y de columna qué fusibles deben fundirse en la matriz AND programable de la Figura 3.88, para implementar cada uno de los siguientes términos producto: $X_1 = \bar{A}BC$, $X_2 = AB\bar{C}$, $X_3 = \bar{A}B\bar{C}$.

SECCIÓN 3.8 Lógica de función fija

25. En la comparación de ciertos dispositivos lógicos se ha observado que la disipación de potencia para un tipo en concreto aumenta cuando aumenta la frecuencia. ¿Se trata de un dispositivo TTL o CMOS?

26. Utilizando las hojas de características de las Figuras 3.65 y 3.66, determinar lo siguiente:

(a) La disipación de potencia del 74LS00 para la máxima tensión de alimentación y un ciclo de trabajo del 50%.

(b) La tensión de salida mínima para el nivel ALTO de un 74LS00.

(c) El retardo de propagación máximo para un 74LS00.

(d) La tensión de salida máxima para el nivel BAJO de un 74HC00A.

(e) El retardo de propagación máximo para un 74HC00A.

27. Determinar t_{PLH} y t_{PHL} para las señales de la pantalla del osciloscopio de la Figura 3.89. Las lecturas están expresadas en V/div y segundos/división para cada canal.

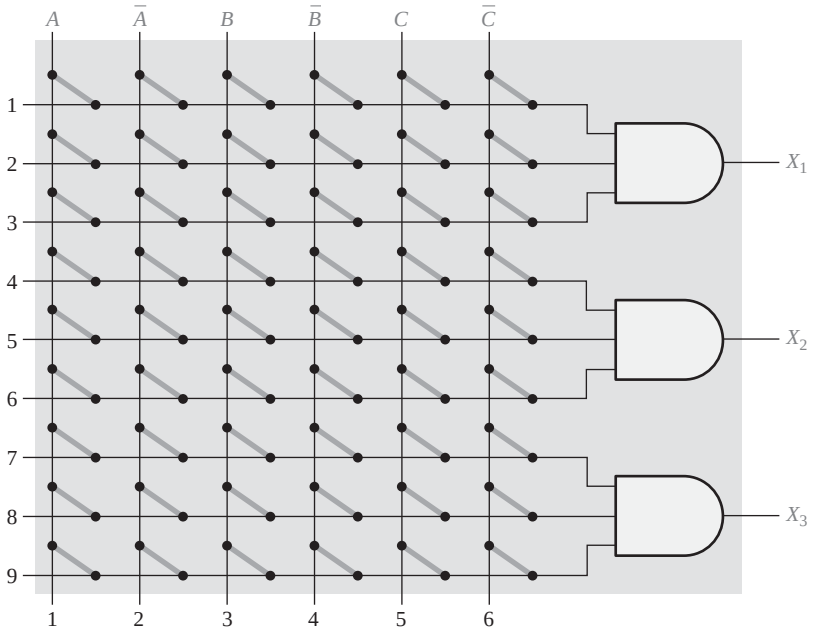


FIGURA 3.88

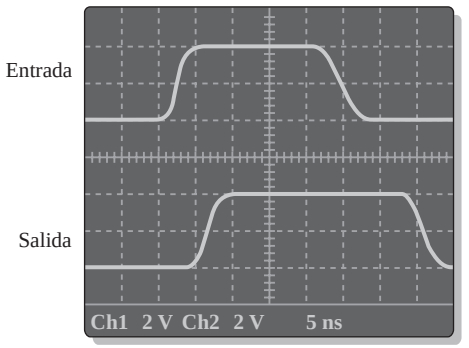


FIGURA 3.89

28. La puerta *A* tiene $t_{PLH} = t_{PHL} = 6$ ns. La puerta *B* tiene $t_{PLH} = t_{PHL} = 10$ ns. ¿Qué puerta puede trabajar a una frecuencia más alta?

29. Si una puerta lógica trabaja con una tensión de alimentación continua de +5 V y circula una corriente media de 4 mA, ¿cuál es la potencia que disipa?

30. La variable I_{CCH} representa la corriente continua de alimentación procedente de V_{CC} cuando todas las salidas de un CI están a nivel ALTO. La variable I_{CCL} representa la corriente de alimentación continua cuando todas las salidas están a nivel BAJO. Para el CI 74LS00, determinar la disipación de potencia típica cuando las salidas de las cuatro puertas están a nivel ALTO. Consulte la hoja de características de la Figura 3.65.

SECCIÓN 3.9 Localización de averías

31. Examinar las condiciones indicadas en la Figura 3.90, e identificar las puertas que fallan.

32. Determinar las puertas que fallan de la Figura 3.91 analizando los cronogramas.

33. Utilizando un osciloscopio, se realizan las observaciones indicadas en la Figura 3.92. Para cada observación, determinar la puerta que es más probable que falle.

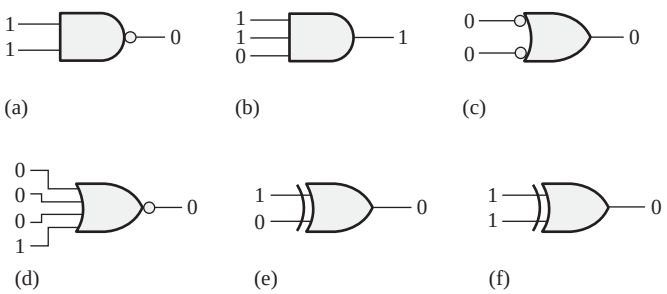


FIGURA 3.90

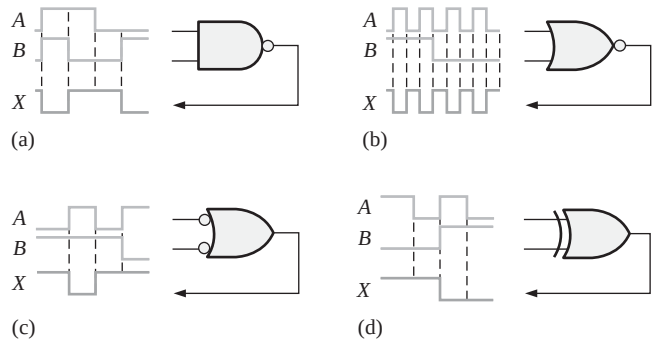


FIGURA 3.91

34. El circuito de alarma de cinturón de seguridad de la Figura 3.16 funciona mal. Se detecta que cuando se enciende el interruptor de arranque y el cinturón está abrochado, la alarma se activa y no se apaga. ¿Cuál será el problema más probable? ¿Cómo lo localizaría?

35. Cada vez que se enciende el interruptor de arranque del circuito de la Figura 3.16, la alarma se activa durante treinta segundos, incluso cuando el cinturón está abrochado. ¿Cuál es la causa más probable de este mal funcionamiento?

36 ¿Qué fallos cree que se pueden haber producido si la salida de una puerta NAND de 3 entradas permanece a nivel ALTO independientemente del nivel de las entradas?

Problemas especiales de diseño

37. Se utilizan sensores para supervisar la presión y la temperatura de una solución química almacenada en un recipiente. La circuitería de cada sensor genera un nivel de tensión ALTO cuando se excede un valor máximo especificado. Cuando se excede la presión o la temperatura, se debe activar una alarma que requiere un nivel de tensión de entrada BAJO. Diseñar un circuito para esta aplicación.

38. En un determinado proceso de fabricación automatizado, se insertan automáticamente los componentes en una tarjeta de circuito impreso. Después de activar la herramienta de inserción, la tarjeta de circuito impreso debe estar correctamente posicionada, y el componente que se va a insertar debe estar en la recámara. Estas condiciones previas se indican mediante un nivel de tensión ALTO. La herramienta de inserción requiere un nivel de tensión BAJO para activarse. Diseñar un circuito para implementar este proceso.

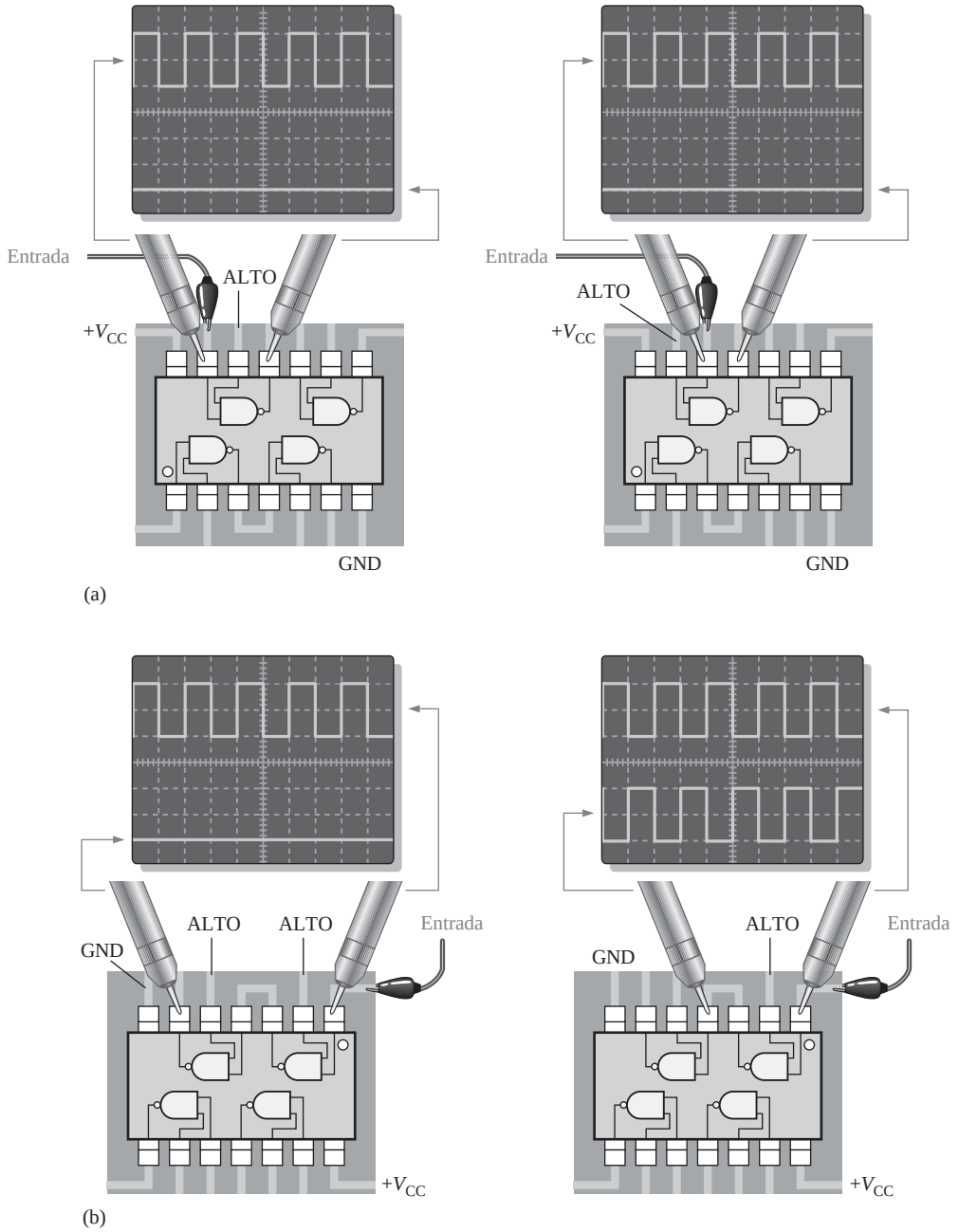


FIGURA 3.92

39. Modificar el contador de frecuencia de la Figura 3.15 para que opere con un impulso de activación (*enable*) que sea activo a nivel BAJO, en lugar de a nivel ALTO, durante el intervalo de 1s.

40. Suponer que la señal de activación de la Figura 3.15 es la forma de onda indicada en la Figura 3.93. Suponer que también se dispone de la señal *B*. Diseñar un circuito que genere un impul-

so de reset activo a nivel ALTO para el contador, sólo durante el tiempo que la señal de activación está a nivel BAJO.

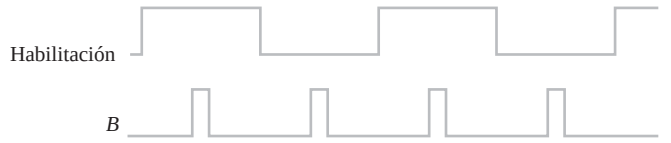


FIGURA 3.93

41. Diseñar un circuito que se colocará en el bloque rayado de la Figura 3.94, que haga que las luces delanteras de un coche se apaguen automáticamente 15 s después de que se apague el interruptor de arranque, en el caso de que el interruptor de las luces se deje activado. Suponer que se necesita un nivel BAJO para apagar las luces.

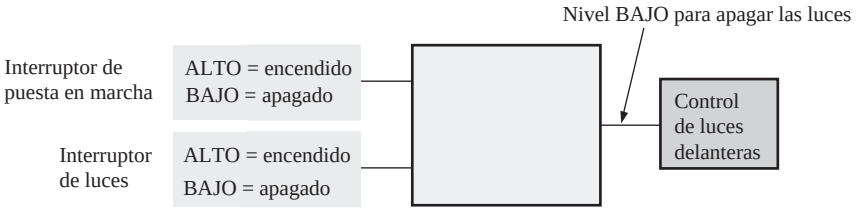


FIGURA 3.94

42. Modificar el circuito lógico de detección de intrusión de la Figura 3.24, para que se puedan proteger dos habitaciones adicionales, cada una de ellas con dos ventanas y una puerta.

43. Modificar el circuito lógico del Problema 42 para realizar un cambio en los sensores de entrada, donde Abierto = nivel BAJO y Cerrado = nivel ALTO.

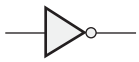
RESPUESTAS

REVISIONES DE CADA SECCIÓN

SECCIÓN 3.1 **El inversor**

1. Cuando la entrada del inversor es 1, la salida es 0.

2. **(a)**



(b) Hay un impulso negativo en la salida (pasa de nivel ALTO a BAJO, y vuelve a nivel ALTO).

SECCIÓN 3.2 **La puerta AND**

1. La salida de una puerta AND es un nivel ALTO cuando todas las entradas están a nivel ALTO.

2. La salida de una puerta AND es un nivel BAJO cuando una o más entradas están a nivel BAJO.

3. Puerta AND de 5 entradas: $X = 1$ cuando $ABCDE = 11111$, y $X = 0$ para las restantes combinaciones de $ABCDE$.

SECCIÓN 3.3 La puerta OR

1. La salida de una puerta OR es un nivel ALTO cuando una o más entradas están a nivel ALTO.
2. La salida de una puerta OR es un nivel BAJO cuando todas las entradas están a nivel BAJO.
3. Puerta OR de 3 entradas: $X = 0$ cuando $ABC = 000$, y $X = 1$ para las restantes combinaciones de ABC.

SECCIÓN 3.4 La puerta NAND

1. La salida de una puerta NAND es un nivel BAJO cuando todas las entradas están a nivel ALTO.
2. La salida de una puerta NAND es un nivel ALTO cuando una o más entradas están a nivel BAJO.
3. NAND: salida activa a nivel BAJO cuando todas las entradas están a nivel ALTO. Negativa–OR: salida activa a nivel ALTO cuando una o más entradas están a nivel BAJO. Ambas tienen la misma tabla de verdad.
4. $X = \overline{ABC}$

SECCIÓN 3.5 La puerta NOR

1. La salida de una puerta NOR es un nivel ALTO cuando todas las entradas están a nivel BAJO.
2. La salida de una puerta NOR es un nivel BAJO cuando una o más entradas están a nivel ALTO.
3. NOR: salida activa a nivel BAJO para una o más entradas a nivel ALTO; negativa–AND: salida activa a nivel ALTO cuando todas las entradas están a nivel BAJO. Ambas tienen la misma tabla de verdad.
4. $X = \overline{A + B + C}$

SECCIÓN 3.6 Puertas exclusiva–OR y exclusiva–NOR

1. La salida de una puerta XOR es un nivel ALTO cuando las entradas están a niveles opuestos.
2. La salida de una puerta XNOR es un nivel ALTO cuando las entradas están al mismo nivel.
3. Aplicar los bits a las entradas de una puerta XOR. Cuando la salida está a nivel ALTO, los bits son diferentes.

SECCIÓN 3.7 Lógica programable

1. Fusible, antifusible, EPROM, EEPROM y SRAM
2. Volátil quiere decir que se pierden todos los datos cuando se desconecta la alimentación y, en consecuencia, el PLD debe reprogramarse; basada en SRAM.
3. Interfaz de texto e interfaz gráfica.
4. JTAG corresponde a *Joint Test Action Group*; el estándar 1149.1 del IEEE para programación y realización de pruebas.

SECCIÓN 3.8 Lógica de función fija

1. CMOS y TTL.
2. **(a)** LS: Schottky de baja potencia. **(b)** ALS: LS avanzada.
   (c) F– TTL rápida **(d)** HC: CMOS de alta velocidad
   (e) AC: CMOS avanzada. **(f)** HCT: HC CMOS TTL compatible
   (g) LV: CMOS de baja tensión.
3. **(a)** 74LS04: inversor séxtuple **(b)** 74HC00: cuádruple NAND de 2 entradas
   (c) 74LV08: cuádruple AND de 2 entradas **(d)** 74ALS10: triple NAND de 3 entradas
   (e) 7432: cuádruple OR de 2 entradas **(f)** 74ACT11: triple AND de 3 entradas

(g) 74AHC02: cuádruple NOR de 2 entradas.

4. Menor disipación de potencia: CMOS.
5. Seis inversores en un encapsulado; cuatro puertas NAND de dos entradas en un encapsulado.
6. $t_{PLH} = 10$ ns; $t_{PHL} = 8$ ns.
7. 18 pJ
8. I_{CCL}: corriente de alimentación continua para el estado de salida BAJO; I_{CCH}: corriente de alimentación continua para el estado de salida ALTO.
9. V_{IL}: tensión de entrada para el nivel BAJO; V_{IH}: tensión de entrada para el nivel ALTO.
10. V_{OL}: tensión de salida para el nivel BAJO; V_{OH}: tensión de salida para el nivel ALTO.

SECCIÓN 3.9 Localización de averías

1. Los fallos más comunes son los circuitos abiertos y los cortocircuitos.
2. Una entrada en circuito abierto se comporta como un nivel de entrada ALTO.
3. Amplitud y período.

PROBLEMAS RELACIONADOS

3.1 El diagrama de tiempos no varía.

3.2 *Véase* la Tabla 3.13.

Entradas *ABCD*	Salida *X*	Entradas *ABCD*	Salida *X*
0000	0	1000	0
0001	0	1001	0
0010	0	1010	0
0011	0	1011	0
0100	0	1100	0
0101	0	1101	0
0110	0	1110	0
0111	0	1111	1

TABLA 3.13

3.3 *Véase* la Figura 3.95.

3.4 La forma de onda de salida es igual que la entrada *A*.

3.5 *Véase* la Figura 3.96.

3.6 *Véase* la Figura 3.97.

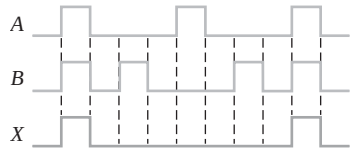


FIGURA 3.95

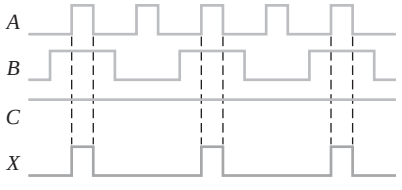


FIGURA 3.96

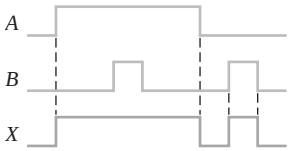


FIGURA 3.97

3.7 *Véase* la Figura 3.98.

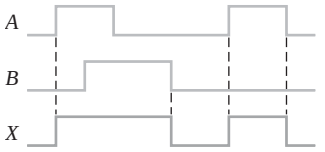


FIGURA 3.98

3.8 *Véase* la Figura 3.99.

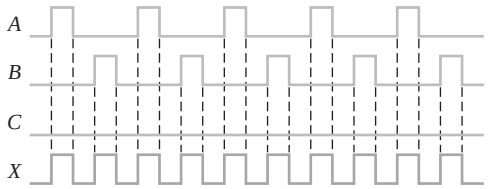


FIGURA 3.99

3.9 *Véase* la Figura 3.100.

3.10 *Véase* la Figura 3.101.

.

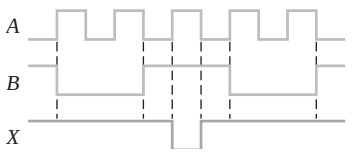


FIGURA 3.100

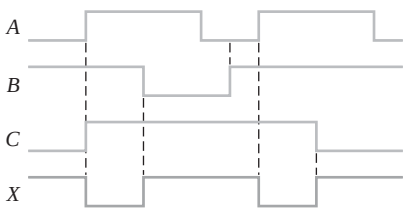


FIGURA 3.101

3.11 Utilizar una puerta NAND de 3 entradas.

3.12 Utilizar una puerta NAND de 4 entradas que funcione como una puerta OR–Negativa.

3.13 *Véase* la Figura 3.102.

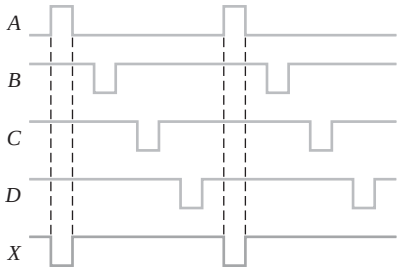


FIGURA 3.102

3.14 *Véase* la Figura 3.103.

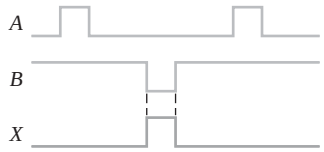


FIGURA 3.103

3.15 *Véase* la Figura 3.104.

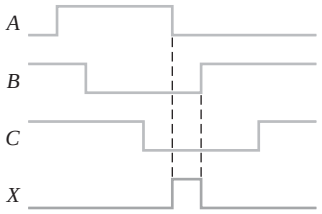


FIGURA 3.104

3.16 Utilizar una puerta NOR de 2 entradas.

3.17 Una puerta NAND de 3 entradas.

3.18 La salida siempre está a nivel BAJO. El cronograma es una línea recta.

3.19 La puerta OR–exclusiva no detectará simultáneamente fallos si ambos circuitos generan la misma salida.

3.20 Las salidas no se ven afectadas.

3.21 6 columnas, 9 filas y tres puertas AND con tres entradas cada una.

3.22 La puerta con t_{PLH} y t_{PHL} igual a 4 ns puede funcionar a la frecuencia más alta.

3.23 10 mW

3.24 La salida de la puerta o el pin 13 de entrada están internamente en circuito abierto.

3.25 El display mostrará una lectura errónea porque el contador continúa hasta que se pone a cero.

3.26 El impulso de habilitación es demasiado corto o el contador se ha puesto a cero demasiado pronto.

AUTOTEST

1. (d) **2.** (d) **3.** (a) **4.** (e) **5.** (c) **6.** (a) **7.** (d)
8. (b) **9.** (d) **10.** (b) **11.** (d) **12.** (c) **13.**(b)
14. (a) **15.** (d) **16.** (c) **17.** (c)

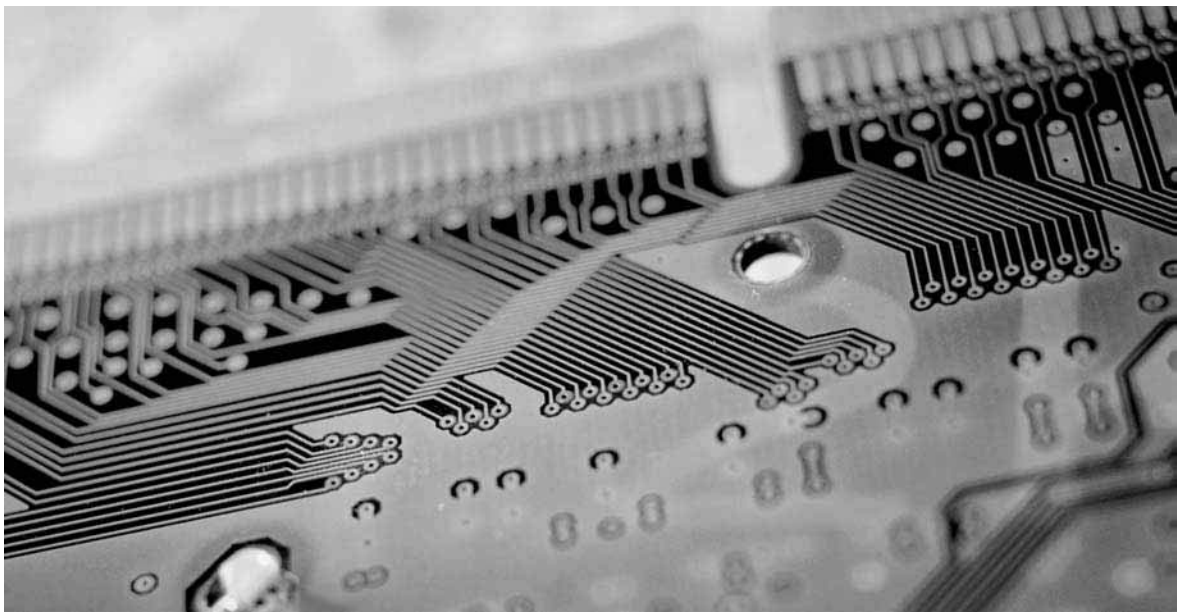
4

ÁLGEBRA DE BOOLE Y SIMPLIFICACIÓN LÓGICA

CONTENIDO DEL CAPÍTULO

OBJETIVOS DEL CAPÍTULO

- Aplicar las leyes y reglas básicas del álgebra de Boole.
- Aplicar los teoremas de DeMorgan a las expresiones booleanas.
- Describir redes de puertas mediante expresiones booleanas.



- Evaluar las expresiones booleanas.
- Simplificar expresiones mediante las leyes y reglas del álgebra booleana.
- Convertir cualquier expresión booleana en una suma de productos.
- Convertir cualquier expresión booleana en un producto de sumas.
- Utilizar el mapa de Karnaugh para simplificar expresiones booleanas.
- Utilizar el mapa de Karnaugh para simplificar tablas de verdad.
- Utilizar condiciones "indiferentes" para simplificar funciones lógicas.
- Aplicar el álgebra de Boole, los mapas de Karnaugh y el lenguaje VHDL a los sistemas digitales.

PALABRAS CLAVE

- Variable
- Complemento
- Término suma
- Término producto
- Suma de productos
- Producto de sumas
- Mapa de Karnaugh
- Indiferente
- VHDL

INTRODUCCIÓN

En 1854, George Boole publicó una obra titulada *Investigación de las leyes del pensamiento, sobre las que se basan las teorías matemáticas de la lógica y la probabilidad*. En esta publicación se formuló la idea de un "álgebra lógica", que se conoce hoy en día como álgebra de Boole. El álgebra de Boole es una forma adecuada y sistemática de expresar y analizar las operaciones de los circuitos lógicos. Claude Shannon fue el primero en aplicar la obra de Boole al análisis y diseño de circuitos. En 1938, Shannon escribió su tesis doctoral en el MIT (*Massachussets Institute of Technology*) titulada *Análisis simbólico de los circuitos de conmutación y relés*.

Este capítulo se ocupa de las leyes, reglas y teoremas del álgebra booleana y sus aplicaciones a los circuitos digitales. Aprenderá a definir un circuito mediante una expresión booleana y a determinar su funcionamiento. También se tratará la simplificación de los circuitos lógicos utilizando el álgebra booleana y los mapas de Karnaugh.

También se presenta el lenguaje de descripción hardware VHDL para la programación de dispositivos lógicos.

APLICACIÓN A LOS SISTEMAS DIGITALES

Esta aplicación a los sistemas digitales ilustra los conceptos que serán explicados a lo largo del capítulo. El funcionamiento del display de 7 segmentos del sistema de control y recuento de pastillas del Capítulo 1 es un buen método para ilustrar la aplicación del álgebra de Boole y de los mapas de Karnaugh, de modo que se obtenga la más sencilla implementación en el diseño de circuitos lógicos. Por tanto, en está aplicación a los sistemas digitales, nos centraremos en la lógica del convertidor BCD-7 segmentos que gobierna los dos displays del sistema indicados en la Figura 1.58.

4.1 OPERACIONES Y EXPRESIONES BOOLEANAS

El álgebra de Boole son las matemáticas de los sistemas digitales. Es indispensable tener unos conocimientos básicos del álgebra booleana para estudiar y analizar los circuitos lógicos. En el capítulo anterior, se han presentado las operaciones y expresiones booleanas para las puertas NOT, AND, OR, NAND y NOR.

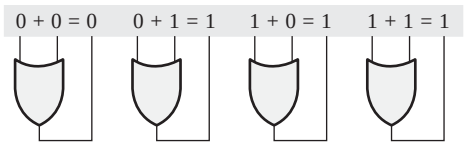
Al finalizar esta sección, el lector deberá ser capaz de:

■ Definir *variable*. ■ Definir *literal*. ■ Identificar un término suma. ■ Evaluar un término suma. ■ Identificar un término producto. ■ Evaluar un término producto. ■ Explicar la adición booleana. ■ Explicar la multiplicación booleana.

Los términos *variable, complemento* y *literal* son términos utilizados en el álgebra booleana. Una ***variable*** es un símbolo (normalmente una letra mayúscula en cursiva) que se utiliza para representar magnitudes lógicas. Cualquier variable puede tener un valor de 0 o de 1. El **complemento** es el inverso de la variable y se indica mediante una barra encima de la misma. Por ejemplo, el complemento de la variable A es $\overline{A}$. Si $A = 1$, entonces $\overline{A} = 0$. Si $A = 0$, entonces $\overline{A} = 1$. El complemento de la variable A se lee "no A" o "A barra". En ocasiones, se emplea un apóstrofe en lugar de la barra para indicar el complemento de una variable; por ejemplo B' indica el complemento de B. En este libro, sólo se utiliza la barra. Un **literal** es una variable o el complemento de una variable.

Suma booleana

Como hemos visto en el Capítulo 3, la **suma booleana** es equivalente a la operación OR y a continuación se muestran sus reglas básicas junto con su relación con la puerta OR:



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

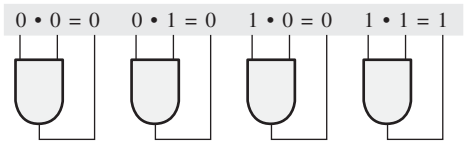
▲ *La puerta OR es un sumador booleano.*

Un término suma es igual a 1 cuando uno o más de los literales del término es 1. Un término suma es igual a 0 sólo si cada uno de los literales son iguales a 0.

Multiplicación booleana

▲ *La puerta AND es un multiplicador booleano.*

En el Capítulo 3 vimos también que la **multiplicación booleana** es equivalente a la operación AND y sus reglas básicas junto con sus relaciones con la puerta AND se ilustran a continuación:



NOTAS INFORMÁTICAS

En un microprocesador, la unidad aritmético lógica (ALU) realiza las operaciones aritméticas y lógicas booleanas sobre los datos digitales mediante instrucciones de programa. Las operaciones lógicas son equivalentes a las operaciones de las puertas básicas con las que ya estamos familiarizados, aunque se trabaja con ocho bits como mínimo a la vez. Ejemplos de instrucciones lógicas booleanas son AND, OR, NOT y XOR, que se denominan *mnemónicos*. Un programa en lenguaje ensamblador utiliza estos mnemónicos para especificar una operación. Y otro programa denominado *ensamblador* traduce los mnemónicos a un código binario que puede entender el microprocesador.

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

Un término producto es igual a 1 sólo si cada uno de los literales del término es 1. Un término producto es igual a 0 cuando uno o más de los literales son iguales a 0.

EJEMPLO 4.1

Determinar los valores de *A, B, C* y *D* que hacen que el término suma $A+\overline{B}+C+\overline{D}$ sea igual a cero.

Solución Para que el término suma sea 0, cada uno de los literales del término debe ser igual a 0. Por tanto, $A = \mathbf{0}$, $B = 1$ (para que $\overline{B} = 0$) y $D = \mathbf{1}$) para que $\overline{D} = 0$).

$$A+\overline{B}+C+\overline{D} = 0+\overline{1}+0+\overline{1} = 0+0+0+0 = 0$$

Problema relacionado* Determinar los valores de A y B de modo que el término suma $\overline{A}+B$ sea igual a 0.

* Las respuestas se encuentran al final del capítulo.

EJEMPLO 4.2

Determinar los valores de *A, B, C* y *D* que hacen que el término producto $A\overline{B}C\overline{D}$ sea igual a 1.

Solución Para que el término producto sea 1, cada uno de los literales del término debe ser igual a 1. Por tanto, $A = \mathbf{1}$, $B = \mathbf{0}$ (para que $\overline{B} = 1$), $C = \mathbf{1}$ y $D = \mathbf{0}$ (para que $\overline{D} = 1$).

$$A\overline{B}C\overline{D} = 1\cdot\overline{0}\cdot1\cdot\overline{0} = 1\cdot1\cdot1\cdot1 = 1$$

Problema relacionado Determinar los valores de A y B de modo que el término suma $\overline{A}\overline{B}$ sea igual a 1.

REVISIÓN DE LA SECCIÓN 4.1

Las respuestas se encuentran al final del capítulo

1. Si $A = 0$, ¿cuánto vale $\overline{A}$?
2. Determinar los valores de A, B y C que hacen que el término suma $\overline{A}+\overline{B}+C$ sea igual a 0.
3. Determinar los valores de A, B y C que hacen que el término producto $A\overline{B}C$ sea igual a 1.

4.2 LEYES Y REGLAS DEL ÁLGEBRA DE BOOLE

Al igual que en otras áreas de las matemáticas, existen en el álgebra de Boole una serie de reglas y leyes bien determinadas que tienen que seguirse para aplicarla correctamente. Las más importantes son las que se presentan en esta sección.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Aplicar las leyes conmutativas de la adición y multiplicación. ■ Aplicar las leyes asociativas de la adición y multiplicación. ■ Aplicar la ley distributiva. ■ Aplicar las doce reglas básicas del álgebra de Boole.

Leyes del álgebra de Boole

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

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

Ecuación 4.1 $$A + B = B + A$$

Esta ley establece que el orden en que se aplica a las variables la operación OR es indiferente. Recuerde que cuando se aplica a los circuitos lógicos, la suma y la operación OR es lo mismo. La Figura 4.1 ilustra la ley conmutativa aplicada a una puerta OR, en la que se puede ver que es indistinto a qué entrada asignemos cada una de las variables. (El símbolo ≡ significa "equivalente a".)

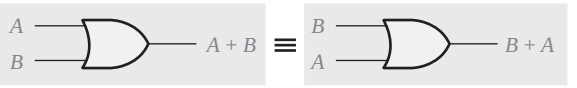


FIGURA 4.1 Aplicación de la ley conmutativa de la suma.

La *ley conmutativa de la multiplicación* para dos variables es

Ecuación 4.2 $$AB = BA$$

Esta ley establece que el orden en que se aplica a las variables la operación AND es indiferente. La Figura 4.2 ilustra esta ley tal y como se aplica a la puerta AND.

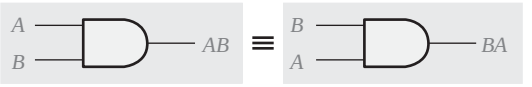


FIGURA 4.2 Aplicación de la ley conmutativa de la multiplicación.

Leyes asociativas La *ley asociativa de la suma* para tres variables se escribe como sigue:

Ecuación 4.3 $$A + (B + C) = (A + B) + C$$

Esta ley establece que cuando se aplica la operación OR a más de dos variables, el resultado es el mismo independientemente de la forma en que se agrupen las variables. La Figura 4.3 ilustra esta ley aplicada a puertas OR de dos entradas.

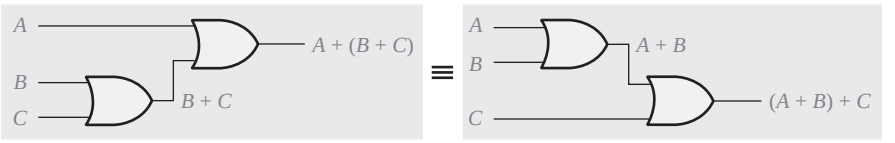


FIGURA 4.3 Aplicación de la ley asociativa de la suma.

La *ley asociativa de la multiplicación* para tres variables se escribe del siguiente modo:

Ecuación 4.4 $$A(BC) = (AB)C$$

Esta ley establece que cuando se aplica la operación AND a más de dos variables, el resultado es el mismo independientemente de la forma en que se agrupen las variables. La Figura 4.4 ilustra esta ley aplicada a puertas AND de dos entradas.

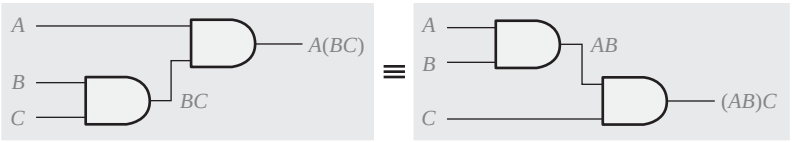


FIGURA 4.4 Aplicación de la ley asociativa de la multiplicación.

Ley distributiva La *ley distributiva* para tres variables se escribe como sigue:

Ecuación 4.5 $$A(B + C) = AB + AC$$

Esta ley establece que aplicar la operación OR a dos o más variables y luego aplicar la operación AND al resultado de esa operación y a otra variable aislada, es equivalente a aplicar la operación AND a la variable aislada con cada uno de los sumandos y luego realizar la operación OR con los productos resultantes. La ley distributiva expresa también el proceso de *sacar factor común* en el que la variable común A se saca como factor de los productos parciales, como por ejemplo, $AB + AC = A(B + C)$. La Figura 4.5 ilustra la ley distributiva mediante su implementación de puertas.

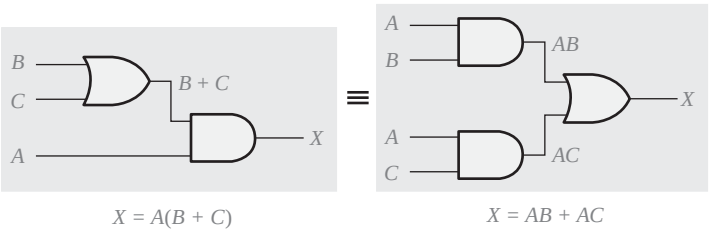


FIGURA 4.5 Aplicación de la ley distributiva.

Reglas del álgebra booleana

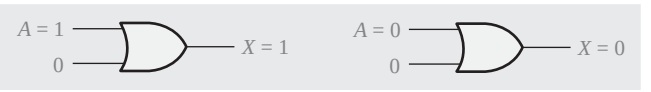
La Tabla 4.1 enumera las doce reglas básicas, muy útiles, para la manipulación y simplificación de **expresiones booleanas**. Las nueve primeras reglas las veremos en términos de su aplicación a las puertas lógicas. Las reglas 10 a 12 se obtendrán a partir de las reglas más sencillas y de las leyes anteriormente explicadas.

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

A, *B* o *C* pueden representar una sola variable o una combinación de variables.

TABLA 4.1 Reglas básicas del Álgebra de Boole.

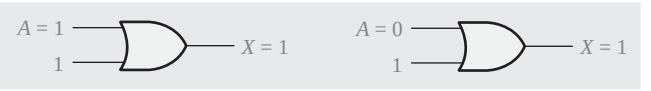
Regla 1. $A + 0 = A$ Si aplicamos la operación OR a una variable cualquiera y a 0, el resultado es siempre igual a la variable. Si *A* es 1, la salida es igual a 1 y, por tanto, igual a *A*. Si *A* es 0, la salida es 0 e igualmente idéntica a *A*. Esta ley se ilustra en la Figura 4.6 en la que la entrada inferior está siempre a 0.



$X = A + 0 = A$

FIGURA 4.6

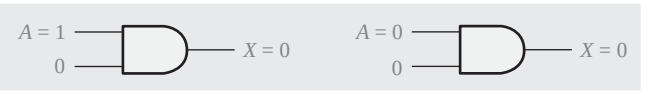
Regla 2. $A + 1 = 1$ Si se aplica la operación OR a una variable y a 1, el resultado es siempre igual a 1. Un 1 en una entrada de una puerta OR produce siempre un 1 en la salida, independientemente del valor de la otra entrada. Esta regla se ilustra en la Figura 4.7, en la que la entrada inferior está siempre a 1.



$X = A + 1 = 1$

FIGURA 4.7

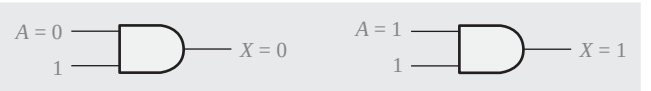
Regla 3. $A \cdot 0 = 0$ Si se aplica la operación AND a una variable y a 0, el resultado es siempre igual a 0. Siempre que una de las entradas de una puerta AND sea 0, la salida siempre es 0, independientemente del valor de la otra entrada. Esta regla se ilustra en la Figura 4.8, en la que la entrada inferior está siempre a 0.



$X = A \bullet 0 = 0$

FIGURA 4.8

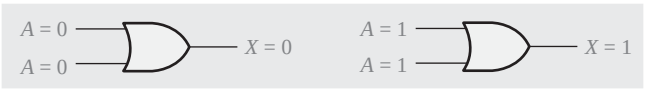
Regla 4. $A \cdot 1 = A$ Si se aplica la operación AND a una variable y a 1, el resultado es siempre igual a la variable. Si la variable *A* es 0, la salida de la puerta AND será siempre 0, mientras que si *A* es 1, la salida será 1, dado que las dos entradas son 1. Esta regla se ilustra en la Figura 4.9, en la que la entrada inferior está siempre a 1.



$X = A \bullet 1 = A$

FIGURA 4.9

Regla 5. $A + A = A$ Si se aplica la operación OR a una variable consigo misma, el resultado es siempre igual a la variable. Si A es 0, entonces $0 + 0 = 0$, mientras que si A es 1, $1 + 1 = 1$. Esto se muestra en la Figura 4.10, en la que se aplica la misma variable a ambas entradas.



$X = A + A = A$

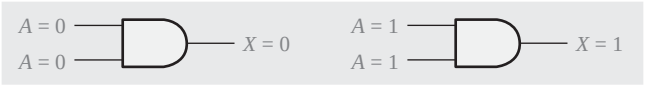
FIGURA 4.10

Regla 6. $A + \overline{A} = 1$ Si se aplica la operación OR a una variable y a su complemento, el resultado es siempre igual a 1. Si A es 0, entonces $0 + \overline{0} = 0 + 1 = 1$. Si A es 1, entonces $1 + \overline{1} = 1 + 0 = 1$. En la Figura 4.11 podemos ver una puerta OR en la que sus entradas son una variable y su complemento.

$X = A + \overline{A} = 1$

FIGURA 4.11

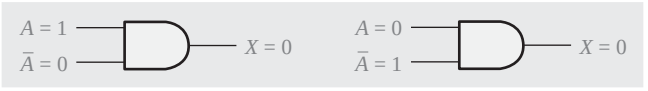
Regla 7. $A \cdot A = A$ Si se aplica la operación AND a una variable consigo misma, el resultado siempre es igual a la variable. Si A = 0, entonces $0 \cdot 0 = 0$, y si $A = 1$, entonces $1 \cdot 1 = 1$. Esta regla se ilustra en la Figura 4.12.



$X = A \bullet A = A$

FIGURA 4.12

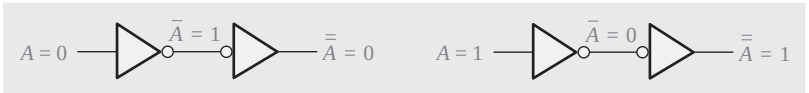
Regla 8. $A \cdot \overline{A} = 0$ Si se aplica la operación AND a una variable y a su complemento, el resultado es siempre igual a 0. Esta regla se basa en que siempre A o $\overline{A}$ será 0, y además en que cuando se aplica un 0 a una de las entradas de una puerta AND, la salida siempre es 0. Esta regla se ilustra en la Figura 4.13.



$X = A \bullet \overline{A} = 0$

FIGURA 4.13

Regla 9. $\overline{\overline{A}} = A$ El complemento del complemento de una variable es siempre la propia variable. El complemento de la variable A es $\overline{A}$ y el complemento de $\overline{A}$ será de nuevo A, que es la variable original. Esta regla se muestra en la Figura 4.14 mediante el uso de dos inversores.



$\overline{\overline{A}} = A$

FIGURA 4.14

Regla 10. $A + AB = A$ Esta regla se puede obtener aplicando la ley distributiva y las reglas 2 y 4, de la siguiente forma:

$A + AB = A(1 + B)$ Sacar factor común (ley distributiva)
$= A \cdot 1$ Regla 2: $(1 + B) = 1$
$= A$ Regla 4: $A \cdot 1 = A$

La demostración se muestra en la Tabla 4.2, la cual incluye la tabla de verdad y la simplificación del circuito lógico resultante.

A	B	AB	$A + AB$
0	0	0	0
0	1	0	0
1	0	0	1
1	1	1	1

igual

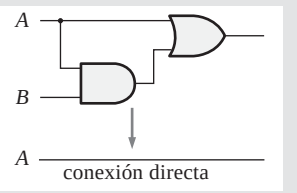


TABLA 4.2 Regla 10: $A + AB = A$.

Regla 11. $A + \bar{A}B = A + B$ Esta regla puede demostrarse de la siguiente forma:

$A + \bar{A}B = (A + AB) + \bar{A}B$ Regla 10: $A = A + AB$
$= (AA + AB) + \bar{A}B$ Regla 7: $A = AA$
$= AA + AB + A\bar{A} + \bar{A}B$ Regla 8: sumar $A\bar{A} = 0$
$= (A + \bar{A})(A + B)$ Sacar factor común
$= 1 \cdot (A + B)$ Regla 6: $A + \bar{A} = 1$
$= A + B$ Regla 4: eliminar el 1

La demostración se muestra en la Tabla 4.3, la cual incluye la tabla de verdad y la simplificación del circuito lógico resultante.

A	B	$\bar{A}B$	$A + \bar{A}B$	$A + B$
0	0	0	0	0
0	1	1	1	1
1	0	0	1	1
1	1	0	1	1

igual

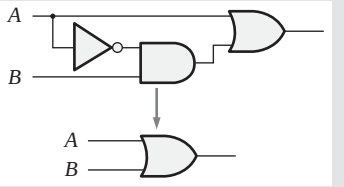


TABLA 4.3 Regla 11: $A + \bar{A}B = A + B$.

Regla 12. $(A + B)(A + C) = A + BC$ Esta regla puede demostrarse de la siguiente forma:

$(A + B)(A + C)$ $= AA + AC + AB + BC$ Ley distributiva
$= A + AC + AB + BC$ Regla 7: $AA = A$
$= A(1 + C) + AB + BC$ Sacar factor común (ley distributiva)

$= A \cdot 1 + AB + BC$ Regla 2: $1 + C = 1$

$= A(1 + B) + BC$ Sacar factor común (ley distributiva)

$= A \cdot 1 + BC$ Regla 2: $1 + B = 1$

$= A + BC$ Regla 4: $A \cdot 1 = A$

La demostración se muestra en la Tabla 4.4, la cual incluye la tabla de verdad y la simplificación del circuito lógico resultante.

A	*B*	*C*	*A* + *B*	*A* + *C*	(*A* + *B*)(*A* + *C*)	*BC*	*A* + *BC*
0	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	1	1	1	1
1	0	0	1	1	1	0	1
1	0	1	1	1	1	0	1
1	1	0	1	1	1	0	1
1	1	1	1	1	1	1	1

igual

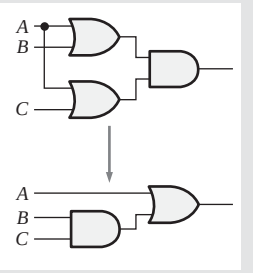


TABLA 4.4 Regla 12: $(A + B)(A + C) = A + BC$.

REVISIÓN DE LA SECCIÓN 4.2

1. Aplicar la ley asociativa de la adición a la expresión $A+(B+C+D)$.
2. Aplicar la ley distributiva a la expresión $A(B+C+D)$.

4.3 TEOREMAS DE DeMORGAN

DeMorgan, matemático que conoció a Boole, propuso dos teoremas que constituyen una parte muy importante del álgebra de Boole. En términos prácticos, los teoremas de DeMorgan proporcionan una verificación matemática de la equivalencia entre las puertas NAND y negativa-OR, y las puertas NOR y negativa-AND, que se han tratado en el Capítulo 3.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Enunciar los teoremas de DeMorgan. ■ Relacionar los teoremas de DeMorgan con la equivalencia entre las puertas NAND y negativa-OR, y entre las puertas NOR y negativa-AND. ■ Aplicar los teoremas de DeMorgan para simplificar las expresiones booleanas.

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

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

O dicho de otra manera

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

La fórmula para expresar este teorema para dos variables es:

Ecuación 4.6 $$\overline{XY} = \overline{X} + \overline{Y}$$

El segundo teorema de DeMorgan se enuncia como sigue:

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

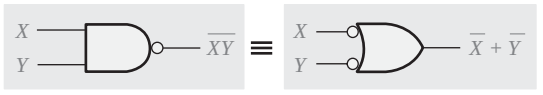
O dicho de otra manera,

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

La fórmula para expresar este teorema es:

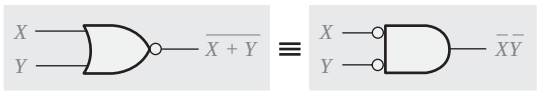
Ecuación 4.7 $$\overline{X + Y} = \overline{X}\,\overline{Y}$$

Las puertas equivalentes y tablas de verdad correspondientes a las Ecuaciones 4.6 y 4.7 se muestran en la Figura 4.15.



NAND Negativa-OR

Entradas		Salida	
X	Y	$\overline{XY}$	$\overline{X} + \overline{Y}$
0	0	1	1
0	1	1	1
1	0	1	1
1	1	0	0



NOR Negativa-AND

Entradas		Salida	
X	Y	$\overline{X + Y}$	$\overline{X}\,\overline{Y}$
0	0	1	1
0	1	0	0
1	0	0	0
1	1	0	0

FIGURA 4.15 Equivalencias de las puertas lógicas y tablas de verdad que ilustran los teoremas de DeMorgan. Observe la igualdad entre las dos columnas de salida de cada tabla. Esto demuestra que las puertas equivalentes realizan la misma función lógica.

Como se ha comentado, los teoremas de DeMorgan se aplican también a expresiones en las que existen más de dos variables. Los siguientes ejemplos ilustran la aplicación de los teoremas de DeMorgan a expresiones de 3 y 4 variables.

EJEMPLO 4.3

Aplicar los teoremas de DeMorgan a las expresiones $\overline{XYZ}$ y $\overline{X + Y + Z}$.

$$\overline{XYZ} = \overline{X} + \overline{Y} + \overline{Z}$$
$$\overline{X + Y + Z} = \overline{X}\,\overline{Y}\,\overline{Z}$$

Problema relacionado Aplicar los teoremas de DeMorgan a la expresión $\overline{\overline{X} + \overline{Y} + \overline{Z}}$.

EJEMPLO 4.4

Aplicar los teoremas de DeMorgan a las expresiones $\overline{WXYZ}$ y $\overline{W+X+Y+Z}$.

$$\overline{WXYZ} = \bar{W}+\bar{X}+\bar{Y}+\bar{Z}$$
$$\overline{W+X+Y+Z} = \bar{W}\bar{X}\bar{Y}\bar{Z}$$

Problema relacionado Aplicar los teoremas de DeMorgan a la expresión $\overline{\bar{W}\bar{X}\bar{Y}\bar{Z}}$

Como se ha establecido en las Ecuaciones 4.6 y 4.7 que enuncian los teoremas de DeMorgan, cada variable puede representar una combinación de otras variables. Por ejemplo, X puede ser igual al término $AB + C$, e Y puede ser igual a $A + BC$. De esta forma, si aplicamos el teorema de DeMorgan para dos variables expresado según $\overline{XY} = \bar{X}+\bar{Y}$ a la expresión $\overline{(AB+C)(A+BC)}$ obtenemos el siguiente resultado:

$$\overline{(AB+C)(A+BC)} = (\overline{AB+C})+(\overline{A+BC})$$

Observe que el resultado anterior tiene dos términos $\overline{AB+C}$ y $\overline{A+BC}$, a los que podemos aplicar individualmente otra vez el teorema de DeMorgan $\overline{X+Y} = \bar{X}\bar{Y}$ del siguiente modo:

$$(\overline{AB+C})+(\overline{A+BC}) = (\overline{AB})\bar{C}+\bar{A}(\overline{BC})$$

De esta manera obtenemos otros dos términos en la expresión a los que de nuevo podemos aplicar el teorema de DeMorgan. Estos términos son $\overline{AB}$ y $\overline{BC}$. Una última aplicación del teorema de DeMorgan nos proporciona el siguiente resultado:

$$(\overline{AB})\bar{C}+\bar{A}(\overline{BC}) = (\bar{A}+\bar{B})\bar{C}+\bar{A}(\bar{B}+\bar{C})$$

Aunque este resultado puede simplificarse aún más utilizando las leyes y reglas de Boole, los teoremas de DeMorgan ya no se pueden aplicar más.

Aplicación de los teoremas de DeMorgan

El siguiente procedimiento ilustra la aplicación de los teoremas de DeMorgan y del álgebra de Boole a la expresión:

$$\overline{\overline{A+B\bar{C}}+D(\overline{E+\bar{F}})}$$

Paso 1. Identificamos los términos a los que se pueden aplicar los teorema de DeMorgan y consideramos cada término como una única variable, por lo que establecemos $\overline{A+B\bar{C}} = X$ y $D(\overline{E+\bar{F}}) = Y$.

Paso 2. Dado que $\overline{X+Y} = \bar{X}\bar{Y}$.

$$\overline{(\overline{A+B\bar{C}})+(\overline{D(E+\bar{F})})} = (\overline{\overline{A+B\bar{C}}})(\overline{D(\overline{E+\bar{F}})})$$

Paso 3. Utilizamos la regla 9 $(\bar{\bar{A}} = A)$ para eliminar la bara doble sobre el término de la izquierda (esto no es parte del teorema de DeMorgan).

$$(\overline{\overline{A+B\bar{C}}})\overline{(D(\overline{E+\bar{F}}))} = (A+B\bar{C})(\overline{D(\overline{E+\bar{F}})})$$

Paso 4. Aplicando el teorema de DeMorgan al segundo término:

$$(A+B\bar{C})(\overline{D(\overline{E+\bar{F}})}) = (A+B\bar{C})(\bar{D}+(\overline{\overline{E+\bar{F}}}))$$

Paso 5. Empleamos la regla 9 $(\bar{\bar{A}} = A)$ para cancelar las barras dobles sobre la parte $E+\bar{F}$ del término.

$$(A+B\bar{C})(\bar{D}+(\overline{\overline{E+\bar{F}}})) = (A+B\bar{C})(\bar{D}+E+\bar{F})$$

Los siguientes tres ejemplos ilustrarán detalladamente cómo emplear los teoremas de DeMorgan.

EJEMPLO 4.5

Aplicar los teoremas de DeMorgan a cada una de las siguientes expresiones:

(a) $\overline{(A+B+C)D}$ **(b)** $\overline{ABC+DEF}$ **(c)** $\overline{A\bar{B}+\bar{C}D+EF}$

Solución **(a)** Sea $A + B + C = X$ y $D = Y$. La expresión $\overline{(A+B+C)D}$ es de la forma $\overline{XY} = \bar{X}+\bar{Y}$ y se puede escribir como sigue:

$$\overline{(A+B+C)D} = \overline{A+B+C}+\bar{D}$$

A continuación, aplicamos el teorema de DeMorgan al término $\overline{A+B+C}$

$$\overline{A+B+C}+\bar{D} = \bar{A}\bar{B}\bar{C}+\bar{D}$$

(b) Sea $ABC = X$ y $DEF = Y$. La expresión $\overline{ABC+DEF}$ es de la forma $\overline{X+Y} = \bar{X}\bar{Y}$ y podemos reescribirla de la forma:

$$\overline{ABC+DEF} = (\overline{ABC})(\overline{DEF})$$

A continuación, aplicamos el teorema de DeMorgan a cada uno de los términos $\overline{ABC}$ y $\overline{DEF}$.

$$(\overline{ABC})(\overline{DEF}) = (\bar{A}+\bar{B}+\bar{C})(\bar{D}+\bar{E}+\bar{F})$$

(c) Sean $A\bar{B} = X, \bar{C}D = Y$ y $EF = Z$. La expresión $\overline{A\bar{B}+\bar{C}D+EF}$ es de la forma $\overline{X+Y+Z} = \bar{X}\bar{Y}\bar{Z}$ y se puede reescribir como:

$$\overline{A\bar{B}+\bar{C}D+EF} = (\overline{A\bar{B}})(\overline{\bar{C}D})(\overline{EF})$$

A continuación, aplicamos el teorema de DeMorgan a cada uno de los términos $\overline{A\bar{B}}$, $\overline{\bar{C}D}$ y $\overline{EF}$.

$$(\overline{A\bar{B}})(\overline{\bar{C}D})(\overline{EF}) = (\bar{A}+B)(C+\bar{D})(\bar{E}+\bar{F})$$

Problema relacionado Aplicar los teoremas de DeMorgan a la expresión $\overline{\overline{ABC}+D+E}$.

EJEMPLO 4.6

Aplicar los teoremas de DeMorgan a cada una de las siguientes expresiones:

(a) $\overline{(\overline{A+B})+\overline{C}}$ **(b)** $\overline{(\overline{A}+B)+CD}$ **(c)** $\overline{(A+B)\overline{C}\overline{D}+E+\overline{F}}$

Solución

(a) $\overline{(\overline{A+B})+\overline{C}} = \overline{(\overline{A+B})}\,\overline{\overline{C}} = (A+B)C$

(b) $\overline{(\overline{A}+B)+CD} = \overline{(\overline{A}+B)}\,\overline{CD} = (\overline{\overline{A}}\,\overline{B})(\overline{C}+\overline{D}) = A\overline{B}(\overline{C}+\overline{D})$

(c) $\overline{(A+B)\overline{C}\overline{D}+E+\overline{F}} = (\overline{(A+B)\overline{C}\overline{D}})(\overline{E+\overline{F}}) = (\overline{A}\overline{B}+C+D)\overline{E}F$

Problema relacionado Aplicar los teoremas de DeMorgan a la expresión $\overline{\overline{A}B(C+\overline{D})+E}$.

EJEMPLO 4.7

La expresión booleana de una puerta OR-exclusiva es $A\overline{B}+\overline{A}B$. Tomando esto como punto de partida, desarrollar una expresión para una puerta NOR-exclusiva, utilizando los teoremas de DeMorgan y aquellas leyes o reglas que puedan aplicarse.

Solución En primer lugar se complementa la expresión OR-exclusiva y luego se aplican los teorema de DeMorgan del siguiente modo:

$$\overline{A\overline{B}+\overline{A}B} = (\overline{A\overline{B}})(\overline{\overline{A}B}) = (\overline{A}+\overline{\overline{B}})(\overline{\overline{A}}+\overline{B}) = (\overline{A}+B)(A+\overline{B})$$

A continuación se aplica la ley distributiva y la regla 8 $(A\cdot\overline{A}=0)$.

$$(\overline{A}+B)(A+\overline{B}) = \overline{A}A+\overline{A}\overline{B}+AB+B\overline{B} = \overline{A}\overline{B}+AB$$

La expresión resultante para una puerta XNOR es $\overline{A}\overline{B}+AB$. Observe que esta expresión es igual a 1 siempre que ambas variables sean 0 o 1.

Problema relacionado A partir de la expresión para una puerta NAND de 4 entradas, utilizar los teoremas de DeMorgan para desarrollar una expresión para una puerta negativa-OR de 4 entradas.

REVISIÓN DE LA SECCIÓN 4.3

1. Aplicar los teoremas de DeMorgan a las siguientes expresiones:

(a) $\overline{ABC}+(\overline{\overline{D}+E})$ **(b)** $\overline{(A+B)C}$ **(c)** $\overline{A+B+C}+\overline{\overline{D}E}$

4.4 ANÁLISIS BOOLEANO DE LOS CIRCUITOS LÓGICOS

El álgebra de Boole proporciona una manera concisa de expresar el funcionamiento de un circuito lógico formado por una combinación de puertas lógicas, de tal forma que la salida puede determinarse por la combinación de los valores de entrada.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Determinar las expresiones booleanas de una combinación de puertas. ■ Evaluar el funcionamiento lógico de un circuito a partir de su expresión booleana. ■ Construir una tabla de verdad.

Expresión booleana de un circuito lógico

▲ *Un circuito lógico se puede describir mediante una ecuación booleana.*

Para obtener la expresión booleana de un determinado circuito lógico, la manera de proceder consiste en comenzar con las entradas situadas más a la izquierda e ir avanzando hasta las líneas de salida, escribiendo la expresión para cada puerta. Para el circuito ejemplo de la Figura 4.16, su expresión booleana se determina de la siguiente manera:

1. La expresión de la puerta AND situada más a la izquierda cuyas entradas son *C* y *D* es *CD*.
2. La salida de la puerta AND situada más a la izquierda es una de las entradas de la puerta OR y *B* es su otra entrada. Por tanto, la expresión para la puerta OR es *B*+*CD*.
3. La salida de la puerta OR es una de las entradas de la puerta AND situada más a la derecha, siendo *A* su otra entrada. Por tanto, la expresión de esta puerta AND será *A*(*B*+*CD*), que es la expresión final de salida del circuito completo.

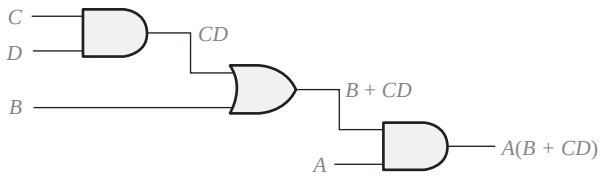


FIGURA 4.16 Circuito lógico que muestra el desarrollo de la expresión booleana para la salida.

Construcción de una tabla de verdad para un circuito lógico

▲ *Un circuito lógico puede describirse mediante una tabla de verdad.*

Una vez que se ha determinado la expresión booleana de un circuito dado, puede desarrollarse una tabla de verdad que represente la salida del circuito lógico para todos los valores posibles de las variables de entrada. El procedimiento requiere que se evalúe la expresión booleana para todas las posibles combinaciones de valores de las variables de entrada. En el caso del circuito de la Figura 4.16, existen cuatro variables de entrada (*A*, *B*, *C* y *D*) y, por tanto, hay dieciséis ($2^4 = 16$) posibles combinaciones de valores.

Evaluación de la expresión. Para evaluar la expresión $A(B + CD)$, en primer lugar hallamos los valores de las variables que hacen que la expresión sea igual a 1, utilizando las reglas de la suma y la multiplicación booleanas. En este caso, la expresión es igual a 1 sólo si $A = 1$ y $B + CD = 1$, ya que:

$$A(B + CD) = 1 \cdot 1 = 1$$

Ahora hay que determinar cuándo el término $B + CD$ es igual a 1. El término $B + CD = 1$ si $B = 1$ o $C = 1$ o si ambas variables son igual a 1, ya que:

$$B + CD = 1 + 0 = 1$$
$$B + CD = 0 + 1 = 1$$
$$B + CD = 1 + 1 = 1$$

El término $CD = 1$ sólo si $C = 1$ y $D = 1$.

Resumiendo, la expresión $A\ (B + CD) = 1$ cuando $A = 1$ y $B = 1$, independientemente de los valores de *C* y *D,* o cuando $A = 1$ y $C = 1$ o cuando $A = 1$ y $C = 1$ y $D = 1$, independientemente del valor de *B*. La expresión $A\ (B + CD) = 0$ para todas las restantes combinaciones de valores de las variables.

Representación de los resultados en una tabla de verdad. El primer paso consiste en enumerar las dieciséis combinaciones de unos y ceros de las variables de entrada en una secuencia binaria, como muestra la Tabla 4.5. A continuación, se pone un 1 en la columna de salida para las combinaciones de variables de entrada que se han determinado en la evaluación de la expresión. Finalmente, se escribe un 0 en la columna de salida para el resto de las combinaciones de las variables de entrada. Estos resultados se muestran en la Tabla 4.5.

Entradas				Salida
A	*B*	*C*	*D*	*A(B + CD)*
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

TABLA 4.5 Tabla de verdad del circuito lógico de la Figura 4.16.

REVISIÓN DE LA SECCIÓN 4.4

1. Reemplazar las puertas AND por puertas OR y la puerta OR por una puerta AND en la Figura 4.16, y determinar la expresión booleana de salida.
2. Elaborar la tabla de verdad del circuito de la cuestión 1.

4.5 SIMPLIFICACIÓN MEDIANTE EL ÁLGEBRA DE BOOLE

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

Al finalizar esta sección, el lector deberá ser capaz de:

■ Aplicar las leyes, reglas y teoremas del álgebra de Boole para simplificar cualquier expresión.

Una expresión booleana simplificada emplea el menor número posible de puertas en la implementación de una determinada expresión. Los Ejemplos 4.8 hasta 4.11 ilustran la simplificación booleana.

EJEMPLO 4.8

Simplificar la siguiente expresión utilizando técnicas del álgebra de Boole:

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

Solución El método que se sigue no es necesariamente el único método posible.

Paso 1. Aplicar la ley distributiva al segundo y tercer término del siguiente modo:

$$AB + AB + AC + BB + BC$$

Paso 2. Aplicar la regla 7 ($BB = B$) al cuarto término.

$$AB + AB + AC + B + BC$$

Paso 3. Aplicar la regla 5 ($AB + AB = AB$) a los dos primeros términos.

$$AB + AC + B + BC$$

Paso 4. Aplicar la regla 10 ($B + BC = B$) a los dos últimos términos.

$$AB + AC + B$$

Paso 5. Aplicar la regla 10 ($AB + B = B$) al primero y tercer término.

$$B + AC$$

En este punto, la expresión ya no puede seguir simplificándose. Según vaya adquiriendo experiencia en la aplicación del álgebra de Boole, podrá combinar muchos de los pasos individuales.

Problema relacionado Simplificar la expresión booleana $A\overline{B} + A(\overline{B+C}) + B(\overline{B+C})$.

▲ *La simplificación consiste en implementar una función con el menor número de puertas posible.*

La Figura 4.17 muestra cómo el proceso de simplificación del Ejemplo 4.8 ha reducido significativamente el número de puertas lógicas necesarias para implementar la expresión. En la parte (a) se puede ver que son necesarias cinco puertas para implementar dicha expresión en su forma original, mientras que sólo se requieren dos para hacerlo una vez simplificada, como se muestra en la parte (b). Es importante resaltar que estos dos circuitos de puertas son equivalentes, es decir, para cualquier combinación de valores en las entradas A, B y C, obtenemos siempre la misma salida en ambos circuitos.

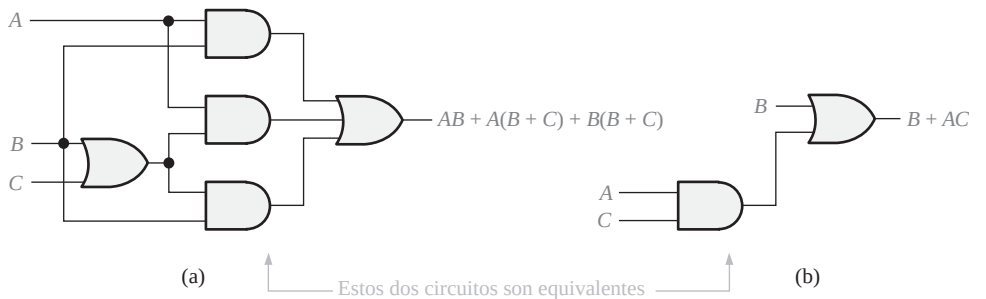


FIGURA 4.17 Circuitos de puertas para el Ejemplo 4.8.

EJEMPLO 4.9

Simplificar la siguiente expresión booleana:

$$[A\overline{B}(C + BD) + \overline{A}\overline{B}]C$$

Tenga en cuenta que los corchetes y paréntesis significan lo mismo: el término en su interior se multiplica (AND) por el término exterior.

Solución

Paso 1. Aplicar la ley distributiva a los términos entre corchetes.

$$(A\overline{B}C + A\overline{B}BD + \overline{A}\overline{B})C$$

Paso 2. Aplicar la regla 8 ($\overline{B}B = 0$) al segundo término entre paréntesis.

$$(A\overline{B}C + A \cdot 0 \cdot D + \overline{A}\overline{B})C$$

Paso 3. Aplicar la regla 3 ($A \cdot 0 \cdot D = 0$) al segundo término contenido dentro de los paréntesis.

$$(A\overline{B}C + 0 + \overline{A}\overline{B})C$$

Paso 4. Aplicar la regla 1 (quitar el 0) dentro del paréntesis

$$(A\overline{B}C + \overline{A}\overline{B})C$$

Paso 5. Aplicar la ley distributiva.

$$A\overline{B}CC + \overline{A}\overline{B}C$$

Paso 6. Aplicar la regla 7 ($CC = C$) al primer término.

$$A\overline{B}C + \overline{A}\overline{B}C$$

Paso 7. Sacar $\overline{B}C$ factor común.

$$\overline{B}C(A + \overline{A})$$

Paso 8. Aplicar la regla 6 ($A + \overline{A} = 1$).

$$\overline{B}C \cdot 1$$

Paso 9. Aplicar la regla 4 (quitar el 1).

$$\overline{B}C$$

Problema relacionado Simplificar la expresión booleana $[AB(C + \overline{BD}) + \overline{AB}]CD$.

EJEMPLO 4.10

Simplificar la siguiente expresión booleana:

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

Solución

Paso 1. Sacar factor común BC del primer y último término.

$$BC(\overline{A}+A)+A\overline{B}\overline{C}+\overline{A}\overline{B}\overline{C}+A\overline{B}C$$

Paso 2. Aplicar la regla 6 $(A+\overline{A}=1)$ al término entre paréntesis y sacar factor común $A\overline{B}$ del segundo y último término.

$$BC\cdot 1+A\overline{B}(\overline{C}+C)+\overline{A}\overline{B}\overline{C}$$

Paso 3. Aplicar la regla número 4 (quitar el 1) al primer término y la regla 6 $(\overline{C}+C=1)$ al término entre paréntesis.

$$BC+A\overline{B}\cdot 1+\overline{A}\overline{B}\overline{C}$$

Paso 4. Aplicar la regla 4 (quitar el 1) al segundo término.

$$BC+A\overline{B}+\overline{A}\overline{B}\overline{C}$$

Paso 5. Sacar $\overline{B}$ factor común al segundo y tercer término.

$$BC+\overline{B}(A+\overline{A}\overline{C})$$

Paso 6. Aplicar la regla 11 $(A+\overline{A}\overline{C}=A+\overline{C})$ al término entre paréntesis.

$$BC+\overline{B}(A+\overline{C})$$

Paso 7. Utilizar las leyes distributiva y conmutativa para obtener la siguiente expresión.

$$BC+A\overline{B}+\overline{B}\overline{C}$$

Problema relacionado Simplificar la expresión booleana $AB\overline{C}+\overline{A}\overline{B}C+\overline{A}BC+\overline{A}\overline{B}\overline{C}$.

EJEMPLO 4.11

Simplificar la siguiente expresión booleana:

$$\overline{AB+AC}+\overline{A}\overline{B}C$$

Solución

Paso 1. Aplicar el teorema de DeMorgan al primer término.

$$(\overline{AB})(\overline{AC})+\overline{A}\overline{B}C$$

Paso 2. Aplicar el teorema de DeMorgan a cada uno de los términos entre paréntesis.

$$(\overline{A}+\overline{B})(\overline{A}+\overline{C})+\overline{A}\overline{B}C$$

Paso 3. Aplicar la ley distributiva a los dos términos entre paréntesis.

$$\overline{A}\overline{A}+\overline{A}\overline{C}+\overline{A}\overline{B}+\overline{B}\overline{C}+\overline{A}\overline{B}C$$

Paso 4. Aplicar la regla número 7 $(\overline{A}\overline{A}=\overline{A})$ al primer término y la regla 10 $[\overline{A}\overline{B}+\overline{A}\overline{B}C=\overline{A}\overline{B}(1+C)=\overline{A}\overline{B}]$ a los términos tercero y último.

$$\overline{A}+\overline{A}\overline{C}+\overline{A}\overline{B}+\overline{B}\overline{C}$$

Paso 5. Aplicar la regla 10, $\bar{A} + \bar{A}\bar{C} = \bar{A}(1 + \bar{C}) = \bar{A}$, a los términos primero y segundo.

$$\bar{A} + \bar{A}\bar{B} + \bar{B}\bar{C}$$

Paso 6. Aplicar la regla 10 $[\bar{A} + \bar{A}\bar{B} = \bar{A}(1 + \bar{B}) = \bar{A}]$ a los términos primero y segundo.

$$\bar{A} + \bar{B}\bar{C}$$

Problema relacionado Simplificar la expresión booleana $\overline{AB} + \overline{AC} + \bar{A}\bar{B}\bar{C}$.

REVISIÓN DE LA SECCIÓN 4.5

1. Simplificar, si es posible, las siguientes expresiones booleanas:
   (a) $A + AB + A\bar{B}C$ **(b)** $(\bar{A} + B)C + ABC$ **(c)** $A\bar{B}C(BD + CDE) + A\bar{C}$
2. Implementar con las puertas lógicas apropiadas cada expresión de la cuestión anterior. Después, implementar la expresión simplificada y comparar el número de puertas empleado en cada caso.

4.6 FORMAS ESTÁNDAR DE LAS EXPRESIONES BOOLEANAS

Todas las expresiones booleanas, independientemente de su forma, pueden convertirse en cualquiera de las dos formas estándar: suma de productos o producto de sumas. La estandarización posibilita que la evaluación, simplificación e implementación de las expresiones booleanas sea mucho más sistemática y sencilla.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Identificar una expresión en forma de suma de productos. ■ Determinar el dominio de una expresión booleana. ■ Convertir cualquier suma de productos a su forma estándar. ■ Evaluar una expresión en forma de suma de productos según los valores binarios. ■ Identificar una expresión en forma de producto de sumas. ■ Convertir cualquier producto de sumas a su forma estándar. ■ Evaluar una expresión en forma de producto de sumas según los valores binarios. ■ Convertir expresiones de una a otra forma estándar.

Suma de productos

▲ *Una suma de productos puede implementarse con una puerta OR y dos o más puertas AND.*

En la Sección 4.1, se ha definido el término producto como un término que es el producto (multiplicación booleana) de literales (variables o sus complementos). Cuando dos o más productos se suman mediante la adición booleana, la expresión resultante se denomina ***suma de productos*** (SOP, *Sum Of Products*). Algunos ejemplos son:

$$AB + ABC$$
$$ABC + CDE + \bar{B}C\bar{D}$$
$$\bar{A}B + \bar{A}B\bar{C} + AC$$

Una suma de productos puede contener también términos de una única variable como en $A + \bar{A}\bar{B}C + BC\bar{D}$. Si volvemos a los ejemplos de simplificación de la sección anterior, puede observarse que cada término de la

expresión resultante era o un producto aislado o una suma de productos. En una expresión con formato de suma de productos, una barra no puede extenderse sobre más de una variable; sin embargo, más de una variable puede tener una barra encima. Por ejemplo, una suma de productos puede contener el término $\bar{A}\bar{B}\bar{C}$ pero no el término $\overline{ABC}$.

Dominio de una expresión booleana. El **dominio** de una expresión booleana es el conjunto de variables contenido en la expresión bien en su forma complementada o no complementada. Por ejemplo, el dominio de la expresión $\bar{A}B + A\bar{B}C$ es el conjunto de variables A, B, C y el dominio de la expresión $AB\bar{C} + C\bar{D}E + \bar{B}C\bar{D}$ es el conjunto de variables A, B, C, D, E.

Implementación AND/OR de una suma de productos. La implementación de una suma de productos simplemente requiere aplicar la operación OR a las salidas de dos o más puertas AND. Una operación AND da lugar a un producto, y la adición de dos o más productos se realiza mediante puertas OR. Por tanto, una expresión suma de productos puede implementarse mediante un circuito lógico AND-OR en el que las salidas de las puertas AND, cuyo número es igual al de productos que contenga la expresión, son las entradas de una puerta OR, como se muestra en la Figura 4.18 para la expresión $AB + BCD + AC$. La salida X de la puerta OR es igual a la suma de productos.

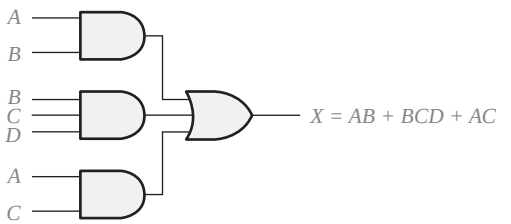


FIGURA 4.18 Implementación de la suma de productos $AB + BCD + AC$.

Implementación NAND/NAND de una suma de productos. Se pueden emplear puertas NAND para implementar una expresión suma de productos. Utilizando sólo puertas NAND se puede obtener una función AND/OR, como se ilustra en la Figura 4.19. El primer nivel de puertas NAND alimenta las entradas de una puerta NAND que actúa como una puerta negativa-OR. Las inversiones de la puerta NAND y las puertas negativa-OR se cancelan y dan como resultado un circuito AND/OR.

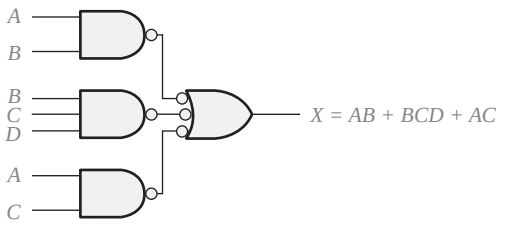


FIGURA 4.19 Esta implementación NAND/NAND es equivalente a la implementación AND/OR de la Figura 4.18.

Conversión de una expresión general a formato suma de productos

Cualquier expresión lógica puede ser transformada a una expresión suma de productos aplicando el álgebra de Boole. Por ejemplo, la expresión $A(B+CD)$ puede convertirse en una suma de productos aplicando la ley distributiva:

$$A\,(B + CD) = AB + ACD$$

EJEMPLO 4.12

Convertir cada una de las siguientes expersiones booleanas a su forma suma de productos:

(a) $AB + B(CD + EF)$ **(b)** $(A + B)(B + C + D)$ **(c)** $\overline{\overline{(A+B)}+C}$

Solución

(a) $AB + B(CD + EF) = AB + BCD + BEF$

(b) $(A + B)(B + C + D) = AB + AC + AD + BB + BC + BD$

(c) $\overline{\overline{(A+B)}+C} = (\overline{\overline{A+B}})\overline{C} = (A+B)\overline{C} = A\overline{C} + B\overline{C}$

Problema relacionado Convertir $\overline{A}B\overline{C} + (A + \overline{B})(B + \overline{C} + A\overline{B})$ a la forma suma de productos.

Forma estándar de la suma de productos

Hasta ahora, hemos estado viendo sumas de productos en las que algunos de los términos no contenían todas las variables del dominio de la expresión. Por ejemplo, la expresión $\overline{A}B\overline{C} + A\overline{B}D + \overline{A}B\overline{C}D$ tiene un dominio formado por las variables A, B, C y D. Sin embargo, el conjunto completo de variables del dominio no está representado en los dos primeros términos de la expresión; es decir, faltan D o $\overline{D}$ en el primer término y C o $\overline{C}$ en el segundo.

Una *suma de productos estándar* es aquella en la que *todas* las variables del dominio aparecen en cada uno de los términos de la expresión. Por ejemplo, $A\overline{B}CD + \overline{A}\overline{B}C\overline{D} + AB\overline{C}\overline{D}$ es una expresión suma de productos estándar. La expresión suma de productos estándar es importante en la construcción de tablas de verdad, lo que se estudiará en la Sección 4.7 y en el método de simplificación de los mapas de Karnaugh, que se aborda en la Sección 4.8. Cualquier expresión suma de productos no estándar (que denominaremos simplemente suma de productos) puede convertirse al formato estándar utilizando el álgebra de Boole.

Conversión de una suma de productos a su forma estándar. Cada término producto de una suma de productos que no contenga todas las variables del dominio puede ampliase a su forma estándar de manera que incluya todas las variables del dominio y sus complementos. Como se muestra en los siguientes pasos, una suma de productos no estándar se convierte a su forma estándar utilizando la regla 6 $(A + \overline{A} = 1)$ de la Tabla 4.1: la suma de una variable y su complemento es igual a 1.

Paso 1. Multiplicar cada término producto no estándar por un término formado por la suma de la variable que falta y su complemento. Con esto se obtienen dos términos producto. Como se sabe, se puede multiplicar por 1 cualquier expresión sin que se altere su valor.

Paso 2. Repetir el paso 1 hasta que todos los términos de la expresión contengan todas las variables o sus complementos del dominio. Al convertir cada producto a su forma estándar, el número de términos producto se duplica por cada variable que falta, como muestra el Ejemplo 4.13.

EJEMPLO 4.13

Convertir la siguiente expresión booleana al formato suma de productos estándar:

$$A\overline{B}C + \overline{A}\overline{B} + AB\overline{C}D$$

Solución El dominio de esta suma de productos es A, B, C, D. Considerando cada término por separado, se comprueba que al primer término, $A\overline{B}C$, le falta la variable D o $\overline{D}$, por lo que multiplicamos dicho término por $D + \overline{D}$ como sigue:

$$A\bar{B}C = A\bar{B}C(D + \bar{D}) = A\bar{B}CD + A\bar{B}C\bar{D}$$

En este caso se obtienen dos productos estándar.

En el segundo término $\bar{A}\bar{B}$ faltan las variables C o $\bar{C}$ y D o $\bar{D}$, por lo que lo multiplicamos por $C + \bar{C}$

$$\bar{A}\bar{B} = \bar{A}\bar{B}(C + \bar{C}) = \bar{A}\bar{B}C + \bar{A}\bar{B}\bar{C}$$

Los dos términos que hemos obtenido carecen de la variable D o $\bar{D}$, por lo que multiplicamos ambos términos por $D + \bar{D}$

$$\bar{A}\bar{B} = \bar{A}\bar{B}C + \bar{A}\bar{B}\bar{C} = \bar{A}\bar{B}C(D + \bar{D}) + \bar{A}\bar{B}\bar{C}(D + \bar{D})$$
$$= \bar{A}\bar{B}CD + \bar{A}\bar{B}C\bar{D} + \bar{A}\bar{B}\bar{C}D + \bar{A}\bar{B}\bar{C}\bar{D}$$

En este caso, el resultado con cuatro productos estándar.

El tercer término, $A\bar{B}\bar{C}D$, ya está en forma estándar. La suma de productos estándar completa que obtenemos finalmente es:

$$A\bar{B}C + \bar{A}\bar{B} + A\bar{B}\bar{C}D = A\bar{B}CD + A\bar{B}C\bar{D} + \bar{A}\bar{B}CD + \bar{A}\bar{B}C\bar{D} + \bar{A}\bar{B}\bar{C}D + \bar{A}\bar{B}\bar{C}\bar{D} + A\bar{B}\bar{C}D$$

Problema relacionado Convertir la expresión $W\bar{X}Y + \bar{X}Y\bar{Z} + WX\bar{Y}$ a su forma de suma de productos estándar.

Representación binaria de un término producto estándar. Un término producto estándar es igual a 1 sólo para una combinación de los valores de las variables. Por ejemplo, el término producto $A\bar{B}C\bar{D}$ es igual a 1 cuando A = 1, B =0, C =1, D = 0, como se muestra a continuación y es igual a 0 para todas las restantes combinaciones de valores de las variables.

$$A\bar{B}C\bar{D} = 1 \cdot \bar{0} \cdot 1 \cdot \bar{0} = 1 \cdot 1 \cdot 1 \cdot 1 = 1$$

En este caso, el término producto tiene un valor binario de 1010 (diez en decimal).

Recuerde que un término producto se implementa mediante una puerta AND cuya salida es 1 si y sólo si cada una de sus entradas está a 1. Para generar el complemento de las variables cuando es necesario se utilizan inversores.

Una expresión suma de productos es igual a 1 si y sólo si uno o más de los términos productos que forman la expresión es igual a 1.

EJEMPLO 4.14

Determinar los valores binarios para los que la siguiente suma de productos estándar sea igual a 1:

$$ABCD + A\bar{B}\bar{C}D + \bar{A}\bar{B}\bar{C}\bar{D}$$

Solución El término $ABCD$ es igual a 1 cuando A = 1, B = 1, C = 1 y D =1.

$$ABCD = 1 \cdot 1 \cdot 1 \cdot 1 = 1$$

El término $A\bar{B}\bar{C}D$ es igual a 1 cuando A = 1, B = 0, C = 0 y D =1.

$$A\bar{B}\bar{C}D = 1 \cdot \bar{0} \cdot \bar{0} \cdot 1 = 1 \cdot 1 \cdot 1 \cdot 1 = 1$$

El término $\bar{A}\bar{B}\bar{C}\bar{D}$ es igual a 1 cuando $A = 0$, $B = 0$, $C = 0$ y $D = 0$.

$$\bar{A}\bar{B}\bar{C}\bar{D} = \bar{0} \cdot \bar{0} \cdot \bar{0} \cdot \bar{0} = 1 \cdot 1 \cdot 1 \cdot 1 = 1$$

La suma de productos es igual a 1 sólo cuando cualquiera de los tres términos o todos son igual a 1.

Problema relacionado Determinar los valores binarios para los que la siguiente expresión suma de productos es igual a 1:

$$\bar{X}YZ + X\bar{Y}Z + XY\bar{Z} + \bar{X}\bar{Y}\bar{Z} + XYZ$$

¿Es una suma de productos estándar?

Producto de sumas

En la Sección 4.1 se ha definido el término suma como un término formado por la suma (adición booleana) de literales (variables o sus complementos). Cuando dos o más términos suma se multiplican, la expresión resultante es un ***producto de sumas*** (POS, *Product Of Sums*). Algunos ejemplos son:

$$(\bar{A} + B)(A + \bar{B} + C)$$

$$(\bar{A} + \bar{B} + \bar{C})(C + \bar{D} + E)(\bar{B} + C + D)$$

$$(A + B)(A + \bar{B} + C)(\bar{A} + C)$$

Un producto de sumas puede contener términos con una única variable como en $\bar{A}(A + \bar{B} + C)(\bar{B} + \bar{C} + D)$. En una expresión producto de sumas, una barra no puede extenderse nunca sobre más de una variable, aunque más de una variable puede tener una barra encima. Por ejemplo, un producto de sumas puede contener el término $\bar{A} + \bar{B} + \bar{C}$ pero no el $\overline{A + B + C}$.

Implementación de un producto de sumas. La implementación de un producto de sumas requiere simplemente la aplicación de la operación AND a las salidas de dos o más puertas OR. Un sumando se origina mediante la operación OR y el producto de varios términos suma se realiza por medio de la operación AND. Por tanto, un producto de sumas puede implementarse a partir de puertas lógicas OR (cuyo número será igual al de sumandos de la expresión) cuyas salidas se conectan a las entradas de una puerta AND, como muestra la Figura 4.20 para la expresión $(A + B)(B + C + D)(A + C)$. La salida X de la puerta AND es igual al producto de sumas.

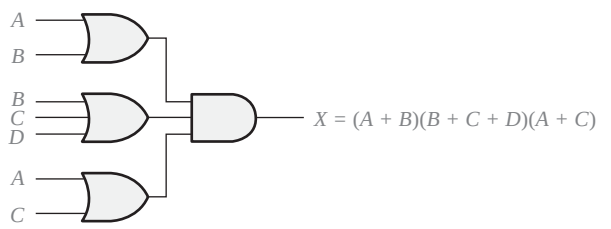


FIGURA 4.20 Implementación del producto de sumas $(A + B)(B + C + D)(A + C)$.

Forma estándar del producto de sumas

Hasta ahora, se han tratado expresiones producto de sumas en las que algunos de los términos no contenían todas las variables del dominio de la expresión. Por ejemplo, la expresión:

$$(A+\overline{B}+C)(A+B+\overline{D})(A+\overline{B}+\overline{C}+D)$$

tiene un dominio formado por las variables *A*, *B*, *C* y *D*. Observe que el conjunto completo de variables del dominio no está representado en los dos primeros términos de la expresión; es decir, faltan *D* o $\overline{D}$ en el primer término y *C* o $\overline{C}$ en el segundo término.

Un producto de sumas estándar es aquel en el que *todas* las variables del dominio o sus complementos aparecen en cada uno de los términos de la expresión. Por ejemplo,

$$(\overline{A}+\overline{B}+\overline{C}+\overline{D})(A+\overline{B}+C+D)(A+B+\overline{C}+D)$$

es un producto de sumas estándar. Cualquier producto de sumas no estándar (que denominaremos simplemente producto de sumas) puede convertirse a su forma estándar mediante el álgebra de Boole.

Conversión de un producto de sumas a su forma estándar. Cada término suma de una expresión producto de sumas que no contenga todas las variables del dominio puede extenderse para obtener su formato estándar incluyendo todas las variables del dominio y sus complementos. Como se establece en los pasos siguientes, un producto de sumas no estándar se convierte a su formato estándar utilizando la regla booleana número 8 $(A\cdot\overline{A}=0)$ de la Tabla 4.1 que establece que una variable multiplicada por su complemento es igual a 0.

Paso 1. Añadir a cada término suma no estándar un término formado por la variable que falta y su complemento. Esto da lugar a la aparición de dos términos suma. Como ya sabemos, se puede sumar 0 a cualquier cosa sin que se altere su valor.

Paso 2. Aplicar la regla 12 de la Tabla 4.1: $A+BC=(A+B)(A+C)$.

Paso 3. Repetir el paso 1 hasta que todos los términos suma resultantes contengan todas las variables del dominio en su forma complementada o no complementada.

EJEMPLO 4.15

Convertir la siguiente expresión booleana a formato producto de sumas:

$$(A+\overline{B}+C)(\overline{B}+C+\overline{D})(A+\overline{B}+\overline{C}+D)$$

Solución

El dominio de este producto de sumas es *A*, *B*, *C*, *D*. Vamos a considerar término por término. En el primero $A+\overline{B}+C$, falta la variable *D* o $\overline{D}$, por lo que añadimos $D\overline{D}$ y aplicamos la regla 12 del siguiente modo:

$$A+\overline{B}+C=A+\overline{B}+C+D\overline{D}=(A+\overline{B}+C+D)(A+\overline{B}+C+\overline{D})$$

En el segundo término, $\overline{B}+C+\overline{D}$ falta la variable *A* o $\overline{A}$, por lo que añadimos $A\overline{A}$ y aplicamos la regla 12 como sigue:

$$\overline{B}+C+\overline{D}=\overline{B}+C+\overline{D}+A\overline{A}=(A+\overline{B}+C+\overline{D})(\overline{A}+\overline{B}+C+\overline{D})$$

El tercer término, $A+\overline{B}+\overline{C}+D$, ya está en formato estándar. El producto de sumas estándar de la expresión original es:

$$(A+\overline{B}+C)(\overline{B}+C+\overline{D})(A+\overline{B}+\overline{C}+D)=$$

$$(A+\overline{B}+C+D)(A+\overline{B}+C+\overline{D})(A+\overline{B}+C+\overline{D})(\overline{A}+\overline{B}+C+\overline{D})(A+\overline{B}+\overline{C}+D)$$

Problema relacionado Convertir la expresión $(A+\overline{B})(B+C)$ a su forma producto de sumas estándar.

Representación binaria de un término suma estándar. Un término suma estándar es igual a 0 sólo para una combinación de los valores de las variables. Por ejemplo, el término suma $A+\overline{B}+C+\overline{D}$ es igual a 1 cuando $A = 0$, $B = 1$, $C = 0$ y $D = 1$, como se muestra a continuación y es igual a 1 para todas las restantes combinaciones de valores de las variables.

$$A+\overline{B}+C+\overline{D}=0+\overline{1}+0+\overline{1}=0+0+0+0=0$$

En este caso, el término suma tiene un valor binario de 0101 (cinco en decimal). Recuerde que un término suma se implementa mediante una puerta OR cuya salida es 0 sólo si cada una de sus entradas está a 0. Para generar el complemento de las variables cuando es necesario se utilizan inversores.

Una expresión producto de sumas es igual a 0 si y sólo si uno o más de los términos suma que forman la expresión es igual a 0.

EJEMPLO 4.16

Determinar los valores binarios de las variables para los que la expresión producto de sumas estándar siguiente es igual a 0:

$$(A+B+C+D)(A+\overline{B}+\overline{C}+D)(\overline{A}+\overline{B}+\overline{C}+\overline{D})$$

Solución El término $A + B + C + D$ es igual a 0 cuando $A = 0$, $B = 0$, $C = 0$ y $D = 0$.

$$A + B + C + D = 0 + 0 + 0 + 0 = 0$$

El término $A+\overline{B}+\overline{C}+D$ es igual a 0 cuando $A = 0$, $B = 1$, $C = 1$ y $D = 0$:

$$A+\overline{B}+\overline{C}+D=0+\overline{1}+\overline{1}+0=0+0+0+0=0$$

El término $\overline{A}+\overline{B}+\overline{C}+\overline{D}$ es igual a 0 cuando $A = 1$, $B = 1$, $C = 1$ y $D = 1$.

$$\overline{A}+\overline{B}+\overline{C}+\overline{D}=\overline{1}+\overline{1}+\overline{1}+\overline{1}=0+0+0+0=0$$

La expresión producto de sumas es igual a 0 cuando cualquiera de los tres términos suma es igual a 0.

Problema relacionado Determinar los valores binarios para los que la siguiente expresión producto de sumas es igual a 0:

$$(X+\overline{Y}+Z)(\overline{X}+Y+Z)(X+Y+\overline{Z})(\overline{X}+\overline{Y}+\overline{Z})(X+\overline{Y}+\overline{Z})$$

¿Es un producto de sumas estándar?

Conversión de una suma de productos estándar en un producto de sumas estándar

Los valores binarios de los términos producto en una suma de productos estándar dada no aparecen en su producto de sumas estándar equivalente. Asimismo, los valores binarios que no están representados en una suma de productos sí aparecen en el producto de sumas equivalente. Por tanto, para pasar de la suma de productos estándar al producto de sumas estándar hay que realizar los siguientes pasos:

Paso 1. Evaluar cada término producto de la expresión suma de productos. Es decir, determinar los números binarios que representan estos términos.

Paso 2. Determinar todos los números binarios no incluidos al realizar la evaluación del paso 1.

Paso 3. Escribir los términos suma equivalente para cada valor binario del paso 2 y expresarlos en forma producto de sumas.

Utilizando un procedimiento similar, se puede pasar de un producto de sumas a una suma de productos.

EJEMPLO 4.17

Convertir la siguiente suma de productos en su expresión equivalente como producto de sumas:

$$\bar{A}\bar{B}\bar{C} + \bar{A}B\bar{C} + \bar{A}BC + A\bar{B}C + ABC$$

Solución El resultado de la evaluación es la siguiente

$$000 + 010 + 011 + 101 + 111$$

Puesto que son tres las variables que conforman el dominio de esta expresión, existe un total de ocho (2^3) posibles combinaciones. La suma de productos contiene cinco de estas combinaciones, luego la expresión producto de sumas debe contener las otras tres que son 001, 100 y 110. Recuerde que estos son los valores binarios que hacen que cada término suma sea igual a cero. La expresión producto de sumas equivalente es la siguiente:

$$(A + B + \bar{C})(\bar{A} + B + C)(\bar{A} + \bar{B} + C)$$

Problema relacionado Sustituyendo los valores binarios en cada término, verificar que las expresiones suma de productos y producto de sumas de este ejemplo son equivalentes.

REVISIÓN DE LA SECCIÓN 4.6

1. Determinar si cada una de las expresiones siguientes es una suma de productos o un producto de sumas. Indicar si se trata de una forma estándar.

(a) $AB + \bar{A}BD + \bar{A}C\bar{D}$

(b) $(A + \bar{B} + C)(A + B + \bar{C})$

(c) $\bar{A}BC + AB\bar{C}$

(d) $A(A + \bar{C})(A + B)$

2. Convertir las sumas de productos de la cuestión 1 a la forma estándar.

3. Convertir los productos de sumas de la cuestión 1 a la forma estándar

4.7 EXPRESIONES BOOLEANAS Y TABLAS DE VERDAD

Todas las expresiones booleanas pueden convertirse fácilmente en tablas de verdad utilizando los valores binarios de cada término de la expresión. La tabla de verdad es una forma muy común, en un formato muy conciso, de expresar el funcionamiento lógico de un circuito. Además, las expresiones suma de productos y producto de sumas pueden determinarse mediante las tablas de verdad. Las tablas de verdad pueden encontrarse en las hojas de especificaciones y en otros textos relativos al funcionamiento de los circuitos y sistemas digitales.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Pasar una expresión suma de productos estándar a su tabla de verdad. ■ Pasar un producto de sumas estándar a su tabla de verdad. ■ Obtener una expresión estándar a partir de su tabla de verdad. ■ Interpretar correctamente los datos de una tabla de verdad.

Conversión de una suma de productos a tabla de verdad

Como se ha establecido en la Sección 4.6, una suma de productos es igual a 1 sólo si y sólo si al menos uno de los productos es igual a 1. Una tabla de verdad es sencillamente la lista de las posibles combinaciones de valores de las variables de entrada y sus correspondientes valores de salida (1 o 0). Para una expresión cuyo dominio es de dos variables, existen cuatro combinaciones distintas de estas variables ($2^2 = 4$). Para una expresión cuyo dominio tiene tres variables, existen ocho ($2^3 = 8$) combinaciones posibles de dichas variables. Para una expresión con un dominio de cuatro variables, existen dieciséis combinaciones diferentes de dichas variables ($2^4 = 16$), etc.

El primer paso para construir una tabla de verdad consiste en enumerar todas las posibles combinaciones de los valores de las variables de la expresión. A continuación, hay que pasar la suma de productos a su formato estándar, si no lo está ya. Por último, se escribe un 1 en la columna de salida (X) para cada valor binario que hace que la suma de productos estándar sea 1, y se escribe un 0 para los restantes valores. Este procedimiento se ilustra en el Ejemplo 4.18.

EJEMPLO 4.18

Desarrollar una tabla de verdad para la expresión suma de productos estándar $\bar{A}\bar{B}C + A\bar{B}\bar{C} + ABC$.

Solución Existen tres variables en el dominio, por lo que hay ocho posibles combinaciones de valores binarios de las variables, como se muestra en las tres columnas

Entradas			Salida	
A	***B***	***C***	***X***	**Término producto**
0	0	0	0	
0	0	1	1	$\bar{A}\bar{B}C$
0	1	0	0	
0	1	1	0	
1	0	0	1	$A\bar{B}\bar{C}$
1	0	1	0	
1	1	0	0	
1	1	1	1	ABC

TABLA 4.6

de la izquierda de la Tabla 4.6. Los valores binarios que hacen que los términos producto de la expresión sean igual a 1 son $\bar{A}\bar{B}C$: 001; $A\bar{B}\bar{C}$: 100 y ABC: 111. Para cada uno de estos valores binarios, se escribe un 1 en la columna de salida, como se indica en la tabla. Para cada una de las restantes combinaciones, se escribe un 0 en la columna de salida.

Problema relacionado Crear una tabla de verdad para la expresión suma de productos estándar $\bar{A}B\bar{C} + A\bar{B}C$.

Conversión de un producto de sumas a tabla de verdad

Recuerde que un producto de sumas es igual a 0 sólo si y sólo si al menos uno de los términos suma es igual a 0. Para construir la tabla de verdad de un producto de sumas, basta con enumerar todas las posibles combinaciones de valores binarios de las variables del mismo modo que se hace para una suma de productos. A continuación, hay que pasar el producto de sumas a su formato estándar, si no lo está ya. Por último, se escribe un 0 en la columna de salida (X) para cada valor binario que hace que la suma de productos estándar sea 0, y se escribe un 1 para los restantes valores binarios. Este procedimiento se ilustra en el Ejemplo 4.19.

EJEMPLO 4.19

Desarrollar una tabla de verdad para la expresión producto de sumas estándar siguiente:

$$(A+B+C)(A+\bar{B}+C)(A+\bar{B}+\bar{C})(\bar{A}+B+\bar{C})(\bar{A}+\bar{B}+C)$$

Solución Existen tres variables en el dominio, por lo que hay ocho posibles combinaciones de valores binarios de las variables, como se muestra en las tres columnas de la izquierda de la Tabla 4.7. Los valores binarios que hacen que los términos suma de la expresión sean igual a 0 son $A + B + C$: 000; $A+\bar{B}+C$: 010; $A+\bar{B}+\bar{C}$: 011; $\bar{A}+B+\bar{C}$: 101 y $\bar{A}+\bar{B}+C$: 110. .Para cada uno de estos valores binarios, se escribe un 0 en la columna de salida, como se indica en la tabla. Para cada una de las restantes combinaciones, se escribe un 1 en la columna de salida.

Entradas			Salida	
A	***B***	***C***	***X***	**Término suma**
0	0	0	0	$(A + B + C)$
0	0	1	1	
0	1	0	0	$(A+\bar{B}+C)$
0	1	1	0	$(A+\bar{B}+\bar{C})$
1	0	0	1	
1	0	1	0	$(\bar{A}+B+\bar{C})$
1	1	0	0	$(\bar{A}+\bar{B}+C)$
1	1	1	1	

TABLA 4.7

Observe que la tabla de verdad de este ejemplo es la misma que la del Ejemplo 4.18. Esto significa que la suma de productos del ejemplo anterior y el producto de sumas de este ejemplo son equivalentes.

Problema relacionado Crear una tabla de verdad para la expresión producto de sumas estándar: $(A+\overline{B}+C)(A+B+\overline{C})(\overline{A}+\overline{B}+\overline{C})$

Determinación de las expresiones estándar a partir de una tabla de verdad

Para determinar la expresión de la suma de productos estándar representada por una tabla de verdad se enumeran todos los valores de las variables de entrada para los que la salida es 1. Cada valor binario se convierte en el correspondientes término producto, reemplazando cada 1 por la variable y cada 0 por la variable complementada. Por ejemplo, el valor binario 1010 se transforma en un término producto de la manera siguiente:

$$1010 \rightarrow A\overline{B}C\overline{D}$$

Si sustituimos podemos comprobar que el término producto es 1:

$$A\overline{B}C\overline{D} = 1 \cdot \overline{0} \cdot 1 \cdot \overline{0} = 1 \cdot 1 \cdot 1 \cdot 1 = 1$$

Para determinar el producto de sumas estándar representado por una tabla de verdad se enumeran todos los valores binarios para los que la salida es 0. A continuación, se convierte cada valor binario en el correspondiente término suma, reemplazando cada 1 por la variable complementada y cada 0 por la variable. Por ejemplo, el número binario 1001 se pasa a término suma de la manera siguiente:

$$1001 \rightarrow \overline{A}+B+C+\overline{D}$$

Si sustituimos podemos comprobar que el término suma es 0:

$$\overline{A}+B+C+\overline{D} = \overline{1}+0+0+\overline{1} = 0+0+0+0 = 0$$

EJEMPLO 4.20

A partir de la tabla de verdad de la Tabla 4.8, determinar la expresión suma de productos estándar y la expresión producto de sumas estándar equivalente.

Entradas			Salida
A	***B***	***C***	***X***
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

TABLA 4.8

Solución

En la columna de salida hay cuatro 1s y los correspondientes valores binarios son 011, 100, 110 y 111. Convertir estos valores binarios a términos producto como sigue:

$$011 \rightarrow \bar{A}BC$$
$$100 \rightarrow A\bar{B}\bar{C}$$
$$110 \rightarrow AB\bar{C}$$
$$111 \rightarrow ABC$$

La expresión suma de productos estándar resultante para la salida X es:

$$X = \bar{A}BC + A\bar{B}\bar{C} + AB\bar{C} + ABC$$

Para el producto de sumas, la salida es 0 para los valores binarios 000, 001, 010 y 101. Estos valores binarios se convierten en términos suma como sigue:

$$000 \rightarrow A + B + C$$
$$001 \rightarrow A + B + \bar{C}$$
$$010 \rightarrow A + \bar{B} + C$$
$$101 \rightarrow A + B + \bar{C}$$

La expresión producto de sumas estándar resultantes para la salida X es:

$$X = (A + B + C)(A + B + \bar{C})(A + \bar{B} + C)(A + B + \bar{C})$$

Problema relacionado

Sustituyendo los valores binarios, demostrar que las expresiones suma de productos y productos de sumas obtenidas en este ejemplo son equivalentes; es decir, para cualquier número binario que se elija ambas deben ser 1 o 0, dependiendo del valor binario.

REVISIÓN DE LA SECCIÓN 4.7

1. Si una determinada expresión booleana tiene un dominio de cinco variables, ¿cuántos valores binarios tendrá su tabla de verdad?
2. En una determinada tabla de verdad, la salida es 1 para el valor binario 0110. Convertir este valor binario en el correspondiente término producto usando las variables W, X, Y y Z.
3. En una determinada tabla de verdad, la salida es 0 para el valor binario 1100. convertir este valor binario en el correspondiente término suma usando las variables W, X, Y y Z.

4.8 MAPAS DE KARNAUGH

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

Al finalizar esta sección, el lector deberá ser capaz de:

■ Construir un mapa de Karnaugh de tres o cuatro variables. ■ Determinar el valor binario de cada celda en un mapa de Karnaugh. ■ Determinar el término producto estándar representado en cada celda de un mapa de Karnaugh. ■ Explicar la adyacencia de celdas e identificar celdas adyacentes.

▲ *El propósito de un mapa de Karnaugh es simplificar una expresión booleana.*

Un ***mapa de Karnaugh*** es similar a una tabla de verdad, ya que muestra todos los valores posibles de las variables de entrada y la salida resultante para cada valor. En lugar de organizar en filas y columnas como una tabla de verdad, el mapa de Karnaugh es una matriz de **celdas** en la que cada celda representa un valor binario de las variables de entrada. Las celdas se organizan de manera que la simplificación de una determinada expresión consiste en agrupar adecuadamente las celdas. Los mapas de Karnaugh se pueden utilizar para expresiones de dos, tres, cuatro y cinco variables, pero nos ocuparemos únicamente de los casos de tres y cuatro variables para ilustrar los principios. La Sección 4.11 aborda el caso de cinco variables utilizando un mapa de Karnaugh de 32 celdas. Existe otro método, que queda fuera del propósito de este libro, denominado método de Quine-McClusky, que puede emplearse para un número mayor de variables.

El número de celdas de un mapa de Karnaugh es igual al número total de posibles combinaciones de las variables de entrada, al igual que el número de filas de una tabla de verdad. Para tres variables, el número de celdas necesarias es de $2^3 = 8$. Para cuatro variables, el número de celdas es de $2^4 = 16$.

Mapa de Karnaugh de tres variables

El mapa de Karnaugh de tres variables es una matriz de ocho celdas, como se muestra en la Figura 4.21(a). En este caso, *A*, *B* y *C* se emplean para denominar a las variables, aunque podían haberse usado cualesquiera otras letras. Los valores binarios de *A* y *B* se encuentran en el lado izquierdo (observe la secuencia) y los valores de *C* se colocan en la parte superior. El valor de una determinada celda es el valor binario de *A* y *B*, en la parte izquierda de la misma fila combinado con el valor de *C* en la parte superior de la misma columna. Por ejemplo, la celda de la esquina superior izquierda tiene un valor binario de 000 y la celda inferior derecha tiene un valor binario de 101. La Figura 4.21(b) muestra los términos producto estándar representados por cada celda del mapa de Karnaugh.

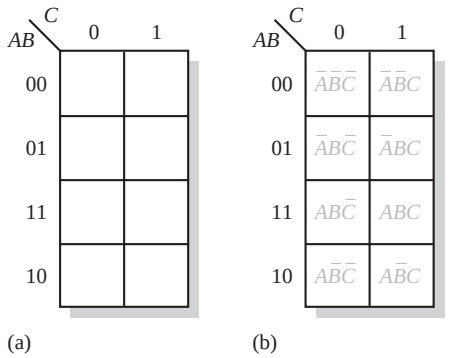


FIGURA 4.21 Mapa de Karnaugh de tres variables que muestra los términos producto.

Mapa de Karnaugh de cuatro variables

El mapa de Karnaugh de cuatro variables es una matriz de dieciséis celdas, como se muestra en la Figura 4.22(a). Los valores binarios de *A* y *B* se encuentran en el lado izquierdo y los valores de *C* y *D* se colocan

en la parte superior. El valor de una determinada celda es el valor binario de A y B, en la parte izquierda de la misma fila combinado con los valores binarios de C y D en la parte superior de la misma columna. Por ejemplo, la celda de la esquina superior derecha tiene un valor binario de 0010 y la celda inferior derecha tiene un valor binario de 1010. En la Figura 4.22(b) se indican los términos producto estándar representados por cada celda del mapa de Karnaugh de cuatro variables.

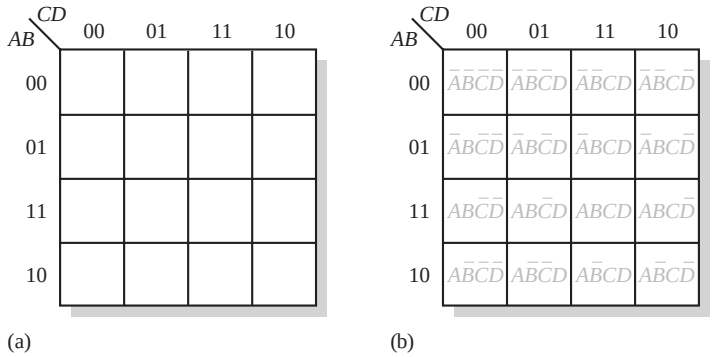


FIGURA 4.22 Mapa de Karnaugh de cuatro variables.

Adyacencia de celdas

▲ *Las celdas que sólo difieren en una variable son adyacentes.*

▲ *Las celdas con valores que difieren en más de una variable no son adyacentes.*

Las celdas de un mapa de Karnaugh se disponen de manera que sólo cambia una única variable entre celdas adyacentes. La **adyacencia** se define por un cambio de una única variable. Las celdas que difieren en una única variable son adyacentes. Por ejemplo, en el mapa de tres variables, la celda 010 es adyacente a las celdas 000, 011 y 110. La celda 010 no es adyacente a la celda 001, ni a la celda 111, ni a la celda 100 ni a la celda 101.

Físicamente, cada celda es adyacente a las celdas que están situadas inmediatas a ella por cualquiera de sus cuatro lados. Un celda no es adyacente a aquellas celdas que tocan diagonalmente alguna de sus esquinas. Además, las celdas de la fila superior son adyacentes a las de la fila inferior y las celdas de la columna izquierda son adyacentes a las situadas en la columna de la derecha. Esto se denomina adyacencia

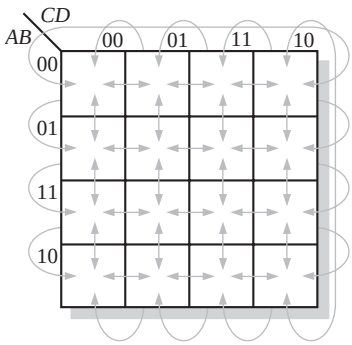


FIGURA 4.23 Celdas adyacentes en un mapa de Karnaugh son aquellas que sólo difieren en una variable. Las flechas apuntan a la celdas adyacentes.

cíclica, ya que podemos pensar que el mapa de Karnaugh se dobla de forma que se toquen los extremos superior e inferior como si fuera un cilindro o los extremos de la derecha e izquierda para formar la misma figura. La Figura 4.23 ilustra la adyacencia de celdas en un mapa de cuatro variables, aunque se aplican las mismas reglas de adyacencia a los mapas de Karnaugh con cualquier número de celdas.

REVISIÓN DE LA SECCIÓN 4.8

1. En un mapa de Karnaugh de 3 variables, ¿cuál es el valor binario de cada una de las siguientes celdas?:
   (a) esquina superior izquierda **(b)** esquina inferior derecha
   (c) esquina inferior izquierda **(d)** esquina superior derecha
2. ¿Cuál es el término producto estándar de cada celda de la cuestión 1 para las variables *X*, *Y* y *Z*?
3. Repetir la cuestión 1 para un mapa de 4 variables.
4. Repetir la cuestión 2 para un mapa de 4 variables utilizando las variables *W*, *X*, *Y* y *Z*.

4.9 MINIMIZACIÓN DE UNA SUMA DE PRODUCTOS MEDIANTE EL MAPA DE KARNAUGH

Como se ha establecido en la sección anterior, el mapa de Karnaugh se utiliza para reducir expresiones booleanas a su expresión mínima. Una expresión suma de productos minimizada está formada por el mínimo número de términos producto posibles con el mínimo número de variables por término. Generalmente, una expresión suma de productos minimizada puede implementarse mediante un número de puertas menor que su expresión estándar, lo cual constituye la finalidad del proceso de simplificación.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Representar una expresión suma de productos en un mapa de Karnaugh. ■ Combinar los unos del mapa en grupos máximos. ■ Determinar el término producto mínimo para cada grupo del mapa. ■ Combinar los términos producto mínimo para formar una expresión suma de productos mínima. ■ Convertir una tabla de verdad en un mapa de Karnaugh para simplificar la expresión representada. ■ Utilizar las condiciones "indiferentes" en un mapa de Karnaugh.

Mapa de Karnaugh de una suma de productos estándar

Por cada término de la expresión suma de productos, se coloca un 1 en el mapa de Kanaugh en la celda correspondiente al valor del producto. Se coloca un 1 en la celda correspondiente al valor de un término producto. Por ejemplo, para el término $A\bar{B}C$, se escribiría un 1 en la celda 101 de un mapa de Karnaugh de tres variables.

Cuando una expresión suma de productos se ha reflejado por completo en el mapa de Karnaugh, en dicho mapa habrá tantos 1s como términos producto tenga la suma de productos estándar. Las celdas que no contienen un 1 son aquellas para las que la expresión es igual a 0. Normalmente, cuando se trabaja con una expresión suma de productos, los 0s no se incluyen en el mapa. Los siguientes pasos y la Figura 4.24 muestra cómo completar los mapas de Karnaugh.

Paso 1. Determinar el valor binario de cada término producto de la suma de productos estándar. Tras un poco de práctica, podrá realizar la evaluación de términos mentalmente.

Paso 2. A medida que evaluamos cada término, colocamos un 1 en el mapa de Karnaugh en la celda que tiene el mismo valor que dicho término producto.

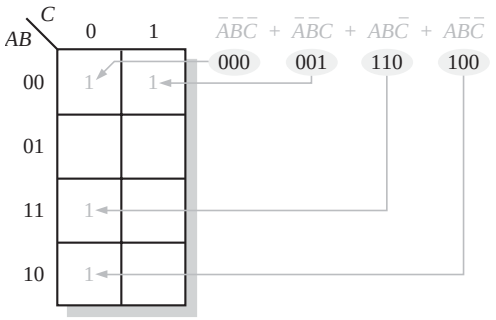


FIGURA 4.24 Ejemplo de transformación a mapa de Karnaugh de una suma de productos estándar.

EJEMPLO 4.21

Transformar la siguiente suma de productos estándar en un mapa de Karnaugh:

$$\bar{A}\bar{B}C + \bar{A}B\bar{C} + AB\bar{C} + ABC$$

Solución La expresión se evalúa como se muestra a continuación. Se escribe un 1 en el mapa de Karnaugh de 3 variables de la Figura 4.24 por cada producto estándar de la expresión.

$$\bar{A}\bar{B}C + \bar{A}B\bar{C} + AB\bar{C} + ABC$$
$$001 \quad 010 \quad 110 \quad 111$$

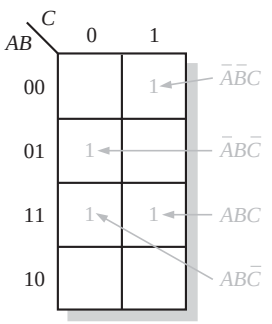


FIGURA 4.25

Problema relacionado Transformar la expresión estándar de la suma de productos $\bar{A}BC + A\bar{B}C + A\bar{B}\bar{C}$ en un mapa de Karnaugh.

EJEMPLO 4.22

Transformar la siguiente suma de productos estándar en un mapa de Karnaugh:

$$\bar{A}\bar{B}CD + \bar{A}B\bar{C}\bar{D} + AB\bar{C}D + ABCD + AB\bar{C}\bar{D} + \bar{A}\bar{B}\bar{C}D + A\bar{B}C\bar{D}$$

Solución La expresión se evalúa como se muestra a continuación. Se coloca un 1 en el mapa de Karnaugh de la Figura 4.26 por cada producto estándar de la expresión.

$$\bar{A}\bar{B}CD + \bar{A}B\bar{C}\bar{D} + AB\bar{C}D + ABCD + AB\bar{C}\bar{D} + \bar{A}\bar{B}\bar{C}D + A\bar{B}C\bar{D}$$
$$0\,0\,1\,1 \quad 0\,1\,0\,0 \quad 1\,1\,0\,1 \quad 1\,1\,1\,1 \quad 1\,1\,0\,0 \quad 0\,0\,0\,1 \quad 1\,0\,1\,0$$

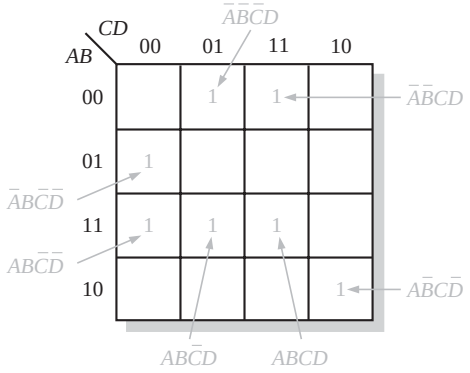


FIGURA 4.26

Problema relacionado Transformar la siguiente expresión estándar suma de productos en un mapa de Karnaugh.

$$\bar{A}BC\bar{D} + ABC\bar{D} + AB\bar{C}\bar{D} + ABCD$$

Mapa de Karnaugh de una suma de productos no estándar

Antes de poder utilizar un mapa de Karnaugh, las expresiones booleanas deben estar en su forma estándar. Si una expresión no lo está, se pasará al formato estándar mediante el procedimiento descrito en la Sección 4.6 o mediante desarrollo numérico. Dado que, en cualquier caso, las expresiones tienen que evaluarse antes de pasarlas al mapa de Karnaugh, el desarrollo numérico es quizá el método más eficaz.

Desarrollo numérico de un producto no estándar. Recuerde que a un término en forma no estándar le faltan una o más variables en su expresión. Por ejemplo, supongamos que uno de los productos de una determinada suma de productos de 3 variables es $A\bar{B}$. Este término se puede desarrollar numéricamente para obtener una expresión estándar de la manera siguiente. En primer lugar, se escribe el valor binario de las dos variables y le añadimos un 0 que corresponde a la variable que falta $\bar{C}$: 100. A continuación, escribimos el valor binario de las dos variables y añadimos un 1 para la variable que falta C: 101. Los dos números binarios resultantes son los valores de los términos de la suma de productos estándar $A\bar{B}\bar{C}$ y $A\bar{B}C$.

Veamos otro ejemplo, supongamos que uno de los términos producto de una expresión de 3 variables es B (recuerde que una variable única se considera como un término producto en una expresión suma de productos). Este término puede expandirse numéricamente a su forma estándar de la siguiente manera: se escribe el valor binario de la variable; a continuación, se añaden todos los posibles valores de las variables que faltan A y C del siguiente modo:

B
010
011
110
111

Los cuatro números binarios resultantes son los valores correspondientes a los términos de la suma de productos estándar $\bar{A}B\bar{C}, \bar{A}BC, AB\bar{C}$ y ABC.

EJEMPLO 4.23

Transformar la siguiente expresión suma de productos en un mapa de Karnaugh: $\bar{A} + A\bar{B} + AB\bar{C}$.

Solución Obviamente, la suma de productos no está en formato estándar, ya que cada término no contiene las tres variables. En el primer término faltan dos variables, en el segundo falta una variable y el tercero sí es un término estándar. En primer lugar, desarrollamos los términos numéricamente de la siguiente manera:

$\bar{A}$	$+A\bar{B}$	$+AB\bar{C}$
000	100	110
001	101	
010		
011		

Cada uno de los valores binarios resultantes se traslada al mapa, colocando un 1 en la celda apropiada del mapa de Karnaugh de 3 variables de la Figura 4.27.

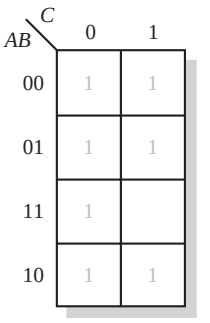


FIGURA 4.27

Problema relacionado Transformar la expresión suma de productos $BC + \bar{A}\bar{C}$ en un mapa de Karnaugh.

EJEMPLO 4.24

Transformar la siguiente expresión suma de productos en un mapa de Karnaugh:

$$\bar{B}\bar{C} + A\bar{B} + AB\bar{C} + A\bar{B}C\bar{D} + \bar{A}\bar{B}\bar{C}D + A\bar{B}CD$$

Solución

Obviamente, la suma de productos no está en formato estándar, ya que cada término no contiene las cuatro variables. En los términos primero y segundo faltan dos variables, en el tercer término falta una variable y el resto de los términos sí son estándar. En primer lugar, desarrollamos los términos numéricamente para incluir las variables que faltan de la siguiente manera:

$\bar{B}\bar{C}$	$A\bar{B}$	$+AB\bar{C}$	$+A\bar{B}C\bar{D}$	$+\bar{A}\bar{B}\bar{C}D$	$+A\bar{B}CD$
0000	1000	1100	1010	0001	1011
0001	1001	1101			
1000	1010				
1001	1011				

Cada uno de los valores binarios resultantes se traslada al mapa, colocando un 1 en la celda apropiada del mapa de Karnaugh de 4 variables de la Figura 4.28. Observe que algunos de los valores de la expresión desarrollada son redundantes.

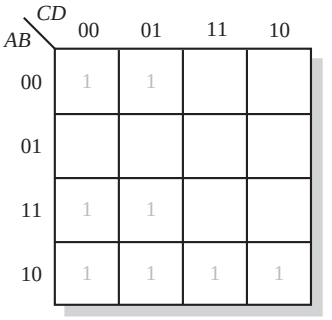


FIGURA 4.28

Problema relacionado Transformar la expresión $A + \bar{C}D + AC\bar{D} + \bar{A}BC\bar{D}$ en un mapa de Karnaugh.

Simplificación de una suma de productos mediante el mapa de Karnaugh

El proceso que genera una expresión que contiene el menor número posible de términos con el mínimo número de variables posibles se denomina ***minimización***. Después de haber obtenido el mapa de Karnaugh de una suma de productos, la expresión suma de productos mínima se obtiene agrupando los 1s y determinando la expresión suma de productos mínima a partir del mapa.

Agrupación de unos. Podemos agrupar los unos del mapa de Karnaugh de acuerdo con las reglas siguientes, rodeando las celdas adyacentes que contengan unos. La finalidad es maximizar el tamaño de los grupos y minimizar el número de estos grupos.

1. Un grupo tiene que contener 1, 2, 4, 8 ó 16 celdas, valores que se corresponden con las potencias de 2. En el caso de un mapa de Karnaugh de 3 variables, el grupo máximo puede contener $2^3 = 8$ celdas.
2. Cada celda de un grupo tiene que ser adyacente a una o más celdas del mismo grupo, pero no todas las celdas del grupo tienen que ser adyacentes entre sí.
3. Incluir siempre en cada grupo el mayor número posible de 1s de acuerdo a la regla número 1.
4. Cada 1 del mapa tiene que estar incluido en al menos un grupo. Los 1s que ya pertenezcan a un grupo pueden estar incluidos en otro, siempre que los grupos que se solapen contengan 1s no comunes.

EJEMPLO 4.25

Agrupar los 1s en cada uno de los mapas de Karnaugh de la Figura 4.29.

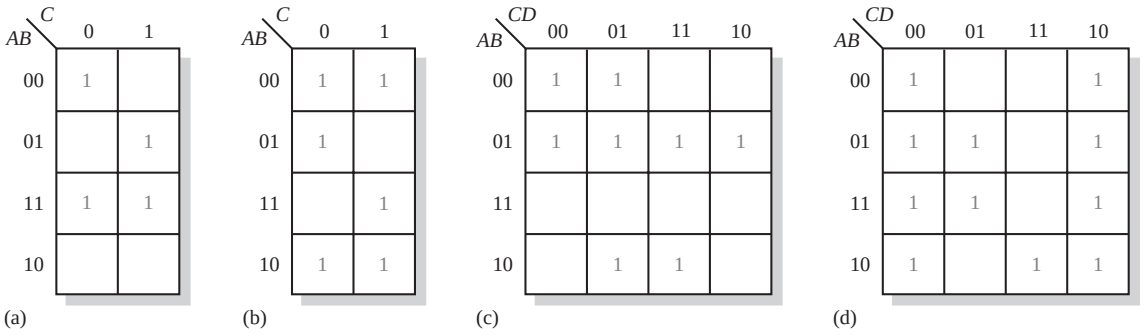


FIGURA 4.29

Solución En la Figura 4.30 se muestran los grupos. En algunos casos, puede existir más de una forma de agrupar los 1s para formar grupos máximos.

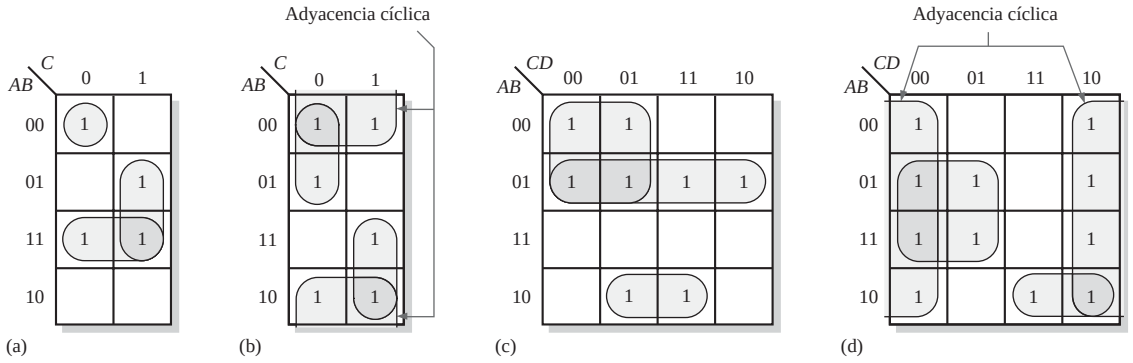


FIGURA 4.30

Problema relacionado Determinar si existen otras formas de agrupar los 1s en la Figura 4.30, para obtener un número mínimo de grupos máximos.

Determinación de la expresión suma de productos mínima a partir del mapa. Cuando todos los 1s que representan los términos productos estándar de una expresión se han trasladado al mapa y se han agrupado adecuadamente, comienza el proceso de obtención de la suma de productos mínima. Para encontrar los términos mínimos y la expresión suma de productos mínima se aplican las siguientes reglas:

1. Agrupar las celdas que contienen 1s. Cada grupo de celdas que contiene 1s da lugar a un término producto compuesto por todas las variables que aparecen en el grupo en sólo una forma (no complementada o complementada). Las variables que aparecen complementadas y sin complementar dentro del mismo grupo se eliminan. A éstas se les denomina *variables contradictorias*.
2. Determinar la operación producto mínima para cada grupo.
   (a) Para un mapa de 3 variables:
   (1) Un grupo formado por 1 celda da lugar a un término producto de 3 variables.
   (2) Un grupo formado por 2 celdas da lugar a un término producto de 2 variables.
   (3) Un grupo formado por 4 celdas da lugar a un término de 1 variable.
   (4) Un grupo formado por 8 celdas indica que la expresión vale 1.
   (b) Para un mapa de 4 variables:
   (1) Un grupo formado por 1 celda da lugar a un término producto de 4 variables.
   (2) Un grupo formado por 2 celdas da lugar a un término producto de 3 variables.
   (3) Un grupo formado por 4 celdas da lugar a un término producto de 2 variables.
   (4) Un grupo formado por 8 celdas da lugar a un término de 1 variable.
   (5) Un grupo formado por 16 celdas indica que la expresión vale 1.
3. Cuando se han obtenido todos los términos producto mínimos a partir del mapa de Karnaugh, se suman para obtener la expresión suma de productos mínima.

EJEMPLO 4.26

Determinar los productos para el mapa de Karnaugh de la Figura 4.31 y escribir la expresión suma de productos mínima resultante.

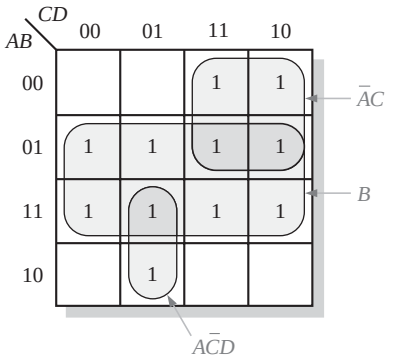


FIGURA 4.31

Solución Se eliminan las variables que aparecen complementadas y no complementadas en un mismo grupo. En la Figura 4.31, el producto para el grupo de 8 celdas es B, ya que las celdas de dicho grupo contienen las variables A y $\bar{A}$, C y $\bar{C}$, y D

y $\overline{D}$, que se eliminan. El grupo de 4 celdas contiene las variables $B, \overline{B}, D$ y $\overline{D}$, quedando las variables $\overline{A}$ y C, que forman el término producto $\overline{A}C$. El grupo de 2 celdas contiene B y $\overline{B}$, quedando las variables A, $\overline{C}$ y D que forman el término producto $A\overline{C}D$. Observe cómo se utiliza el solapamiento para maximizar el tamaño de los grupos. La suma de productos mínima resultantes es la suma de estos términos producto:

$$B + \overline{A}C + A\overline{C}D$$

Problema relacionado En el mapa de Karnaugh de la Figura 4.31, añadir un 1 a la celda inferior derecha (1010) y determinar la expresión suma de productos resultante.

EJEMPLO 4.27

Determinar los productos para cada uno de los mapas de Karnaugh de la Figura 4.32 y escribir las correspondientes expresiones suma de productos mínima resultante.

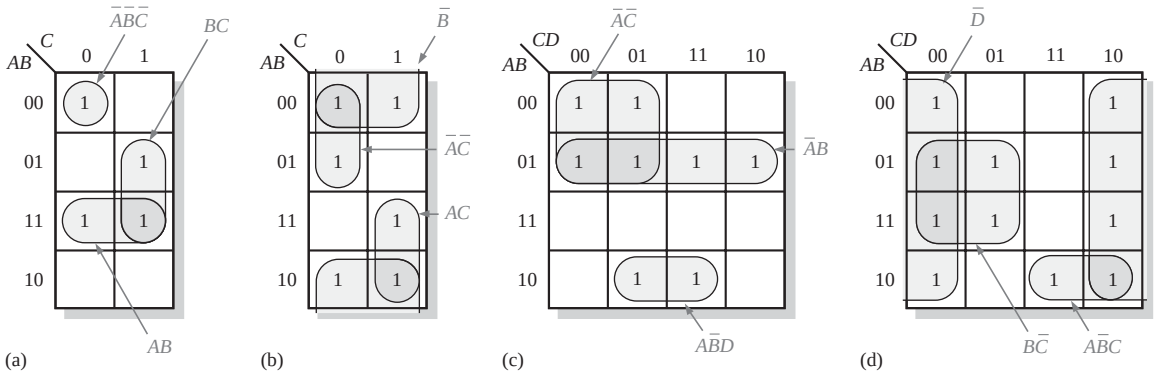


FIGURA 4.32

Solución En la Figura 4.32 se muestran los productos mínimos resultantes para cada grupo. La expresión suma de productos mínima para cada uno de los mapas de Karnaugh de la figura son:

(a) $AB + BC + \overline{A}\overline{B}\overline{C}$ **(b)** $\overline{B} + \overline{A}\overline{C} + AC$

(c) $\overline{A}B + \overline{A}\overline{C} + A\overline{B}D$ **(d)** $\overline{D} + A\overline{B}C + B\overline{C}$

Problema relacionado En el mapa de Karnaugh de la Figura 4.32(d), añadir un 1 a la celda 0111 y determinar la expresión suma de productos resultante.

EJEMPLO 4.28

Utilizar un mapa de Karnaugh para minimizar la siguiente expresión suma de productos estándar:

$$A\bar{B}C + \bar{A}BC + \bar{A}\bar{B}C + \bar{A}\bar{B}\bar{C} + A\bar{B}\bar{C}$$

Solución

Los valores binarios de la expresión son:

$$101 + 011 + 011 + 000 + 100$$

La suma de productos estándar se pasa al mapa y las celdas se agrupan como se muestra en la Figura 4.33.

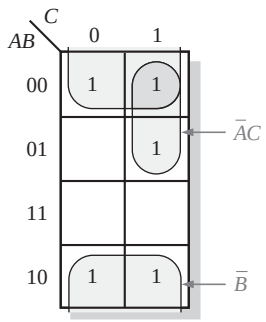


FIGURA 4.33

Observe que el grupo de 4 celdas en los extremos del mapa que incluye las filas superior e inferior de 1s. El 1 restante se incluye en otro grupo superpuesto de dos celdas. El grupo de los cuatro 1s da lugar a un término de una sola variable, $\bar{B}$. Esto se deduce del hecho de que, dentro del grupo, $\bar{B}$ es la única variable que no cambia de celda a celda. El grupo de los dos 1s da lugar al producto de dos variables $\bar{A}C$. Este término se determina observando que, dentro de este grupo, $\bar{A}$ y C no cambian de una celda a la siguiente. Se ha indicado el término producto correspondiente a cada grupo. La expresión suma de productos mínima resultante es:

$$\bar{B} + \bar{A}C$$

Tenga presente que esta expresión mínima es equivalente a la expresión estándar original.

Problema relacionado

Utilizando un mapa de Karnaugh, simplificar la siguiente expresión suma de productos estándar:

$$X\bar{Y}Z + XY\bar{Z} + \bar{X}YZ + \bar{X}Y\bar{Z} + X\bar{Y}\bar{Z} + XYZ$$

EJEMPLO 4.29

Utilizar un mapa de Karnaugh para minimizar la siguiente expresión suma de productos:

$$\bar{B}\bar{C}\bar{D} + \bar{A}B\bar{C}\bar{D} + AB\bar{C}\bar{D} + \bar{A}\bar{B}CD + A\bar{B}CD + \bar{A}\bar{B}C\bar{D} + \bar{A}BC\bar{D} + ABC\bar{D} + A\bar{B}C\bar{D}$$

Solución

El primer término $\bar{B}\bar{C}\bar{D}$ tiene que desarrollarse en los términos $A\bar{B}\bar{C}\bar{D}$ y $\bar{A}\bar{B}\bar{C}\bar{D}$ para obtener la suma de productos estándar, que a continuación se trasladará a un mapa, donde se agrupan las celdas como se muestra en la Figura 4.34.

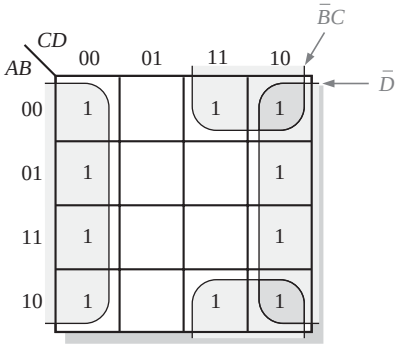


FIGURA 4.34

Observe que ambos grupos tienen adyacencia "cíclica" de celdas. Se puede formar el grupo de ocho celdas, ya que las dos columnas exteriores son adyacentes. El grupo de cuatro celdas se forma tomando los dos restantes 1s, puesto que las celdas superior e inferior son adyacentes. Se indica el término producto para cada grupo y la expresión suma de productos mínima resultante es:

$$\bar{D} + \bar{B}C$$

Tenga presente que esta expresión mínima es equivalente a la expresión estándar original.

Problema relacionado Mediante un mapa de Karnaugh simplifique la siguiente expresión suma de productos:

$$\bar{W}\bar{X}\bar{Y}\,\bar{Z} + W\bar{X}YZ + W\bar{X}\bar{Y}Z + \bar{W}YZ + W\bar{X}\bar{Y}\,\bar{Z}$$

Obtención directa del mapa de Karnaugh a partir de la tabla de verdad

Hemos visto cómo las expresiones booleanas se transforman en mapas de Karnaugh. Ahora aprenderá cómo pasar de una tabla de verdad a un mapa de Karnaugh. Recuerde que una tabla de verdad proporciona la salida de una expresión booleana para todas las posibles combinaciones de las variables de entrada. En la Figura 4.35 se facilita un ejemplo de expresión booleana junto con su tabla de verdad. Observe que la salida X es 1 para cuatro distintas combinaciones de las variables de entrada. Los 1s de la columna de salida de la tabla de verdad se trasladan directamente al mapa de Karnaugh, a las celdas correspondientes a los valores asociados de las combinaciones de variables de entrada, como muestra la Figura 4.35. En esta figura puede ver que tanto la expresión booleana, la tabla de verdad como el mapa de Karnaugh son sólo distintas maneras de representar una función lógica.

Condiciones indiferentes

Algunas veces se producen situaciones en las que algunas combinaciones de las variables de entrada no están permitidas. Por ejemplo, recuerde que en el código BCD, visto en el Capítulo 2, existían seis combinaciones no válidas: 1010, 1011, 1100, 1101, 1110 y 1111. Dado que estos estados no permitidos no ocurren nunca en una aplicación que emplee el código BCD, pueden considerarse como términos ***indiferentes*** con respecto a su efecto en la salida. Esto significa que a estos términos se les puede asignar tanto un 1 como un 0 en la salida; realmente no son importantes dado que nunca van a generarse.

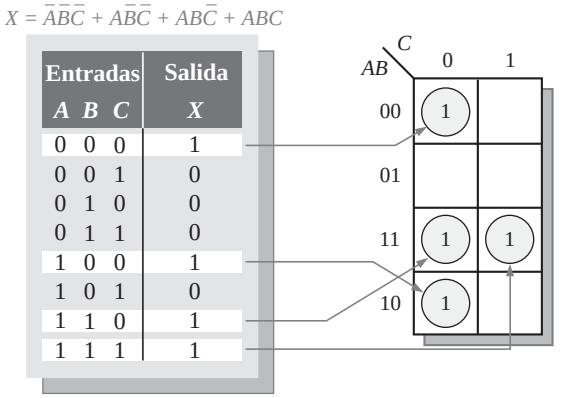
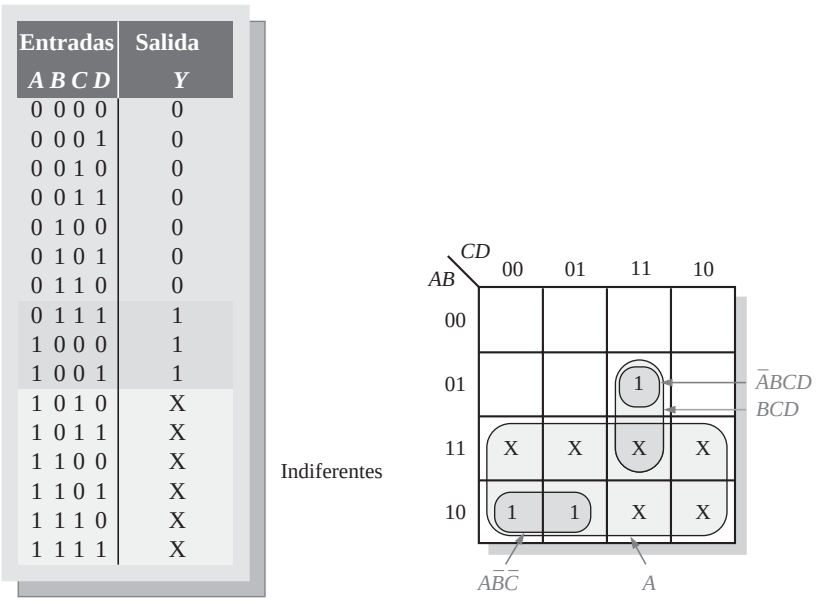


FIGURA 4.35 Ejemplo de obtención directa de un mapa de Karnaugh a partir de una tabla de verdad.

Los términos "indiferentes" pueden utilizarse para aprovechar mejor el método del mapa de Karnaugh. La Figura 4.36 muestra que, para cada término indiferente, se escribe una X en la celda. Cuando se agrupan los 1s, las X pueden ser consideradas también como 1s para agrandar los grupos, o como 0s si no obtenemos ninguna ventaja. Cuanto mayor sea el grupo, más sencillo será el término resultante.

La tabla de verdad de la Figura 4.36(a) describe una función lógica que tiene sólo la salida igual a 1 cuando el código BCD correspondiente al 7, 8 o 9 está presente en las entradas. Si las condiciones "indiferentes" se emplean como 1s, la expresión resultante para la función es $A + BCD$, como se indica en la parte (b) de la figura. Si las condiciones "indiferentes" no se establecen como 1s, la expresión resultante es $A\bar{B}\bar{C} + \bar{A}BCD$, por lo que puede concluirse que los términos indiferentes pueden aprovecharse para obtener una expresión más sencilla.



(a) Tabla de verdad

(b) Sin condiciones indiferentes $Y = A\bar{B}\bar{C} + \bar{A}BCD$
Con condiciones indiferentes $Y = A + BCD$

FIGURA 4.36 Ejemplo de la utilización de las condiciones "indiferentes" para simplificar una expresión.

REVISIÓN DE LA SECCIÓN 4.9

1. Explicar los mapas de Karnaugh de 3 y 4 variables.
2. Agrupar los 1s y escribir la expresión suma de productos simplificada para el mapa de Karnaugh de la Figura 4.25.
3. Escribir la expresión estándar de la suma de productos original de cada uno de los mapas de Karnaugh de la Figura 4.32.

4.10 MINIMIZACIÓN DE UN PRODUCTO DE SUMAS MEDIANTE EL MAPA DE KARNAUGH

En la sección anterior estudiamos la minimización de una expresión suma de productos mediante los mapas de Karnaugh. En esta sección, nos vamos a centrar en las expresiones producto de sumas. Los métodos son muy similares, excepto que ahora se trata de productos de sumas, en los que los 0s representan los términos suma estándar y se colocan en el mapa de Karnaugh en lugar de los 1s.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Transformar un producto de sumas estándar en un mapa de Karnaugh. ■ Combinar los 0s del mapa para formar grupos máximos. ■ Determinar el término suma mínimo para cada grupo del mapa. ■ Combinar los términos suma mínimos para formar el producto de sumas mínimo. ■ Utilizar el mapa de Karnaugh para convertir productos de sumas en sumas de productos.

Conversión de una expresión producto de sumas estándar a mapa de Karnaugh

Para un producto de sumas en forma estándar, se introduce un 0 en el mapa de Karnaugh por cada término suma de la expresión. Cada 0 se sitúa en la celda correspondiente al valor de un término suma. Por ejemplo, para la suma $A+\overline{B}+C$, se escribe un 0 en la celda 010 del mapa de Karnaugh de 3 variables.

Cuando un producto de sumas se ha trasladado por completo al mapa, habrá tantos 0s en el mapa de Karnaugh, como términos suma en la expresión del producto de sumas estándar. Las celdas que no contienen un 0 son aquellas para las que la expresión vale 1. Generalmente, cuando se trabaja con productos de sumas, los 1s no se escriben. Los siguientes pasos junto con la Figura 4.37 ilustran este proceso.

Paso 1. Determinar el valor binario de cada término suma del producto de sumas estándar. Éste es el valor binario que hace que dicho término sea igual a 0.

Paso 2. Cada vez que se evalúa un término suma, se introduce un 0 en la correspondiente celda del mapa de Karnaugh.

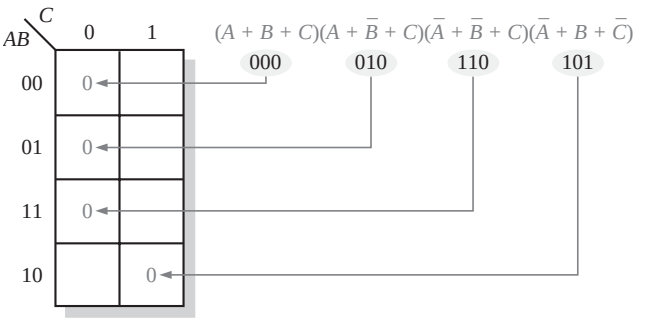


FIGURA 4.37 Ejemplo de obtención del mapa de Karnaugh de un producto de sumas estándar.

EJEMPLO 4.30

Transformar la siguiente expresión suma de productos estándar en un mapa de Karnaugh:

$$(\overline{A}+\overline{B}+C+D)(\overline{A}+B+\overline{C}+\overline{D})(A+B+\overline{C}+D)(\overline{A}+\overline{B}+\overline{C}+\overline{D})(A+B+\overline{C}+\overline{D})$$

Solución La expresión se evalúa como se indica a continuación y se coloca un 0 en el mapa de Karnaugh de 4 variables de la Figura 4.38 por cada término suma estándar de la expresión.

$$(\overline{A}+\overline{B}+C+D)(\overline{A}+B+\overline{C}+\overline{D})(A+B+\overline{C}+D)(\overline{A}+\overline{B}+\overline{C}+\overline{D})(A+B+\overline{C}+\overline{D})$$
$$1100 \qquad 1011 \qquad 0010 \qquad 1111 \qquad 0011$$

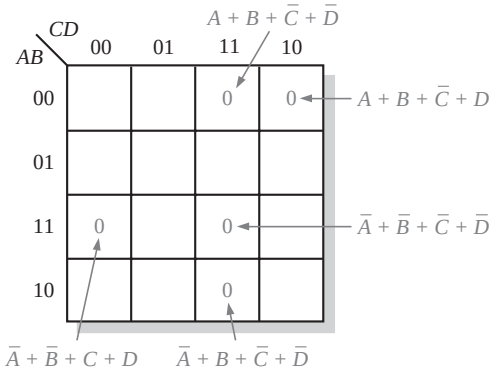


FIGURA 4.38

Problema relacionado Transformar la siguiente expresión suma de productos estándar en un mapa de Karnaugh.

$$(A+\overline{B}+\overline{C}+D)(A+B+C+\overline{D})(A+B+C+D)(\overline{A}+B+\overline{C}+D)$$

Simplificación mediante el mapa de Karnaugh de expresiones producto de sumas

El proceso de minimización de un producto de sumas es básicamente el mismo que para una expresión suma de productos, excepto que ahora hay que agrupar los ceros para generar el mínimo número de términos suma, en lugar de los 1s para obtener el número mínimo de términos producto. Las reglas para agrupar los 0s son las mismas que para agrupar los 1s, y son las que se han estudiado en la Sección 4.9.

EJEMPLO 4.31

Utilizar un mapa de Karnaugh para minimizar la siguiente expresión producto de sumas estándar:

$$(A+B+C)(A+B+\overline{C})(A+\overline{B}+C)(A+\overline{B}+\overline{C})(\overline{A}+\overline{B}+C)$$

Deducir también la expresión suma de productos equivalente.

Solución

Las combinaciones de valores binarios de la expresión son

$$(0 + 0 + 0)\ (0 + 0 + 1)\ (0 + 1 + 0)\ (0 + 1 + 1)\ (1 + 1 + 0)$$

La expresión de la suma de productos estándar se traslada al mapa de Karnaugh y las celdas se agrupan como se muestra en la Figura 4.39

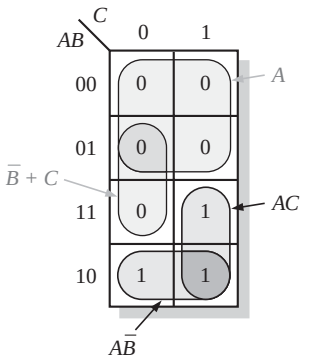


FIGURA 4.39

Observe que el 0 de la celda 110 se incluye en un grupo de dos celdas, utilizando el 0 del grupo de cuatro celdas. El término suma para cada grupo se muestra en la figura y la expresión suma de productos mínima resultante es:

$$A(\overline{B}+C)$$

Tenga en cuenta que esta expresión suma de productos mínima es equivalente a la expresión suma de productos estándar.
Agrupando los 1s como se indica en la áreas de color gris se obtiene una expresión suma de productos que es equivalente a agrupar los ceros.

$$AC + A\overline{B} = A(\overline{B}+C)$$

Problema relacionado

Utilizar un mapa de Karnaugh para simplificar la siguiente suma de productos estándar:

$$(X+\overline{Y}+Z)(X+\overline{Y}+\overline{Z})(\overline{X}+\overline{Y}+Z)(\overline{X}+Y+Z)$$

EJEMPLO 4.32

Utilizar un mapa de Karnaugh para minimizar la siguiente expresión producto de sumas:

$$(B+C+D)(A+B+\overline{C}+D)(\overline{A}+B+C+\overline{D})(A+\overline{B}+C+D)(\overline{A}+\overline{B}+C+D)$$

Solución

El primer término tiene que desarrollarse en los términos $\overline{A}+B+C+D$ y $A + B + C + D$ para obtener una expresión producto de sumas estándar, que luego debe pasarse a un mapa de Karnaugh, y agrupar las celdas como se muestra en la Figura 4.40. El término suma correspondiente a cada grupo se indica en la figura y la expresión producto de sumas mínima resultante es:

$$(C+D)(A+B+D)(\overline{A}+B+C)$$

Tenga en cuenta que este producto de sumas mínimo es equivalente al producto de sumas estándar original.

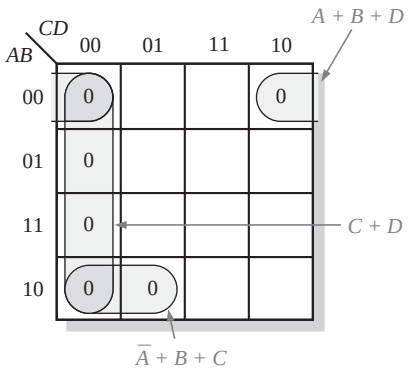


FIGURA 4.40

Problema relacionado Utilizar un mapa de Karnaugh para simplificar la siguiente expresión producto de sumas:

$$(W+\overline{X}+Y+\overline{Z})(W+X+Y+Z)(W+\overline{X}+\overline{Y}+Z)(\overline{W}+\overline{X}+Z)$$

Conversión entre suma de productos y productos de sumas mediante el mapa de Karnaugh

Cuando un producto de sumas se traslada a un mapa de Karnaugh, puede fácilmente pasarse a la suma de productos equivalente directamente a partir de dicho mapa. También, dado un mapa de Karnaugh de una suma de productos, el producto de sumas equivalente puede obtenerse directamente a partir del mapa. Esto proporciona una excelente manera de comparar ambas formas mínimas de una expresión, para determinar si una de ellas puede implementarse con menos puertas que la otra.

Para un producto de sumas, todas las celdas que no contienen 0s contienen 1s, de lo que se deriva su expresión suma de productos. De igual manera, para una suma de productos, todas las celdas que no contienen 1s contendrán 0s, de los que se obtiene la expresión producto de sumas. El Ejemplo 4.33 ilustra esta conversión.

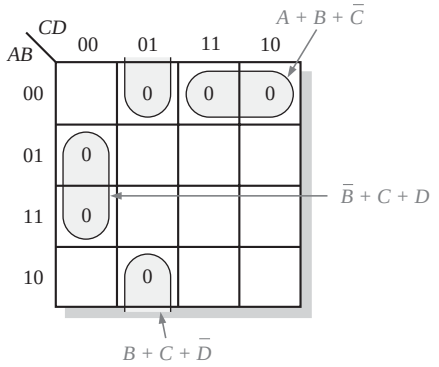
EJEMPLO 4.33

Utilizando un mapa de Karnaugh, convertir el siguiente producto de sumas estándar en un producto de sumas mínimo, una suma de productos estándar y una suma de productos mínima.

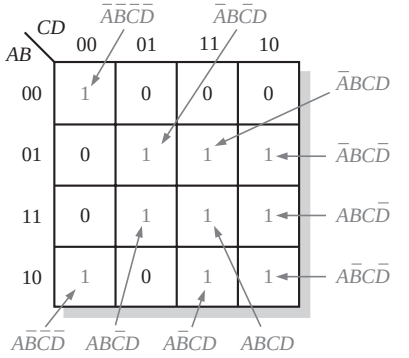
$$(\overline{A}+\overline{B}+C+D)(A+\overline{B}+C+D)(A+B+C+\overline{D})$$
$$(A+B+\overline{C}+\overline{D})(\overline{A}+B+C+\overline{D})(A+B+\overline{C}+D)$$

Solución Los ceros de la expresión producto de sumas estándar se transforman y agrupan para obtener el producto de sumas mínimo, como se indica en la Figura 4.41(a). En la Figura 4.41(b), se añaden 1s en las celdas que no contienen 0s.

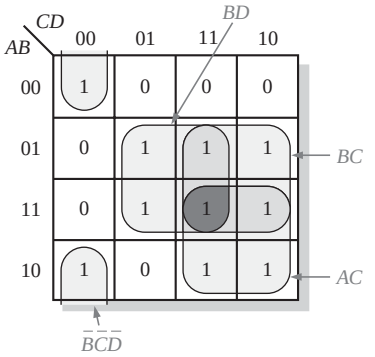
De cada celda que contenga un 1, se obtiene un término producto estándar, como se indica. Estos términos producto forman la expresión suma de productos estándar. En la Figura 4.41(c), se agrupan los 1s y se obtiene una expresión suma de productos mínima.



(a) Producto de sumas mínimo:
$(A + B + \overline{C})(\overline{B} + C + D)(B + C + \overline{D})$



(b) Suma de productos estándar:
$\overline{A}\overline{B}\overline{C}\overline{D} + \overline{A}\overline{B}C\overline{D} + \overline{A}BCD + \overline{A}BC\overline{D} + ABC\overline{D} + AB\overline{C}\overline{D} +$
$AB\overline{C}\overline{D} + ABC\overline{D} + AB\overline{C}D + ABCD$



(c) Suma de productos mínima:
$AC + BC + BD + \overline{B}\overline{C}\overline{D}$

FIGURA 4.41

Problema relacionado Utilizar un mapa de Karnaugh para convertir la siguiente expresión a su forma suma de productos mínima:

$$(W + \overline{X} + Y + \overline{Z})(\overline{W} + X + \overline{Y} + \overline{Z})(\overline{W} + \overline{X} + \overline{Y} + Z)(\overline{W} + \overline{X} + \overline{Z})$$

REVISIÓN DE LA SECCIÓN 4.10

1. ¿Cuál es la diferencia entre pasar a mapa de Karnaugh un producto de sumas y una suma de productos?
2. ¿Cuál es el término suma estándar expresado con las variables A, B, C y D para un 0 en la celda 1011 del mapa de Karnaugh?
3. ¿Cuál es el término producto estándar expresado con las variables A, B, C y D para un 1 en la celda 0010 del mapa de Karnaugh?

4.11 MAPA DE KARNAUGH DE CINCO VARIABLES

Las funciones booleanas de cinco variables pueden simplificarse mediante un mapa de Karnaugh de 32 celdas. Realmente, para construir un mapa de 5 variables se utilizan dos mapas de cuatro variables (con 16 celdas cada uno). Ya conocemos la adyacencia de celdas en los mapas de 4 variables y cómo se forman los grupos de celdas que contengan 1s para simplificar una suma de productos. Luego todo lo que se necesita aprender para manejar cinco variables es la adyacencia de celdas entre los dos mapas de 4 variables y cómo agruparlas.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Determinar la adyacencia de celdas en un mapa de 5 variables. ■ Formar grupos que contengan el máximo número de celdas en un mapa de 5 variables. ■ Minimizar las expresiones booleanas de 5 variables utilizando el mapa de Karnaugh.

Un mapa de Karnaugh de cinco variables (*ABCDE*) puede crearse utilizando dos mapas de 4 variables, con los que ya estamos familiarizados. Cada mapa contiene 16 celdas con todas las posibles combinaciones de las variables *B, C, D* y *E*. Un mapa es para *A*=0, mientras que el otro es para *A*=1, como se muestra en la Figura 4.42.

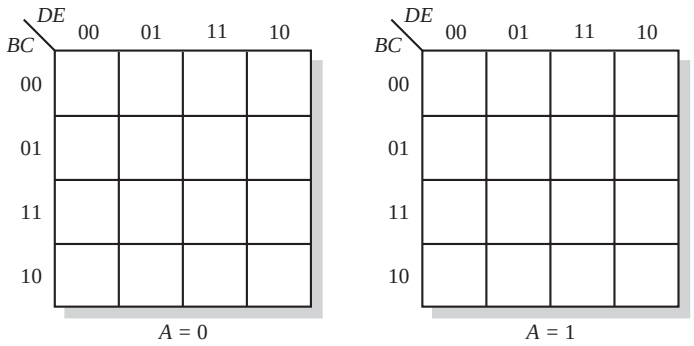


FIGURA 4.42 Mapa de Karnaugh de 5 variables.

Adyacencia de celdas

Ya sabemos cómo determinar celdas adyacentes dentro de un mapa de cuatro variables. La mejor manera de visualizar la adyacencia de celdas entre los dos mapas de 16 celdas consiste en imaginar que el mapa $A = 0$ está colocado encima del mapa $A = 1$. Cada celda del mapa $A = 0$ es adyacente con la celda que está justo debajo en el mapa $A = 1$.

Para ilustrar esto, la Figura 4.43 muestra un ejemplo con cuatro grupos, con los mapas en disposición tridimensional. Los 1s de las celdas en gris más claro forman un grupo de 8 bits (cuatro correspondientes al mapa $A = 0$ combinadas con cuatro del mapa $A = 1$). Los 1s de las celdas marcadas con un degradado de grises forman un grupo de 4 bits. Los 1s de las celdas de la esquina inferior izquierda constituyen un grupo de 4 bits sólo en el mapa $A = 0$. El 1 de la celda gris oscuro del mapa $A = 1$ se agrupa con el 1 de la celda gris más claro de la parte inferior derecha del mapa $A = 0$ para formar un grupo de 2 bits.

Determinación de la expresión booleana. La expresión booleana suma de productos original que está dibujada en el mapa de Karnaugh de la Figura 4.43 contiene diecisiete términos de cinco variables, ya que existen dieci-

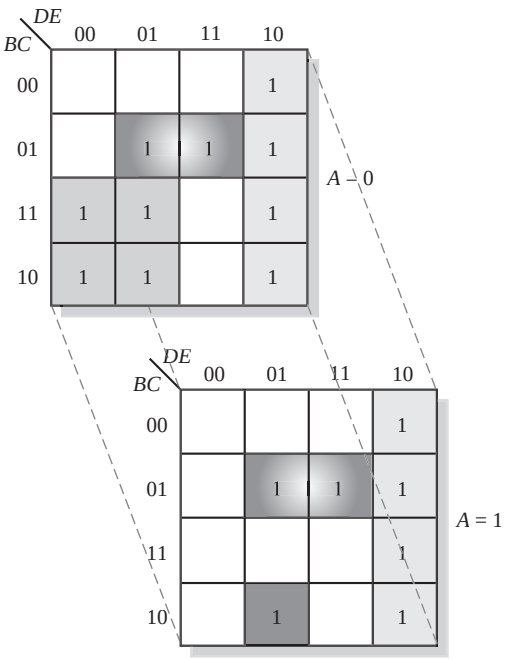


FIGURA 4.43 Ilustración de la agrupación de 1s en celdas adyacentes de un mapa de 5 variables.

siete 1s en el mapa. Como ya sabemos, sólo las variables que no cambian de no complementada a complementada dentro de un grupo permanecen en la expresión correspondiente a ese grupo. La expresión simplificada se obtiene a partir del mapa de la manera siguiente:

- El término para el grupo de ocho 1s marcado en gris claro es $D\bar{E}$.
- El término para el grupo de cuatro 1s marcado en gris degradado es $\bar{B}CE$.
- El término para el grupo de cuatro 1s de la esquina inferior izquierda del mapa $A = 0$ es $\bar{A}B\bar{D}$.
- El término para la celda gris más oscuro agrupada con la celda en gris más claro es $B\bar{C}\bar{D}E$.

Combinando estos términos en la expresión suma de productos simplificada tenemos

$$X = D\bar{E} + \bar{B}CE + \bar{A}B\bar{D} + B\bar{C}\bar{D}E$$

EJEMPLO 4.34

Utilizar un mapa de Karnaugh para minimizar la siguiente expresión suma de productos de 5 variables:

$$X = \bar{A}\bar{B}\bar{C}\bar{D}\bar{E} + \bar{A}\bar{B}C\bar{D}\bar{E} + \bar{A}BC\bar{D}\bar{E} + \bar{A}B\bar{C}\bar{D}\bar{E} + \bar{A}\bar{B}\bar{C}\bar{D}E + \bar{A}BC\bar{D}E$$
$$+ \bar{A}BCDE + AB\bar{C}\bar{D}\bar{E} + AB\bar{C}\bar{D}E + ABC\bar{D}E + ABCDE + A\bar{B}CDE$$

Solución En la Figura 4.44, se traslada la suma de productos al mapa de Karnaugh y se realizan las agrupaciones indicando los términos correspondientes. Combinando estos términos se obtiene la siguiente expresión suma de productos minimizada:

$$X + \bar{A}\bar{D}\bar{E} + \bar{B}\bar{C}\bar{D} + BCE + ABCE$$

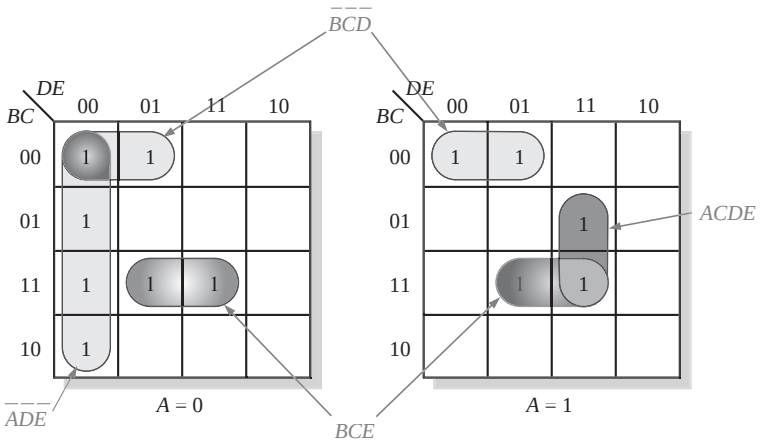


FIGURA 4.44

Problema relacionado Minimizar la siguiente expresión:

$$X = \bar{A}B\bar{C}\bar{D}\bar{E} + \bar{A}B\bar{C}D\bar{E} + \bar{A}BC\bar{D}\bar{E} + \bar{A}BCD\bar{E} + A\bar{B}\bar{C}\bar{D}\bar{E} + A\bar{B}C\bar{D}E + ABC\bar{D}\bar{E} + AB\bar{C}\bar{D}\bar{E}$$
$$+ \bar{A}B\bar{C}\bar{D}E + \bar{A}B\bar{C}DE + \bar{A}BCDE + \bar{A}BC\bar{D}E + A\bar{B}C\bar{D}E + A\bar{B}CDE + ABCDE + AB\bar{C}DE$$

REVISIÓN DE LA SECCIÓN 4.11

1. ¿Por qué un mapa de Karnaugh de 5 variables requiere 32 celdas?
2. ¿Cuál es la expresión representada por un mapa de Karnaugh de cinco variables en el que cada celda contiene un 1?

4.12 VHDL (OPCIONAL)

Esta sección opcional proporciona una breve introducción al lenguaje VHDL y su finalidad no es la de enseñar la estructura y la sintaxis completas del lenguaje. Para obtener información más detallada, consulte las referencias incluidas en la nota a pie de página. Los lenguajes de descripción hardware (HDL, hardware description language) son herramientas que permiten introducir los diseños lógicos, utilizando texto, que se emplean para implementar circuitos lógicos en dispositivos lógicos programables. Aunque el VHDL proporciona múltiples métodos para describir un circuito lógico, aquí sólo vamos a ver los ejemplos de programación más sencillos y directos de introducción del diseño mediante texto.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Establecer los elementos fundamentales del VHDL. ■ Escribir un programa VHDL simple.

La V de VHDL* viene de VHSIC (*Very High Speed Integrated Circuit*) y, claro está, HDL es el acrónimo de *Hardware Description Language*. Como ya hemos mencionado, ***VHDL*** es un lenguaje estándar adoptado por

---

* ***Véase*** **Floyd, Thomas. 2003. *Digital Fundamentals with VHDL.* Prentice Hall; Pellerin, David y Taylor, Douglas. 1997. *VHDL Made Easy!* Prentice Hall; Bhasker, Jayaram. 1999. *A VHDL Primer,* 3 ed. Prentice Hall.**

el IEEE (*Institute of Electrical and Electronics Engineers*) y se designa como IEEE Std. 1076-1993. VHDL es un lenguaje complejo y exhaustivo y utilizarlo para sacarle el máximo partido posible exige un gran esfuerzo y tener experiencia.

VHDL proporciona tres métodos básicos para describir un circuito digital por software: *comportamental*, *flujo de datos* y *estructural.* Esta exposición vamos a restringirla al método de flujo de datos en el que se escriben instrucciones de tipo booleano para describir un circuito lógico. Tenga en cuenta que el VHDL, asi como otros lenguajes HDL, es una herramienta que permite implementar diseños digitales y es, por tanto, un medio para conseguir un fin y no un fin en sí mismo.

Es relativamente fácil escribir programas para describir circuitos lógicos simples en VHDL. Los operadores lógicos son las siguientes palabras clave VHDL: **and**, **or, not, nand, nor, xor** y **xnor**. Los dos elementos fundamentales en cualquier programa VHDL son la entidad y la arquitectura y deben utilizarse juntos. La **entidad** describe una determinada función lógica en función de sus entradas externas y sus salidas, denominadas puertos. La **arquitectura** describe la operación interna de la función lógica.

En su forma más simple, el elemento entidad (**entity**) consta de tres instrucciones. La primera de ellas asigna un nombre a una función lógica; la segunda instrucción, denominada instrucción *port* y que se indenta, especifica las entradas y las salidas; y la tercera instrucción es la instrucción *end*. Aunque probablemente nunca escriba un programa VHDL para una única puerta, es instructivo empezar con un ejemplo sencillo como por ejemplo una puerta AND. La declaración entity para una puerta AND de 2 entradas es:

▲ *Las comas y puntos y comas deben utilizarse de forma apropiada en todos los programas VHDL.*

```
entity AND_Gate2 is
   port (A, B: in bit; X: out bit);
end entity AND_Gate2;
```

Los términos en negrita son las palabras clave VHDL; los demás términos son identificadores que el usuario define; la sintaxis del VHDL exige el uso de paréntesis, comas y puntos y comas. Como puede ver, A y B se especifican como bits de entrada y X se especifica como un bit de salida. Los identificadores de puerto A, B y X, así como el nombre de la entidad, AND_Gate2, son definidos por el usuario y, por tanto, su nombre puede cambiarse. Como en todos los lenguajes HDL, la colocación de las comas y de los puntos y comas es crucial y deben cumplirse de forma estricta.

El elemento arquitectura (**architecture**) VHDL del programa para una puerta AND de 2 entradas descrito mediante el elemento entidad anterior es

```
architecture LogicFunction of AND_Gate2 is
begin
   X ⇐ A and B;
end architecture LogicFunction;
```

De nuevo, las palabras clave VHDL se han escrito en negrita; la sintaxis impone el uso de los puntos y comas y del símbolo ⇐. La primera instrucción de este elemento debe hacer referencia al nombre de la entidad.

La entidad y la arquitectura se combinan en un único programa VHDL para describir una puerta AND, como se ilustra en la Figura 4.45.

Escritura de expresiones booleanas en VHDL. Como hemos visto, la expresión para una puerta AND de 2 entradas, $X = AB$, en VHDL se escribe como X ⇐ A **and** B;. Cualquier expresión booleana puede escribirse utilizando las palabras clave VHDL **not, and, or, nand, nor, xor** y **xnor**. Por ejemplo, la expresión booleana $X = A + B + C$ puede escribirse en VHDL como X ⇐ A **or** B **or** C;. La expresión booleana $X = A\overline{B} + \overline{C}D$ puede escribirse en VHDL como X ⇐ (A **and not** B) **or** (**not** C **and** D);. Veamos otro ejemplo, la instrucción VHDL para una puerta NAND de 2 entradas puede escribirse como X ⇐ **not** (A **and** B); o también podría escribirse como X ⇐ A **nand** B);.

```
entity AND_Gate2 is
  port (A, B: in bit; X: out bit);
end entity AND_Gate2;

architecture LogicFunction of AND_Gate2 is
begin
  X <= A and B;
end architecture LogicFunction;
```

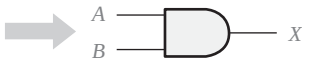


FIGURA 4.45 Un programa VHDL para una puerta AND de 2 entradas.

EJEMPLO 4.35

Escribir un programa VHDL para describir el circuito lógico de la Figura 4.46.

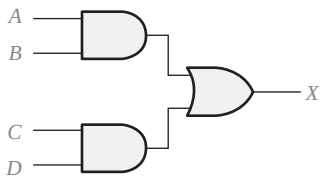


FIGURA 4.46

Solución En álgebra de Boole, este circuito lógico se describe como sigue:

$$X = AB + CD$$

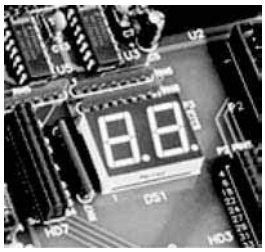
El programa VHDL es el siguiente. La entidad se denomina AND_OR.

```
entity AND_OR is
  port (A, B, C, D: in bit; X: out bit);
end entity AND_OR;
architecture LogicFunction of AND_OR is
begin
  X <= (A and B) or (C and D);
end architecture LogicFunction;
```

Problema relacionado Escribir la instrucción VHDL que describa el circuito lógico si una puerta NOR reemplaza a la puerta OR de la Figura 4.46.

REVISIÓN DE LA SECCIÓN 4.12

1. ¿Qué es el HDL?
2. Nombrar los dos elementos de diseño fundamentales en un programa VHDL.
3. ¿Para qué sirve la entidad?
4. ¿Para qué sirve la arquitectura?



APLICACIÓN A LOS SISTEMAS DIGITALES

Los displays de 7 segmentos se utilizan en toda clase de productos. El sistema de control y recuento de pastillas, que se ha descrito en el Capítulo 1, tiene dos displays de 7 segmentos. Estos displays se utilizan junto con circuitos lógicos que decodifican un número BCD y activan los dígitos adecuados del display. En esta sección, nos vamos a centrar en un diseño con un número mínimo de puertas para ilustrar las aplicaciones de las expresiones booleanas y de los mapas de Karnaugh. De forma opcional, también se aplica el lenguaje VHDL.

El display de 7 segmentos

La Figura 4.47 muestra un display común formado por siete elementos o segmentos. Excitando determinadas combinaciones de estos segmentos, se pueden obtener cada uno de los diez dígitos decimales. La Figura 4.48 muestra este tipo de display digital para cada uno de los diez dígitos, en el que se utiliza un segmento gris oscuro para indicar cuál está excitado. Para generar un 1, se excitan los segmentos *b* y *c*; para producir un 2, se excitan los segmentos *a, b, g, e* y *d*, y así sucesivamente.

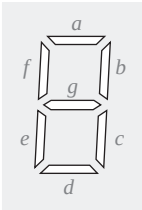
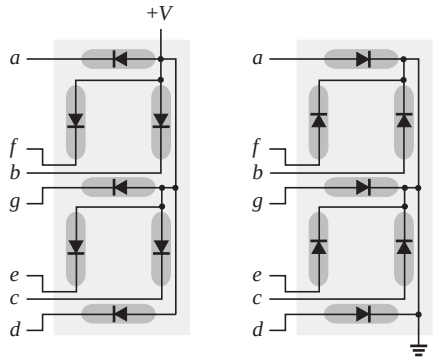


FIGURA 4.47 Disposición de los segmentos en un display de 7 segmentos.

Displays de LED. Un tipo muy común de display de 7 segmentos es el de diodos emisores de luz (*Light-Emitting Diode*, **LED**), colocados como se muestra en la Figura 4.49. Cada segmento es un LED que emite luz cuando lo atraviesa una corriente eléctrica. En la Figura 4.49(a), la configuración en ánodo común requiere un circuito de excitación, que proporcione un nivel de tensión bajo para activar un determinado segmento. Cuando se aplica un nivel BAJO a la entrada de un segmento, el LED se enciende y circula corriente a su través. En la Figura 4.49(b), la configuración en cátodo común requiere un circuito de excitación que proporcione un nivel de tensión alto para activar un cierto segmento. Cuando se aplica un nivel ALTO a la entrada del segmento, el LED se enciende y circula corriente a su través.



(a) Ánodo común (b) Cátodo común

FIGURA 4.49 Configuraciones de los display de LED de 7 segmentos.

Displays de LCD. Otro tipo común de displays de 7-segmentos es el de cristal líquido, (**LCD**, *Liquid Crystal Display*). Los LCD funcionan polarizando la luz de forma que un segmento que no está activado refleja la luz incidente, por lo que se ilumina. Un segmento activado no

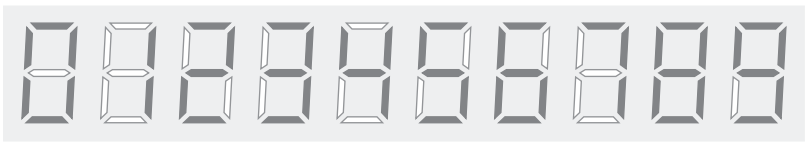


FIGURA 4.48 Display para dígitos decimales mediante un dispositivo de 7 segmentos.

refleja la luz incidente y, por tanto, permanece oscuro. Los LCD consumen mucha menos potencia que los LED, pero no se pueden ver en la oscuridad, mientras que los LED sí.

Lógica de los segmentos

Cada segmento se utiliza para varios dígitos decimales, pero ninguno de ellos se emplea para representar los diez dígitos; por tanto, cada segmento tiene que activarse mediante su propio circuito de decodificación que detecta la aparición de cualquier número en el que haya que usar ese segmento. A partir de las Figuras 4.47 y 4.48, se determinan los segmentos que hay que activar para representar cada uno de los dígitos, los cuales se enumeran en la Tabla 4.9.

Dígito	Segmentos activados
0	*a, b, c, d, e, f*
1	*b, c*
2	*a, b, d, e, g*
3	*a, b, c, d, g*
4	*b, c, f, g*
5	*a, c, d, f, g*
6	*a, c, d, e, f, g*
7	*a, b, c*
8	*a, b, c, d, e, f, g*
9	*a, b, c, d, f, g*

TABLA 4.9 Segmentos activados para cada dígito decimal.

Tabla de verdad de la lógica de segmentos. La lógica de decodificación de segmentos requiere cuatro entradas en código decimal binario (BCD) y siete salidas, una para cada segmento del display, como se indica en el diagrama de bloques de la Figura 4.50. La tabla de verdad de salida múltiple, que se muestra en la Tabla 4.10, corresponde en realidad a siete tablas de verdad, que podrían separarse en una tabla por segmento. Si aparece un 1 en las columnas de salida de la tabla, indica que el segmento está activado.

Puesto que el código BCD no incluye los valores binarios 1010, 1011, 1100, 1101, 1110 y 1111, estas combinaciones no van nunca a aparecer en las entradas y pueden, por tanto, tratarse como condiciones indiferentes (X), como se indica en la tabla de verdad. Para coincidir con la mayoría de fabricantes de circuitos integrados, en esta aplicación una *A* representa el bit menos significativo y una *D* indica el bit más significativo.

Expresiones booleanas de la lógica de segmentos. A partir de la tabla de verdad se puede escribir para cada segmento una expresión suma de productos o producto de sumas. Por ejemplo, la expresión suma de productos estándar para el segmento *a* es:

$$a = \bar{D}\bar{C}\bar{B}\bar{A} + \bar{D}\bar{C}B\bar{A} + \bar{D}\bar{C}BA + \bar{D}C\bar{B}A + \bar{D}CB\bar{A} + \bar{D}CBA + D\bar{C}\bar{B}\bar{A} + D\bar{C}\bar{B}A$$

y la expresión suma de productos estándar para el segmento *e* es

$$e = \bar{D}\bar{C}\bar{B}\bar{A} + \bar{D}\bar{C}B\bar{A} + \bar{D}CB\bar{A} + D\bar{C}\bar{B}\bar{A}$$

De forma similar, se pueden desarrollar expresiones para los restantes segmentos. Como se puede ver, la expresión para el segmento *a* consta de ocho productos y la expresión para el segmento *e* tienen cuatro términos productos que representan cada una de las entradas BCD que activan dicho segmento. Esto significa que la implementación de la suma de productos estándar de la lógica del segmento *a* requiere un circuito AND-OR formado por ocho puertas AND de 4 entradas y una puerta OR de ocho entradas. La implementación de la lógica correspondiente al segmento *e* requiere cuatro puertas AND de 4 entradas y una puerta OR de 4 entradas. En ambos casos, se necesitan cuatro inversores para generar el complemento de cada una de las variables.

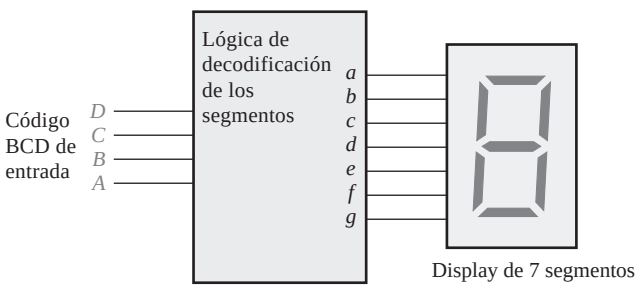


FIGURA 4.50 Diagrama de bloques de la lógica y el display de 7-segmentos.

Dígito	Entradas				Salidas de segmentos						
decimal	*D*	*C*	*B*	*A*	*a*	*b*	*c*	*d*	*e*	*f*	*g*
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	1
10	1	0	1	0	X	X	X	X	X	X	X
11	1	0	1	1	X	X	X	X	X	X	X
12	1	1	0	0	X	X	X	X	X	X	X
13	1	1	0	1	X	X	X	X	X	X	X
14	1	1	1	0	X	X	X	X	X	X	X
15	1	1	1	1	X	X	X	X	X	X	X

Salida = 1 quiere decir segmento activado (encendido)
Salida = 0 quiere decir segmento desactivado (apagado)
Salida = X significa indiferente.

TABLA 4.10 Tabla de verdad para la lógica de 7 segmentos.

Minimización mediante el mapa de Karnaugh de la lógica de segmentos. Vamos a comenzar obteniendo una expresión suma de productos mínima para el segmento *a*. En la Figura 4.51 se muestra un mapa de Karnaugh correspondiente al segmento *a*. Los pasos que hay que seguir son los siguientes:

Paso 1. Los 1s de la Tabla 4.10 se pasan directamente al mapa de Karnaugh.

Paso 2. Se introducen en el mapa todas las condiciones "indiferentes" (X).

Paso 3. Se agrupan los 1s como se muestra. Se utilizan las condiciones "indiferentes" y superposiciones de celdas para conseguir los grupos más grandes posibles.

Paso 4. Se escribe el término producto mínimo para cada grupo y se suman para obtener la expresión suma de productos mínima.

No olvide que las condiciones "indiferentes" no tienen porqué incluirse en un grupo, pero en este caso se utilizan todas ellas. También hay que fijarse en que los 1s de las celdas de las esquinas se agrupan con condiciones indiferentes utilizando la adyacencia cíclica.

Implementación mínima de la lógica del segmento a. La expresión mínima suma de productos a partir del mapa de Karnaugh de la Figura 4.52 para la lógica del segmento *a* es:

$$D + B + CA + \overline{C}\,\overline{A}$$

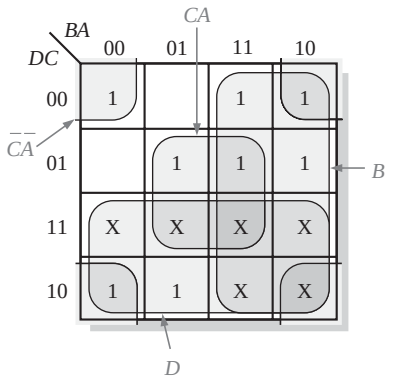
Esta expresión puede ser implementada mediante dos puertas AND de 2 entradas, una puerta OR de 4 entradas y dos inversores, como se muestra en la Figura 4.52. Compare este circuito con la implementación de la expresión estándar del segmento *a* vista anteriormente. Comprobará que el número de puertas e inversores se ha reducido de trece a cinco, disminuyendo significativamente el número de interconexiones necesarias.

La lógica mínima necesaria para cada uno de los restantes seis segmentos (*b, c, d, e, f* y *g*) puede obtenerse mediante un método similar.

Implementación VHDL (opcional)

Toda la lógica de los segmentos puede describirse utilizando VHDL para llevar a cabo la implementación en un dispositivo lógico programable. La lógica correspondiente al

Suma de productos estándar:
$\bar{D}\bar{C}\bar{B}\bar{A} + \bar{D}\bar{C}\bar{B}A + \bar{D}\bar{C}B\bar{A} + \bar{D}\bar{C}BA + \bar{D}C\bar{B}\bar{A} + \bar{D}C\bar{B}A + DC\bar{B}\bar{A} + DC\bar{B}A$



Suma de productos mínima: $D + B + CA + \overline{C}\overline{A}$

FIGURA 4.51 Minimización de la expresión lógica del segmento *a* mediante el mapa de Karnaugh.

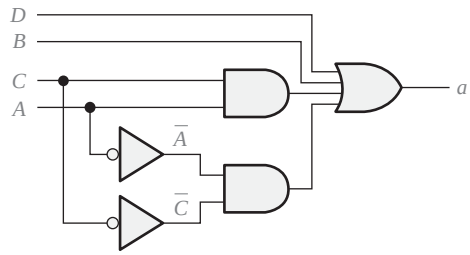


FIGURA 4.52 Implementación lógica mínima del segmento *a* de un display de 7-segmentos.

segmento *a* puede describirse mediante el siguiente programa VHDL:

```
entity SEGLOGIC is
  port (A, B, C, D: in bit; SEGa: out bit);
end entity SEGLOGIC;
architecture LogicFunction of
SEGLOGIC is
begin
  SEGa <= (A and C) or (not A
     and not C) or B or D;
end architecture LogicFunction;
```

Prácticas de sistemas

- ***Actividad 1*** Determinar la lógica mínima para el segmento *b*.
- ***Actividad 2*** Determinar la lógica mínima para el segmento *c*.
- ***Actividad 3*** Determinar la lógica mínima para el segmento *d*.
- ***Actividad 4*** Determinar la lógica mínima para el segmento *e*.
- ***Actividad 5*** Determinar la lógica mínima para el segmento *f*.
- ***Actividad 6*** Determinar la lógica mínima para el segmento *g*.
- ***Actividad opcional*** Completar el programa VHDL para los siete segmentos incluyendo en la arquitectura la descripción lógica de cada segmento.

RESUMEN

- En la Figura 4.53 se muestran los símbolos y las expresiones booleanas de salida para un inversor y puertas de dos entradas.

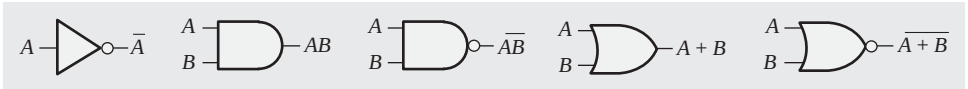


FIGURA 4.53

- Leyes conmutativas: $A + B = B + A$
  $AB = BA$

- Leyes asociativas: $A + (B + C) = (A + B) + C$
  $A(BC) = (AB)C$
- Ley distributiva: $A(B + C) = AB + AC$
- Reglas del álgebra booleana:

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

- Teoremas de DeMorgan:
  1. El complemento de un producto es igual a la suma de los complementos de los términos del producto.
  $\overline{XY} = \overline{X} + \overline{Y}$
  2. El complemento de una suma es igual al producto de los complementos de los términos de la suma,
  $\overline{X + Y} = \overline{X}\,\overline{Y}$
- En la Figura 4.54 se muestran los mapas de Karnaugh para 3 y 4 variables. Un mapa de 5 variables se forma a partir de dos tablas de 4 variables.

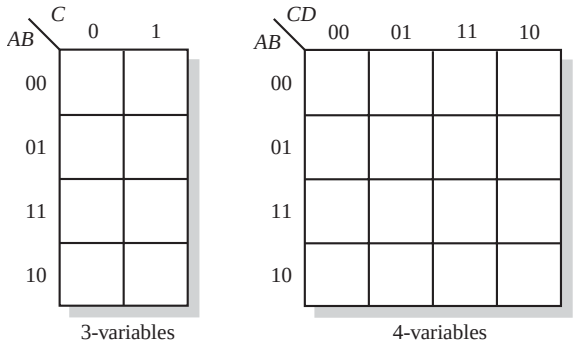


FIGURA 4.54

- El elemento de diseño básico en VHDL es una pareja entidad/arquitectura.

PALABRAS CLAVE

Las palabras clave y otros términos que se han resaltado en negrita se encuentran en el glosario final del libro.

Complemento El inverso u opuesto de un número. En el álgebra de Boole, la función inversa expresada mediante una barra sobre una variable. El complemento de 1 es 0, y viceversa.

Indiferente Combinación de literales de entrada que no pueden ocurrir y que se utilizan como 1s o 0s en un mapa de Karnaugh.

Mapa de Karnaugh Disposición de celdas que representa las combinaciones de literales en una expresión booleana y que se utiliza para la simplificación sistemática de la expresión.

Minimización El proceso que da como resultado una expresión booleana suma de productos o un producto de sumas que contiene el menor número de literales posible por término.

Producto de sumas Expresión booleana que consiste simplemente en multiplicar (operación AND) términos suma (operación OR).

Suma de productos Expresión booleana que consiste simplemente en sumar (operación OR) términos que contienen productos (operación AND).
Término producto Producto booleano de dos o más literales equivalente a una operación AND.
Término suma Suma booleana de dos o más literales equivalente a la operación OR.
Variable Símbolo utilizado para representar una magnitud lógica que puede tener tanto un valor 1 como 0; generalmente se designa mediante una letra cursiva.
VHDL Lenguaje estándar de descripción hardware. IEEE Std. 1076-1993.

AUTOTEST

Las respuestas se encuentran al final del capítulo.

1. El complemento de una variable siempre es:
(a) 0 **(b)** 1 **(c)** igual a la variable **(d)** el inverso de la variable

2. La expresión booleana $A + \overline{B} + C$ es:
(a) un término suma **(b)** un literal
(c) un término producto **(d)** un término complementado

3. La expresión booleana $A\overline{B}C\overline{D}$ es:
(a) un término suma **(b)** un término producto
(c) un literal **(d)** siempre 1

4. El dominio de la expresión $A\overline{B}CD + A\overline{B} + \overline{C}D + B$ es:
(a) A y D **(b)** Sólo B **(c)** A, B, C y D **(d)** ninguno de los anteriores

5. De acuerdo con la ley conmutativa de la suma,
(a) $AB = BA$ **(b)** $A = A + A$
(c) $A + (B + C) = (A + B) + C$ **(d)** $A + B = B + A$

6. De acuerdo con la ley asociativa de la multiplicación,
(a) $B = BB$ **(b)** $A\ (BC) = (AB)C$
(c) $A + B = B + A$ **(d)** $B + B\ (B + 0)$

7. De acuerdo con la ley distributiva,
(a) $A\ (B + C) = AB + AC$ **(b)** $A\ (BC) = ABC$
(c) $A(A + 1) = A$ **(d)** $A + AB = A$

8. ¿Cuál de las siguientes no es una regla válida del álgebra booleana?
(a) $A + 1 = 1$ **(b)** $A = \overline{A}$
(c) $AA = A$ **(d)** $A + 0 = A$

9. ¿Cuál de las siguientes reglas establece que si una entrada de una puerta AND es siempre 1, la salida es igual a la otra entrada?
(a) $A + 1 = 1$ **(b)** $A + A = A$
(c) $A \cdot A = A$ **(d)** $A \cdot 1 = A$

10. De acuerdo con los teoremas de DeMorgan, ¿cuáles de las siguientes igualdades son correctas?
(a) $\overline{AB} = \overline{A} + \overline{B}$ **(b)** $\overline{XYZ} = \overline{X} + \overline{Y} + \overline{Z}$
(c) $\overline{A + B + C} = \overline{A}\overline{B}\overline{C}$ **(d)** Todas las respuestas

11. La expresión booleana $X = AB + CD$ representa

(a) dos operaciones OR multiplicadas (AND). **(b)** Una puerta AND de 4 entradas

(c) dos operaciones AND sumadas (OR) **(d)** una operación OR-exclusiva

12. Un ejemplo de una expresión suma de productos es

(a) $A + B(C + D)$ **(b)** $\overline{A}B + A\overline{C} + A\overline{B}C$

(c) $(\overline{A} + B + C)(A + \overline{B} + C)$ **(d)** Las respuestas (a) y (b)

13. Un ejemplo de una expresión producto de sumas es

(a) $A(B + C) + A\overline{C}$ **(b)** $(A + B)(\overline{A} + B + \overline{C})$

(c) $\overline{A} + \overline{B} + BC$ **(d)** Las respuestas (a) y (b)

14. Un ejemplo de una expresión suma de productos estándar es

(a) $\overline{A}B + A\overline{B}C) + AB\overline{D}$ **(b)** $A\overline{B}C + A\overline{C}D$

(c) $A\overline{B} + \overline{A}B + AB$ **(d)** $A\overline{B}C\overline{D} + \overline{A}B + \overline{A}$

15. Un mapa de Karnaugh de 3 variables tiene

(a) ocho celdas **(b)** tres celdas **(c)** dieciséis celdas **(d)** cuatro celdas

16. En un mapa de Karnaugh de 4 variables, un término producto de dos variables se obtiene de un

(a) grupo de 2 celdas de 1s **(b)** grupo de 8 celdas de 1s

(c) grupo de 4 celdas de 1s **(d)** grupo de 4 celdas de 0s

17. En un mapa de Karnaugh, la agrupación de 0s produce

(a) una expresión producto de sumas **(b)** una expresión suma de productos

(c) una condición "indiferente" **(d)** un circuito lógico AND-OR

18. Un mapa de Karnaugh de 5 variables tiene

(a) dieciséis celdas **(b)** treinta y dos celdas **(c)** sesenta y cuatro celdas

19. Un SPLD que tiene una matriz AND programable y una matriz OR fija es una

(a) PROM **(b)** PLA **(c)** PAL **(d)** GAL

20. VHDL es un tipo de

(a) circuito lógico programable **(b)** lenguaje de descripción hardware

(c) matriz programable **(d)** matemáticas lógicas

21. En VHDL, un puerto es

(a) un tipo de entidad **(b)** un tipo de arquitectura

(c) una entrada o una salida **(d)** un tipo de variable

PROBLEMAS

Las respuestas a los problemas impares se encuentran al final del libro.

SECCIÓN 4.1 Operaciones y expresiones booleanas

1. Utilizando la notación booleana, escribir una expresión que sea 1 siempre que una o más de sus variables (A, B, C y D) sean 1.

2. Escribir una expresión que sea 1 sólo si todas sus variables (A, B, C, D y E) son 1.

3. Escribir una expresión que sea 1 cuando una o más variables (A, B y C) son 0.

4. Evaluar las siguientes operaciones:

(a) $0 + 0 + 1$ **(b)** $1 + 1 + 1$ **(c)** $1 \cdot 0 \cdot 0$

(d) $1 \cdot 1 \cdot 1$ **(e)** $1 \cdot 0 \cdot 1$ **(f)** $1 \cdot 1 + 0 \cdot 1 \cdot 1$

5. Hallar los valores de las variables que hacen que cada término producto sea 1 y que cada suma sea 0.

(a) AB **(b)** $A\overline{B}C$ **(c)** $A + B$ **(d)** $\overline{A} + B + \overline{C}$

(e) $\overline{A} + \overline{B} + C$ **(f)** $\overline{A} + B$ **(g)** $A\overline{B}\overline{C}$

6. Hallar los valores de X para todos los posibles valores de las variables.

(a) $X = (A + B)C + B$ **(b)** $X = (\overline{A + B})C$ **(c)** $X = A\overline{B}C + AB$

(e) $X = (A + B)(\overline{A} + B)$ **(f)** $X = (A + BC)(\overline{B} + \overline{C})$

SECCIÓN 4.2 Leyes y reglas del álgebra booleana

7. Identificar la ley del álgebra de Boole en que está basada cada una de las siguientes igualdades.

(a) $A\overline{B} + CD + A\overline{C}D + B = B + A\overline{B} + A\overline{C}D + CD$

(b) $AB\overline{C}D + \overline{ABC} = D\overline{C}BA + \overline{CBA}$

(c) $AB(CD + E\overline{F} + GH) = ABCD + ABE\overline{F} + ABGH$

8. Identificar la regla o reglas del álgebra de Boole en que está basada cada una de las siguientes igualdades.

(a) $\overline{\overline{AB + CD}} + \overline{EF} = AB + CD + \overline{EF}$ **(b)** $A\overline{A}B + AB\overline{C} + AB\overline{B} = AB\overline{C}$

(c) $A(BC + BC) + AC = A(BC) + AC$ **(d)** $AB(C + \overline{C}) + AC = AB + AC$

(e) $A\overline{B} + A\overline{B}C = A\overline{B}$ **(f)** $ABC + \overline{AB} + \overline{ABCD} = ABC + \overline{AB} + D$

SECCIÓN 4.3 Teoremas de DeMorgan

9. Aplicar los teoremas de DeMorgan a cada expresión:

(a) $\overline{A + \overline{B}}$ **(b)** $\overline{\overline{A}B}$ **(c)** $\overline{A + B + C}$ **(d)** $\overline{ABC}$

(e) $\overline{A(B + C)}$ **(f)** $\overline{AB} + \overline{CD}$ **(g)** $\overline{AB + CD}$ **(h)** $\overline{(A + \overline{B})(\overline{C} + D)}$

10. Aplicar los teoremas de DeMorgan a cada expresión:

(a) $\overline{A\overline{B}(C + \overline{D})}$ **(b)** $\overline{AB(CD + EF)}$

(c) $\overline{(A + \overline{B} + C + \overline{D})} + \overline{ABC\overline{D}}$ **(d)** $\overline{(\overline{A} + B + C + D)}\,\overline{(A\overline{B}\overline{C}D)}$

(e) $\overline{\overline{AB}(CD + \overline{E}F)}\,\overline{(\overline{AB} + \overline{CD})}$

11. Aplicar los teoremas de DeMorgan a las siguientes expresiones:

(a) $\overline{\overline{(ABC)}\,\overline{(EFG)}} + \overline{\overline{(HIJ)}\,\overline{(KLM)}}$ **(b)** $\overline{(A + \overline{B\overline{C}} + CD)} + \overline{\overline{BC}}$

(c) $\overline{\overline{(A + B)}\,\overline{(C + D)}\,\overline{(E + F)}\,\overline{(G + H)}}$

SECCIÓN 4.4 Análisis booleano de los circuitos lógicos

12. Escribir la expresión booleana para cada puerta lógica de la Figura 4.55.

13. Escribir la expresión booleana para cada uno de los circuitos lógicos de la Figura 4.56.

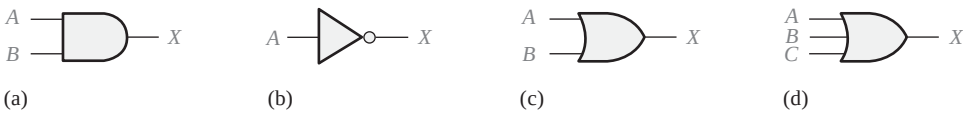


FIGURA 4.55

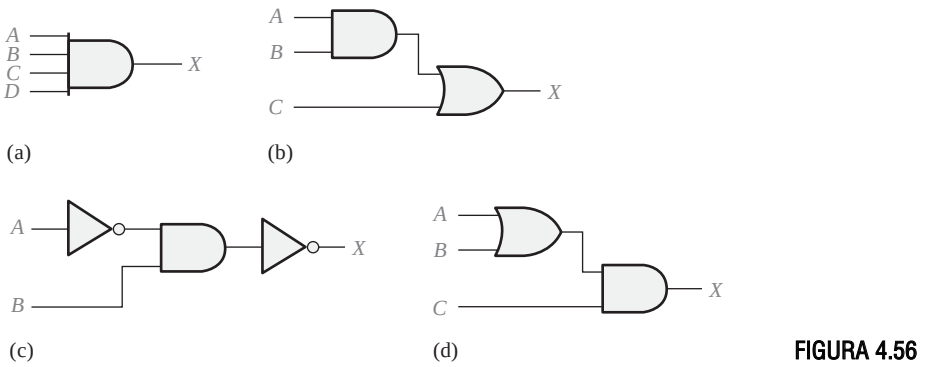


FIGURA 4.56

14. Dibujar el circuito lógico representado por cada una de las siguientes expresiones.

(a) $A + B + C$ **(b)** ABC **(c)** $AB + C$ **(d)** $AB + CD$

15. Dibujar el circuito lógico representado por cada una de las siguientes expresiones.

(a) $A\bar{B} + \bar{A}B$ **(b)** $AB + \bar{A}\bar{B} + \bar{A}BC$

(c) $\bar{A}B(C + \bar{D})$ **(d)** $A + (B[C + D(B + \bar{C})]$

16. Construir una tabla de verdad para cada una de las siguientes expresiones booleanas.

(a) $A + B$ **(b)** AB **(c)** $AB + BC$

(d) $(A + B)C$ **(e)** $(A + B)(\bar{B} + C)$

SECCIÓN 4.5 Simplificación mediante el álgebra de Boole

17. Mediante las técnicas del álgebra de Boole, simplificar las siguientes expresiones lo máximo posible:

(a) $A(A + B)$ **(b)** $A(\bar{A} + AB)$ **(c)** $BC + \bar{B}C$

(d) $A(A + \bar{A}B)$ **(e)** $A\bar{B}C + \bar{A}BC + \bar{A}\bar{B}C$

18. Mediante las técnicas del álgebra de Boole, simplificar las siguientes expresiones:

(a) $(A + \bar{B})(A + C)$ **(b)** $\bar{A}B + \bar{A}B\bar{C} + \bar{A}BCD + \bar{A}B\bar{C}\bar{D}E$

(c) $AB + \overline{A\bar{B}}C + A$ **(d)** $(A + \bar{A})(AB + AB\bar{C})$

(e) $AB + (\bar{A} + \bar{B})C + AB$

19. Mediante las técnicas del álgebra de Boole, simplificar las siguientes expresiones:

(a) $BD + B(D + E) + \bar{D}(D + F)$ **(b)** $\bar{A}\bar{B}C + \overline{(A + B + \bar{C})} + \bar{A}\bar{B}\bar{C}D$

(c) $(B + BC)(B + \bar{B}C)(B + D)$ **(d)** $ABCD + AB(\overline{CD}) + (\overline{AB})CD$

(e) $ABC[AB + \bar{C}(BC + AC)]$

20. Determinar cuáles de los circuitos lógicos de la Figura 4.57 son equivalentes.

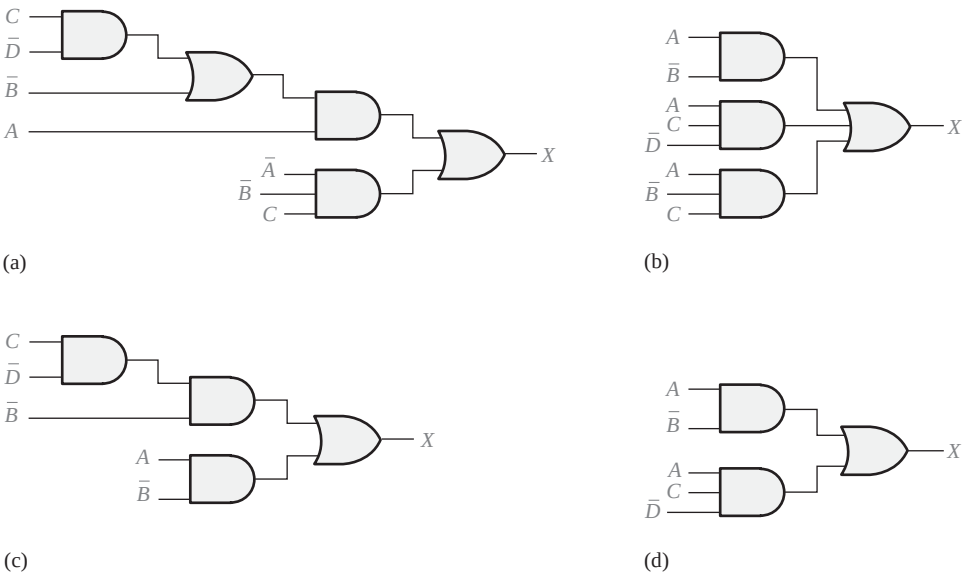


FIGURA 4.57

SECCIÓN 4.6 Formas estándar de las expresiones booleanas

21. Convertir las siguientes expresiones en sumas de productos:

(a) $(A+B)(C+\bar{B})$ **(b)** $(A+\bar{B}C)C$ **(c)** $(A+C)(AB+AC)$

22. Convertir las siguientes expresiones en sumas de productos:

(a) $AB+CD(A\bar{B}+CD)$ **(b)** $AB(\bar{B}\bar{C}+BD)$ **(c)** $A+B[AC+(B+\bar{C})D]$

23. Definir el dominio de cada suma de productos del Problema 21 y convertir la expresión a su forma estándar.

24. Convertir cada suma de productos del Problema 22 a su forma estándar.

25. Determinar el valor binario de cada término en las expresiones suma de productos del Problema 23.

26. Determinar el valor binario de cada término en las expresiones suma de productos del Problema 24.

27. Convertir cada una de las expresiones suma de productos estándar del Problema 23 a su forma producto de sumas estándar.

28. Convertir cada una de las expresiones suma de productos estándar del Problema 24 a su forma producto de sumas estándar.

SECCIÓN 4.7 Expresiones booleanas y tablas de verdad

29. Desarrollar la tabla de verdad de cada una de las siguientes expresiones suma de productos estándar:

(a) $A\bar{B}C+\bar{A}B\bar{C}+ABC$ **(b)** $\overline{XYZ}+\bar{X}\bar{Y}Z+XY\bar{Z}+X\bar{Y}Z+\bar{X}YZ$

30. Desarrollar la tabla de verdad de cada una de las siguientes expresiones suma de productos estándar:

(a) $\bar{A}B\bar{C}D+\bar{A}BC\bar{D}+A\bar{B}\bar{C}D+\bar{A}\bar{B}\bar{C}\bar{D}$

(b) $WXYZ+WXY\bar{Z}+\bar{W}XYZ+W\bar{X}YZ+WX\bar{Y}Z$

31. Desarrollar la tabla de verdad de cada una de las siguientes expresiones suma de productos estándar:

(a) $\bar{A}B + AB\bar{C} + \bar{A}\bar{C} + A\bar{B}C$ **(b)** $\bar{X} + Y\bar{Z} + WZ + X\bar{Y}Z$

32. Desarrollar la tabla de verdad de cada una de las siguientes expresiones producto de sumas estándar:

(a) $(\bar{A} + \bar{B} + \bar{C})(A + B + C)(A + \bar{B} + C)$

(b) $(\bar{A} + B + \bar{C} + D)(A + \bar{B} + C + \bar{D})(A + \bar{B} + \bar{C} + D)(\bar{A} + B + C + \bar{D})$

33. Desarrollar la tabla de verdad de cada una de las siguientes expresiones producto de sumas estándar:

(a) $(A + B)(A + C)(A + B + C)$

(b) $(A + \bar{B})(A + \bar{B} + \bar{C})(B + C + \bar{D})(\bar{A} + B + \bar{C} + D)$

34. Para cada tabla de verdad de la Figura 4.58, obtener una expresión suma de productos estándar y un producto de sumas estándar.

ABC	*X*
0 0 0	0
0 0 1	1
0 1 0	0
0 1 1	0
1 0 0	1
1 0 1	1
1 1 0	0
1 1 1	1

(a)

ABC	*X*
0 0 0	0
0 0 1	0
0 1 0	0
0 1 1	0
1 0 0	0
1 0 1	1
1 1 0	1
1 1 1	1

(b)

ABCD	*X*
0 0 0 0	1
0 0 0 1	1
0 0 1 0	0
0 0 1 1	1
0 1 0 0	0
0 1 0 1	1
0 1 1 0	1
0 1 1 1	0
1 0 0 0	0
1 0 0 1	1
1 0 1 0	0
1 0 1 1	0
1 1 0 0	1
1 1 0 1	0
1 1 1 0	0
1 1 1 1	0

(c)

ABCD	*X*
0 0 0 0	0
0 0 0 1	0
0 0 1 0	1
0 0 1 1	0
0 1 0 0	1
0 1 0 1	1
0 1 1 0	0
0 1 1 1	1
1 0 0 0	0
1 0 0 1	0
1 0 1 0	0
1 0 1 1	1
1 1 0 0	1
1 1 0 1	0
1 1 1 0	0
1 1 1 1	1

(d)

FIGURA 4.58

SECCIÓN 4.8 Mapas de Karnaugh

35. Dibujar un mapa de Karnaugh de 3 variables y etiquetar cada celda según su valor binario.

36. Dibujar un mapa de Karnaugh de 4 variables y etiquetar cada celda según su valor binario.

37. Escribir los términos producto estándar correspondientes a cada celda de un mapa de Karnaugh de 3 variables.

SECCIÓN 4.9 Minimización de una suma de productos mediante el mapa de Karnaugh

38. Utilizar un mapa de Karnaugh para hallar la suma de productos mínima para cada una de las expresiones siguientes.

(a) $\bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}C + A\bar{B}C$ **(b)** $AC(\bar{B} + C)$

(c) $\bar{A}(BC + B\bar{C}) + A(BC + B\bar{C})$ **(d)** $\bar{A}\bar{B}\bar{C} + A\bar{B}\bar{C} + \bar{A}B\bar{C} + AB\bar{C}$

39. Utilizar un mapa de Karnaugh para simplificar las expresiones siguientes a su forma suma de productos mínima.

(a) $\bar{A}\bar{B}\bar{C} + A\bar{B}C + \bar{A}BC + AB\bar{C}$ **(b)** $AC[\bar{B} + B(B + \bar{C})]$

(c) $DE\bar{F} + \bar{D}E\bar{F} + \bar{D}\bar{E}\bar{F}$

40. Expandir las expresiones siguientes a su forma suma de productos estándar.

(a) $AB + A\bar{B}C + ABC$ **(b)** $A + BC$

(c) $A\bar{B}\bar{C}D + AC\bar{D} + B\bar{C}D + \bar{A}BC\bar{D}$ **(d)** $A\bar{B} + A\bar{B}\bar{C}D + CD + B\bar{C}D + ABCD$

41. Minimizar las expresiones del Problema 40 utilizando un mapa de Karnaugh.

42. Utilizar un mapa de Karnaugh para reducir las expresiones siguientes a su forma suma de productos mínima.

(a) $A + B\bar{C} + CD$

(b) $\bar{A}\bar{B}\bar{C}\bar{D} + \bar{A}\bar{B}\bar{C}D + ABCD + ABC\bar{D}$

(c) $\bar{A}B(\bar{C}\bar{D} + \bar{C}D) + AB(\bar{C}\bar{D} + \bar{C}D) + A\bar{B}\bar{C}D$

(d) $(\bar{A}\bar{B} + A\bar{B})(CD + C\bar{D})$

(e) $\bar{A}\bar{B} + A\bar{B} + \bar{C}\bar{D} + C\bar{D}$

43. Reducir la función especificada en la tabla de verdad de la Figura 4.59 a su forma suma de productos mínima mediante un mapa de Karnaugh.

44. Utilizar el mapa de Karnaugh para implementar la forma suma de productos mínima de la función lógica especificada en la tabla de verdad de la Figura 4.60.

45. Resolver el Problema 44 para una situación en que las seis últimas combinaciones binarias no están permitidas.

Entradas			Salida
A	*B*	*C*	*X*
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

FIGURA 4.59

Entradas				Salida
A	*B*	*C*	*D*	*X*
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	0
1	1	1	1	1

FIGURA 4.60

SECCIÓN 4.10 Minimización de un producto de sumas mediante el mapa de Karnaugh

46. Utilizar un mapa de Karnaugh para hallar la suma de productos mínima de las siguientes expresiones:

(a) $(A+B+C)(\bar{A}+\bar{B}+\bar{C})(A+\bar{B}+C)$

(b) $(X+\bar{Y})(\bar{X}+Z)(X+\bar{Y}+\bar{Z})(\bar{X}+\bar{Y}+Z)$

(c) $A(B+\bar{C})(\bar{A}+C)(A+\bar{B}+C)(\bar{A}+B+\bar{C})$

47. Utilizar un mapa de Karnaugh para simplificar las siguientes expresiones a su forma producto de sumas mínima:

(a) $(A+\bar{B}+C+\bar{D})(\bar{A}+B+\bar{C}+D)(\bar{A}+\bar{B}+\bar{C}+\bar{D})$

(b) $(X+\bar{Y})(W+\bar{Z})(\bar{X}+\bar{Y}+\bar{Z})(W+X+Y+Z)$

48. Para la función especificada en la tabla de verdad de la Figura 4.59, determinar el producto de sumas mínimo mediante el mapa de Karnaugh.

49. Determinar el producto de sumas mínimo para la función de la tabla de verdad de la Figura 4.60.

50. Convertir cada una de las siguientes expresiones producto de sumas mínimo a la forma de suma de productos mínima utilizando un mapa de Karnaugh.

(a) $(A+\bar{B})(A+\bar{C})(\bar{A}+\bar{B}+C)$

(b) $(\bar{A}+B)(\bar{A}+\bar{B}+\bar{C})(B+\bar{C}+D)(A+\bar{B}+C+\bar{D})$

SECCIÓN 4.11 Mapa de Karnaugh de cinco variables

51. Minimizar la siguiente suma de productos utilizando un mapa de Karnaugh.

$$X = \bar{A}B\bar{C}D\bar{E} + \bar{A}\bar{B}\bar{C}DE + A\bar{B}\bar{C}DE + AB\bar{C}\bar{D}\bar{E} + \bar{A}BCD\bar{E} + \bar{A}BC\bar{D}E + \bar{A}\bar{B}\bar{C}\bar{D}\bar{E} + \bar{A}\bar{B}CDE + AB\bar{C}D\bar{E} + AB\bar{C}DE$$

52. Aplicar el mapa de Karnaugh para minimizar la siguiente suma de productos.

$$A = \bar{V}WXYZ + V\bar{W}XYZ + VW\bar{X}YZ + VWX\bar{Y}Z + VWXY\bar{Z} + \bar{V}\bar{W}\bar{X}\bar{Y}\bar{Z} + \bar{V}\bar{W}\bar{X}Y\bar{Z} + \bar{V}\bar{W}X\bar{Y}\bar{Z} + \bar{V}W\bar{X}\bar{Y}\bar{Z}$$

SECCIÓN 4.12 VHDL (opcional)

53. Escribir un programa VHDL para el circuito lógico de la Figura 4.61.

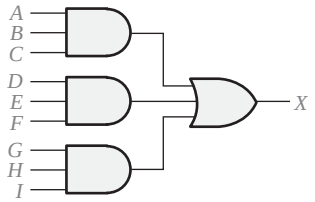


FIGURA 4.61

54. Escribir un programa VHDL para la expresión

$$Y = A\bar{B}C + \bar{A}\bar{B}C + A\bar{B}\bar{C} + \bar{A}BC$$

Aplicación a los sistemas digitales

55. Si es necesario elegir un tipo de display para trabajar bajo condiciones de baja luminosidad, ¿cuál se seleccionaría, un display de 7 segmentos de diodos LED o de cristal líquido? ¿Por qué?

56. Explicar por qué los códigos 1010, 1011, 1100, 1101, 1110 y 1111 pertenecen a la categoría de condiciones "indiferentes" en las aplicaciones con displays de 7 segmentos.

57. Para el segmento *b*, ¿cuántas puertas e inversores menos se necesitan para implementar la suma de productos mínima con respecto a la suma de productos estándar?

58. Repetir el Problema 57 para la lógica de los segmentos *c* hasta *g*.

Problemas especiales de diseño

59. La lógica del segmento *a* de la Figura 4.52 produce una salida a nivel ALTO para activar el segmento, ocurriendo lo mismo para el resto de los segmentos. Si se utiliza un display de 7-segmentos que requiere un nivel BAJO para activar cada segmento, modificar adecuadamente la lógica del segmento.

60. Rediseñar la lógica del segmento *a* utilizando un producto de sumas mínimo ¿Cuál es más sencilla, la suma de productos mínima o el producto de sumas mínimo?

61. Repetir el Problema 60 para los segmentos *b* hasta *g*.

62. Resumir los resultados del rediseño que se ha hecho en los Problemas 60 y 61, y recomendar el mejor diseño en función del mínimo número de circuitos integrados. Especificar los tipos de CI que se utilizarían.

RESPUESTAS

REVISIONES DE CADA SECCIÓN

SECCIÓN 4.1 **Operaciones y expresiones booleanas**

1. $\overline{A} = \overline{0} = 1$ **2.** $A = 1, B = 1, C = 0;\ \overline{A} + \overline{B} + C = \overline{1} + \overline{1} + 0 = 0 + 0 + 0 = 0$

3. $A = 1, B = 0, C = 1; A\overline{B}C = 1 \cdot \overline{0} \cdot 1 = 1 \cdot 1 \cdot 1 = 1$

SECCIÓN 4.2 **Leyes y reglas del álgebra booleana**

1. $A + (B + C + D) = (A + B + C) + D$ **2.** $A(B + C + D) = AB + AC + AD$

SECCIÓN 4.3 **Teoremas de DeMorgan**

1. (a) $\overline{ABC} + (\overline{\overline{D} + E}) = \overline{A} + \overline{B} + \overline{C} + D\overline{E}$ **(b)** $\overline{(A + B)C} = \overline{A}\overline{B} + \overline{C}$

(c) $\overline{A + B + C} + \overline{\overline{D}E} = \overline{A}\overline{B}\overline{C} + D + \overline{E}$

SECCIÓN 4.4 **Análisis booleano de los circuitos lógicos**

1. $(C + D)B + A$

2. Tabla de verdad abreviada: la expresión es 1 cuando *A* es 1 o cuando *B* y *C* son 1, o cuando *B* y *D* son 1. La expresión es 0 para todas las demás combinaciones.

SECCIÓN 4.5 **Simplificación mediante el álgebra de Boole**

1. **1. (a)** $A + AB + A\overline{B}C = A$ **(b)** $(\overline{A} + B)C + ABC = C(\overline{A} + B)$

(c) $A\overline{B}C(BD + CDE) + A\overline{C} = A(\overline{C} + \overline{B}DE)$

2. **(a)** *Original:* 2 puertas AND, 1 puerta OR, 1 inversor. *Simplificada:* sin puertas (conexión directa).

(b) *Original:* 2 puertas OR, 2 puertas AND, 1 inversor. *Simplificada:* 1 puerta OR, 1 puerta AND, 1 inversor.

(c) *Original:* 5 puertas AND, 2 puertas OR, 2 inversores. *Simplificada:* 1 puertas AND, 1 puerta OR, 2 inversores.

SECCIÓN 4.6 Formas estándar de las expresiones booleanas

1. **(a)** Suma de productos **(b)** Producto de sumas estándar
(c) Suma de productos estándar **(d)** Producto de sumas

2. **(a)** $AB\overline{C}\overline{D} + AB\overline{C}D + ABC\overline{D} + ABCD + \overline{A}B\overline{C}D + \overline{A}BCD + \overline{A}\overline{B}C\overline{D} + \overline{A}BC\overline{D}$
(c) Ya está en forma estándar

3. **(b)** Ya está en forma estándar
(d) $(A + \overline{B} + \overline{C})(A + \overline{B} + C)(A + B + \overline{C})(A + B + C)$

SECCIÓN 4.7 Expresiones booleanas y tablas de verdad

1. $2^5 = 32$

2. $0110 \rightarrow \overline{W}XY\overline{Z}$

3. $1100 \rightarrow \overline{W} + \overline{X} + Y + Z$

SECCIÓN 4.8 Mapas de Karnaugh

1. **(a)** celda superior izquierda: 000 **(b)** celda inferior derecha: 101
(c) celda inferior izquierda: 100 **(d)** celda superior derecha: 001

2. **(a)** celda superior izquierda: $\overline{X}\overline{Y}\overline{Z}$ **(b)** celda inferior derecha: $X\overline{Y}Z$
(c) celda inferior izquierda: $X\overline{Y}\overline{Z}$ **(d)** celda superior derecha: $\overline{X}\overline{Y}Z$

3. **(a)** celda superior izquierda: 0000 **(b)** celda inferior derecha: 1010
(c) celda inferior izquierda: 1000 **(d)** celda superior derecha: 0010

4. **(a)** celda superior izquierda: $\overline{W}\overline{X}\overline{Y}\overline{Z}$ **(b)** celda inferior derecha: $W\overline{X}YZ$
(c) celda inferior izquierda: $W\overline{X}\overline{Y}\overline{Z}$ **(d)** celda superior derecha: $\overline{W}\overline{X}YZ$

SECCIÓN 4.9 Minimización de una suma de productos mediante el mapa de Karnaugh

1. Mapa de 8 celdas para 3 variables; mapa de 16 celdas para 4 variables.

2. $AB + B\overline{C} + \overline{A}\overline{B}C$

3. **(a)** $\overline{A}\overline{B}\overline{C} + \overline{A}BC + ABC + AB\overline{C}$
(b) $\overline{A}\overline{B}\overline{C} + \overline{A}\overline{B}C + \overline{A}B\overline{C} + ABC + A\overline{B}\overline{C} + A\overline{B}C$
(c) $\overline{A}\overline{B}\overline{C}\overline{D} + \overline{A}\overline{B}\overline{C}D + \overline{A}B\overline{C}\overline{D} + \overline{A}B\overline{C}D + \overline{A}BCD + \overline{A}BC\overline{D} + A\overline{B}\overline{C}D + A\overline{B}CD$
(d) $\overline{A}\overline{B}\overline{C}\overline{D} + \overline{A}\overline{B}\overline{C}D + \overline{A}B\overline{C}\overline{D} + \overline{A}B\overline{C}D + \overline{A}BC\overline{D} + \overline{A}BCD + A\overline{B}C\overline{D} + A\overline{B}CD + \overline{A}\overline{B}C\overline{D} + \overline{A}BC\overline{D} + ABC\overline{D} + A\overline{B}C\overline{D}$

SECCIÓN 4.10 Minimización de un producto de sumas mediante el mapa de Karnaugh

1. Cuando se pasa a un mapa de Karnaugh un producto de sumas, los 0s se colocan en las celdas cuyos valores hacen que el término suma estándar sea cero; cuando se pasa a un mapa de Karnaugh una suma de productos, los 1s se colocan en la celdas que tienen los mismos valores que los términos producto.

2. 0 en la celda 1011: $\overline{A} + B + \overline{C} + \overline{D}$ **3.** 1 en la celda 0010: $\overline{A}\overline{B}C\overline{D}$

SECCIÓN 4.11 Mapas de Karnaugh de cinco variables

1. Existen 32 combinaciones de las 5 variables ($2^5 = 32$).

2. $X = 1$, ya que la función es 1 para todas las posibles combinaciones de las 5 variables.

SECCIÓN 4.12 VHDL (opcional)

1. HDL es un lenguaje de descripción hardware para dispositivos lógicos programables.

2. Entidad y arquitectura.

3. La entidad especifica las entradas y las salidas de una función lógica.

4. La arquitectura especifica la operación de una función lógica.

PROBLEMAS RELACIONADOS

4.1 $\bar{A} + B = 0$ cuando $A = 1$ y $B = 0$.

4.2 $\bar{A}\bar{B} = 1$ cuando $A = 0$ y $B = 0$.

4.3 XYZ

4.4 $W + X + Y + Z$

4.5 $ABC\bar{D}\bar{E}$

4.6 $(A + \bar{B} + \bar{C}D)\bar{E}$

4.7 $\overline{ABCD} = \bar{A} + \bar{B} + \bar{C} + \bar{D}$

4.8 $A\bar{B}$

4.9 CD

4.10 $AB\bar{C} + \bar{A}C + \bar{A}\bar{B}$

4.11 $\bar{A} + \bar{B} + \bar{C}$

4.12 $\bar{A}B\bar{C} + AB + A\bar{C} + A\bar{B} + \bar{B}\bar{C}$

4.13 $W\bar{X}YZ + W\bar{X}Y\bar{Z} + W\bar{X}\bar{Y}\bar{Z} + \bar{W}\bar{X}\bar{Y}\bar{Z} + WX\bar{Y}Z + WX\bar{Y}\bar{Z}$

4.14 011, 101, 110, 010, 111. Sí

4.15 $(A + \bar{B} + C)(A + \bar{B} + \bar{C})(A + B + C)(\bar{A} + B + C)$

4.16 010, 100, 001, 111, 011. Sí

4.17 Las expresiones suma de productos y producto de sumas son equivalentes.

4.18 *Véase* la Tabla 4.11

4.19 *Véase* la Tabla 4.12.

A	*B*	*C*	*X*
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

TABLA 4.11

A	*B*	*C*	*X*
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

TABLA 4.12

4.20 Las expresiones suma de productos y producto de sumas son equivalentes.

4.21 *Véase* la Figura 4.62.

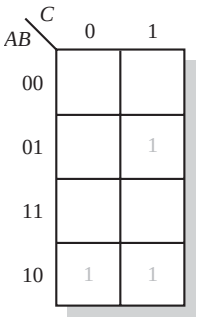


FIGURA 4.62

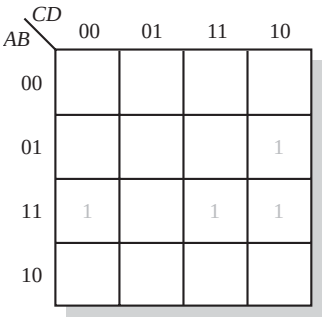


FIGURA 4.63

4.22 *Véase* la Figura 4.63.

4.23 *Véase* la Figura 4.64.

4.24 *Véase* la Figura 4.65.

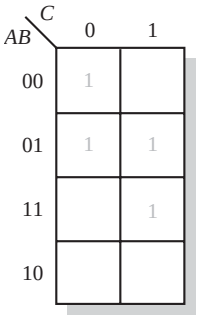


FIGURA 4.64

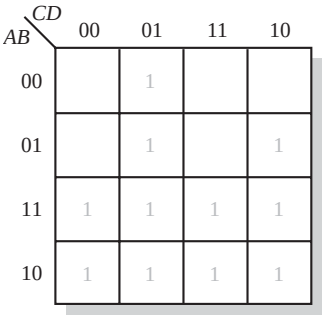


FIGURA 4.65

4.25 Ninguna otra forma.

4.26 $X = B + \bar{A}C + A\bar{C}D + C\bar{D}$

4.27 $X = \bar{D} + A\bar{B}C + B\bar{C} + \bar{A}B$

4.28 $Q = X + Y$

4.29 $Q = \bar{X}\bar{Y}\bar{Z} + W\bar{X}Z + \bar{W}YZ$

4.30 *Véase* la Figura 4.66.

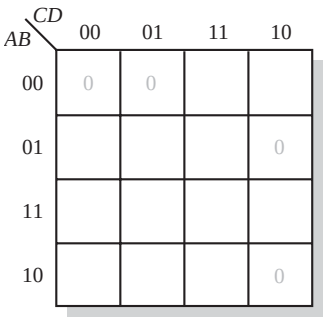


FIGURA 4.66

4.31 $Q = (X + \overline{Y})(X + \overline{Z})(\overline{X} + Y + Z)$

4.32 $Q = (\overline{X} + \overline{Y} + Z)(\overline{W} + \overline{X} + Z)(W + X + Y + Z)(W + \overline{X} + Y + \overline{Z})$

4.33 $Q = \overline{YZ} + \overline{XZ} + \overline{WY} + \overline{X}\overline{Y}Z$

4.34 $Y = \overline{D}\overline{E} + \overline{A}\overline{E} + \overline{B}\overline{C}\overline{E}$

4.35 X ⇐ (A **and** B)(C **and** D);

AUTOTEST

1. (d) **2.** (a) **3.** (b) **4.** (c) **5.** (d) **6.** (b) **7.** (a) **8.** (b)
9. (d) **10.** (d) **11.** (c) **12.** (b) **13.** (b) **14.** (c) **15.** (a) **16.** (c)
17. (a) **18.** (b) **19.** (c) **20.** (b) **21.** (c)

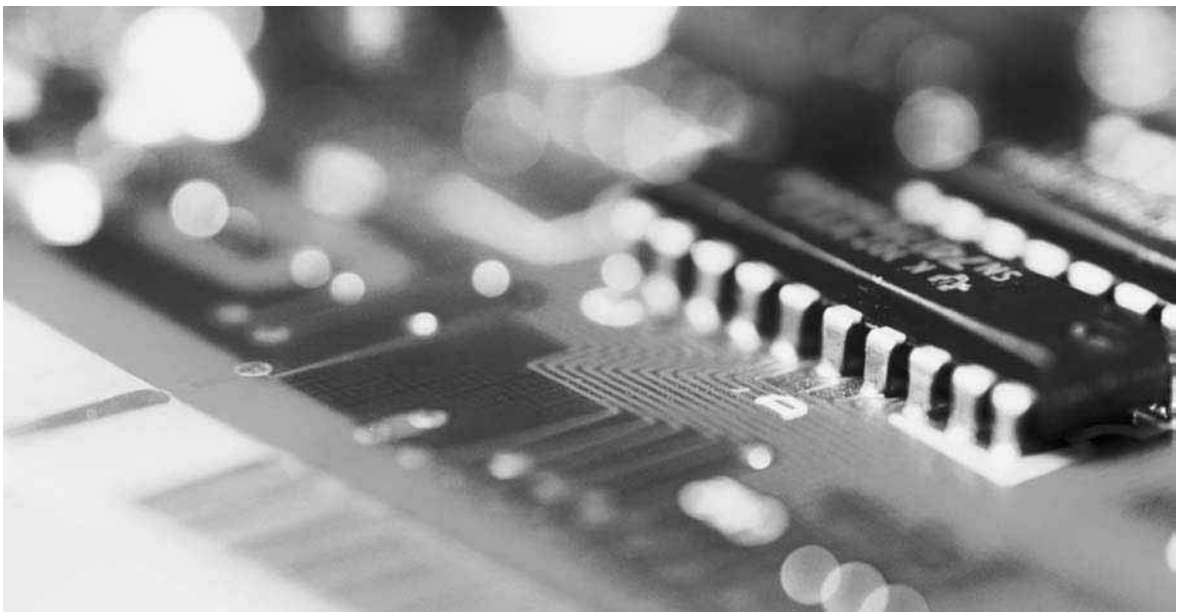
5

ANÁLISIS DE LA LÓGICA COMBINACIONAL

CONTENIDO DEL CAPÍTULO

OBJETIVOS DEL CAPÍTULO

- Analizar los circuitos lógicos combinacionales básicos, tales como AND-OR, AND-OR-Inversor, OR-exclusiva y NOR-exclusiva.
- Utilizar los circuitos AND-OR y AND-OR-Inversor para implementar expresiones como suma de productos y producto de sumas.
- Escribir la expresión booleana de salida de cualquier circuito lógico combinacional.
- Desarrollar la tabla de verdad a partir de la expresión de salida de un circuito lógico combinacional.



- Utilizar el mapa de Karnaugh para expandir una expresión de salida que contenga variables suprimidas en una suma de productos completa.
- Diseñar un circuito lógico combinacional para una expresión booleana de salida dada.
- Diseñar un circuito lógico combinacional para una tabla de verdad dada.
- Simplificar un circuito lógico combinacional a su forma mínima.
- Utilizar puertas NAND para implementar cualquier función lógica combinacional.
- Utilizar puertas NOR para implementar cualquier función lógica combinacional.
- Escribir programas VHDL para circuitos lógicos simples.
- Localizar fallos en los circuitos lógicos.
- Localizar fallos en los circuitos lógicos utilizando el seguimiento de señales y el análisis de las formas de onda.

PALABRAS CLAVE

- Puerta universal
- negativa-OR
- negativa-AND
- Componente
- Señal
- Nodo
- Seguimiento de señales

INTRODUCCIÓN

En los Capítulos 3 y 4, nos hemos ocupado de las puertas lógicas como elementos individuales y en sencillas combinaciones. Se han introducido las implementaciones de suma de productos y producto de sumas, que son las formas básicas de la lógica combinacional. Cuando se conectan puertas lógicas entre sí, con el fin de generar una determinada salida específica para determinadas combinaciones específicas de las variables de entrada, sin que haya implicado almacenamiento, el circuito resultante se califica como **lógica combinacional**. En la lógica combinacional, el nivel de salida depende siempre de la combinación de los niveles de entrada. Este capítulo amplía el material presentado en los capítulos anteriores y cubre el análisis, diseño y localización de fallos de diversos circuitos lógicos combinacionales. Se presenta el método estructural del VHDL y se aplica a la lógica combinacional.

APLICACIÓN A LOS SISTEMAS DIGITALES

Esta aplicación a un sistema digital ilustra los conceptos que se enseñan en este capítulo, mostrando cómo se puede usar la lógica combinacional para un propósito específico en una aplicación práctica. Se emplea un circuito lógico para controlar el nivel y la temperatura de un fluido en un tanque de almacenamiento. Manipulando las válvulas interna y externa, se controla la entrada y salida de flujo basándose en la información proporciona por las entradas del sensor de nivel. De forma opcional, también se explica cómo usar VHDL para describir la lógica.

5.1 CIRCUITOS LÓGICOS COMBINACIONALES BÁSICOS

En el Capítulo 4, ha aprendido que las expresiones suma de productos se implementan con una puerta AND para cada término producto y una puerta OR para sumar todos los términos producto. Esta implementación de la suma de productos se llama lógica AND-OR y es la forma básica para realizar las funciones estándar booleanas. En esta sección, se estudian las combinaciones AND-OR y AND-OR-Inversor y las puertas OR-exclusiva y NOR-exclusiva, que realmente son una forma de la lógica AND-OR.

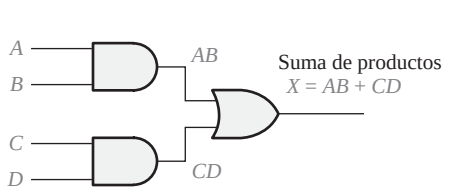
Al finalizar esta sección, el lector deberá ser capaz de:

■ Analizar y aplicar los circuitos AND-OR. ■ Analizar y aplicar los circuitos AND-OR-Inversor. ■ Analizar y aplicar las puertas OR-exclusiva. ■ Analizar y aplicar las puertas NOR-exclusiva.

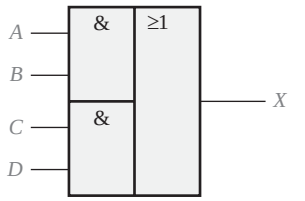
Lógica AND-OR

▲ *La lógica AND-OR genera expresiones suma de productos.*

La Figura 5.1(a) muestra un circuito AND-OR formado por dos puertas AND de dos entradas y una puerta OR de dos entradas. La Figura 5.1(b) corresponde al símbolo estándar rectangular ANSI. En el diagrama se indica la expresión booleana para las salidas de las puertas AND y la suma de productos resultante para la salida *X*. En general, un circuito AND-OR puede tener cualquier número de puertas AND, cada una de ellas con cualquier número de entradas.



(a) Diagrama lógico (símbolos distintivos del estándar ANSI).



(b) Símbolo rectangular del estándar ANSI.

FIGURA 5.1 Ejemplo de circuito lógico AND-OR.

En la Tabla 5.1 se presenta la tabla de verdad para el circuito lógico AND-OR de cuatro entradas. Las salidas de la puerta AND intermedia (las columnas *AB* y *CD*) también se indican en dicha tabla.

Un circuito AND-OR implementa directamente una suma de productos, suponiendo que se dispone de los complementos de las variables. La operación lógica del circuito AND-OR de la Figura 5.1 se enuncia así:

En un circuito lógico AND-OR de 4 entradas, la salida *X* es un nivel ALTO (1) sólo si las dos entradas *A* y *B* están a nivel ALTO (1) o si las dos entradas *C* y *D* están a nivel ALTO (1).

Circuito lógico AND-OR-Inversor

Cuando se complementa (invierte) la salida de un circuito AND-OR, se obtiene el circuito AND-OR-Inversor. Recuerde que el circuito AND-OR implementa directamente la suma de productos. El producto de sumas puede implementarse con un circuito lógico AND-OR-Inversor. Esto se ilustra de la forma siguiente, partiendo del producto de sumas y desarrollando la expresión AND-OR-Inversor correspondiente.

$$X = (\overline{A} + \overline{B})(\overline{C} + \overline{D}) = (\overline{AB})(\overline{CD}) = \overline{\overline{(\overline{AB})(\overline{CD})}} = \overline{\overline{AB} + \overline{CD}} = \overline{AB + CD}$$

Entradas						Salida
A	*B*	*C*	*D*	*AB*	*CD*	*X*
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	0	1	0	0	0	0
0	0	1	1	0	1	1
0	1	0	0	0	0	0
0	1	0	1	0	0	0
0	1	1	0	0	0	0
0	1	1	1	0	1	1
1	0	0	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	0	0	0
1	0	1	1	0	1	1
1	1	0	0	1	0	1
1	1	0	1	1	0	1
1	1	1	0	1	0	1
1	1	1	1	1	1	1

TABLA 5.1 Tabla de verdad para el circuito lógico AND-OR de la Figura 5.1.

EJEMPLO 5.1

En una determinada planta de procesamiento químico se emplea un elemento químico líquido en un proceso de fabricación. Dicho elemento químico se almacena en tres tanques diferentes. Un sensor de nivel en cada tanque genera una tensión a nivel ALTO cuando el nivel de líquido en el tanque cae por debajo de un punto especificado.

Diseñar un circuito para supervisar el nivel del elemento químico en cada tanque, que indique cuándo el nivel de dos tanques cualesquiera cae por debajo del punto especificado.

Solución El circuito AND-OR de la Figura 5.2 tiene entradas procedentes de los sensores de los tanques *A*, *B* y *C*. La puerta AND G_1 comprueba el nivel en los

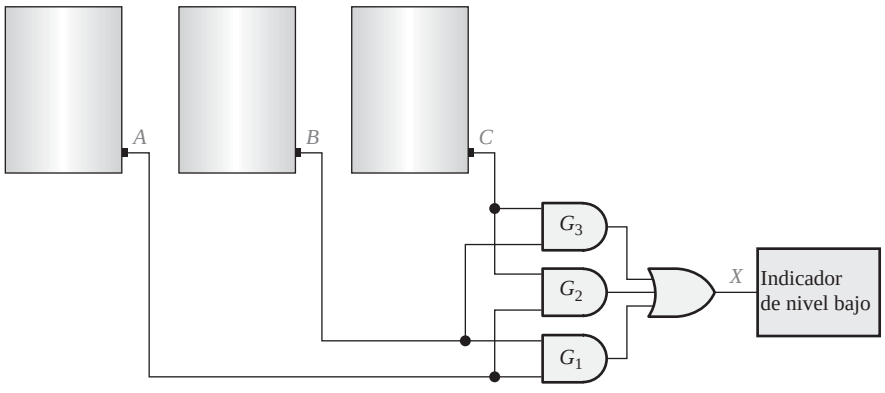


FIGURA 5.2

tanques *A* y *B*, la puerta G_2 comprueba los tanques *A* y *C*, y la puerta G_3 comprueba los tanques *B* y *C*. Cuando el nivel del elemento químico en dos tanques cualesquiera desciende demasiado, una de las puertas AND tendrá a nivel ALTO ambas entradas, haciendo que su salida sea un nivel ALTO, por lo que la salida final *X* de la puerta OR estará a nivel ALTO. Esta salida a nivel ALTO se usa entonces para activar un indicador, tal como una alarma luminosa o audible, como muestra la figura.

Problema relacionado* Escribir la expresión booleana en forma de suma de productos para el circuito lógico AND-OR de la Figura 5.2.

* Las respuestas se encuentran al final del capítulo.

El diagrama lógico de la Figura 5.3(a) muestra un circuito AND-OR-Inversor y el desarrollo de la expresión de salida como producto de sumas. En la parte (b) de la figura se presenta el símbolo rectangular estándar ANSI. En general, un circuito AND-OR-Inversor puede tener cualquier número de puertas AND, y cada una de ellas puede tener cualquier número de entradas.

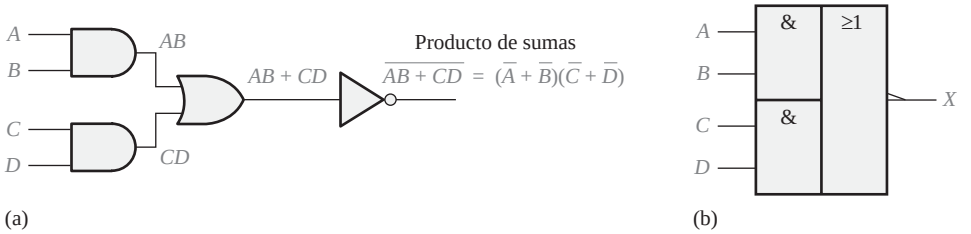


FIGURA 5.3 Un circuito AND-OR-Inversor genera una salida que es un producto de sumas.

La operación lógica del circuito AND-OR-Inversor de la Figura 5.3 se define como:

En un circuito lógico AND-OR-Inversor de 4 entradas, la entrada *X* es un nivel BAJO (0) si las dos entradas *A* y *B* están a nivel ALTO (1), o si las dos entradas *C* y *D* están a nivel ALTO (1).

Se puede desarrollar la tabla de verdad a partir de la Tabla 5.1, tabla de verdad del circuito AND-OR, cambiando simplemente, en la columna de salida, todos los 1s por 0s y todos los 0s por 1s.

EJEMPLO 5.2

Los sensores colocados en los tanques químicos del Ejemplo 5.1 se reemplazan por un nuevo modelo que genera una tensión a nivel BAJO en lugar de una tensión a nivel ALTO cuando el nivel de líquido en el tanque cae por debajo del punto crítico.

Modificar el circuito de la Figura 5.2 para trabajar con los diferentes niveles de entrada y generar una salida a nivel ALTO que active el indicador cuando el nivel de dos tanques caiga por debajo del punto crítico. Realizar el diagrama lógico.

Solución Los sensores de los tanques *A, B* y *C* se conectan a las entradas del circuito AND-OR-Inversor, como se muestra en la Figura 5.4. La puerta AND G_1 comprueba el nivel en los tanques *A* y *B*, la puerta G_2 comprueba los tanques *A* y *C*, y la puerta G_3 comprueba los tanques *B* y *C*. Cuando el nivel del elemento

químico en dos tanques cualesquiera desciende, al menos una de las entradas de cada una de las puertas AND estará a nivel BAJO, haciendo que su salida sea un nivel BAJO, por lo que la salida final X del inversor estará a nivel ALTO. Esta salida a nivel ALTO se usa entonces para activar un indicador.

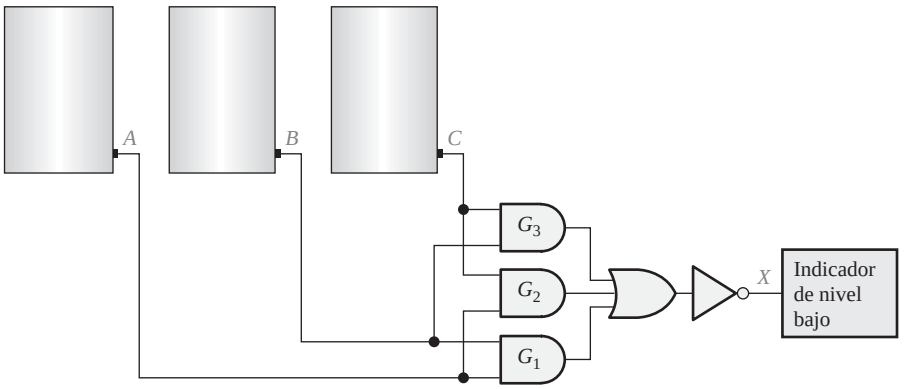
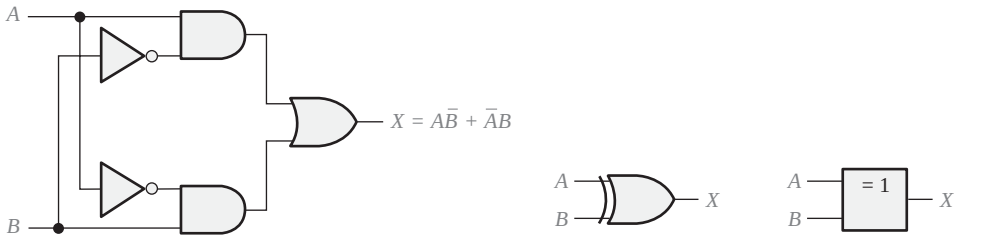


FIGURA 5.4

Problema relacionado Escribir la expresión booleana para el circuito lógico AND-OR-Inversor de la Figura 5.4 y mostrar que la salida es un nivel ALTO (1) cuando dos entradas cualesquiera de entre A, B y C estén a nivel BAJO (0).

Circuito lógico OR-exclusiva

En el Capítulo 3 se ha presentado la puerta OR-exclusiva. Aunque, debido a su importancia este circuito se considera como una puerta lógica con su propio símbolo distintivo, realmente es una combinación de dos puertas AND, una puerta OR y dos inversores, tal y como muestra la Figura 5.5(a). En las Figuras 5.5 (b) y (c) se presentan los dos símbolos lógicos estándar ANSI.



(a) Diagrama lógico

(b) Símbolo distintivo ANSI

(c) Símbolo rectangular ANSI

FIGURA 5.5 Diagrama lógico y símbolos del circuito OR-exclusiva.

La expresión de salida para el circuito de la Figura 5.5 es

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

La evaluación de esta expresión se muestra en la tabla de verdad de la Tabla 5.2. Observe que la salida está a nivel ALTO sólo cuando las dos entradas están a niveles opuestos. A menudo, se emplea el operador especial OR-exclusiva ⊕, por lo que la expresión $X = A\overline{B} + \overline{A}B$ puede enunciarse como "*X* es igual a *A* OR-Exclusiva *B*" y puede expresarse como:

$$X = A \oplus B$$

A	*B*	*X*
0	0	0
0	1	1
1	0	1
1	1	0

TABLA 5.2 Tabla de verdad para la puerta OR-exclusiva.

Circuito lógico NOR-exclusiva

Como sabemos, el complemento de la función OR-exclusiva es la función NOR-exclusiva, la cual se obtiene del siguiente modo:

$$X = \overline{A\overline{B} + \overline{A}B} = \overline{(A\overline{B})}\,\overline{(\overline{A}B)} = (\overline{A} + B)(A + \overline{B}) = \overline{A}\overline{B} + AB$$

Observe que la salida *X* es un nivel ALTO sólo cuando las dos entradas, *A* y *B*, están al mismo nivel.

La función NOR-exclusiva puede implementarse invirtiendo la salida de un circuito OR-exclusiva, como muestra la Figura 5.6(a), o bien se puede implementar directamente a partir de la expresión $\overline{A}\overline{B} + AB$, como muestra la parte (b) de la figura.

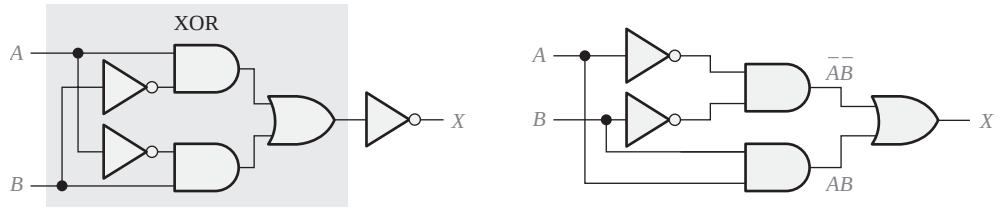


FIGURA 5.6 Dos formas equivalentes de implementar el circuito NOR-exclusiva.

REVISIÓN DE LA SECCIÓN 5.1

Las respuestas se encuentran al final del capítulo.

1. Determinar la salida (1 o 0) de un circuito AND-OR-Inversor de 4 variables para cada una de las siguientes condiciones de entrada:
   (a) $A = 1, B = 0, C = 1, D = 0$
   (b) $A = 1, B = 1, C = 0, D = 1$
   (c) $A = 0, B = 1, C = 1, D = 1$
2. Determinar la salida (1 o 0) de una puerta OR-exclusiva para cada una de las siguientes condiciones de entrada:

(a) $A = 1, B = 0$ **(b)** $A = 1, B = 1$

(c) $A = 0, B = 1,$ **(d)** $A = 0, B = 0$

3. Desarrollar la tabla de verdad para un determinado circuito lógico de 3 entradas con la siguiente expresión de salida $X = A\bar{B}C + \bar{A}BC + \bar{A}\bar{B}\bar{C} + AB\bar{C} + ABC$.

4. Dibujar un diagrama lógico para un circuito NOR-exclusiva.

5.2 IMPLEMENTACIÓN DE LA LÓGICA COMBINACIONAL

En esta sección se emplean ejemplos para ilustrar cómo implementar un circuito lógico a partir de una expresión booleana o una tabla de verdad. También se tratará la minimización de un circuito lógico utilizando los métodos vistos en el Capítulo 4.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Implementar un circuito lógico a partir de una expresión booleana. ■ Implementar un circuito lógico a partir de una tabla de verdad. ■ Minimizar un circuito lógico.

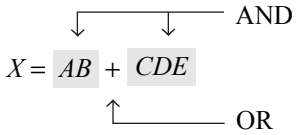
Obtención del circuito lógico a partir de una expresión booleana

▲ *Para toda expresión booleana existe un circuito lógico y para todo circuito lógico existe una expresión booleana.*

Examinemos la siguiente expresión booleana:

$$X = AB + CDE$$

Una rápida inspección revela que esta expresión está formada por dos términos, *AB* y *CDE*, y tienen un dominio de cinco variables. El primer término está definido por la operación AND entre *A* y *B*, y el segundo término queda definido por la multiplicación (AND) de *C*, *D* y *E*. Después, los dos términos se suman (operación OR) para dar lugar a la salida *X*. Estas operaciones se indican en la siguiente estructura de la expresión:



Observe que, en esta expresión, las operaciones AND son dos términos individuales, *AB* y *CDE*, que deben efectuarse *antes* de aplicar la operación OR.

Para implementar esta expresión booleana se requiere una puerta AND de 2 entradas para obtener el término *AB*, y una puerta AND de tres entradas para generar el término *CDE*. Para combinar los dos términos obtenidos en las puertas AND se requiere una puerta OR de 2 entradas. El resultado se muestra en la Figura 5.7.

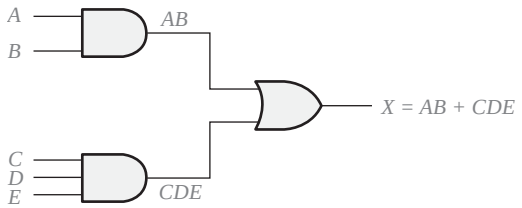


FIGURA 5.7 Circuito lógico para $X = AB + CDE$.

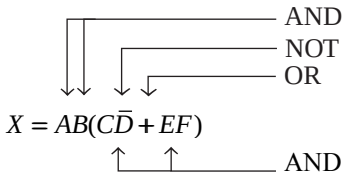
NOTAS INFORMÁTICAS

Muchos programas de control requieren que una computadora realice operaciones lógicas. Un controlador es un programa de control que se emplea con los periféricos de la computadora. Por ejemplo, un controlador de ratón requiere pruebas lógicas para determinar si se ha pulsado un botón y más operaciones lógicas para determinar si se ha movido en sentido horizontal o en sentido vertical. En el núcleo de un procesador se encuentra la unidad aritmético lógica (ALU), que realiza dichas operaciones lógicas mediante instrucciones de programa. Toda la lógica que se describe en este capítulo también puede ser realizada por la ALU, proporcionando las instrucciones adecuadas.

Veamos otro ejemplo, vamos a implementar la siguiente expresión:

$$X = AB(C\overline{D} + EF)$$

Un examen de esta expresión muestra que se aplica la operación AND a los términos AB y $(C\overline{D} + EF)$. El término $C\overline{D} + EF$ se obtiene aplicando primero la operación AND a C y $\overline{D}$ y luego a E y F, y aplicando por último la operación OR a estos dos términos. Esta estructura se indica respecto de la expresión del siguiente modo:



Antes de poder implementar la expresión completa, hay que crear el término suma $C\overline{D} + EF$. Para ello, previamente hay que disponer de los términos $C\overline{D}$ y EF, pero antes de obtener el término $C\overline{D}$ es necesario tener $\overline{D}$. Luego, como puede ver, las operaciones lógicas deben efectuarse en el orden adecuado.

Las puertas lógicas necesarias para implementar $X = AB(C\overline{D} + EF)$ son las siguientes:

1. Un inversor para obtener $\overline{D}$.
2. Dos puertas AND de 2 entradas para obtener $C\overline{D}$ y EF.
3. Una puerta OR de 2 entradas para obtener $C\overline{D} + EF$.
4. Una puerta AND de 3 entradas para generar X.

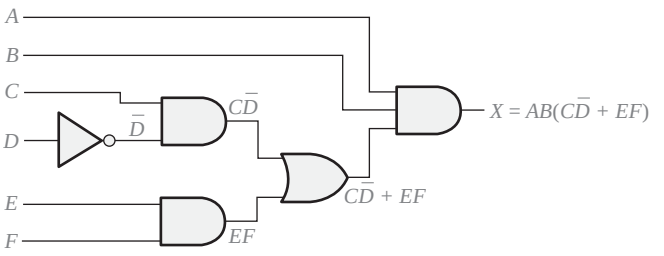
En la Figura 5.8(a) se muestra el circuito lógico correspondiente a esta expresión. Observe que hay un máximo de tres puertas y un inversor entre una entrada y la salida del circuito (de la entrada D a la salida). A menudo, el retardo de propagación a través del circuito lógico es una consideración de gran importancia. Los retardos de propagación se suman, luego cuantos más inversores y puertas haya entre la entrada y la salida, mayor será el retardo de propagación.

A menos que un término intermedio, tal como $C\overline{D} + EF$ de la Figura 5.8(a), se requiera como salida para algún otro propósito, usualmente lo mejor es reducir el circuito a su suma de productos con el fin de reducir el tiempo de retardo de propagación total. La expresión se convierte en suma de productos como sigue, y el circuito resultante se muestra en la Figura 5.8(b).

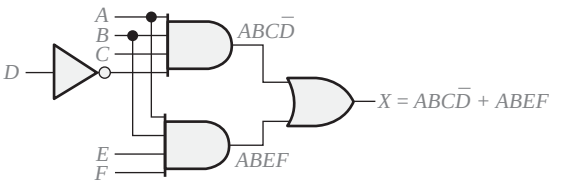
$$AB(C\overline{D} + EF) = ABC\overline{D} + ABEF$$

Obtención del circuito lógico a partir de la tabla de verdad

Si en lugar de partir de una expresión se parte de una tabla de verdad, puede escribirse la suma de productos que se obtiene de la tabla de verdad, y luego implementar el circuito lógico. La Tabla 5.3 especifica una función lógica.



(a)



(b) Implementación de la suma de productos del circuito de la parte (a)

FIGURA 5.8 Circuitos lógicos para la expresión $X = AB(C\overline{D}+EF) = ABC\overline{D} + ABEF$.

Entradas			Salida	
A	***B***	***C***	***X***	**Término producto**
0	0	0	0	
0	0	1	0	
0	1	0	0	
0	1	1	1	$\overline{A}BC$
1	0	0	1	$A\overline{B}\overline{C}$
1	0	1	0	
1	1	0	0	
1	1	1	0	

TABLA 5.3

La expresión booleana suma de productos que se obtiene a partir de la tabla de verdad, haciendo la suma (OR) de los productos para los que $X = 1$ es

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

El primer término de la expresión se obtiene calculando el producto lógico (AND) de las tres variables $\overline{A}, B$ y C. El segundo término se obtiene multiplicando (operación AND) las tres variables A, $\overline{B}$ y $\overline{C}$.

Las puertas lógicas necesarias para implementar esta expresión son: tres inversores para obtener las variables $\overline{A}, \overline{B}$ y $\overline{C}$; dos puertas AND de 3 entradas para obtener los términos $\overline{A}BC$ y $A\overline{B}\overline{C}$ y una puerta OR de 2 entradas para obtener la forma de la función final de salida $\overline{A}BC + A\overline{B}\overline{C}$.

La Figura 5.9 ilustra la implementación de esta función lógica.

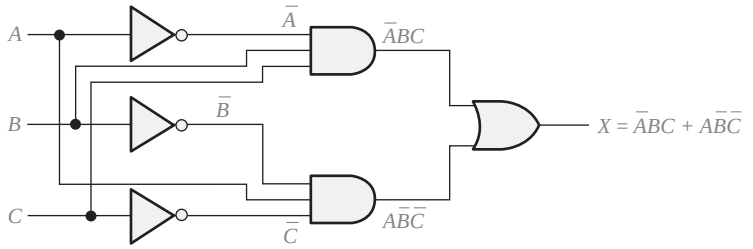


FIGURA 5.9 Circuito lógico para $X = \overline{A}BC + A\overline{B}\overline{C}$.

EJEMPLO 5.3

Diseñar un circuito lógico para implementar la operación especificada en la tabla de verdad indicada en la Tabla 5.4.

Entradas			Salida	
A	***B***	***C***	***X***	**Término producto**
0	0	0	0	
0	0	1	0	
0	1	0	0	
0	1	1	1	$\overline{A}BC$
1	0	0	0	
1	0	1	1	$A\overline{B}C$
1	1	0	1	$AB\overline{C}$
1	1	1	0	

TABLA 5.4

Solución Observe que $X = 1$ sólo para tres de las condiciones de entrada. Por tanto, la expresión lógica es: $X = \overline{A}BC + A\overline{B}C + AB\overline{C}$.

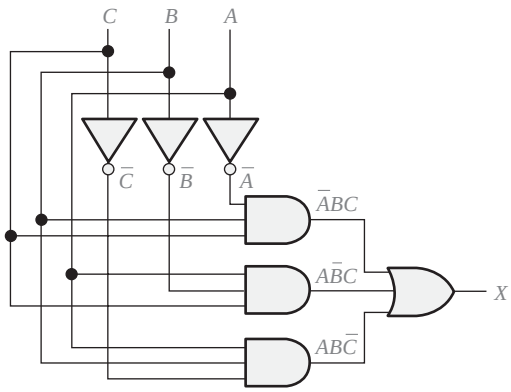


FIGURA 5.10

Las puertas lógicas requeridas son tres inversores, tres puertas AND de 3 entradas y una puerta OR de 3 entradas. El circuito lógico es el que se muestra en la Figura 5.10.

Problema relacionado Determinar si el circuito lógico de la Figura 5.10 puede simplificarse.

EJEMPLO 5.4

Desarrollar un circuito lógico con cuatro variables de entrada que sólo genera un 1 en la salida cuando tres variables de entrada son 1.

Solución Para cuatro variables, existen dieciséis posibles combinaciones; de éstas las que contienen tres 1s son las que se enumeran en la Tabla 5.5, junto con el correspondiente término producto.

A	*B*	*C*	*D*	Término producto
0	1	1	1	$\bar{A}BCD$
1	0	1	1	$A\bar{B}CD$
1	1	0	1	$AB\bar{C}D$
1	1	1	0	$ABC\bar{D}$

TABLA 5.5

Se aplica la operación OR a los productos para obtener la siguiente expresión:

$$X = \bar{A}BCD + A\bar{B}CD + AB\bar{C}D + ABC\bar{D}$$

Esta expresión se implementa con el circuito lógico AND-OR de la Figura 5.11.

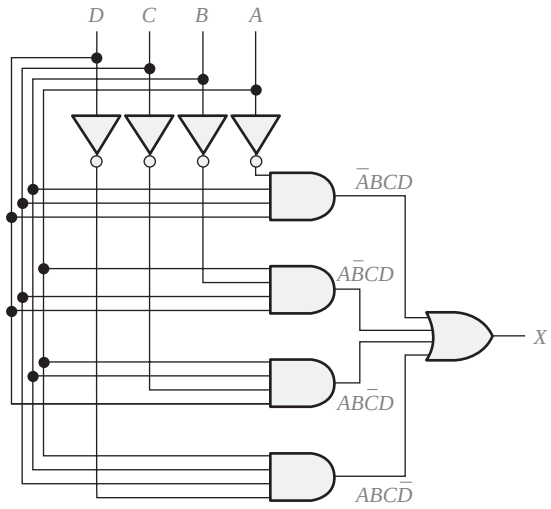


FIGURA 5.11

Problema relacionado Determinar si el circuito lógico de la Figura 5.11 puede simplificarse.

EJEMPLO 5.5

Reducir el circuito lógico combinacional de la Figura 5.12 a una forma mínima.

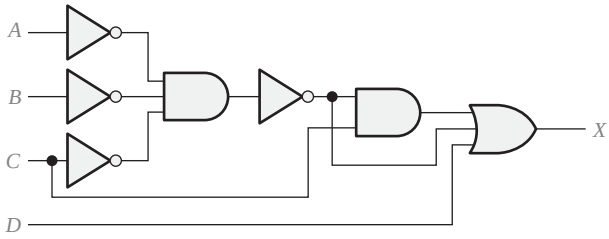


FIGURA 5.12

Solución La expresión para la salida del circuito es:

$$X = (\overline{\overline{A}\overline{B}\overline{C}})C + \overline{\overline{A}\overline{B}\overline{C}} + D$$

Aplicando el teorema de DeMorgan y el álgebra booleana se tiene

$$\begin{aligned} X &= (\overline{\overline{A}} + \overline{\overline{B}} + \overline{\overline{C}})C + \overline{\overline{A}} + \overline{\overline{B}} + \overline{\overline{C}} + D \\ &= AC + BC + CC + A + B + C + D \\ &= AC + BC + C + A + B + \not{C} + D \\ &= C(A + B + 1) + A + B + D \\ X &= A + B + C + D \end{aligned}$$

El circuito simplificado es una puerta OR de cuatro entradas, como muestra la Figura 5.13.

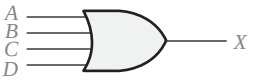


FIGURA 5.13

Problema relacionado Verificar la expresión minimizada $A + B + C + D$ utilizando un mapa de Karnaugh.

EJEMPLO 5.6

Minimizar el circuito lógico combinacional de la Figura 5.14. Los inversores para las variables complementadas no se muestran.

Solución La expresión de salida es:

$$X = A\overline{B}\overline{C} + AB\overline{C}\overline{D} + \overline{A}\overline{B}\overline{C}D + \overline{A}\overline{B}\overline{C}\overline{D}$$

Desarrollando el primer término para incluir las variables que faltan D y $\overline{D}$ tenemos,

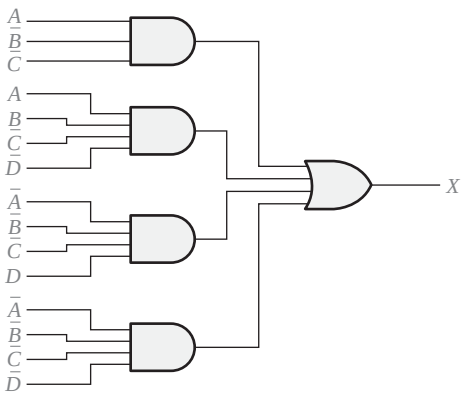


FIGURA 5.14

$$X = A\bar{B}\bar{C}(D + \bar{D}) + AB\bar{C}\bar{D} + \bar{A}\bar{B}\bar{C}D + \bar{A}\bar{B}\bar{C}\bar{D}$$

$$= A\bar{B}\bar{C}D + A\bar{B}\bar{C}\bar{D} + AB\bar{C}\bar{D} + \bar{A}\bar{B}\bar{C}D + \bar{A}\bar{B}\bar{C}\bar{D}$$

Esta suma de productos completa se traslada y simplifica en el mapa de Karnaugh de la Figura 5.15(a). La implementación simplificada se muestra en la parte (b). Los inversores no se muestran.

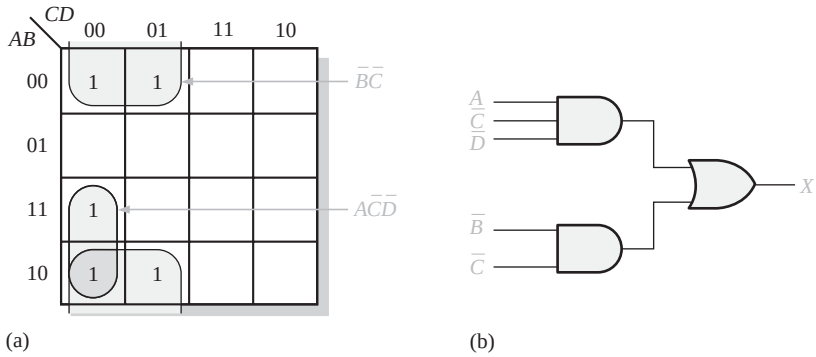


FIGURA 5.15

Problema relacionado Desarrollar el producto de sumas equivalente del circuito de la Figura 5.15(b).

REVISIÓN DE LA SECCIÓN 5.2

1. Implementar las siguientes expresiones booleanas tal y como se definen.
   (a) $X = ABC + AB + AC$ **(b)** $X = AB(C + DE)$
2. Desarrollar un circuito lógico que genere un 1 en su salida cuando sus tres entradas están a 1 o cuando sus tres entradas están a 0.
3. Simplificar los circuitos de la cuestión 1 utilizando la suma de productos mínima.

5.3 LA PROPIEDAD UNIVERSAL DE LAS PUERTAS NAND Y NOR

Hasta este momento se han estudiado los circuitos combinacionales que se implementan con puertas AND, puertas OR e inversores. En esta sección, se va a tratar la propiedad universal de la puerta NAND y de la puerta NOR. La universalidad de la puerta NAND significa que puede utilizarse como un inversor y que pueden emplearse combinaciones de la puerta NAND para implementar las operaciones AND, OR y NOR. Del mismo modo, la puerta NOR se puede utilizar para implementar el inversor (NOT) y las operaciones AND, OR y NAND.

Al finalizar esta sección, el lector deberá ser capaz de:

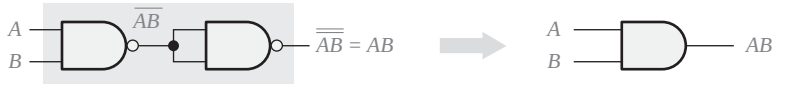
■ Utilizar las puertas NAND para implementar el inversor, la puerta AND, la puerta OR y la puerta NOR. ■ Utilizar las puertas NOR para implementar el inversor, la puerta AND, la puerta OR y la puerta NAND.

La puerta NAND como elemento lógico universal

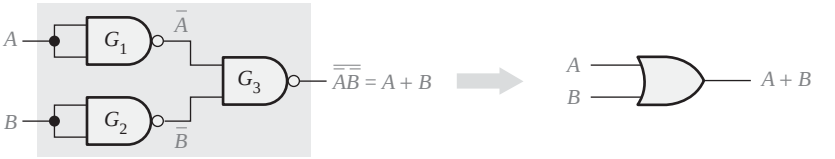
▲ *Las puertas NAND pueden emplearse para generar cualquier función lógica.*

La puerta NAND es una ***puerta universal*** porque puede utilizarse para generar las funciones NOT, AND, OR y NOR. Se puede obtener un inversor a partir de una puerta NAND conectando juntas todas las entradas, dando lugar a una única entrada, como se muestra en la Figura 5.16(a) con una puerta de 2 entradas. La operación

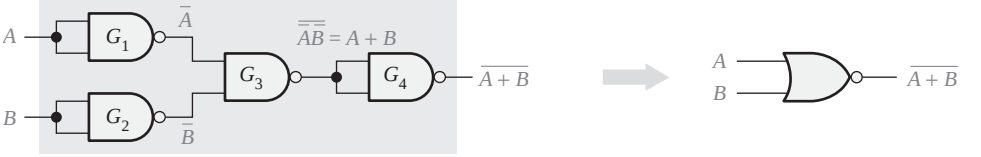
(a) Una puerta AND utilizada como inversor



(b) Dos puertas NAND utilizadas como puerta AND



(c) Tres puertas NAND utilizadas como puerta OR



(d) Cuatro puertas NAND utilizadas como puerta NOR

FIGURA 5.16 Aplicación universal de las puertas NAND.

AND se puede generar utilizando sólo puertas NAND, como muestra la Figura 5.16(b). La operación OR se puede obtener con varias puertas NAND, como ilustra la parte (c). Por último, la operación NOR se obtiene como se indica en la parte (d) de la figura.

En la Figura 5.16(b), se utiliza una puerta NAND para invertir (complementar) la salida de una puerta NAND para obtener una función AND, como indica la siguiente ecuación:

$$X = \overline{\overline{AB}} = AB$$

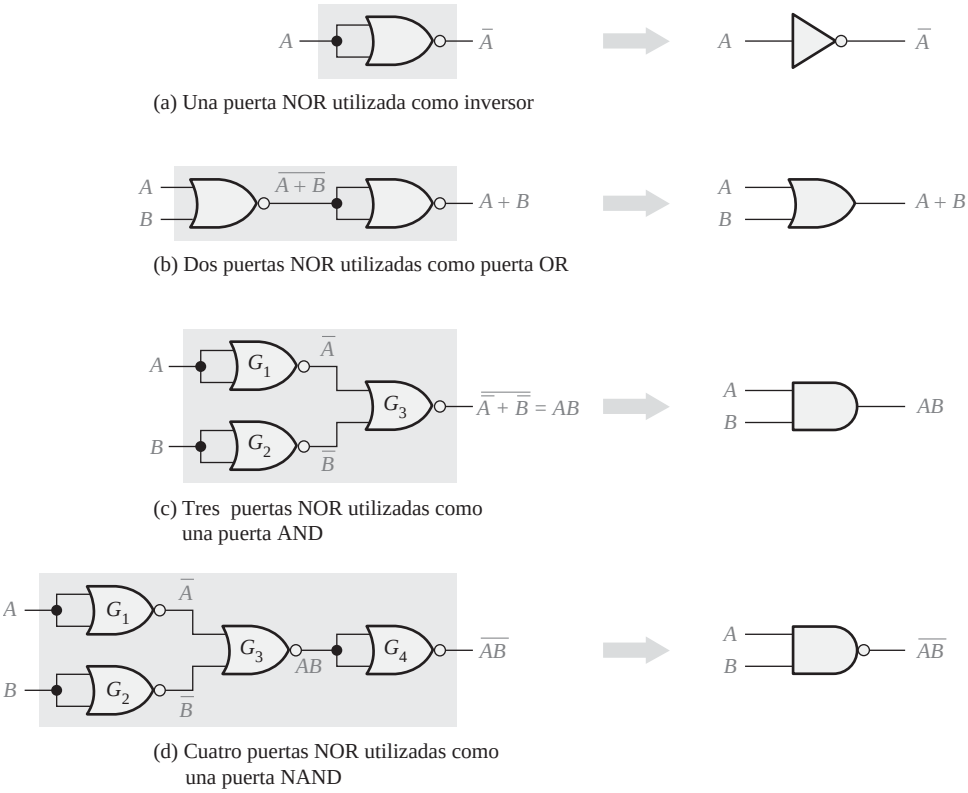
En la Figura 5.16(c), las puertas NAND G_1 y G_2 se emplean para invertir las dos variables de entrada antes de aplicarlas a la puerta NAND G_3. La salida final de la puerta OR se obtiene aplicando el teorema de DeMorgan del siguiente modo:

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

En la Figura 5.16(d), la puerta NAND G_4 se utiliza como un inversor conectado al circuito de la parte (c) con el fin de obtener la operación NOR $A + B$.

La puerta NOR como un elemento lógico universal

Al igual que la puerta NAND, la puerta NOR se puede utilizar para generar las funciones NOT, AND, OR y NAND. Un circuito NOT, o inversor, puede obtenerse a partir de una puerta NOR conectando todas sus



(a) Una puerta NOR utilizada como inversor

(b) Dos puertas NOR utilizadas como puerta OR

(c) Tres puertas NOR utilizadas como una puerta AND

(d) Cuatro puertas NOR utilizadas como una puerta NAND

FIGURA 5.17 Aplicación universal de las puertas NOR.

▲ *Las puertas NOR pueden emplearse para generar cualquier función lógica.*

entradas juntas para tener una única puerta, como se muestra en la Figura 5.17(a) con una puerta de 2 entradas. También puede obtenerse una puerta OR a partir de puertas NOR, como se ilustra en la Figura 5.17(b). Un puerta AND puede construirse utilizando puertas NOR como muestra la Figura 5.17(c). En este caso, las puertas NOR G_1 y G_2 se usan como inversores y la salida final se obtiene aplicando el teorema de DeMorgan del siguiente modo:

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

La Figura 5.17(d) muestra cómo se usan las puertas NOR para obtener una función NAND.

REVISIÓN DE LA SECCIÓN 5.3

1. Utilizando puertas NAND implementar las siguientes expresiones:
   (a) $X = \overline{A} + B$ **(b)** $X = A\overline{B}$
2. Utilizando puertas NOR implementar las siguientes expresiones:
   (a) $X = \overline{A} + B$ **(b)** $X = A\overline{B}$

5.4 LÓGICA COMBINACIONAL CON PUERTAS NAND Y NOR

En esta sección se verá cómo se usan las puertas NAND y NOR para implementar una función lógica. Recuerde del Capítulo 3 que la puerta NAND tiene una operación equivalente denominada negativa-OR, y que la puerta NOR tiene una operación equivalente denominada negativa-AND. Veremos cómo el uso de los símbolos adecuados para representar las operaciones equivalentes hace la "lectura" del diagrama lógico más fácil.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Utilizar puertas NAND para implementar una función lógica. ■ Utilizar puertas NOR para implementar una función lógica. ■ Utilizar el símbolo apropiado en un diagrama lógico.

Circuito lógico NAND

Como ya se ha dicho, una puerta NAND puede expresarse como una función NAND o una función negativa-OR, ya que por el teorema de DeMorgan:

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

NAND ↑ ↑ negativa-OR

Considerando el circuito lógico de la Figura 5.18, la expresión de salida se desarrolla según los pasos siguientes:

$$\begin{aligned} X &= \overline{(\overline{AB})(\overline{CD})} \\ &= \overline{(\overline{A} + \overline{B})(\overline{C} + \overline{D})} \\ &= \overline{(\overline{A} + \overline{B})} + \overline{(\overline{C} + \overline{D})} \\ &= \overline{\overline{A}}\,\overline{\overline{B}} + \overline{\overline{C}}\,\overline{\overline{D}} \\ &= AB + CD \end{aligned}$$

Como puede ver en la Figura 5.18, la expresión de salida, $AB + CD$, corresponde a la forma de dos términos que se multiplican (AND) y luego se suman (OR). Esta expresión muestra que las puertas G_2 y G_3 actúan

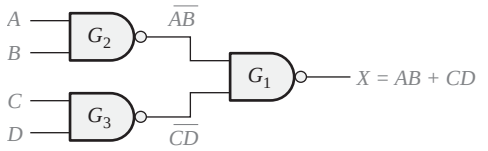
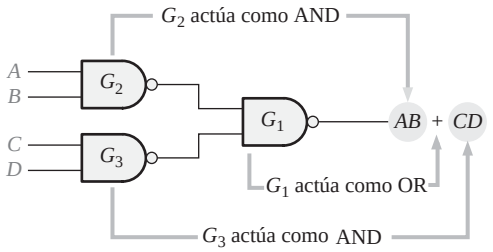
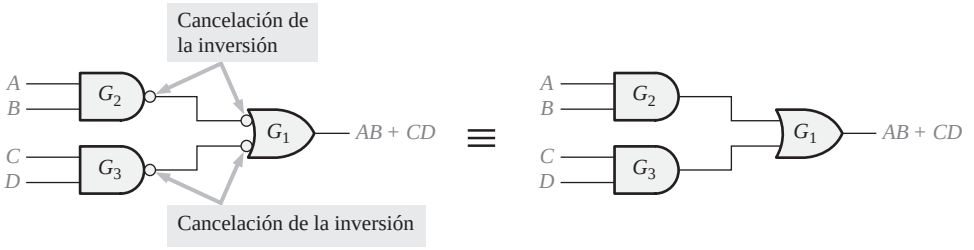


FIGURA 5.18 Circuito lógico NAND para $X = AB + CD$.

como puertas AND, y la puerta G_1 actúa como puerta OR, como ilustra la Figura 5.19(a). En la parte (b) de esta figura se presenta este circuito con los símbolos NAND para las puertas G_2 y G_3, y un símbolo de la puerta negativa-OR para la puerta G_1.



(a) Diagrama lógico NAND original que muestra la operación de la puerta correspondiente a la expresión de salida.



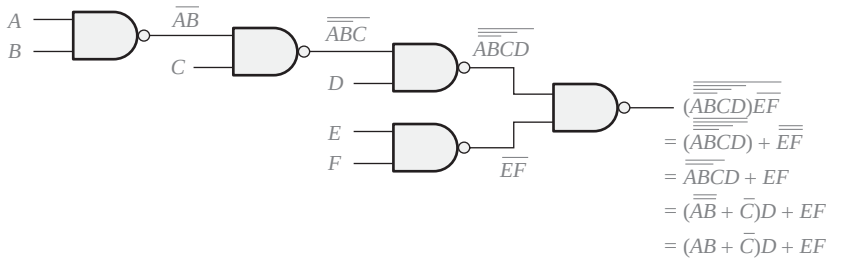
(b) Diagrama lógico equivalente NAND/Negativa-OR. (c) Equivalente AND-OR.

FIGURA 5.19 Desarrollo del equivalente AND-OR del circuito de la Figura 5.18.

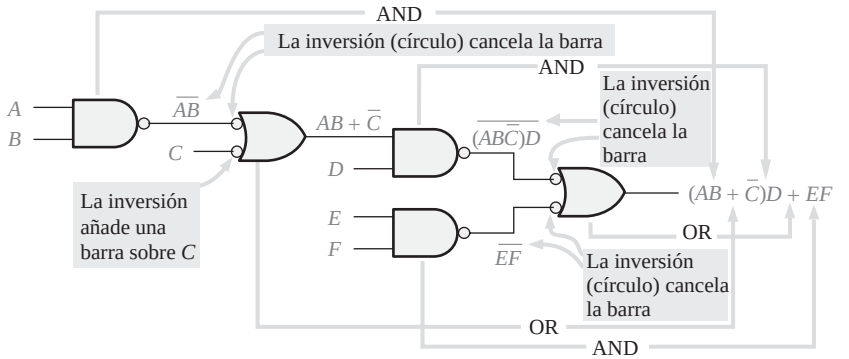
Fíjese en la Figura 5.19(b) en las conexiones círculo-círculo entre las salidas de las puertas G_2 y G_3, y las entradas de G_1. *Puesto que un círculo indica una inversión, dos círculos conectados representan una doble inversión y, por tanto, se cancelan entre sí*. Esta cancelación de inversión se ha podido ver en el desarrollo anterior para la expresión de salida $AB + CD$, y se indica por la ausencia, en la misma, de términos con una barra encima. Luego el circuito de la Figura 5.19(b) es *efectivamente* un circuito AND-OR, como se muestra en la Figura 5.19(c).

Diagrama lógico NAND utilizando símbolos duales. Todos los diagramas lógicos que utilizan puertas NAND deberían dibujarse utilizando el símbolo NAND o el símbolo equivalente negativa-OR para representar cada puerta, con el fin de reflejar la operación de la puerta dentro del circuito lógico. Los símbolos NAND y negativa-OR se denominan símbolos duales. Cuando se dibuja un diagrama lógico NAND, siempre se emplean los símbolos de puerta de tal forma que cada una de las conexiones entre la salida de una puerta y la entrada de otra sea una conexión círculo-círculo o una conexión no círculo-no círculo.

La Figura 5.20 ilustra el procedimiento de utilización de los símbolos duales adecuados para un circuito NAND con varios niveles de puertas. Aunque es correcto utilizar siempre símbolos NAND, como muestra la



(a) Se necesitan varios pasos del álgebra booleana para llegar a la expresión de salida final.



(b) La expresión de salida puede obtenerse directamente a partir de la función del símbolo de cada puerta del diagrama.

FIGURA 5.20 Ejemplo de utilización de los símbolos duales apropiados en un diagrama lógico NAND.

Figura 5.20(a), el diagrama de la parte (b) es más fácil de "leer" y es preferible. Como puede verse en la Figura 5.20(b), la puerta de salida se ha representado con un símbolo negativa-OR. El símbolo NAND se emplea para los niveles de puertas anteriores a la puerta de salida y los símbolos para los sucesivos niveles de puertas se alternan según se alejan de la salida.

La forma de las puertas indica cómo aparecerán sus entradas en la ecuación de salida y, por tanto, cómo funciona la puerta dentro del circuito lógico. Cuando se usa el símbolo NAND, las entradas aparecen multiplicadas (AND) en la expresión de salida, y cuando se usa el símbolo negativa-OR las entradas aparecen sumadas (OR), como ilustra la Figura 5.20(b). Puede ver que en el diagrama de símbolos duales de la parte (b) de la figura es mucho más fácil determinar directamente la expresión de salida, ya que cada símbolo de puerta indica las relaciones de sus variables de entrada, tal y como aparecen en la expresión de salida.

EJEMPLO 5.7

Volver a dibujar el diagrama lógico y desarrollar la expresión de salida para el circuito de la Figura 5.21, utilizando los símbolos duales adecuados.

Solución Dibujamos de nuevo el diagrama lógico de la Figura 5.21 utilizando símbolos equivalentes negativa-OR como se muestra en la Figura 5.22. La expresión de X obtenida directamente de la operación lógica que indica cada puerta es:

$$X = (\overline{A} + \overline{B})C + (\overline{D} + \overline{E})F$$

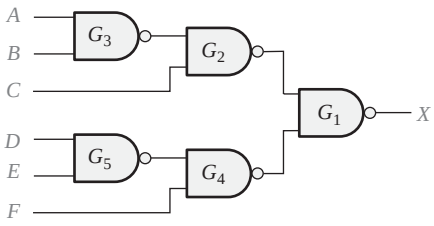


FIGURA 5.21

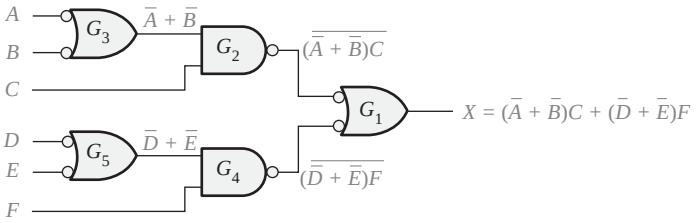


FIGURA 5.22

Problema relacionado Obtener la expresión de salida a partir de la Figura 5.21, y demostrar que es equivalente a la expresión obtenida como solución.

EJEMPLO 5.8

Implementar las siguientes expresiones mediante lógica NAND usando los símbolos duales apropiados:
(a) $ABC + DE$ **(b)** $ABC + \overline{D} + \overline{E}$

Solución *Véase* la Figura 5.23.

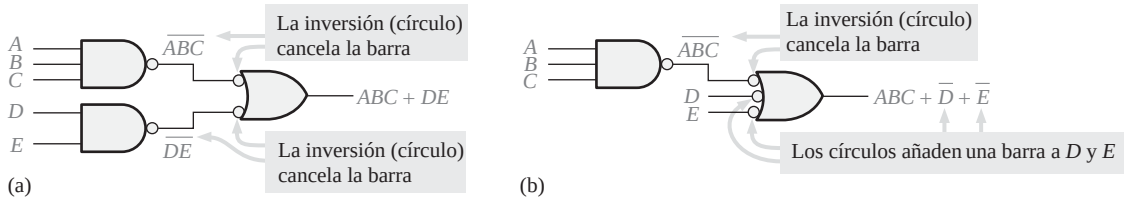


FIGURA 5.23

Problema relacionado Convertir los circuitos NAND de las Figuras 5.23(a) y (b) a su equivalente lógico AND-OR.

Lógica NOR

Una puerta NOR puede funcionar como NOR o como ***negativa-AND***, como demuestra el teorema de DeMorgan:

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

NOR ↑ ↑ negativa-AND

Consideremos el diagrama lógico NOR de la Figura 5.24. La expresión de salida se desarrolla así:

$$X = \overline{\overline{A+B}+\overline{C+D}} = (\overline{\overline{A+B}})(\overline{\overline{C+D}}) = (A+B)(C+D)$$

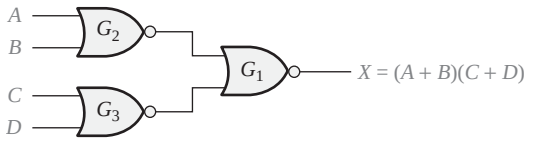


FIGURA 5.24 Diagrama lógico NOR para $X = (A+B)\,(C + D)$.

Como puede verse en la Figura 5.24, la expresión de salida $(A + B)(C + D)$ está formada por dos términos a los que primero se les aplica la operación OR y luego la operación AND. Esto implica que las puertas G_2 y G_3 operan como puertas OR, y la puerta G_1 como puerta AND, como muestra la Figura 5.25(a). Este circuito se ha dibujado de nuevo en la parte (b) de la figura con un símbolo negativa-AND para la puerta G_1.

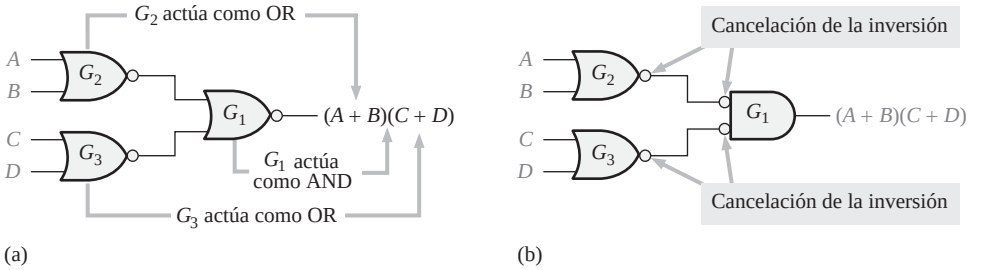
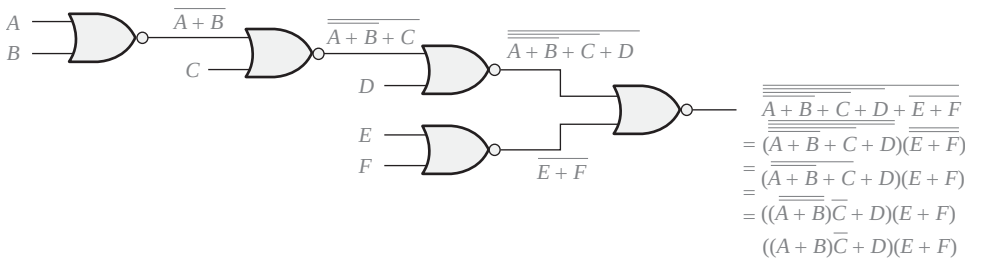


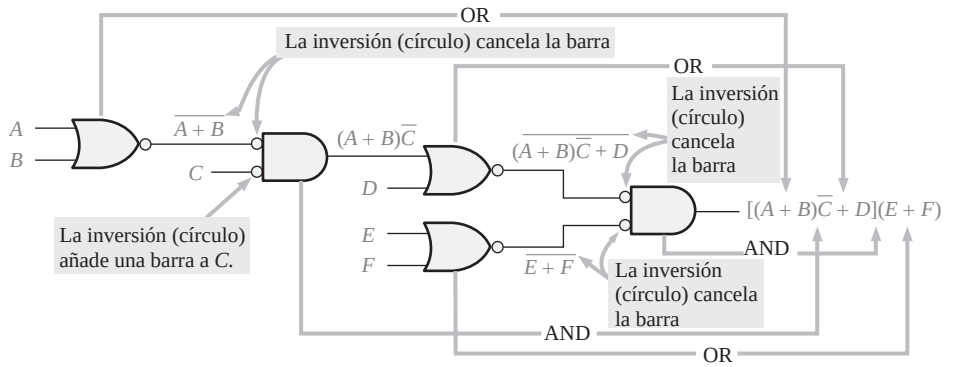
FIGURA 5.25

Diagrama lógico NOR utilizando símbolos duales. Como en el diagrama lógico NAND, el propósito de utilizar los símbolos duales es hacer más fácil la lectura y el análisis del diagrama, lo que se ilustra en el circuito lógico NOR de la Figura 5.26. Cuando el circuito de la parte (a) se redibuja con símbolos duales, se obtiene el circuito de la parte (b) de la figura; observe que todas las conexiones de salida-entrada son círculo-círculo o no círculo-no círculo. De nuevo, puede comprobar que la forma de cada puerta indica el tipo de término (AND u OR) que producirá en la expresión de salida, lo que hace que sea más fácil determinar la expresión de salida y más fácil también analizar el diagrama lógico.



(a) La expresión de salida final se obtiene después de aplicar varios pasos del álgebra booleana.

FIGURA 5.26 Ilustración del uso de los símbolos duales apropiados en un diagrama lógico NOR. (*Continúa*)



(b) La expresión de salida puede obtenerse directamente de la función de cada símbolo de puerta del diagrama.

FIGURA 5.26 (*Continuación*).

EJEMPLO 5.9

Utilizando los símbolos duales apropiados, dibujar de nuevo el diagrama lógico y desarrollar la expresión de salida para el circuito de la Figura 5.27.

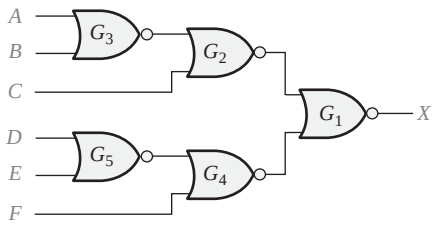


FIGURA 5.27

Solución En la Figura 5.28 se muestra el nuevo diagrama lógico utilizando el símbolo de la puerta negativa-AND equivalente. La expresión *X* se obtiene directamente a partir de la función lógica de cada puerta.

$$X = (\overline{A}\overline{B} + C)(\overline{D}\overline{E} + F)$$

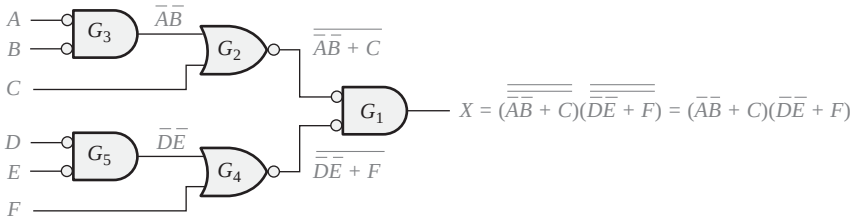


FIGURA 5.28

Problema relacionado Demostrar que la salida del circuito NOR de la Figura 5.27 es igual que la del circuito de la Figura 5.28.

REVISIÓN DE LA SECCIÓN 5.4

1. Implementar la expresión $X = \overline{(\bar{A} + \bar{B} + \bar{C})DE}$ usando un diagrama lógico NAND.
2. Implementar la expresión $X = \overline{\bar{A}\bar{B}\bar{C} + (D + E)}$ usando un diagrama lógico NOR.

5.5 FUNCIONAMIENTO DE LOS CIRCUITOS LÓGICOS CON TRENES DE IMPULSOS

En esta sección se examinan varios ejemplos de circuitos lógicos combinacionales con entradas que son trenes de impulsos. Hay que tener en mente que la operación lógica de las puertas es la misma para impulsos que para niveles continuos de entrada. En cualquier instante determinado, la salida de un circuito lógico depende de sus entradas en ese instante, por lo que son de importancia capital las variaciones con el tiempo de las entradas.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Analizar cualquier circuito lógico combinacional con trenes de impulsos como entradas. ■ Desarrollar el diagrama de tiempos para cualquier circuito lógico combinacional de acuerdo con las entradas especificadas.

La operación lógica que realiza una puerta es la misma independientemente de que a sus entradas se apliquen impulsos o niveles constantes. La naturaleza de las entradas (impulsos o niveles constantes) no altera la tabla de verdad de un circuito. Los ejemplos de esta sección ilustran el análisis de circuitos lógicos combinacionales con impulsos de entrada.

A continuación, se presenta un repaso de la operación lógica que realiza cada puerta, con el fin de analizar los circuitos combinacionales con trenes de impulsos en sus entradas.

1. La salida de una puerta AND es un nivel ALTO sólo cuando todas las entradas están a nivel ALTO en el mismo instante.
2. La salida de una puerta OR es un nivel ALTO siempre que al menos una de sus entrada esté a nivel ALTO.
3. La salida de una puerta NAND es un nivel BAJO sólo cuando todas las entradas están a nivel ALTO en el mismo instante.
4. La salida de una puerta NOR es un nivel BAJO siempre que al menos una de las entradas esté a nivel ALTO.

EJEMPLO 5.10

Determinar la forma de onda de salida *X* para el circuito de la Figura 5.29, cuando se aplican las entradas *A, B* y *C* que se indican.

Solución La expresión de salida, $\overline{AB + AC}$, indica que la salida *X* es un nivel BAJO cuando *A* y *B* están a nivel ALTO, o cuando *A* y *C* están a nivel ALTO, o cuando todas las entradas están a nivel ALTO. En la Figura 5.29 se muestra el diagrama de tiempos correspondiente a la señal de salida *X*. La señal de salida intermedia *Y*, salida de la puerta OR, también se indica en el diagrama.

Problema relacionado Determinar la forma de onda de salida si la entrada *A* es un nivel ALTO constante.

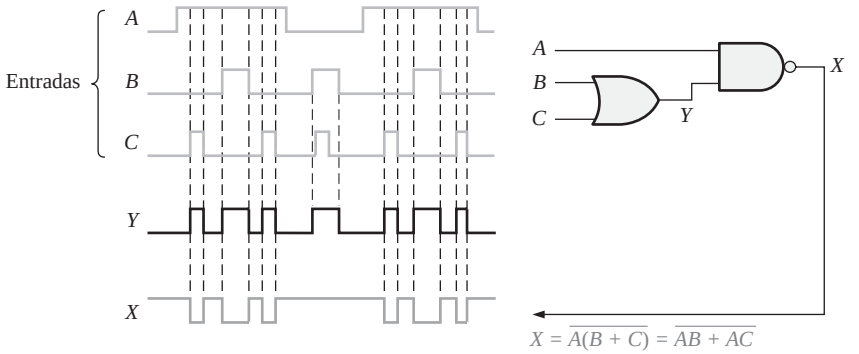


FIGURA 5.29

EJEMPLO 5.11

Dibujar el diagrama de tiempos para el circuito de la Figura 5.30, especificando las salidas de las puertas G_1, G_2 y G_3, siendo las entradas las formas de onda A y B que se indican.

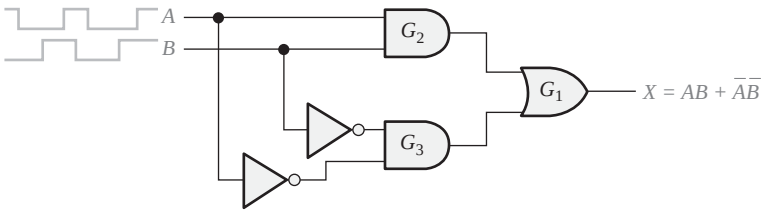


FIGURA 5.30

Solución

Cuando las dos entradas están a nivel ALTO o a nivel BAJO, la salida X se pone a nivel ALTO, como muestra la Figura 5.31. Observe que se trata de un circuito NOR-exclusiva. Las salidas intermedias de las puertas G_2 y G_3 también se muestran en dicha figura.

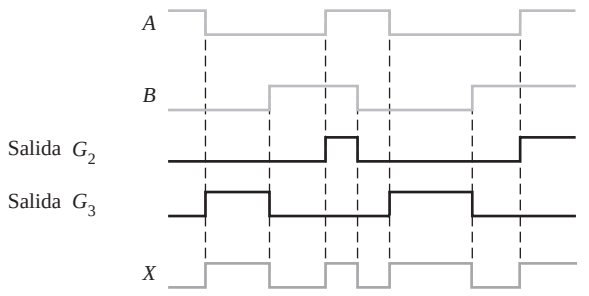


FIGURA 5.31

Problema relacionado

Determinar la salida X correspondiente al circuito de la Figura 5.30, si se invierte la entrada B.

EJEMPLO 5.12

Determinar la forma de onda de salida X para el circuito lógico de la Figura 5.32(a), hallando en primer lugar las formas de onda intermedias en los puntos Y_1, Y_2, Y_3 e Y_4. Las formas de onda de entrada son las que se indican en la Figura 5.32(b).

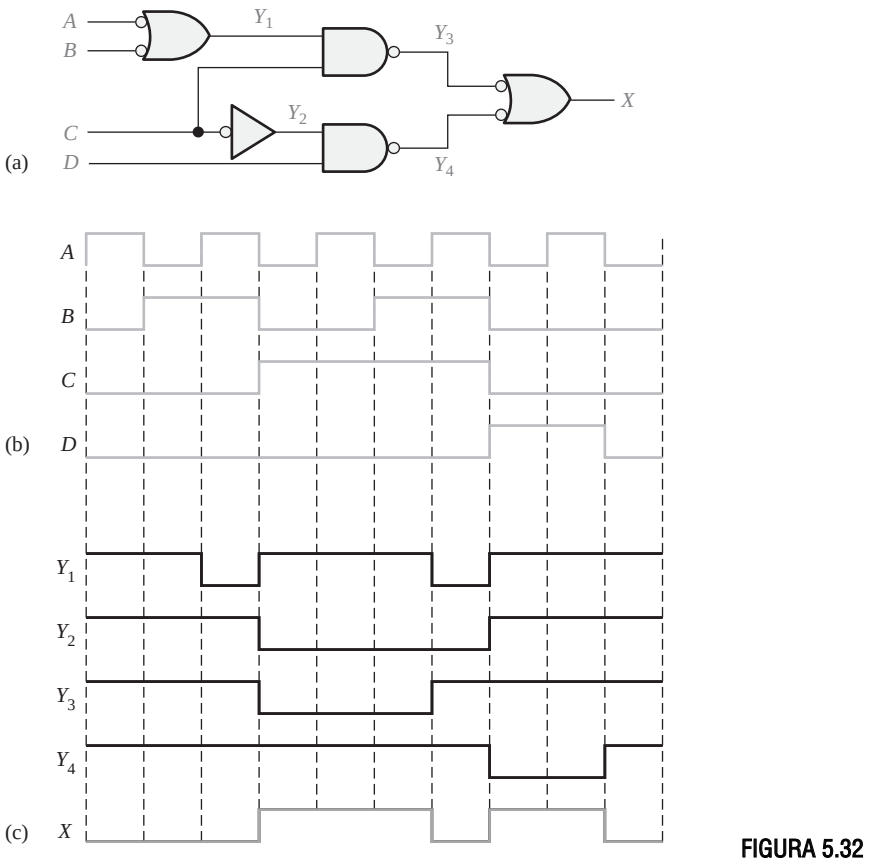


FIGURA 5.32

Solución Todas las formas de onda intermedias y la forma de onda de salida se muestran en el diagrama de tiempos de la Figura 5.32(c).

Problema relacionado Determinar las formas de onda Y_1, Y_2, Y_3 Y_4 y X si se invierte la entrada A.

EJEMPLO 5.13

Determinar la forma de onda de salida X para el circuito del Ejemplo 5.12, Figura 5.32(a), directamente a partir de la expresión de salida.

Solución En la Figura 5.33 se desarrolla la expresión de salida para el circuito especificado. La suma de productos indica que la salida es un nivel ALTO cuando A está a nivel BAJO y C está a nivel ALTO, o cuando B está a nivel BAJO y C está a nivel ALTO, o cuando C está a nivel BAJO y D está a nivel ALTO.

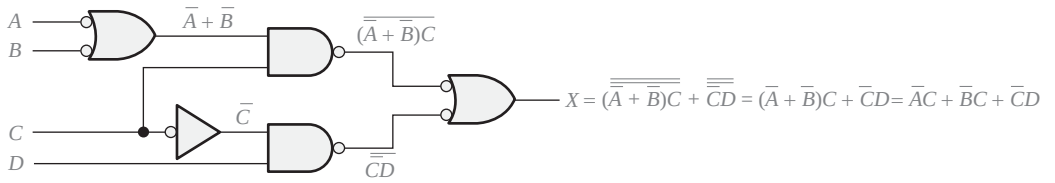


FIGURA 5.33

El resultado se muestra en la Figura 5.34, y es el mismo que se ha obtenido por el método de las señales intermedias del Ejemplo 5.12. Se indican los términos producto para cada forma de onda que da lugar a un nivel de salida ALTO.

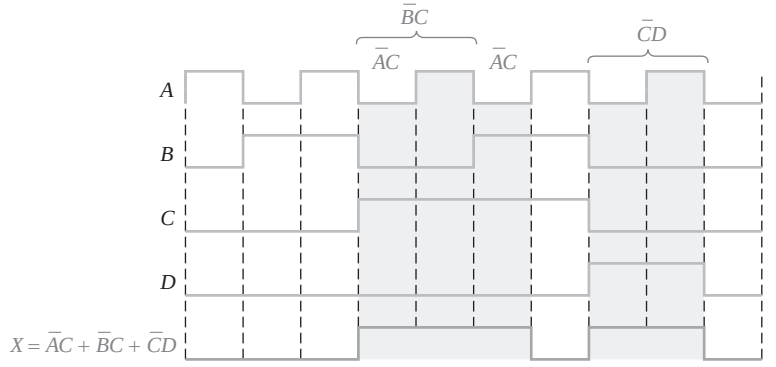


FIGURA 5.34

Problema relacionado Repetir este ejemplo si se invierten todas las formas de onda de entrada.

REVISIÓN DE LA SECCIÓN 5.5

1. A una de las entradas de un circuito OR-exclusiva se aplica un impulso con t_W = 50 μs. A la otra entrada, 15 μs después de que se genere el flanco de subida del primer impulso, se aplica un impulso positivo con t_W = 10 μs. Dibujar la salida en función de las entradas.
2. Los trenes de impulsos *A* y *B* de la Figura 5.29 se aplican al circuito NOR-exclusiva de la Figura 5.30. Desarrollar el diagrama de tiempos completo.

5.6 LÓGICA COMBINACIONAL CON VHDL (opcional)

El propósito de describir la lógica utilizando VHDL es que puede programarse en un PLD. En el Capítulo 4 se ha descrito el método del flujo de datos para escribir un programa VHDL. En esta sección opcional, vamos a usar tanto el método de flujo de datos usando expresiones booleanas como el método estructural para desarrollar el código VHDL que describe los circuitos lógicos. Vamos a presentar y a utilizar el componente VHDL para ilustrar las descripciones estructurales. También se abordan algunos aspectos de las herramientas de desarrollo software.

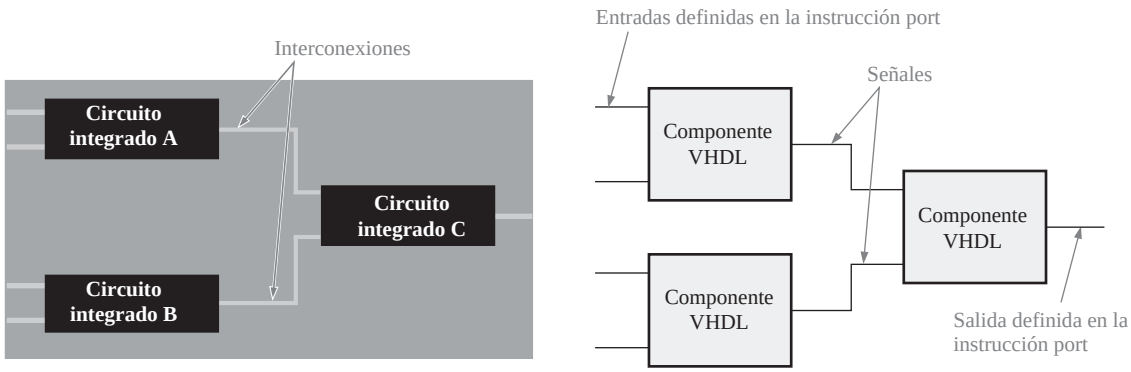
Al finalizar esta sección, el lector deberá ser capaz de:

■ Describir un componente VHDL y explicar cómo se emplea en un programa. ■ Aplicar el método estructural y el método de flujo de datos para escribir código VHDL. ■ Describir dos herramientas básicas de desarrollo software.

Método estructural de programación en VHDL

El método estructural para escribir una descripción VHDL de una función lógica puede compararse con el montaje de circuitos integrados en una tarjeta de circuito y el establecimiento de las interconexiones entre ellos mediante cables. Con el método estructural, se describen las funciones lógicas y se especifica cómo se conectan entre sí. El ***componente*** VHDL es una forma de predefinir una función lógica para poder emplearla repetidas veces en un mismo programa o en otros programas. El componente puede utilizarse para describir cualquier cosa, desde una simple puerta lógica a una función lógica compleja. La ***señal*** VHDL es una forma de especificar una conexión mediante un "cable" entre componentes.

La Figura 5.35 proporciona una comparación simplificada del método estructural con una implementación hardware en una tarjeta de circuito.



(a) Implementación hardware con lógica de función fija.

(b) Implementación estructural VHDL.

FIGURA 5.35 Comparación simplificada del método estructural VHDL con una implementación hardware. Las señales VHDL se corresponden con las interconexiones de la tarjeta de circuito y los componentes VHDL con los circuitos integrados.

Componentes VHDL

Un componente VHDL describe la lógica predefinida que puede almacenarse como una declaración de paquete en una biblioteca VHDL y puede llamarse tantas veces como sea necesario dentro de un programa. Puede emplear componentes para evitar repetir el mismo código una y otra vez dentro de un programa. Por ejemplo, puede crear un componente VHDL para una puerta AND y utilizarlo tantas veces como desee sin tener que escribir un programa para una puerta AND cada vez que lo necesite.

Los componentes VHDL se almacenan y están disponibles para su uso cuando se escribe un programa. Esto es similar, por ejemplo, a disponer de una bandeja de almacenamiento de circuitos integrados mientras se está montando un circuito. Cada que vez que se necesita un dispositivo integrado, se toma de la bandeja de almacenamiento y se coloca sobre la tarjeta de circuito impreso.

El programa VHDL para cualquier función lógica puede ser un componente y puede emplearse cuando sea necesario en un programa más largo mediante la declaración del componente, que tiene el formato general siguiente. **component** es una palabra clave VHDL.

component nombre_del_componente **is**
port (definiciones de puerto);
end component nombre_del_componente;

Para simplificar, como se muestra en la Figura 5.36 suponemos que tenemos descripciones de flujo de datos VHDL predefinidas para una puerta AND de 2 entradas con el nombre de entidad AND_gate y para una puerta OR de 2 entradas con el nombre de entidad OR_gate.

Puerta AND de 2 entradas

```
entity AND_gate is
  port (A, B: in bit; X: out bit);
end entity AND_gate;

architecture ANDfunction of AND_gate is
begin
  X <= A and B;
end architecture ANDfunction;
```

Puerta OR de 2 entradas

```
entity OR_gate is
  port (A, B: in bit; X: out bit);
end entity OR_gate;

architecture ORfunction of OR_gate is
begin
  X <= A or B;
end architecture ORfunction;
```

FIGURA 5.36 Programas predefinidos para una puerta AND de 2 entradas y para una puerta OR de 2 entradas que se emplearán como componentes en el método de flujo de datos.

A continuación suponemos que estamos escribiendo un programa para un circuito lógico que tiene varias puertas AND. En lugar de reescribir el programa de la Figura 5.36 una y otra vez podemos usar una declaración de componente para especificar la puerta AND. La instrucción *port* de la declaración del componente debe corresponderse con la instrucción *port* de la declaración de entidad de la puerta AND.

component AND_gate **is**
port (A, B: **in** bit; X: **out** bit);
end component AND_gate;

Utilización de componentes en un programa. Para emplear un componente en un programa, hay que escribir una instrucción de instantación de componente por cada instancia en la que se utilice el componente. Una instantación de componente es como una solicitud o llamada al componente que se va a usar en el programa principal. Por ejemplo, el circuito lógico en forma suma de productos simple de la Figura 5.37 tiene dos puertas AND y una puerta OR. Por tanto, el programa VHDL para este circuito tendrá dos componentes y tres instantaciones o llamadas a componente.

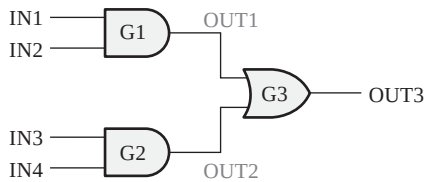


FIGURA 5.37

Señales. En VHDL, las señales son como los hilos que interconectan los componentes montados en una tarjeta de circuito impreso. En la Figura 5.37 las señales se han etiquetado como OUT1 y OUT2. Las señales son las conexiones *internas* del circuito lógico y se tratan de forma diferente que las entradas y salidas. Mientras que las entradas y salidas se declaran en la declaración de entidad utilizando la instrucción *port*, las señales se declaran dentro de la arquitectura mediante la instrucción *signal*, (**signal** es una palabra clave VHDL).

El programa. El programa para el circuito de la Figura 5.37 comienza con la siguiente declaración de entidad:

```
-- Programa para el circuito lógico de la Figura 5.37
entity AND_OR_Logic is
  port (IN1, IN2, IN3, IN4: in bit; OUT3: out bit);
end entity AND_OR_Logic;
```

La declaración de arquitectura contiene las declaraciones de los componentes para la puerta AND y la puerta OR, las definiciones de las señales y las instantaciones de los componentes.

```
architecture LogicOperation of AND_OR_Logic is
  component AND_gate is
    port (A, B: in bit); X: out bit);
  end component AND_gate;

  component OR_gate is
    port (A, B: in bit; X: out bit);
  end component OR_gate;

  signal OUT1, OUT2: bit;
begin
  G1: AND_gate port map (A => IN1, B => IN2, X => OUT1);
  G2: AND_gate port map (A => IN3, B => IN4, X => OUT2);
  G3: OR_gate port map (A => OUT1, B => OUT2, X => OUT3);
end architecture LogicOperation;
```

Declaración de componente para la puerta AND

Declaración de componente para la puerta OR

Declaración de las señales

Instantaciones de los componentes

Instantaciones de los componentes. Centrémonos en las instantaciones de los componentes. En primer lugar, fíjese en que las instantaciones de los componentes aparecen entre la palabra clave **begin** y la instrucción **end**. Para cada instantación, se define un identificador, en este caso, G1, G2 y G3. A continuación se especifica el nombre del componente. La instrucción *port map* establece, fundamentalmente, todas las conexiones de la función lógica utilizando el operador =>. Por ejemplo, la primera instantación,

```
G1: AND_gate port map (A => IN1, B => IN2, X => OUT1);
```

puede explicarse de la forma siguiente: *la entrada A de la puerta AND G1 se conecta a la entrada IN1, la entrada B de la puerta se conecta a la entrada IN2 y la salida X de la puerta se conecta a la señal OUT1.*

Las tres instrucciones de instantación describen completamente el circuito lógico de la Figura 5.37, como se ilustra en la Figura 5.38.

Aunque el método de flujo de datos usando expresiones booleanas hubiera sido más fácil y, probablemente, la mejor forma de describir este circuito concreto, hemos empleado este simple circuito para explicar el concepto del método estructural. El Ejemplo 5.14 compara los métodos de flujo de datos y estructural para escribir un programa VHDL para un circuito lógico en forma suma de productos.

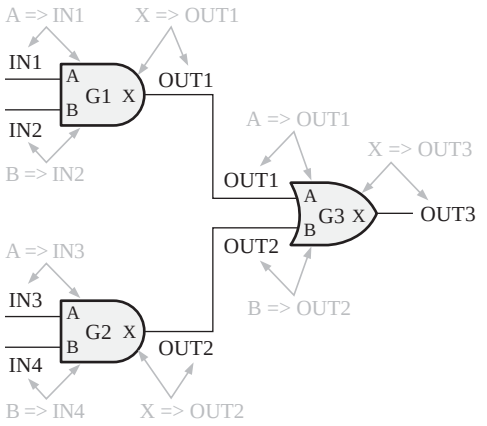


FIGURA 5.38 Ilustración de las instrucciones de instantación y la correspondencia de puertos (port map) aplicada a la lógica AND-OR. Las señales se indican en negro.

EJEMPLO 5.14

Escribir un programa VHDL para el circuito lógico en forma de suma de productos de la Figura 5.39 utilizando el método estructural. Suponemos que se dispone de los componentes para una puerta NAND de 3 entradas y para una puerta NAND de 2 entradas. Observe que la puerta NAND G4 se muestra como una puerta negativa-OR.

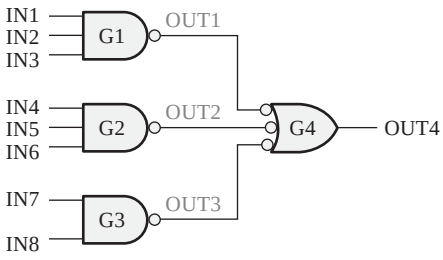


FIGURA 5.39

Solución

Los componentes y las instantaciones de los componentes se muestran en negrita.

```
-- Programa para el circuito lógico de la Figura 5.39
entity SOP_Logic is
  port (IN1, IN2, IN3, IN4, IN5, IN6, IN7, IN8: in bit; OUT4: out bit);
end entity SOP_Logic;
architecture LogicOperation of SOP_Logic is
-- declaración de componente para una puerta NAND de 3 entradas
  component NAND_gate3 is
   port (A, B, C: in bit X: out bit);
  end component NAND_gate3;
```

```
-- declaración de componente para una puerta NAND de 2 entradas
  component NAND_gate2 is
    port (A, B: in bit; X: out bit);
  end component NAND_gate;
  signal OUT1, OUT2, OUT3: bit;
begin
  G1: NAND_gate3 port map (A => IN1, B => IN2, C => IN3, X => OUT1);
  G2: NAND_gate3 port map (A => IN4, B => IN5, C => IN6, X => OUT2);
  G3: NAND_gate2 port map (A => IN7, B => IN8, X => OUT3);
  G4: NAND_gate3 port map (A => OUT1, B => OUT2, C => OUT3,
                           X => OUT4);
end architecture LogicOperation;
```

Con fines de comparación, escribimos el programa para el circuito lógico de la Figura 5.39 utilizando el método de flujo de datos.

```
entity SOP_Logic is
  port (IN1, IN2, IN3, IN4, IN5, IN6, IN7, IN8: in bit; OUT4: out bit);
end entity SOP_Logic;
architecture LogicOperation of SOP_Logic is
begin
  OUT4 <= (IN1 and IN2 and IN3) or (IN4 and IN5 and IN6) or (IN7 and IN8);
end architecture LogicOperation;
```

Como puede ver, el método de flujo de datos da lugar a un código mucho más sencillo en el caso de esta función lógica en concreto. Sin embargo, en situaciones en las que una función lógica consta de muchos bloques de lógica compleja, el método estructural puede resultar más ventajoso que el método de flujo de datos.

Problema relacionado Si se añade otra puerta NAND con entradas IN9 e IN10 al circuito de la Figura 5.39, escribir una instantación de componente para añadirla al programa. Especifique cualquier otro cambio que sea necesario realizar en el programa.

Aplicación de herramientas de desarrollo software

Como ya sabemos, debe utilizarse un paquete de desarrollo software para implementar un diseño HDL en un dispositivo objetivo. Una vez que la lógica se ha descrito empleando un lenguaje de descripción hardware y se ha introducido mediante una herramienta software, denominada editor de código o de texto, puede probarse mediante simulación con el fin de verificar que funciona correctamente antes de programar realmente el dispositivo objetivo. El uso de herramientas de desarrollo software permite diseñar, desarrollar y probar la lógica combinacional antes de implementarla en el hardware. En el Capítulo 11 se estudian más en detalle las herramientas de desarrollo software.

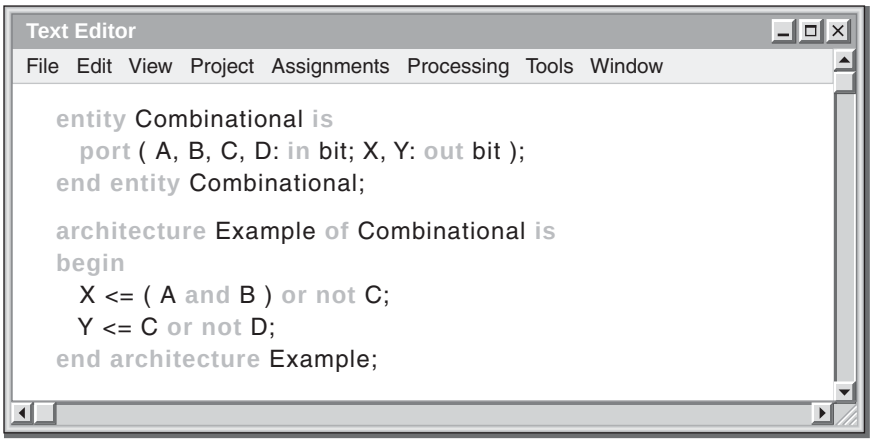


FIGURA 5.40 Programa VHDL para un circuito lógico combinacional como aparece en una pantalla de un editor de texto genérico, que forma parte de una herramienta de desarrollo software.

Las herramientas de desarrollo software típicas permiten introducir el código VHDL en un editor de texto específico de la herramienta de desarrollo concreta que se esté utilizando. En la Figura 5.40 se muestra una pantalla de computadora que contiene el código VHDL de un circuito lógico combinacional escrito en un editor de texto genérico. Como puede verse, muchos editores de código proporcionan funcionalidades mejoradas tales como resaltar las palabras clave.

Una vez que se ha escrito el programa en el editor de texto, se pasa al compilador. El compilador toma el código VHDL de alto nivel y lo convierte en un archivo que puede descargarse en el dispositivo objetivo. Una vez que el programa se ha compilado, se puede crear una simulación para probarlo. Los valores de entrada simulados se insertan en el diseño lógico y se puede proceder a la verificación de las salidas.

Las formas de onda de entrada se especifican mediante una herramienta software denominada editor de formas de onda, como la mostrada en la Figura 5.41. Las formas de onda de salida se generan mediante una simulación del código VHDL que se ha escrito en el editor de texto de la Figura 5.40. La simulación de la forma de onda proporciona las salidas resultantes X e Y para todas las combinaciones desde 0000_2 hasta 1111_2 de las entradas A, B, C y D.

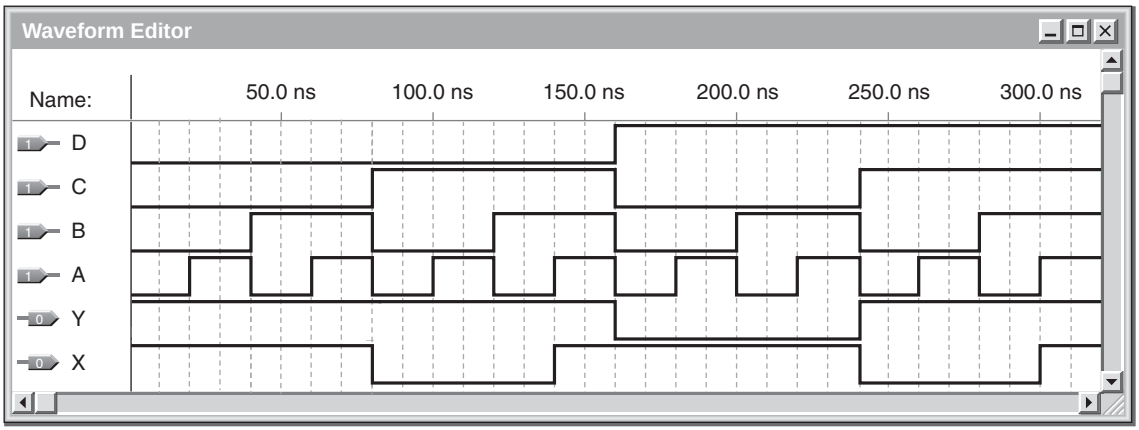


FIGURA 5.41 Un editor de formas de onda típico que muestra las formas de onda simuladas para el circuito lógico descrito por el código VHDL de la Figura 5.40.

Como se ha establecido en el Capítulo 3, no debe olvidar que en la creación de cualquier sistema digital deben tenerse en cuenta diversas características de funcionamiento de los circuitos lógicos. Por ejemplo, el retardo de propagación determina la velocidad o frecuencia a la que el circuito lógico puede funcionar. Puede utilizarse una simulación de temporización para simular el retardo de propagación a través del diseño lógico en el dispositivo objetivo.

REVISIÓN DE LA SECCIÓN 5.6

1. ¿Qué es un componente VHDL?
2. Explicar cuál es el propósito de una instantación de componente en una arquitectura de programa.
3. ¿Cómo se establecen las interconexiones entre los componentes en VHDL?
4. ¿Qué método representa el uso de componentes en un programa VHDL?

5.7 LOCALIZACIÓN DE AVERÍAS

Las secciones anteriores nos han introducido en el modo de operación de los circuitos lógicos combinacionales y las relaciones entre las entradas y las salidas. Estos conocimientos son esenciales cuando hay que localizar una avería en un circuito digital, ya que se debe conocer qué niveles lógicos o señales hay que buscar en el circuito para un conjunto dado de condiciones de entrada.

En esta sección, se utiliza un osciloscopio para localizar averías en un circuito lógico de función fija cuando una salida de una puerta está conectada a varias entradas de otras puertas. También se presenta un ejemplo de métodos de análisis y seguimiento de señales utilizando un oscilosopio o un analizador lógico para localizar un fallo en un circuito lógico combinacional.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Definir nodo de circuito. ■ Utilizar un osciloscopio para encontrar un fallo en un nodo de circuito. ■ Utilizar un osciloscopio para encontrar un circuito abierto en la salida de una puerta. ■ Utilizar un osciloscopio para encontrar un cortocircuito en la salida o la entrada de una puerta. ■ Utilizar un osciloscopio o un analizador lógico para seguir una señal en un circuito lógico combinacional.

En un circuito lógico combinacional, la salida de una puerta puede conectarse a dos o más entradas de otra puerta, como muestra la Figura 5.42. Las interconexiones se cruzan en un punto eléctrico común que se denomina **nodo**.

La puerta G_1 de la Figura 5.42 excita al nodo, y las demás puertas representan las cargas conectadas al nodo. Una puerta excitadora puede excitar a un determinado número de entradas de puertas de carga, hasta el máximo determinado por su *fan-out* específico. En esta situación, se pueden producir diversos tipos de fallos. Algunos de estos tipos de fallos son difíciles de aislar en una puerta, ya que todas las puertas conectadas al nodo se ven afectadas. Los fallos más comunes son los siguientes:

1. *Salida en circuito abierto en la puerta excitadora*. Este fallo da lugar a pérdida de la señal en todas las puertas de carga.
2. *Entrada en circuito abierto en una puerta de carga*. Este fallo no afectará al funcionamiento de ninguna otra puerta conectada al nodo, pero hará que no se detecte señal de salida en la puerta que falla.
3. *Salida cortocircuitada de la puerta excitadora*. Este fallo puede dar lugar a que el nodo permanezca en estado BAJO (cortocircuitado a masa) o en estado ALTO (cortocircuitado a V_{CC}).
4. *Entrada cortocircuitada en una puerta de carga*. Este fallo también hace que el nodo se mantenga a nivel BAJO (cortocircuitado a masa) o en estado ALTO (cortocircuitado a V_{CC}).

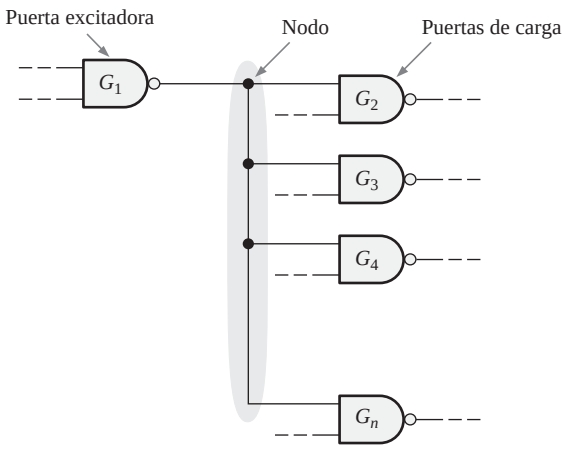


FIGURA 5.42 Ilustración de un nodo en un circuito lógico.

CONSEJOS PRÁCTICOS

Cuando se están localizando averías en circuitos lógicos, se debe empezar por una comprobación visual con el fin de localizar los problemas obvios. La inspección visual debería incluir los conectores además de los componentes. Los conectores externos se usan frecuentemente para llevar a una tarjeta de circuito las señales, la alimentación y la masa. Las superficies de contacto de los conectores deben estar limpias y deben tener una buena fijación mecánica. Un conector sucio puede producir un fallo intermitente o completo del circuito. Los conectores externos se pueden limpiar con un borrador de lápiz normal y un bastoncillo humedecido en alcohol. También se deberían comprobar todos los conectores para localizar los pines que no estén bien ajustados.

Localización de los fallos más comunes

Salida en circuito abierto en la puerta excitadora. En este caso no se detectan impulsos en el nodo. Con el circuito alimentado, un nodo en circuito abierto dará lugar, normalmente, a un nivel "flotante", lo que puede indicarse mediante ruido, como se ilustra en la Figura 5.43.

Entrada en circuito abierto en una puerta de carga. Si la salida de la puerta excitadora no está en circuito abierto, entonces hay que probar si la entrada de una puerta de carga está en circuito abierto. Permaneciendo las entradas de las puertas no pertenecientes al nodo a nivel ALTO, se comprueba con el osciloscopio la salida de cada una de las puertas, como se indica en la Figura 5.44. Si una de las entradas conectadas al nodo está en circuito abierto, no se detectarán impulsos en la salida de la misma.

Salida o entrada cortocircuita a masa. Si la salida está cortocircuitada a masa en la puerta excitadora o la entrada a una puerta de carga está cortocircuitada a masa, esto hará que el nodo permanezca a nivel BAJO, como se ha dicho anteriormente. Una rápida comprobación con la sonda del osciloscopio lo detectará, como se muestra en la Figura 5.45. Un cortocircuito a masa en la salida de la puerta excitadora o en la entrada de cualquier puerta de carga dará lugar a este síntoma, por lo que deben hacerse más comprobaciones para aislar el cortocircuito en una puerta en concreto.

Análisis y seguimiento de señales

Aunque los métodos de aislamiento de cortocircuitos y circuitos abiertos en un nodo son muy útiles en ocasiones, una técnica más general para la localización de fallos es el ***seguimiento de señales***, la cual tiene un

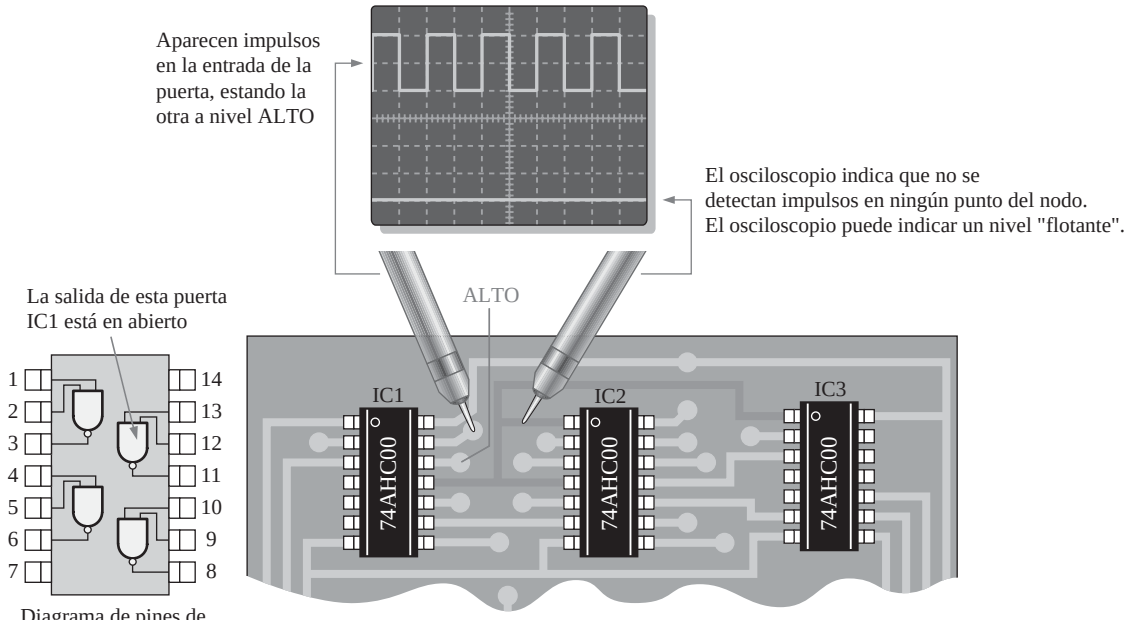


FIGURA 5.43 Salida en circuito abierto en la puerta excitadora. Para simplificar, se supone que hay un nivel ALTO en la entrada de una puerta.

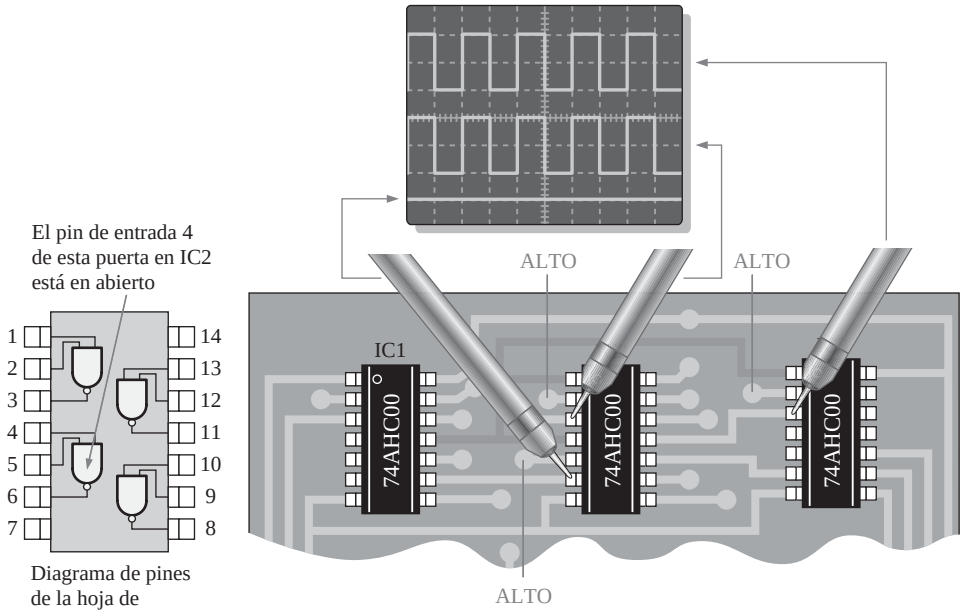


FIGURA 5.44 Entrada en circuito abierto de una puerta de carga.

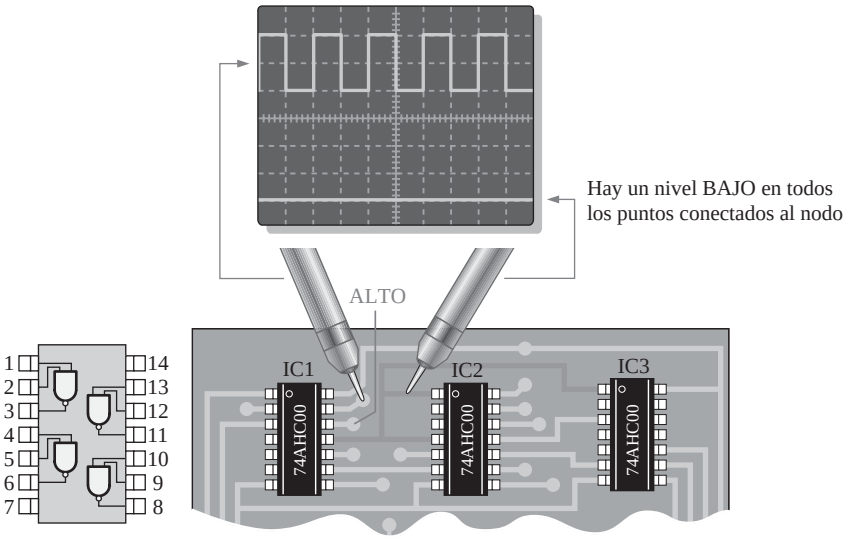


FIGURA 5.45 Salida cortocircuitada en una puerta excitadora o entrada cortocircuitada en una puerta de carga.

gran valor para el técnico en casi todos los casos de fallo. La medida de la señal se realiza con un osciloscopio o un analizador lógico.

Básicamente, el método de seguimiento de señales requiere que se observen las formas de onda y sus relaciones temporales en todos los puntos accesibles del circuito lógico. Se puede comenzar por las entradas y, a partir del análisis del diagrama de tiempos de la señal en cada punto, determinar cuál es el primer punto en que la señal es incorrecta. Normalmente, con este procedimiento se puede aislar el fallo en una puerta específica. También se puede usar el método de comenzar por la salida y continuar hacia atrás hasta las entradas.

El procedimiento general del seguimiento de señales comenzando por las entradas es el siguiente:

- Dentro del sistema, definir la sección del circuito lógico que se sospecha que está fallando.
- Comenzar en las entradas de la sección que se va a examinar. Para este estudio, suponemos que las formas de onda de entrada proceden de otras partes del sistema que son correctas.
- Para cada puerta, empezando por la entrada y yendo hacia la salida del circuito lógico, se observa la forma de onda de salida de la puerta y se compara con las formas de onda de entrada, utilizando el osciloscopio o el analizador lógico.
- Determinar si la señal de salida es correcta utilizando nuestros conocimientos sobre la operación lógica de la puerta.
- Si la salida es incorrecta, en la puerta bajo prueba puede estar el fallo. Extraiga el CI que contiene la puerta de la que se sospecha que produce el fallo, y compruébelo fuera del circuito. Si la puerta falla, reemplace el CI. Si funciona correctamente, el fallo está en la circuitería externa o en otro CI al que está conectado el que se está probando.
- Si la salida es correcta, pase a la puerta siguiente. Continúe comprobando cada puerta hasta observar una forma de onda incorrecta.

La Figura 5.46 es un ejemplo que ilustra el procedimiento general para un circuito lógico específico, en el que se siguen los pasos que a continuación se indican:

Paso 1. Observar la salida de la puerta G_1 (punto de prueba 5) respecto a sus entradas. Si es correcta, probar el inversor siguiente. Si la salida no es correcta, la puerta o sus conexiones están mal; o, si la salida está a nivel BAJO, la entrada de la puerta G_2 puede estar cortocircuitada.

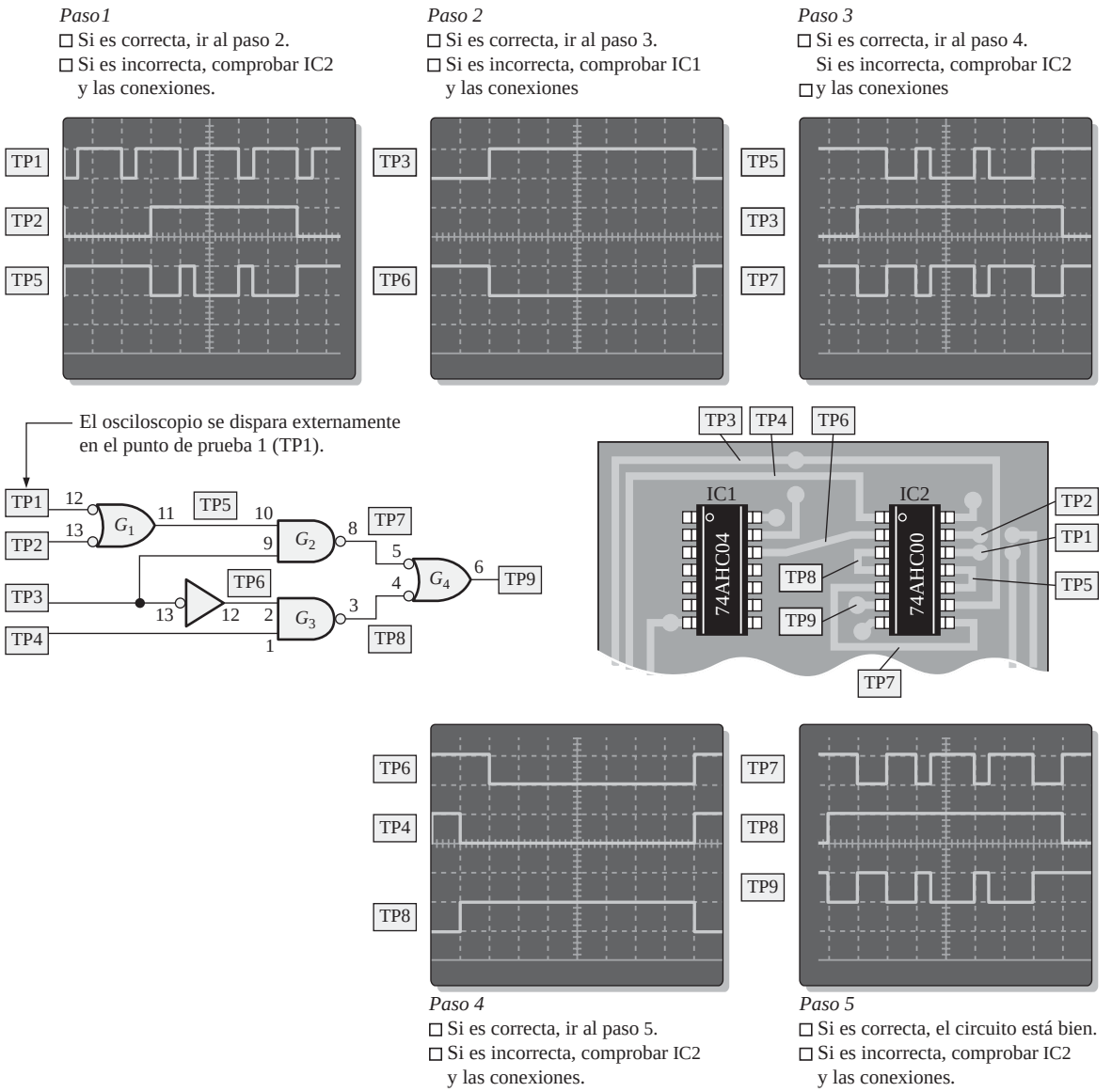


FIGURA 5.46 Ejemplo de análisis y seguimiento de señales en una parte de una tarjeta de circuito impreso. TP (test point) indica punto de prueba.

Paso 2. Observar la salida del inversor (TP6) respecto a la entrada. Si es correcta, probar la puerta siguiente, G_2. Si la salida no es correcta, el inversor o sus conexiones están mal; o, si la salida está a nivel BAJO, la entrada de la puerta G_3 puede estar cortocircuitada.

Paso 3. Observar la salida de la puerta G_2 (TP7) respecto a las entradas. Si es correcta, probar la puerta siguiente, G_3. Si la salida no es correcta, el inversor o sus conexiones están mal; o, si la salida está a nivel BAJO, la entrada de la puerta G_4 puede estar cortocircuitada.

Paso 4. Observar la salida de la puerta G_3 (TP8) respecto a sus entradas. Si es correcta, probar la puerta G_4. Si la salida no es correcta, la puerta o sus conexiones están mal; o, si la salida está a nivel BAJO, la entrada de la puerta G_4 (TP9) puede estar cortocircuitada.

Paso 5. Observar la salida de la puerta G_4 (TP9) respecto a sus entradas. Si es correcta, el circuito está bien. Si la salida no es correcta, la puerta o sus conexiones están mal.

EJEMPLO 5.15

Determinar el fallo en el circuito lógico de la Figura 5.47(a) utilizando el análisis de señales. Debe observar las formas de onda de color gris de la Figura 5.47(b). Las formas de onda en negro son correctas y se proporcionan con propósitos de comparación.

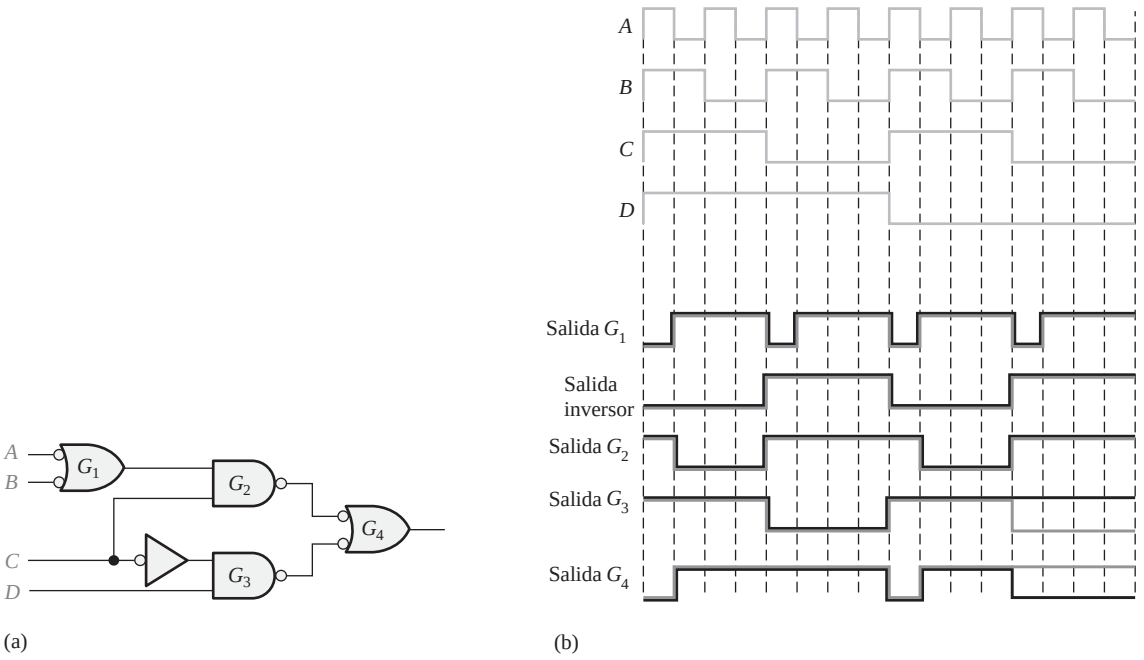


FIGURA 5.47

Solución

1. Determinar cuál es la forma de onda correcta para cada puerta. Las formas de onda correctas se muestran en color negro en la Figura 5.47(b), sobreimpresas a las formas de onda medidas realmente.

2. Comparar las formas de onda puerta por puerta hasta encontrar una señal medida que no se corresponda con la señal correcta.

En este ejemplo, todas las comprobaciones son correctas hasta llegar a la puerta G_3. La salida de ésta es incorrecta de acuerdo con las diferencias que se indican. Un análisis de la señal pone de manifiesto que, si la entrada D de la puerta G_3 está en circuito abierto y opera como un nivel ALTO, se obtendría la forma de onda de salida medida (mostrada en negro). Observe que la salida de G_4 también es incorrecta ya que la entrada procedente de G_3 es incorrecta.

Reemplace el circuito integrado que contiene a G_3, y pruebe de nuevo el funcionamiento del circuito.

Problema relacionado Para las entradas de la Figura 5.47(b), determinar la señal de salida del circuito lógico (salida de G_4), si la salida del inversor está en circuito abierto.

CONSEJOS PRÁCTICOS

Como ya sabemos, probar y localizar los fallos en circuitos lógicos frecuentemente requiere observar y comparar dos formas de onda digitales de forma simultánea, tal como una entrada y la salida de una puerta, en un osciloscopio de doble canal. Para las formas de onda digitales, el osciloscopio siempre debería configurarse con acoplamiento DC en cada entrada de canal, para evitar los "desplazamientos" del nivel de tierra. Deberá determinar dónde se encuentra el nivel de 0 V en la pantalla para ambos canales.

Para comparar la temporización de las formas de onda, el osciloscopio debería dispararse sólo desde un canal (no utilice el disparo en modo vertical o compuesto). El canal seleccionado para disparo debería ser siempre aquél que tenga la frecuencia más baja, cuando sea posible.

REVISIÓN DE LA SECCIÓN 5.7

1. Enumerar cuatro tipos de fallos comunes en las puertas lógicas.
2. Una entrada de una puerta NOR está externamente cortocircuitada a $+V_{CC}$. ¿Cómo afecta esta condición al funcionamiento de la puerta?
3. Determinar la salida de la puerta G_4 de la Figura 5.47(a), si se aplican las entradas de la Figura 5.47(b), para los fallos siguientes:
   (a) Una entrada de G_1 cortocircuitada a masa.
   (b) La entrada del inversor cortocircuitada a masa.
   (c) Una salida en circuito abierto en G_3.

APLICACIÓN A LOS SISTEMAS DIGITALES

En esta aplicación a los sistemas digitales, se desarrolla la lógica de control de un sistema digital que permite controlar el fluido que hay en un tanque de almacenamiento. El propósito de la lógica es mantener un nivel apropiado de fluido controlando las válvulas de entrada y de salida. La lógica también tiene que controlar la temperatura del fluido dentro de un determinado rango y disparar una alarma si el sensor de nivel o el sensor de temperatura falla.

Funcionamiento básico del sistema

Las salidas de la lógica de control del sistema controlan la entrada de fluido, la salida de fluido y la temperatura del mismo. La lógica de control actúa sobre una válvula de entrada que permite que el fluido entre en el tanque hasta que el sensor de nivel alto se activa al quedar sumergido en el fluido. Cuando el sensor de nivel alto está sumergido (activado) la lógica de control cierra la válvula de entrada. El fluido contenido en el tanque debe mantenerse dentro de un rango de temperatura especificado, el cual queda determinado por dos sensores de temperatura. Uno de los sensores de temperatura indica si el fluido está demasiado caliente y el otro si el fluido está demasiado frío. La lógica de control activa un elemento de calefacción si los sensores de temperatura indican que el fluido está demasiado

frío. La lógica de control mantiene abierta la válvula de salida siempre que el sensor de nivel bajo esté sumergido y el fluido se encuentre a la temperatura adecuada. Cuando el nivel de fluido cae por debajo del sensor de nivel bajo, la lógica de control cierra la válvula de salida.

Requisitos de operación

Los niveles máximo y mínimo de fluido quedan determinados por las posiciones de los sensores de nivel del tanque. La salida de cada sensor estará a nivel ALTO mientras que esté sumergido en el fluido y estará a nivel BAJO cuando no quede sumergido. Cuando la salida del sensor de nivel alto está a nivel BAJO, la lógica de control genera un nivel ALTO y abre la válvula de entrada. Cuando la salida del sensor de nivel alto está a nivel ALTO, la lógica de control genera un nivel BAJO y cierra la válvula de entrada.

Antes de abrir la válvula de salida, el fluido debe encontrarse dentro del rango de temperatura especificado. Un sensor genera un nivel ALTO cuando el fluido está muy caliente y el otro sensor de temperatura genera un nivel ALTO cuando la temperatura es demasiado baja. La lógica de control genera un nivel ALTO para activar el elemento de calefacción cuando se tiene la indicación de temperatura baja; en caso contrario, el elemento de calefacción está apagado. Cuando aparece la condición de temperatura alta, se activa una alarma.

Cuando el sensor de nivel bajo genera una salida a nivel ALTO (lo que indica que está sumergido) y la salida de los dos sensores de temperatura están a nivel BAJO (lo que indica que el fluido está a la temperatura correcta), la lógica de control abre la válvula de salida. Si la salida del sensor de nivel bajo pasa a nivel BAJO o si las salidas de los sensores de temperatura pasan a nivel ALTO, la lógica de control cierra la válvula de salida.

Si la lógica de control detecta un fallo en cualquiera de los sensores o una condición de temperatura muy alta, se activa una alarma. Un fallo en un sensor de nivel se produce cuando el sensor de nivel alto está activado y el sensor de nivel bajo no lo está. Un fallo en un sensor de temperatura se indica mediante la activación de los dos sensores a un mismo tiempo. La Figura 5.48 muestra el sistema de control del tanque.

En la Tabla 5.6 se resumen las entradas y las salidas del sistema y en la Tabla 5.7 se ilustra la tabla de verdad.

Diseño de la lógica de control

Hay cuatro salidas diferentes, una para la válvula de entrada, una para la válvula de salida, una para el sistema de calefacción y una para la alarma. Vamos a abordar el diseño como cuatro circuitos lógicos separados.

Lógica de la válvula de entrada Comenzamos diseñando el circuito lógico para la válvula de entrada. La salida de este circuito lógico es la variable V_{ENTRADA}. El primer paso consiste en transferir los datos de la tabla de verdad a un mapa de Karnaugh y desarrollar una expresión suma de productos.

Las variables de entrada L_H, L_L, T_H y T_L son las variables del mapa y los estados de V_{ENTRADA} se dibujan y agrupan como se muestra en la Figura 5.49(a). Los 0s del mapa son las condiciones de entrada cuando la válvula de entrada está cerrada y los 1s son las condiciones de entrada

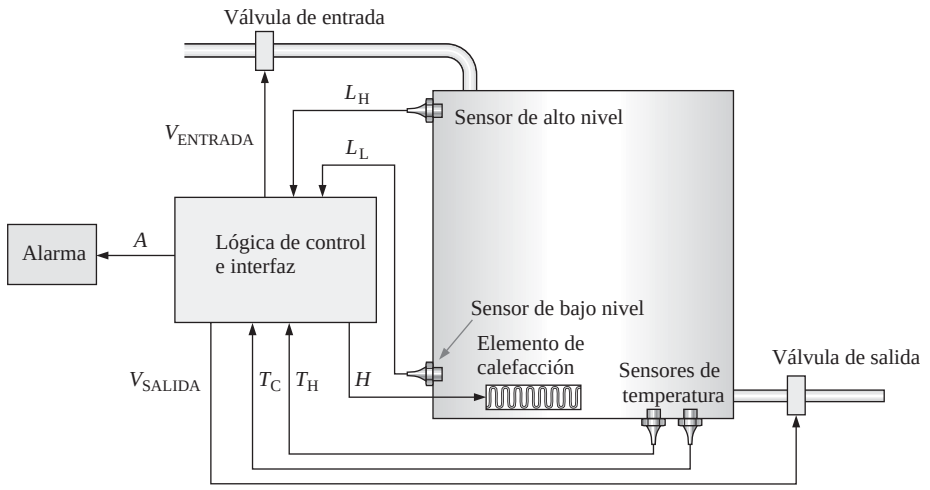


FIGURA 5.48 Tanque de almacenamiento de fluido con controles y sensores de nivel y de temperatura.

ENTRADAS DE LA LÓGICA DE CONTROL

Variable	Descripción	Nivel activo	Comentarios
L_H	Sensor de nivel alto	ALTO (1)	El sensor está sumergido
L_L	Sensor de nivel bajo	ALTO (1)	El sensor está sumergido
T_H	Sensor de temperatura alta	ALTO (1)	Temperatura muy alta
T_L	Sensor de temperatura baja	ALTO (1)	Temperatura muy baja

SALIDAS DE LA LÓGICA DE CONTROL

Variable	Descripción	Nivel activo	Comentarios
$V_{ENTRADA}$	Válvula de entrada	ALTO (1)	Válvula abierta
V_{SALIDA}	Válvula de salida	ALTO (1)	Válvula abierta
H	Elemento de calefacción	ALTO (1)	Calefacción activada
A	Alarma	ALTO (1)	Fallo de sensor o condición de temperatura alta

TABLA 5.6 Entradas y salidas de la lógica de control del tanque.

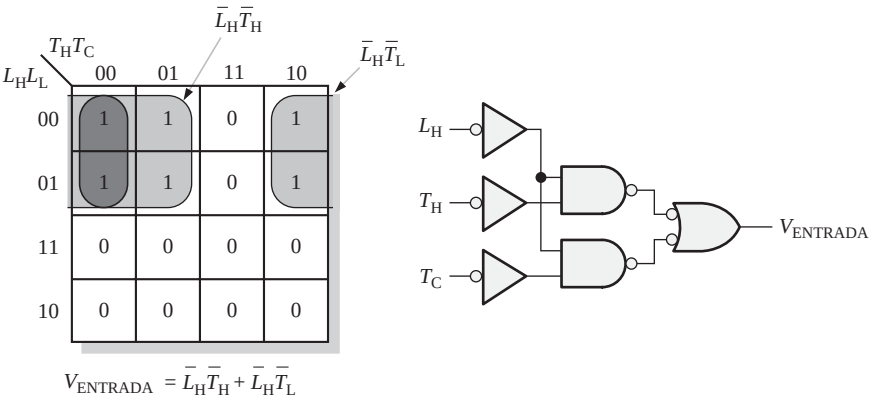
Entradas				Salidas				
L_H	L_L	T_H	T_L	$V_{ENTRADA}$	V_{SALIDA}	H	A	Comentarios
0	0	0	0	1	0	0	0	Rellenar/calefacción apagada
0	0	0	1	1	0	1	0	Rellenar/calefacción encendida
0	0	1	0	1	0	0	1	Rellenar/calefacción apagada/alarma
0	0	1	1	0	0	0	1	Fallo sensor de temp./alarma
0	1	0	0	1	1	0	0	Rellenar y vaciar/ calefacción apagada
0	1	0	1	1	0	1	0	Rellenar/calefacción encendido
0	1	1	0	1	0	0	1	Rellenar/calefacción apagada/alarma
0	1	1	1	0	0	0	1	Fallo sensor de temp./alarma
1	0	0	0	0	0	0	1	Fallo sensor de nivel/alarma
1	0	0	1	0	0	0	1	Fallo sensor de nivel/alarma
1	0	1	0	0	0	0	1	Fallo sensor de nivel/alarma
1	0	1	1	0	0	0	1	Fallo de varios sensores /alarma
1	1	0	0	0	1	0	0	Vaciar/calefacción apagada
1	1	0	1	0	0	1	0	Calefacción encendida
1	1	1	0	0	0	0	1	Calefacción apagada/alarma
1	1	1	1	0	0	0	1	Fallo de sensor de temp/alarma

TABLA 5.7 Tabla de verdad para la lógica de control del tanque.

cuando dicha válvula está abierta. La expresión suma de productos resultante para la lógica de la válvula de entrada da lugar a la implementación NAND mostrada en la parte (b) de la figura.

Lógica de la válvula de salida A continuación diseñamos el circuito lógico para la válvula de salida. La salida de este circuito lógico es V_{SALIDA}. De nuevo, el primer paso consiste en transferir los datos de la tabla de verdad a un mapa de Karnaugh y desarrollar una expresión suma de productos.

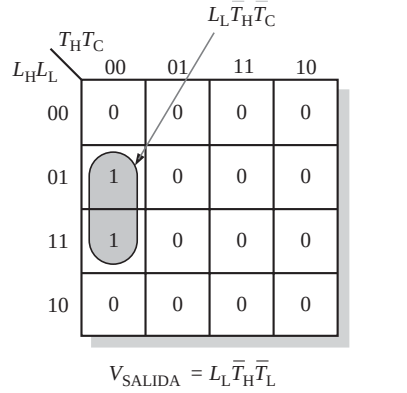
La variables de entrada, L_H, L_L, T_H y T_L son las variables del mapa y los estados de V_{SALIDA} se dibujan y agrupan



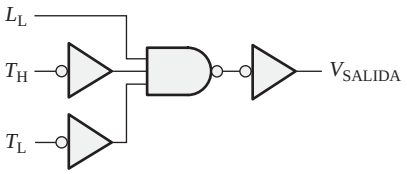
(a) Mapa de $V_{ENTRADA}$ (b) Circuito lógico

FIGURA 5.49 Simplificación mediante el mapa de Karnaugh e implementación de la lógica de la válvula de entrada.

como se muestra en la Figura 5.50(a). Los 0s del mapa son las condiciones de entrada cuando la válvula de salida está cerrada y los 1s son las condiciones de entrada cuando dicha válvula está abierta. La expresión suma de productos resultante para la lógica de la válvula de salida da lugar a la implementación NAND mostrada en la parte (b).



(a) Mapa de V_{SALIDA}



(b) Circuito lógico

FIGURA 5.50 Simplificación mediante el mapa de Karnaugh e implementación de la lógica de la válvula de salida.

Código VHDL para la lógica de las válvulas de entrada y de salida (opcional)

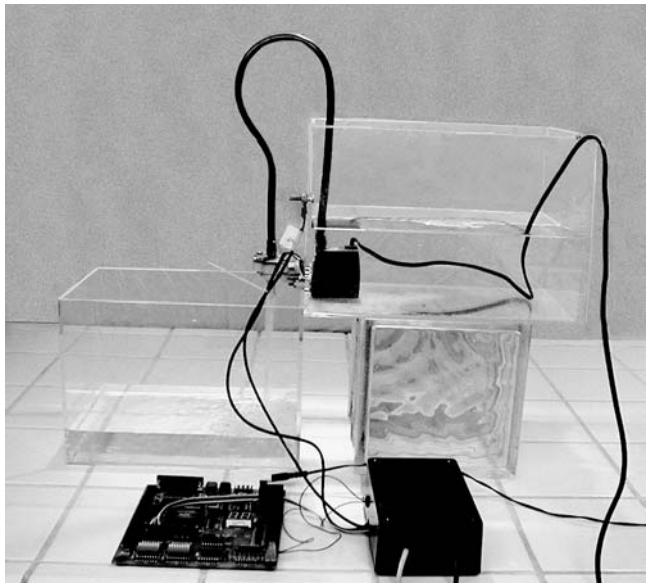
Una misma entidad y arquitectura describen la lógica de la válvula de entrada y la lógica de la válvula de salida utilizando el método de flujo de datos, como muestra el siguiente programa.

```
entity TankControl is
  port (LL, LH, TH, TL in bit; Ventrada,
    Vsalida: out bit);
end entity TankControl;
architecture ValveLogic of TankControl is
begin
    Ventrada <= (not LH and not TH) or
      (not LH and not TL);
    Vsalida <= LL and not TH and not TL;
end architecture ValveLogic;
```

Se ha diseñado y desarrollado el código VHDL para la lógica de las válvulas de entrada y de salida. Ahora, es el momento de que complete el resto del diseño de la lógica de control para el elemento de calefacción y la alarma, y de que escriba el programa VHDL para implementar la lógica en un dispositivo objetivo.

Práctica de sistemas

■ ***Actividad 1.*** Utilizando la Tabla 5.7 y el método de mapa de Karnaugh, diseñar la lógica para controlar el



Una fotografía de una réplica de un tanque de almacenamiento del laboratorio de electrónica del Yuba College en California. La lógica de control se ha programado en un PLD sobre una tarjeta de desarrollo y se ha conectado al tanque para controlar el llenado y el vaciado del mismo. Cortesía de Doug Joksch.

elemento de calefacción del tanque. Utilice puertas NAND e inversores para implementar el circuito.

- ***Actividad 2.*** Diseñar la lógica para activar la alarma.
- ***Actividad 3.*** Combinar la lógica de cada una de las cuatro funciones de control del tanque en un diagrama lógico completo.
- ***Actividad opcional.*** Escribir la arquitectura y la entidad VHDL para la lógica completa, modificando el código anteriormente desarrollado para la lógica de las válvulas de entrada y de salida.

RESUMEN

- La lógica AND-OR genera una expresión de salida en forma de suma de productos.
- La lógica AND-OR_Inversor genera una forma suma de productos complementada, la cual realmente es una forma producto de sumas.
- El símbolo operacional para la operación OR-Exclusiva es $\oplus$. Una expresión OR-Exclusiva puede expresarse de dos formas equivalentes

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

- Para hacer un análisis de un circuito lógico, se parte del circuito lógico y se desarrolla la expresión de salida booleana o la tabla de verdad, o ambas.
- La implementación de un circuito lógico es el proceso por el que, partiendo de las expresiones booleanas de salida o de la tabla de verdad, se desarrolla un circuito que genera la función de salida.
- Todos los diagrama lógicos NAND y NOR deben dibujarse empleando los símbolos duales apropiados, de modo que las salidas invertidas (con círculo) se conecten a entradas invertidas y las salida no invertidas (sin círculo) se conecten a entradas no invertidas.
- Cuando se conectan dos indicadores de negación (círculos), se cancelan entre sí.

- Un componente VHDL es una función lógica predefinida que se almacena para utilizarla a lo largo de un programa o en otros programas.
- Una instantación de componente se utiliza para llamar a un componente en un programa.
- Una señal VHDL actúa como una interconexión interna en una descripción estructural VHDL.

PALABRAS CLAVE

Las palabras clave y otros términos que se han resaltado en negrita se encuentran en el glosario final del libro.

Componente Una funcionalidad VHDL que puede utilizarse para predefinir una función lógica que puede emplearse a lo largo de un programa o programas.
Negativa-AND La operación dual de una puerta NOR cuando las entradas son activas a nivel BAJO.
Negativa-OR La operación dual de una puerta NAND cuando las entradas son activas a nivel BAJO.
Nodo Punto de conexión común en un circuito, en el que la salida de una puerta se conecta a una o más entradas de puerta.
Puerta universal Tanto una puerta NAND como NOR. El término *universal* se refiere a la propiedad de aquellas puertas que permiten que cualquier operación lógica pueda ser implementada mediante ellas o mediante una combinación de puertas de ese tipo.
Seguimiento de señales Técnica de localización de averías mediante la cual se observan las señales paso a paso, comenzando en la entrada y siguiéndolas hasta la salida, o viceversa. En cada punto, las formas de ondas observadas se comparan con la señal correcta que debería haber en ese punto.
Señal Una forma de onda; un tipo de objeto VHDL que almacena datos.

AUTOTEST

Las respuestas se encuentran al final del capítulo.

1. La expresión de salida para un circuito AND-OR que tiene una puerta AND con las entradas A, B, C y D y una puerta AND con las entradas E y F es
(a) $ABCDEF$ **(b)** $A + B + C + D + E + F$
(c) $(A + B + C + D)(E + F)$ **(d)** $ABCD + EF$

2. Un circuito lógico con una salida $A\overline{B}C + A\overline{C}$ consta de
(a) Dos puertas AND y una puerta OR
(b) Dos puertas AND, una puerta OR y dos inversores
(c) Dos puertas OR, una puerta AND y dos inversores
(d) Dos puertas AND, una puerta OR y un inversor.

3. Para implementar la expresión $\overline{A}BCD + A\overline{B}CD + AB\overline{C}\overline{D}$, se necesita una puerta OR y
(a) una puerta AND
(b) tres puertas AND
(c) tres puertas AND y cuatro inversores
(d) tres puertas AND y tres inversores

4. La expresión $\overline{A}BCD + ABC\overline{D} + A\overline{B}\overline{C}D$
(a) no puede simplificarse
(b) puede simplificarse a $\overline{A}BC + A\overline{B}$
(c) puede simplificarse a $ABC\overline{D} + \overline{A}B\overline{C}$

(d) ninguna de las respuestas anteriores es correcta

5. La expresión de salida de un circuito AND-OR_Inversor que tiene una puerta AND con entradas A, B, C y D y una puerta AND con entradas E y F es

(a) $ABCD + EF$

(b) $\overline{A} + \overline{B} + \overline{C} + \overline{D} + \overline{E} + \overline{F}$

(c) $\overline{(A + B + C + D)(E + F)}$

(d) $(\overline{A} + \overline{B} + \overline{C} + \overline{D})(\overline{E} + \overline{F})$

6. Una expresión OR-Exclusiva se expresa como

(a) $\overline{A}\overline{B} + AB$

(b) $\overline{A}B + A\overline{B}$

(c) $(\overline{A} + B)(A + \overline{B})$

(d) $(\overline{A} + \overline{B}) + (A + B)$

7. La operación AND se puede generar con

(a) dos puertas NAND

(b) tres puertas NAND

(c) una puerta NOR

(d) tres puertas NOR

8. La operación OR se puede generar con

(a) dos puertas NOR **(b)** tres puertas NAND

(c) cuatro puerta NAND **(d)** las respuestas (a) y (b)

9. **C**uando se usan símbolos duales en un diagrama lógico

(a) las salida invertidas (con círculo) se conectan a las entradas invertidas (con círculo)

(b) los símbolos NAND generan las operaciones AND

(c) los símbolos negativa-OR generan las operaciones OR

(d) todas las respuestas son verdaderas

(e) ninguna respuesta es verdadera

10. Todas las expresiones booleanas pueden implementarse con

(a) sólo puertas NAND

(b) sólo puertas NOR

(c) combinaciones de puertas NAND y NOR

(d) combinaciones de puertas AND, puertas OR e inversores

(e) todas las anteriores

11. Un componente VHDL

(a) puede usarse sólo una vez en cada programa

(b) es una descripción predefinida de una función lógica

(c) puede utilizarse múltiples veces en un programa

(d) es parte de una descripción de flujo de datos

(e) las respuestas (b) y (c)

12. En un programa, se llama a un componente utilizando

(a) una señal **(b)** una variable

(c) una instantación de componente **(d)** una declaración de arquitectura

PROBLEMAS

Las respuestas a los problemas impares se encuentran al final del libro.

SECCIÓN 5.1 Circuitos lógicos combinacionales básicos

1. Dibujar el diagrama lógico con símbolos distintivos ANSI de un circuito AND-OR-Inversor de 4 entradas y triple anchura. Dibujar también el diagrama utilizando los símbolos rectangulares ANSI estándar.

2. Escribir la expresión de salida de los circuitos de la Figura 5.51.

3. Escribir la expresión de salida de los circuitos de la Figura 5.52.

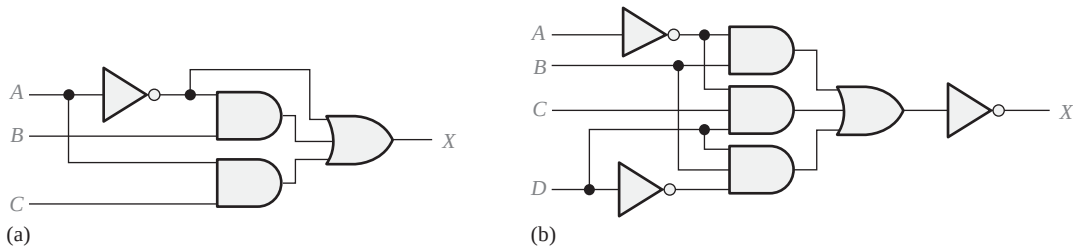


FIGURA 5.51

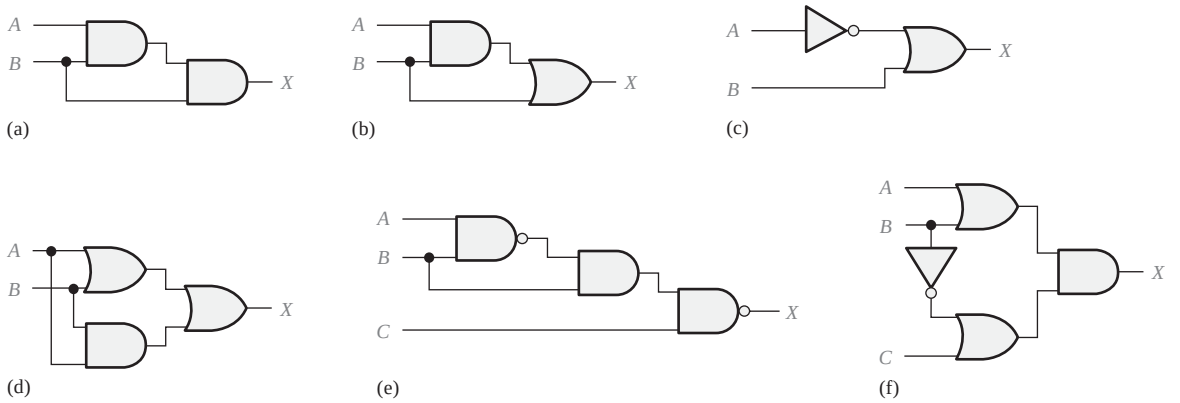


FIGURA 5.52

4. Escribir la expresión de salida de los circuitos de la Figura 5.53, y dibujar los circuitos equivalentes utilizando la configuración AND-OR.

5. Desarrollar la tabla de verdad de cada uno de los circuitos de la Figura 5.52.

6. Desarrollar la tabla de verdad de cada uno de los circuitos de la Figura 5.53.

7. Demostrar que un circuito NOR-exclusiva genera una salida que es un producto de sumas.

SECCIÓN 5.2 Implementación de la lógica combinacional

8. Utilizando puertas AND, puertas OR o combinaciones de ambas, implementar las siguientes expresiones lógicas:

(a) $X = AB$

(b) $X = A + B$

(c) $X = AB + C$

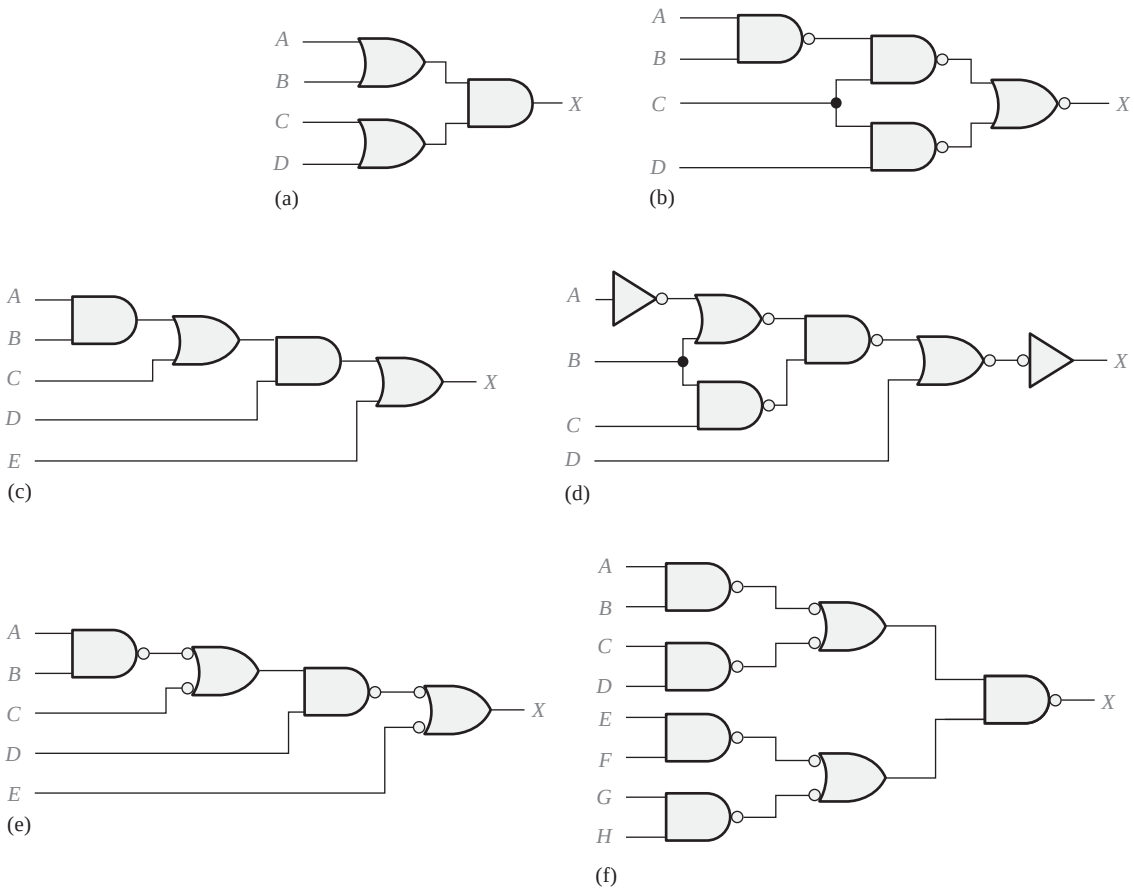


FIGURA 5.53

(d) $X = ABC + D$

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

(f) $X = ABC\,D$

(g) $X = A(CD + B)$

(h) $X = AB(C + DEF) + CE(A + B + F)$

9. Utilizando puertas AND, puertas OR e inversores cuando sea necesario, implementar las siguientes expresiones lógicas:

(a) $X = AB + \overline{B}C$ **(b)** $X = A(B + \overline{C})$

(c) $X = A\overline{B} + AB$ **(d)** $X = \overline{ABC} + B(EF + \overline{G})$

(e) $X = A[BC(A + B + C + D)]$ **(f)** $X = B(C\overline{D}E + \overline{E}FG)(\overline{AB} + C)$

10. Utilizando puertas NAND, puertas NOR o combinaciones de ambas, implementar las siguientes expresiones lógicas:

(a) $X = \overline{A}B + CD + (\overline{A + B})(ACD + \overline{BE})$

(b) $X = AB\overline{C}\overline{D} + D\overline{E}F + \overline{AF}$

(c) $X = A[B + \overline{C}(D + E)]$

11. Implementar un circuito lógico para la tabla de verdad de la Tabla 5.8.

Entradas			Salida
A	*B*	*C*	*X*
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

TABLA 5.8

12. Implementar un circuito lógico para la tabla de verdad de la Tabla 5.9.

Entradas				Salida
A	*B*	*C*	*D*	*X*
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

TABLA 5.9

13. Simplificar el circuito de la Figura 5.54 tanto como sea posible, y verificar que el circuito simplificado es equivalente al original, demostrando que las tablas de verdad son idénticas.

14. Repetir el Problema 13 para el circuito de la Figura 5.55.

15. Minimizar las puertas requeridas para implementar las funciones de cada apartado del Problema 9 en forma de suma de productos.

16. Minimizar las puertas requeridas para implementar las funciones de cada apartado del Problema 10 en forma de suma de productos.

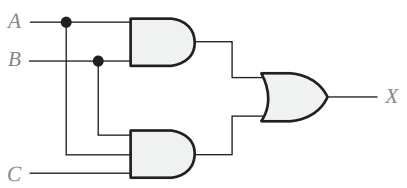


FIGURA 5.54

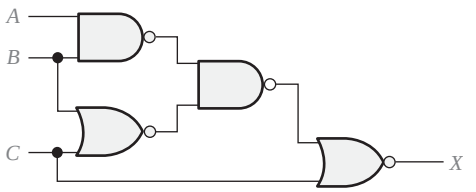


FIGURA 5.55

17. Minimizar las puertas requeridas para implementar la función de los circuitos de cada apartado de la Figura 5.53 en forma de suma de productos.

SECCIÓN 5.3 La propiedad universal de las puertas NAND y NOR

18. Implementar los circuitos lógicos de la Figura 5.51 utilizando sólo puertas NAND.

19. Implementar los circuitos lógicos de la Figura 5.55 utilizando sólo puertas NAND.

20. Repetir el Problema 18 utilizando sólo puertas NOR.

21. Repetir el Problema 19 utilizando sólo puertas NOR.

SECCIÓN 5.4 Lógica combinacional con puertas NAND y NOR

22. Mostrar cómo pueden implementarse las siguientes expresiones utilizando sólo puertas NOR:

(a) $X = ABC$ **(b)** $X = \overline{ABC}$ **(c)** $X = A + B$

(d) $X = A + B + \overline{C}$ **(e)** $X = \overline{AB} + \overline{CD}$ **(f)** $X = (A + B)(C + D)$

(g) $X = AB[C(\overline{DE} + \overline{AB}) + \overline{BCE}]$

23. Repetir el Problema 23 utilizando sólo puertas NAND.

24. Implementar cada una de las funciones del Problema 8 utilizando sólo puertas NAND.

25. Implementar cada una de las funciones del Problema 9 utilizando sólo puertas NAND.

SECCIÓN 5.5 Funcionamiento de los circuitos lógicos con trenes de impulsos

26. Dados el circuito lógico y las formas de onda de entrada de la Figura 5.56, dibujar la forma de onda de salida.

27. Para el circuito lógico de la Figura 5.57, dibujar la forma de onda de salida con respecto a las entradas.

FIGURA 5.56

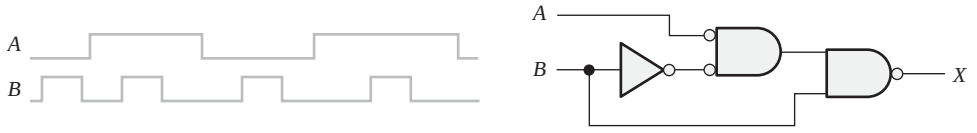


FIGURA 5.57

28. Para las formas de onda de entrada de la Figura 5.58, ¿qué circuito lógico generará la señal de salida mostrada?

29. Repetir el Problema 28 para la señal de la Figura 5.59.

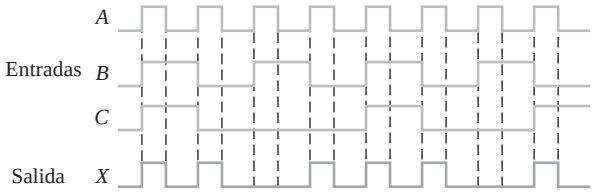


FIGURA 5.58

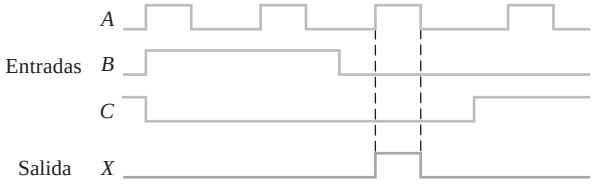


FIGURA 5.59

30. Para el circuito de la Figura 5.60, dibujar las formas de onda para los puntos numerados, indicando la relación de tiempos entre ellos.

31. Suponiendo un tiempo de propagación en cada puerta de 10 nanosegundos (ns), determinar si las entradas indicadas generarán la forma de onda de salida *X deseada* de la Figura 5.61 (impulso con un mínimo t_W = 25 ns como el mostrado).

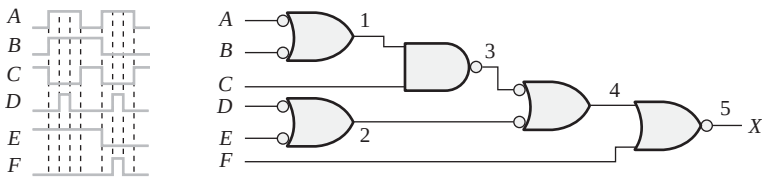


FIGURA 5.60

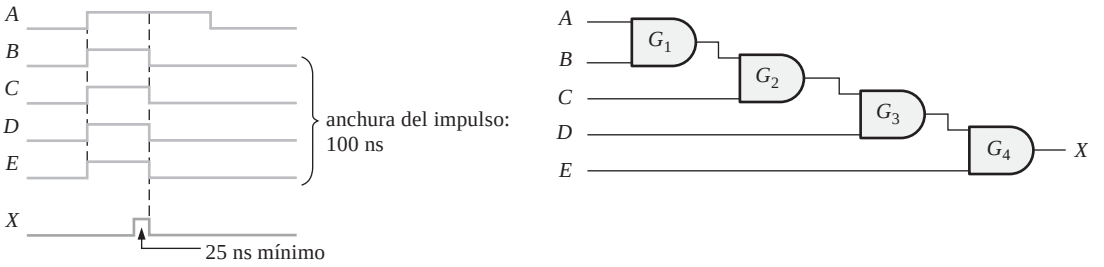


FIGURA 5.61

SECCIÓN 5.6 Lógica combinacional con VHDL (opcional)

32. Escribir un programa VHDL usando el método de flujo de datos (expresiones booleanas) para describir el circuito lógico de la Figura 5.51(b).

33. Escribir un programa VHDL usando el método de flujo de datos (expresiones booleanas) para describir los circuitos lógicos de las Figuras 5.52(e) y (f).

34. Escribir un programa VHDL usando el método estructural para describir el circuito lógico de la Figura 5.53(d). Suponer que están disponibles las declaraciones de componentes para cada tipo de puerta.

35. Repetir el Problema 34 para el circuito lógico de la Figura 5.53(f).

36. Describir la lógica representada por la tabla de verdad de la Tabla 5.8 utilizando VHDL, pasándola primero a una forma suma de productos.

37. Desarrollar un programa VHDL para la lógica de la Figura 5.64, utilizando los métodos de flujo de datos y estructural. Comparar los programas resultantes.

38. Desarrollar un programa VHDL para la lógica de la Figura 5.68, utilizando los métodos de flujo de datos y estructural. Comparar los programas resultantes.

39. Dado el siguiente programa VHDL, crear la tabla de verdad que describe el circuito lógico.

```
entity CombLogic is
  port (A, B, C, D: in bit; X: out bit);
end entity CombLogic;
architecture Example of CombLogic is
  begin
    X <= not((not A and not B) or (not A and not C) or (not A and not D) or
          (not B and not C) or(not B and not D) or (not D and not C));
end architecture Example;
```

40. Describir el circuito lógico mostrado en la Figura 5.62 con un programa VHDL, utilizando el método de flujo de datos.

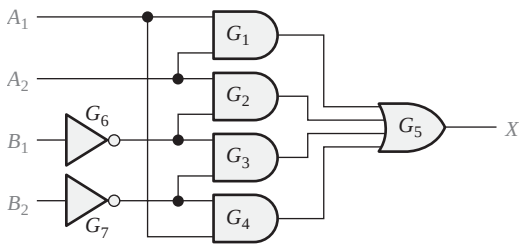


FIGURA 5.62

41. Repetir el Problema 40 utilizando el método estructural.

SECCIÓN 5.7 Localización de averías

42. Para el circuito lógico y la señal de entrada de la Figura 5.63, se observa la señal de salida indicada. Determinar si esta señal de salida es correcta.

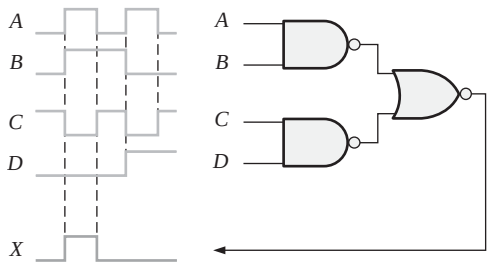


FIGURA 5.63

43. La forma de onda de salida de la Figura 5.64 es incorrecta para las entradas que se aplican al circuito. Suponiendo que una puerta del circuito está fallando, con su salida a un nivel ALTO o BAJO constante, determinar la puerta que falla y el tipo de fallo (circuito abierto o cortocircuito).

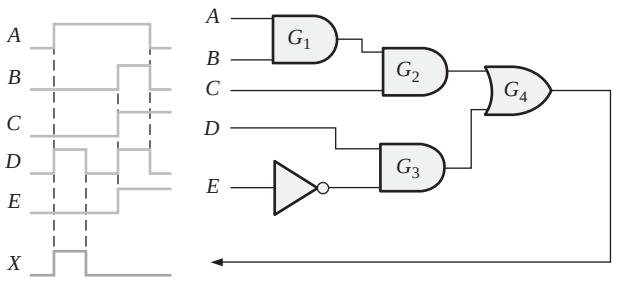


FIGURA 5.64

44. Repetir el Problema 43 para el circuito de la Figura 5.65 para las señales de entrada y salida dadas.

45. Examinando las conexiones de la Figura 5.66, determinar la puerta excitadora y las puertas de carga. Especificar por dispositivo y números de pines.

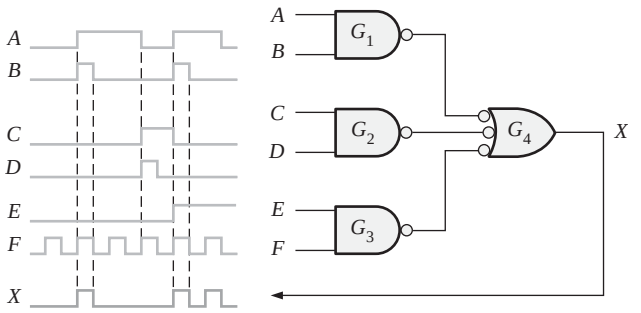


FIGURA 5.65

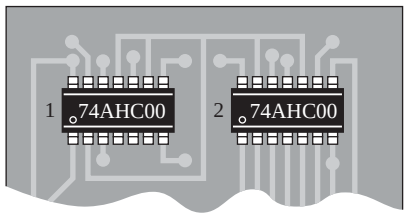
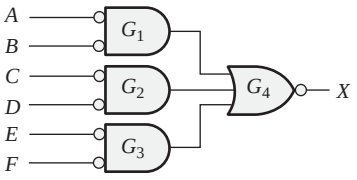
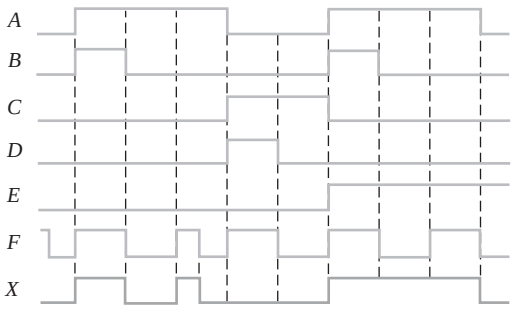


FIGURA 5.66

46. La Figura 5.67(a) es un circuito lógico bajo prueba. La Figura 5.67(b) muestra las formas de onda que se observan en el analizador lógico. Para las entradas que se aplican al circuito, la salida es incorrecta. Suponiendo que una puerta del circuito ha fallado, estando su salida a un nivel ALTO o a nivel BAJO constante, determinar la puerta que falla y el tipo de fallo.



(a)



(b)

FIGURA 5.67

47. Al circuito lógico de la Figura 5.68 se le aplican las formas de onda de entrada mostradas.

(a) Determinar la señal de salida correcta con respecto a las entradas.

(b) Determinar la señal de salida si la salida de la puerta G_3 está en circuito abierto.

(c) Determinar la señal de salida si la entrada superior de la puerta G_3 está cortocircuitada a masa.

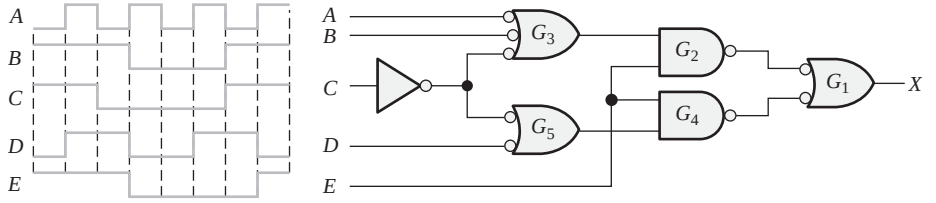


FIGURA 5.68

48. El circuito lógico de la Figura 5.69 tiene disponible un único punto de prueba intermedio próximo a la salida. Para las entradas indicadas, se observa la señal dada en el punto de prueba. ¿Es correcta esta forma de onda? Si no lo es, ¿cuáles son los posibles fallos que podrían generar dicha señal?

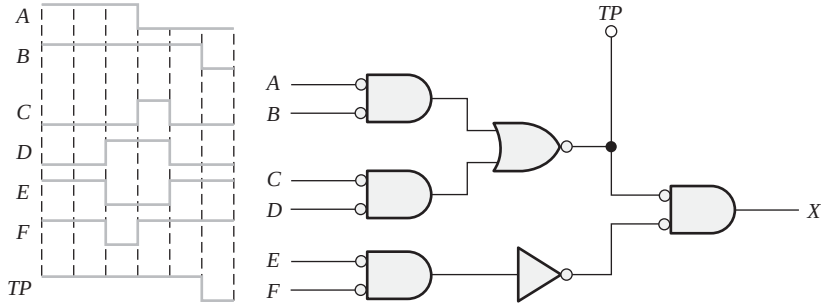


FIGURA 5.69

ICONO APLICACIONES A SISTEMAS

Aplicación a los sistemas digitales

49. Implementar la lógica de la válvula de entrada de la Figura 5.49(b) con puertas NOR e inversores.

50. Repetir el Problema 49 para la lógica de la válvula de salida de la Figura 5.50(b).

51. Implementar la lógica del elemento de calefacción y de la alarma usando puertas NOR e inversores.

Problemas especiales de diseño

52. Diseñar un circuito lógico para generar una salida a nivel ALTO si y sólo si la entrada, representada por un número binario de 4 bits, es mayor que doce o menor que tres. Desarrolle primero la tabla de verdad y después dibuje el diagrama lógico.

53. Desarrollar el circuito lógico que cumpla los siguientes requisitos:

Una lámpara situada en una habitación puede accionarse mediante dos interruptores, uno colocado detrás de la puerta y el otro frente a la puerta. La lámpara se enciende si se activa el interruptor frente a la puerta y el de detrás de la misma no se activa, o en el caso contrario. La lám-

para está apagada si ambos interruptores están desactivados o si ambos están activados. Una salida a nivel ALTO representa una condición de encendido y una salida a nivel BAJO representa la condición de apagado.

54. Diseñar un circuito que permite introducir un aditivo químico en el fluido a través de otra válvula de entrada sólo cuando la temperatura no sea ni demasiado baja ni demasiado alta y el fluido se encuentre por encima del sensor de nivel alto.

55. Desarrollar el diagrama lógico NAND para un codificador hexadecimal de teclado que convierta cada pulsación a binario.

RESPUESTAS

REVISIONES DE CADA SECCIÓN

SECCIÓN 5.1 Circuitos lógicos combinacionales básicos

1. **(a)** $\overline{AB+CD}=\overline{1\cdot 0+1\cdot 0}=1$

(b) $\overline{AB+CD}=\overline{1\cdot 1+0\cdot 1}=0$

(c) $\overline{AB+CD}=\overline{0\cdot 1+1\cdot 1}=0$

2. **(a)** $A\overline{B}+\overline{A}B=1\cdot\overline{0}+\overline{1}\cdot 0=1$

(b) $A\overline{B}+\overline{A}B=1\cdot\overline{1}+\overline{1}\cdot 1=0$

(c) $A\overline{B}+\overline{A}B=0\cdot\overline{1}+\overline{0}\cdot 1=1$

(d) $A\overline{B}+\overline{A}B=0\cdot\overline{0}+\overline{0}\cdot 0=0$

3. $X = 1$ cuando $ABC = 000, 011, 101, 110$ y 111. $X = 0$ cuando $ABC = 001, 010$ y 100.

4. $X=AB+\overline{A}\overline{B}$; el circuito está constituido por dos puertas AND, una puerta OR y dos inversores. Consulte el diagrama de la Figura 5.6(b).

SECCIÓN 5.2 Implementación de la lógica combinacional

1. **(a)** $X = ABC + AB + AC$; tres puertas AND, una puerta OR.

(b) $X = AB\ (C + DE)$; tres puertas AND, una puerta OR.

2. $X=ABC+\overline{A}\overline{B}\overline{C}$; dos puertas AND, una puerta OR y tres inversores.

3. **(a)** $X = AB(C + 1) + AC = AB + AC$

(b) $X = AB\ (C + DE) = ABC + ABDE$

SECCIÓN 5.3 La propiedad universal de las puertas NAND y NOR

1. **(a)** $X=\overline{A}+B$: es una puerta NAND con A y $\overline{B}$ en sus entradas.

(b) $X=A\overline{B}$: es una puerta NAND con A y $\overline{B}$ en sus entradas, seguida de una puerta NAND utilizada como inversor.

2. **(a)** $X=\overline{A}+B$: es una puerta NOR con $\overline{A}$ y B en sus entradas, seguida de una puerta NOR utilizada como inversor.

(b) $X=A\overline{B}$: es una puerta NOR con $\overline{A}$ y B en sus entradas.

SECCIÓN 5.4 Lógica combinacional con puertas NAND y NOR

1. $X=\overline{(\overline{A}+\overline{B}+\overline{C})DE}$: una puerta NAND de 3 entradas con las entradas A, B y C, con su salida conectada a una segunda puerta NAND de 3 entradas con otras dos entradas D y E.

2. $X=\overline{\overline{A}\overline{B}\overline{C}+(D+E)}$: una puerta NOR de 3 entradas con las entradas A, B y C, con su salida conectada a una segunda puerta NOR de 3 entradas con otras dos entradas D y E.

SECCIÓN 5.5 Funcionamiento de los circuitos lógicos con trenes de impulsos

1. La salida de la puerta OR-exclusiva es un impulso de 15 μs seguido de un impulso de 25 μs, con una separación de 10 μs entre los impulsos.
2. La salida de la puerta NOR-exclusiva es un nivel ALTO cuando ambas entradas están a nivel ALTO, o cuando ambas entradas están a nivel BAJO.

SECCIÓN 5.6 Lógica combinacional con VHDL (opcional)

1. Un componente VHDL es un programa predefinido que describe una función lógica especificada.
2. Una instantación de componente se utiliza para llamar a un componente especificado en una arquitectura de programa.
3. Las interconexiones entre componentes se hacen utilizando señales VHDL.
4. Los componentes se emplean en el método estructural.

SECCIÓN 5.7 Localización de averías

1. Los fallos más comunes en las puertas son entrada o salida en circuito abierto, y entrada o salida cortocircuitada a masa.
2. La entrada cortocircuitada a V_{CC} hace que la salida se mantenga a nivel BAJO.
3. **(a)** La salida de G_4 está a nivel alto hasta el flanco de bajada del séptimo impulso; luego pasa a nivel bajo.

   (b) La salida de G_4 es igual a la entrada D.

   (c) La salida de G_4 es la misma que la salida de G_2, mostrada en la Figura 5.47(b).

PROBLEMAS RELACIONADOS

5.1 $X = AB + AC + BC$

5.2 $X = \overline{AB + AC + BC}$

Si $A = 0$ y $B = 0, X = \overline{0 \cdot 0 + 0 \cdot 1 + 0 \cdot 1} = \overline{0} = 1$

Si $A = 0$ y $C = 0, X = \overline{0 \cdot 1 + 0 \cdot 0 + 1 \cdot 0} = \overline{0} = 1$

Si $B = 0$ y $C = 0, X = \overline{1 \cdot 0 + 1 \cdot 0 + 0 \cdot 0} = \overline{0} = 1$

5.3 No se puede simplificar.

5.4 No se puede simplificar.

5.5 $X = A + B + C + D$ es válida.

5.6 *Véase* la Figura 5.70.

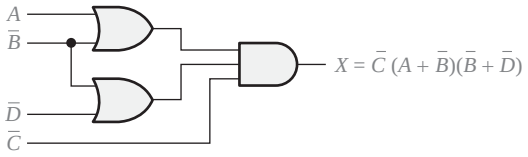


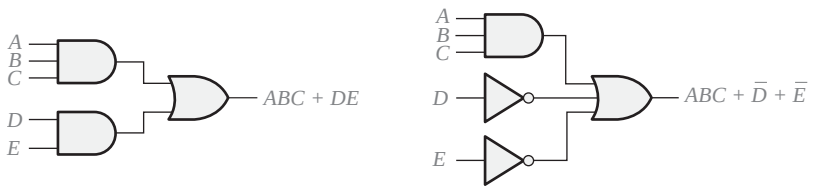
FIGURA 5.70

5.7 $X = \overline{(\overline{\overline{AB}C})(\overline{\overline{DEF}})} = (\overline{AB})C + (\overline{DE})F = (\overline{A} + \overline{B})C + (\overline{D} + \overline{E})F$

5.8 *Véase* la Figura 5.71.

5.9 $X = \overline{(\overline{\overline{A + B + C}}) + (\overline{\overline{D + E + F}})} = (\overline{A + B} + C)(\overline{D + E} + F) = (\overline{A}\overline{B} + C)(\overline{D}\overline{E} + F)$

5.10 *Véase* la Figura 5.72.



(a) (b)

FIGURA 5.71

5.11 *Véase* la Figura 5.73.

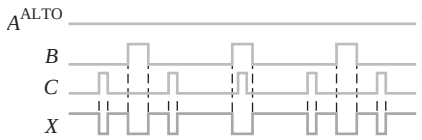


FIGURA 5.72

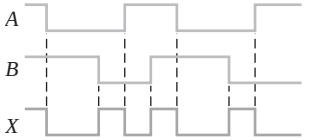


FIGURA 5.73

5.12 *Véase* la Figura 5.74.

5.13 *Véase* la Figura 5.75.

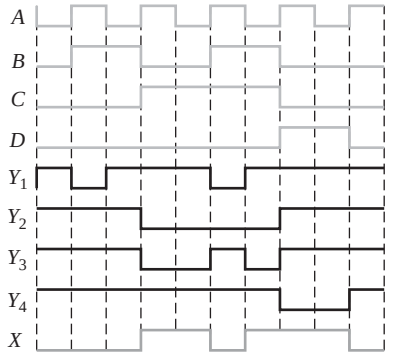


FIGURA 5.74

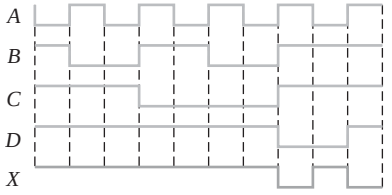


FIGURA 5.75

5.14 G5: NAND_gate2 **port map** (A => IN9, B =>IN10, X = > OUT4);

5.15 *Véase* la Figura 5.76.

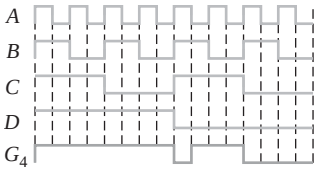


FIGURA 5.76

AUTOTEST

1. (d) **2.** (b) **3.** (c) **4.** (a) **5.** (d) **6.** (b) **7.** (a) **8.** (d)
9. (d) **10.** (e) **11.** (e) **12.** (c)

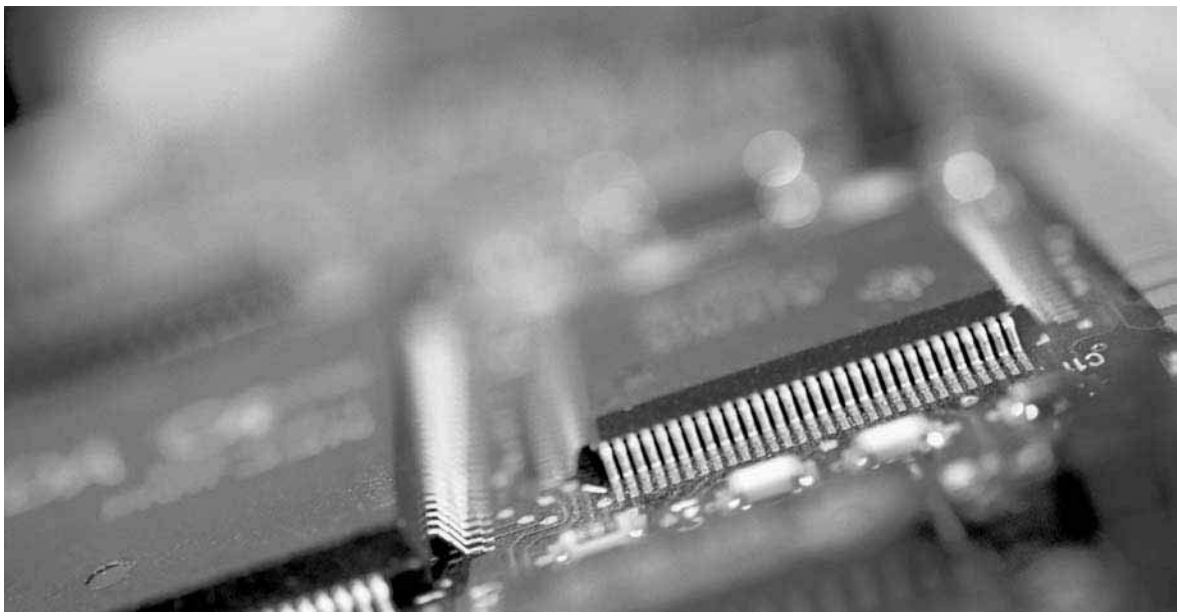
6

FUNCIONES DE LA LÓGICA COMBINACIONAL

CONTENIDO DEL CAPÍTULO

OBJETIVOS DEL CAPÍTULO

- Distinguir entre semi-sumadores y sumadores completos.



- Utilizar sumadores completos para implementar sumadores en paralelo binarios de múltiples bits.
- Explicar las diferencias entre sumadores con acarreo serie y sumadores paralelo con acarreo anticipado.
- Utilizar los comparadores de magnitud para determinar la relación entre dos números binarios y utilizar los comparadores en cascada para realizar comparaciones de números más grandes.
- Implementar un decodificador binario básico.
- Utilizar decodificadores BCD a 7-segmentos en sistemas con displays.
- Emplear un codificador de prioridad BCD-binario en un sencillo sistema con teclado.
- Convertir, utilizando dispositivos lógicos, números en código binario a código Gray, y códigos Gray a números binarios.
- Aplicar multiplexores para la selección de datos, los displays multiplexados, la generación de funciones lógicas y sistemas sencillos de comunicaciones.
- Utilizar decodificadores como demultiplexores.
- Explicar el significado de paridad.
- Usar generadores y comprobadores de paridad para detectar errores de bits en los sistemas digitales.
- Implementar un sencillo sistema de comunicación de datos.
- Identificar *glitches* (impulsos no deseados de muy corta duración), errores muy comunes en los sistemas digitales.

PALABRAS CLAVE

- Semi-sumador
- Conexión en cascada
- Acarreo serie
- Acarreo anticipado
- Decodificador
- Codificador
- Codificador de prioridad
- Multiplexor
- Demultiplexor
- Bit de paridad
- *Glitch*

INTRODUCCIÓN

En este capítulo se presentan distintos tipos de circuitos lógicos combinacionales, incluyendo sumadores, comparadores, decodificadores, codificadores, convertidores de código, multiplexores (selectores de datos), demultiplexores y generadores/comprobadores de paridad. También se incluyen ejemplos de circuitos integrados de función fija.

DISPOSITIVOS LÓGICOS DE FUNCIÓN FIJA

74XX42	74XX47	74XX85
74XX138	74XX139	74XX147
74XX148	74XX151	74XX154
74XX157	74XX280	74XX283

APLICACIÓN A LOS SISTEMAS DIGITALES

Esta aplicación a un sistema digital ilustra los conceptos que se enseñan en este capítulo y consiste en una parte de un sistema de control de semáforos. Las secciones "Aplicación a los sistemas digitales" de los Capítulos 6, 7 y 8 se centran en las distintas partes de este mismo sistema. Básicamente, este sistema se encarga de controlar el tráfico en la intersección de una calle con mucho tráfico y otra más despejada. El sistema está formado por una parte en la que se aplica la lógica combinacional, explicada en este capítulo, un circuito de temporización, que se verá en el Capítulo 7 y un sistema de lógica secuencial, al que se aplican los conceptos del Capítulo 8.

6.1 SUMADORES BÁSICOS

Los sumadores son muy importantes no solamente en las computadoras, sino en muchos tipos de sistemas digitales en los que se procesan datos numéricos. Comprender el funcionamiento de un sumador básico es fundamental en el estudio de los sistemas digitales. En esta sección se presentan el semi-sumador y el sumador completo.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Describir el funcionamiento de un semi-sumador. ■ Dibujar el diagrama lógico de un semi-sumador. ■ Describir el funcionamiento de un sumador completo. ■ Dibujar el diagrama lógico de un sumador completo utilizando semi-sumadores. ■ Implementar un sumador completo mediante lógica AND-OR.

El semi-sumador

Recordemos las reglas básicas de la suma binaria expuestas en el Capítulo 2:

$$0 + 0 = 0$$
$$0 + 1 = 1$$
$$1 + 0 = 1$$
$$1 + 1 = 10$$

▲ *Un semi-sumador suma dos bits y genera una suma y una salida de acarreo.*

Todas estas operaciones se realizan mediante un circuito lógico denominado ***semi-sumador***.

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

Los semi-sumadores se representan mediante el símbolo lógico de la Figura 6.1.

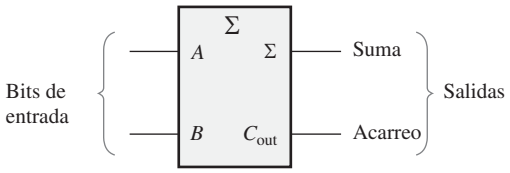


FIGURA 6.1 Símbolo lógico de un semi-sumador.

Lógica del semi-sumador. A partir del funcionamiento lógico de un semi-sumador, expuesto en la Tabla 6.1, las expresiones correspondientes a la suma y al acarreo de salida se pueden obtener como funciones de las entradas. Observe que la salida de acarreo (C_{out}) es 1 sólo cuando A y B son 1; por tanto, C_{out} puede expresarse como una operación AND de las variables de entrada.

Ecuación 6.1 $C_{out} = AB$

Observe ahora que la salida correspondiente a la suma (Σ) es 1 sólo si las variables A y B son distintas. Por tanto, la suma puede expresarse como una operación OR-exclusiva de las variables de entrada.

Ecuación 6.2 $\Sigma = A \oplus B$

A partir de las Ecuaciones (6.1) y (6.2), se puede desarrollar la implementación lógica del funcionamiento de un semi-sumador. La salida de acarreo se produce mediante una puerta AND, siendo A y B sus dos entra-

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

Σ = suma
C_{out} = acarreo de salida
A y B = variables de entrada (operandos)

TABLA 6.1 Tabla de verdad de un semi-sumador.

das, y la salida de la suma se obtiene mediante una puerta OR-exclusiva, como muestra la Figura 6.2. Recuerde que la operación OR-exclusiva se implementa con puertas AND, una puerta OR e inversores.

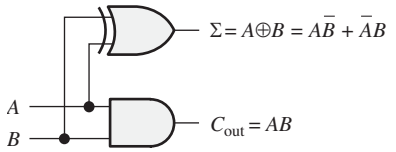


FIGURA 6.2 Diagrama lógico de un semi-sumador.

El sumador completo

▲ *Un sumador completo tiene un acarreo de entrada mientras que el semi-sumador no.*

El segundo tipo de sumador es el ***sumador completo.***

Un sumador acepta dos bits de entrada y un acarreo de entrada, y genera una salida de suma y un acarreo de salida.

La diferencia principal entre un sumador completo y un semi-sumador es que el sumador completo acepta un acarreo de entrada. El símbolo lógico de un sumador completo se muestra en la Figura 6.3, y la tabla de verdad mostrada en la Tabla 6.2 describe su funcionamiento.

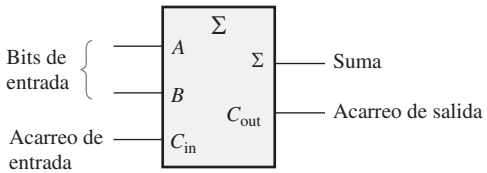


FIGURA 6.3 Símbolo lógico de un sumador completo.

Lógica del sumador completo. El sumador completo tiene que sumar dos bits de entrada y un acarreo de entrada. Del semi-sumador sabemos que la suma de los bits de entrada A y B es la operación OR-exclusiva de esas dos variables, $A \oplus B$. Para sumar el acarreo de entrada (C_{in}) a los bits de entrada, hay que aplicar de nuevo la operación OR-exclusiva, obteniéndose la siguiente ecuación para la salida de suma del sumador completo:

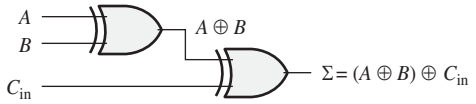
Ecuación 6.3 $\Sigma = (A \oplus B) \oplus C_{in}$

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

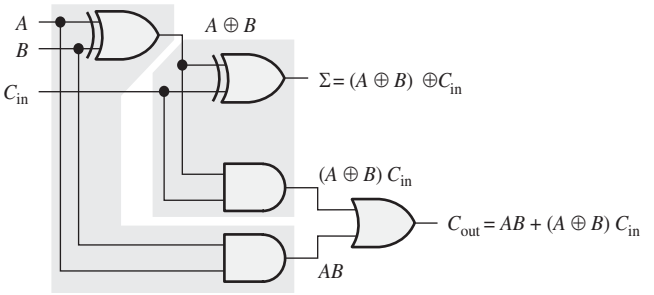
C_{in} = acarreo de entrada. Algunas veces se designa como *CI*.
C_{out} = acarreo de salida. Algunas veces se designa como *CO*.
Σ = suma
A y B = variables de entrada (operandos)

TABLA 6.2 Tabla de verdad de un sumador completo.

Esto significa que para implementar la función del sumador completo se pueden utilizar dos puertas OR-exclusiva de 2 entradas. La primera tiene que generar el término $A \oplus B$, y la segunda tiene como entradas la salida de la primera puerta XOR y el acarreo de entrada, como se ilustra en la Figura 6.4(a).



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



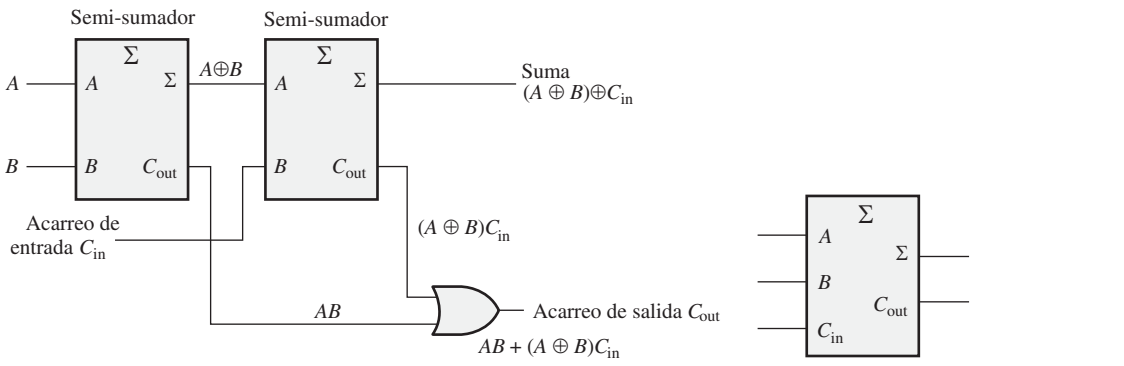
(b) Circuito lógico de un sumador completo (cada semi-sumador se representa por un área sombreada)

FIGURA 6.4 Lógica de un sumador completo.

El acarreo de salida es 1 cuando las dos entradas de la primera puerta XOR están a 1 o cuando las dos entradas de la segunda puerta XOR están a 1. Puede comprobar esto estudiando la Tabla 6.2. El acarreo de salida del sumador completo se obtiene por tanto del producto lógico (AND) de la entradas A y B, y del producto lógico de $A \oplus B$ y C_{in}. Después se aplica la operación OR a estos dos términos, como muestra la Ecuación 6.4. Esta función se implementa y se combina con la lógica de la suma para formar un circuito sumador completo, como se muestra en la Figura 6.4(b).

Ecuación 6.4 $$C_{out} = AB + (A \oplus B)\, C_{in}$$

Observe que, en la Figura 6.4(b), existen dos semi-sumadores conectados, como se muestra en el diagrama de bloques de la Figura 6.5(a), cuyos acarreos de salida se aplican a una puerta OR. El símbolo lógico mostrado en la Figura 6.5(b) será el que normalmente empleemos para representar un sumador completo.



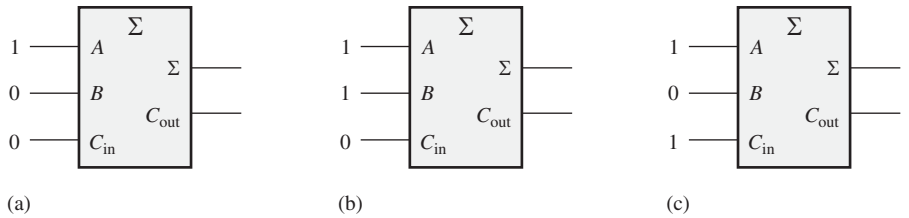
(a) Dos semisumadores para formar un sumador completo

(b) Símbolo lógico del sumador completo

FIGURA 6.5 Sumador completo implementado mediante semi-sumadores.

EJEMPLO 6.1

Para cada uno de los tres sumadores completos de la Figura 6.6, determinar las salidas para las entradas indicadas.



(a) (b) (c)

FIGURA 6.6

Solución

(a) Los bits de entrada son $A = 1$, $B = 0$ y $C_{in} = 0$.

$$1 + 0 + 0 = 1 \text{ sin acarreo}$$

Por tanto, $\mathbf{\Sigma = 1}$ y $\boldsymbol{C_{out} = 0}$

(b) Los bits de entrada son $A = 1$, $B = 1$ y $C_{in} = 0$.

$$1 + 1 + 0 = 0 \text{ con acarreo de 1}$$

Por tanto, $\mathbf{\Sigma = 0}$ y $\boldsymbol{C_{out} = 1}$

(c) Los bits de entrada son $A = 1$, $B = 0$ y $C_{in} = 1$.

$$1 + 0 + 1 = 0 \text{ con acarreo de 1}$$

Por tanto, $\mathbf{\Sigma = 0}$ y $\boldsymbol{C_{out} = 1}$

Problema relacionado* ¿Cuáles serán las salidas del sumador completo para $A = 1$, $B = 1$ y $C_{in} = 1$?

* Las respuestas se encuentran al final del capítulo.

REVISIÓN DE LA SECCIÓN 6.1

1. Determinar la suma (Σ) y el acarreo de salida (C_{out}) de un semi-sumador para cada uno de los siguientes grupos de bits de entrada:
   (a) 01 **(b)** 00 **(c)** 10 **(d)** 11
2. Un sumador completo tiene $C_{in} = 1$. ¿Cuánto vale la suma (Σ) y el acarreo de salida (C_{out}) cuando $A = 1$ y $B = 1$?

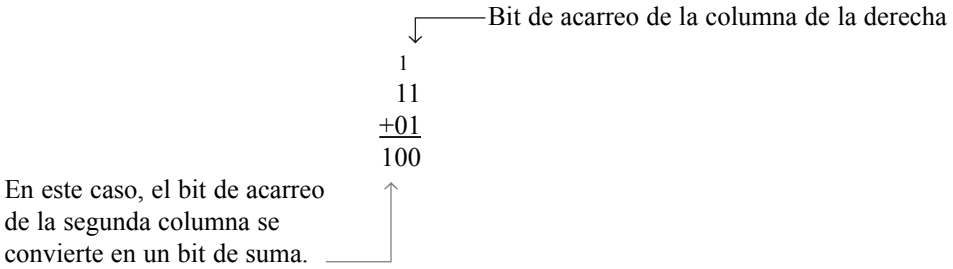
6.2 SUMADORES BINARIOS EN PARALELO

Para formar un sumador binario en paralelo se conectan dos o más sumadores completos. En esta sección aprenderemos los principios básicos de este tipo de sumador, de manera que podamos entender todas las funciones necesarias de entrada y salida cuando se trabaja con este tipo de dispositivos.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Utilizar sumadores completos para implementar un sumador binario en paralelo. ■ Explicar el proceso de adición en un sumador binario en paralelo. ■ Emplear la tabla de verdad para un sumador en paralelo de 4 bits. ■ Utilizar dos dispositivos 74LS283 para sumar dos números binarios de 4 bits. ■ Ampliar el sumador de 4 bits para poder realizar adiciones de 8 bits o 16 bits.

Como se ha visto en la Sección 6.1, un único sumador completo es capaz de sumar dos números binarios de 1 bit y un acarreo de entrada. Para sumar números binarios de más de un bit, se tienen que utilizar sumadores completos adicionales. Cuando se suman dos números binarios, cada columna genera un bit de suma y un 1 ó 0, correspondiente al bit de acarreo, que se añade a la columna inmediata de la izquierda, como se muestra a continuación con dos números de 2 bits.



Para sumar dos números binarios, se necesita un sumador completo por cada bit que tengan los números que se quieren sumar. Así, para números de dos bits se necesitan dos sumadores, para números de cuatro bits hacen falta cuatro sumadores, y así sucesivamente. La salida de acarreo de cada sumador se conecta a la entrada de acarreo del sumador de orden inmediatamente superior, como se muestra en la Figura 6.7 para un sumador de 2 bits. Téngase en cuenta que se puede usar un semi-sumador para la posición menos significativa, o bien se puede poner a 0 (masa) la entrada de acarreo de un sumador completo, ya que no existe entrada de acarreo en la posición del bit menos significativo.

En la Figura 6.7 los bits menos significativos (LSB) de los dos números se representan como A_1 y B_1. Los siguientes bits de orden superior se representan como A_2 y B_2. Los tres bits de suma son Σ_1, Σ_2 y Σ_3. Observe

NOTAS INFORMÁTICAS

Las computadoras realizan la operación de suma con dos números a un tiempo, denominados *operandos*. El *operando fuente* es un número que se añade a un número existente denominado *operando de destino*, que es el que se almacena en un registro de la UAL, tal como el acumulador. A continuación, la suma de los dos números se almacena de nuevo en el acumulador. La adición se realiza con números enteros o números en coma flotante utilizando, respectivamente, las instrucciones ADD o FADD.

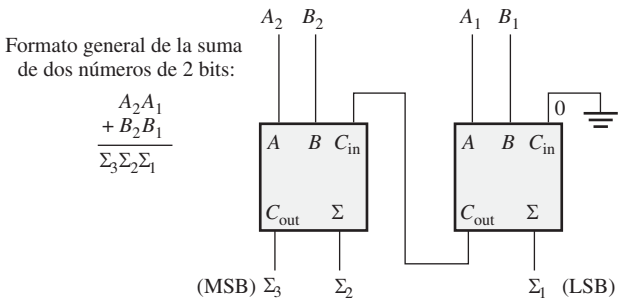


FIGURA 6.7 Diagrama de bloques de un sumador paralelo de 2 bits básico utilizando dos sumadores completos.

que el acarreo de salida del sumador completo de más a la izquierda se convierte en el bit más significativo (MSB) en la suma Σ_3.

EJEMPLO 6.2

Determinar la suma generada por el sumador paralelo de tres bits mostrado en la Figura 6.8 e indicar los acarreos intermedios cuando se están sumando los números 101 y 011.

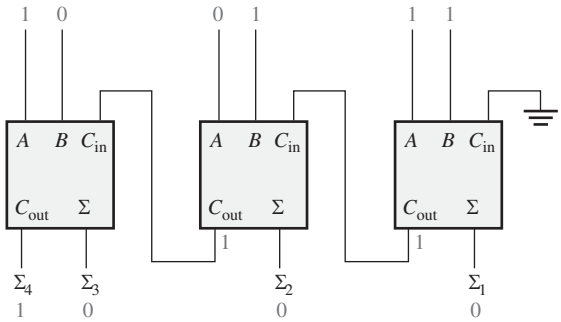


FIGURA 6.8

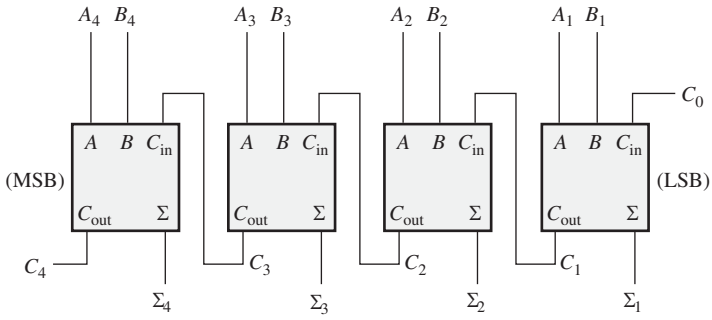
Solución Los bits menos significativos (LSB) de los dos números se suman en el sumador completo situado más a la derecha. En la Figura 6.8 se indican los bits de suma y los acarreos intermedios en negrita.

Problema relacionado Cuando se utiliza un sumador paralelo de 3 bits para sumar los números 111 y 101, ¿cuáles son las salidas de suma?

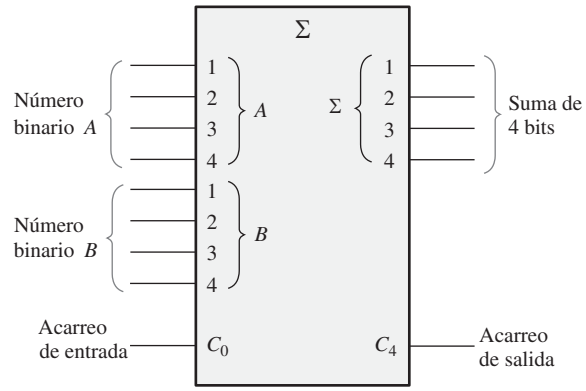
Sumadores en paralelo de cuatro bits

Un grupo de cuatro bits se denomina **nibble**. Un sumador básico en paralelo de 4 bits se implementa mediante cuatro sumadores completos, como se muestra en la Figura 6.9. De nuevo, los bits menos significativos (A_1 y B_1) de cada número que se suma, se introducen en el sumador completo que está más a la derecha; los bits

de orden más alto se introducen sucesivamente en los siguientes sumadores, aplicando los bits más significativos de cada número (A_4 y B_4) al sumador que está más a la izquierda. La salida de acarreo de cada sumador se conecta a la entrada de acarreo del siguiente sumador de orden superior. Estos acarreos se denominan *acarreos internos*.



(a) Diagrama de bloques



(b) Símbolo lógico

FIGURA 6.9 Sumador en paralelo de 4 bits.

En la mayoría de las hojas de características suministradas por los fabricantes, se denomina C_0 al acarreo de entrada del sumador del bit menos significativo; C_4, en el caso de cuatro bits, sería el acarreo de salida del sumador del bit más significativo; Σ_1 (LSB) hasta Σ_4 (MSB) son las sumas de salida. El símbolo lógico correspondiente se muestra en la Figura 6.9(b).

En función del método utilizado para manipular los acarreos en un sumador paralelo, existen dos tipos: el sumador de *acarreo serie* y el sumador *de acarreo anticipado*, que se estudian en la Sección 6.3.

Tabla de verdad de un sumador en paralelo de 4 bits

La Tabla 6.3 es la tabla de verdad de un sumador de 4 bits. En algunas hojas de características, las tablas de verdad se denominan *tablas de función* o *tablas de verdad funcionales*. El subíndice *n* representa los bits del sumador y puede ser igual a 1, 2, 3 o 4 para un sumador de 4 bits. C_{n-1} es el acarreo del sumador previo. Los acarreos C_1, C_2 y C_3 se generan internamente. C_0 es un acarreo de entrada externo y C_4 es una salida. El Ejemplo 6.3 ilustra cómo utilizar la Tabla 6.3.

C_{n-1}	A_n	B_n	Σ_n	C_n
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

TABLA 6.3 Tabla de verdad para cada etapa de un sumador en paralelo de 4 bits.

EJEMPLO 6.3

Utilizar la tabla de verdad del sumador en paralelo de 4 bits (Tabla 6.3) para hallar la suma y el acarreo de salida correspondientes a los siguientes dos números binarios de 4 bits, siendo el acarreo de entrada (C_{n-1}) igual a 0:

$$A_4A_3A_2A_1 = 1100 \quad \text{y} \quad B_4B_3B_2B_1 = 1100$$

Solución

Para $n = 1$: $A_1 = 0$, $B_1 = 0$ y $C_{n-1} = 0$. Según la primera fila de la tabla,

$$\mathbf{\Sigma_1 = 0} \text{ y } C_1 = 0$$

Para $n = 2$: $A_2 = 0$, $B_2 = 0$ y $C_{n-1} = 0$. Según la primera fila de la tabla,

$$\mathbf{\Sigma_2 = 0} \text{ y } C_2 = 0$$

Para $n = 3$: $A_3 = 1$, $B_3 = 1$ y $C_{n-1} = 0$. Según la cuarta fila de la tabla,

$$\mathbf{\Sigma_3 = 0} \text{ y } C_3 = 1$$

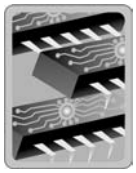
Para $n = 4$: $A_4 = 1$, $B_4 = 1$ y $C_{n-1} = 1$. Según la última fila de la tabla,

$$\mathbf{\Sigma_4 = 0} \text{ y } C_4 = \mathbf{1}$$

C_4 será el acarreo de salida; la suma de 1100 y 1100 es 11000.

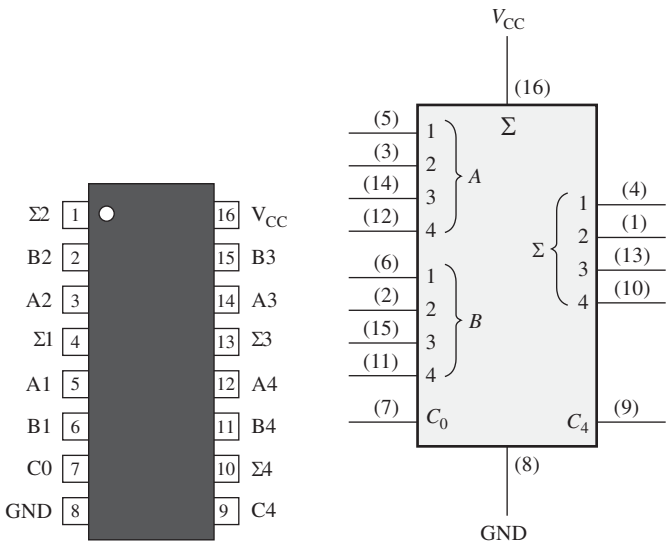
Problema relacionado Utilizar la tabla de verdad (Tabla 6.3) para calcular la suma de los números binarios 1011 y 1010.

SUMADORES PARALELO DE 4 BITS 74LS283



Un ejemplo de un sumador paralelo de 4 bits que está disponible como circuito integrado es el 74LS283. Para el 74LS83, V_{CC} es el pin 16 y tierra es el pin 8, que es una configuración estándar. El diagrama de pines y el símbolo lógico de este dispositivo se muestran en la Figura 6.10. Este dispositivo está disponible en las familias TTL y CMOS. Consulte el sitio web de Texas Instruments en www.ti.com.

Hoja de características del CI. Recuerde que las puertas lógicas tienen un retardo de propagación especificado, t_P, desde una entrada a la salida. Para estos dispositivos lógicos, existen varias especificaciones diferentes para este parámetro. El sumador paralelo de 4 bits dispone de 4 especificaciones para t_P, como se muestra en la Figura 6.11, tabla que es parte de una hoja de características del 74LS283.



(a) Diagrama de pines del 74LS283 (b) Símbolo lógico del 74LS283

FIGURA 6.10 Sumador en paralelo de 4 bits.

Símbolo	Parámetro	Límites			Unidad
		Mínimo	Típico	Máximo	
t_{PLH} t_{PHL}	Retardo de propagación; entrada de acarreo (C_0) a cualquier salida de suma Σ.		16 15	24 24	ns
t_{PLH} t_{PHL}	Retardo de propagación; cualquier entrada A o B a salidas de suma Σ.		15 15	24 24	ns
t_{PLH} t_{PHL}	Retardo de propagación; entrada de acarreo (C_0), a la salida de acarreo (C_4).		11 11	17 22	ns
t_{PLH} t_{PHL}	Retardo de propagación; cualquier entrada A o B a la salida de acarreo C_4.		11 12	17 17	ns

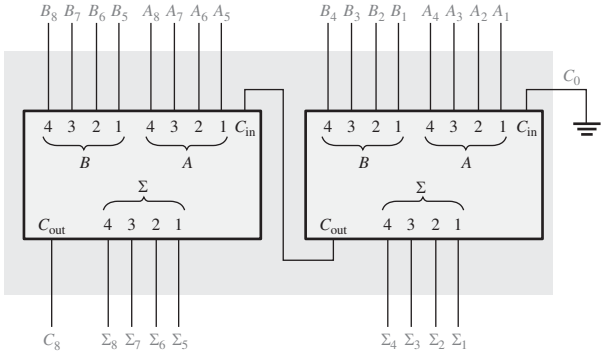
FIGURA 6.11 Características del retardo de propagación del 74LS283.

Expansión de sumadores

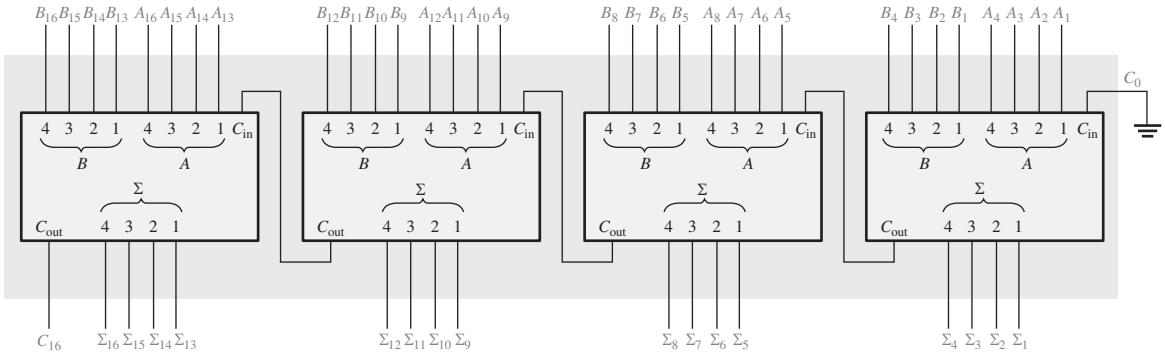
▲ *Los sumadores pueden ampliarse conectándose en cascada para trabajar con más bits.*

Un sumador en paralelo de 4 bits se puede expandir para realizar sumas de dos números de 8 bits, utilizando dos sumadores de cuatro bits. La entrada de acarreo del sumador de menor orden (C_0) se conecta a tierra, ya que no existe acarreo en la posición del bit menos significativo, y la salida de acarreo del sumador de menor orden se conecta a la entrada de acarreo del sumador de orden superior, como se muestra en la Figura 6.12(a). Este proceso se denomina ***conexión en cascada***. Observe que, en este caso, el acarreo de salida se designa como C_8, dado que se genera a partir del bit que se encuentra en la posición número ocho. El sumador de menor orden es el que realiza la suma de los cuatro bits menos sig-

nificativos, mientras que el sumador de orden superior es el que suma los cuatro bits más significativos de los dos números binarios de 8 bits.



(a) Sumadores de 4 bits conectados en cascada que forman un sumador de 8 bits



(b) Sumadores de 4 bits conectados en cascada que forman un sumador de 16 bits

FIGURA 6.12 Ejemplos de expansión de sumadores.

De forma similar, se pueden emplear cuatro sumadores de 4 bits en cascada para sumar dos números de 16 bits, como se muestra en la Figura 6.12(b). Observe que el acarreo de salida se designa como C_{16}, ya que se genera a partir del bit que se encuentra en la posición dieciséis.

EJEMPLO 6.4

Mostrar cómo se pueden conectar dos sumadores 74LS283 para formar un sumador en paralelo de 8 bits. Obtener los bits de salida para los siguientes números de entrada de 8 bits:

$$A_8\,A_7\,A_6\,A_5\,A_4\,A_3\,A_2\,A_1 = 10111001 \quad \text{y} \quad B_8\,B_7\,B_6\,B_5\,B_4\,B_3\,B_2\,B_1 = 10011110$$

Solución Se utilizan dos sumadores paralelo de 4 bits 74LS283 para implementar el sumador de 8 bits. La única conexión entre los dos 74LS283 es la que une la salida de acarreo (pin 9) del sumador de menor orden a la entrada de acarreo (pin 7) del sumador de orden superior, como se muestra en la Figura 6.13. El pin 7 del sumador de orden inferior se conecta a masa (no hay entrada de acarreo).

La suma de los dos números de 8 bits es:

$$\Sigma_9\Sigma_8\Sigma_7\Sigma_6\Sigma_5\Sigma_4\Sigma_3\Sigma_2\Sigma_1 = 101010111$$

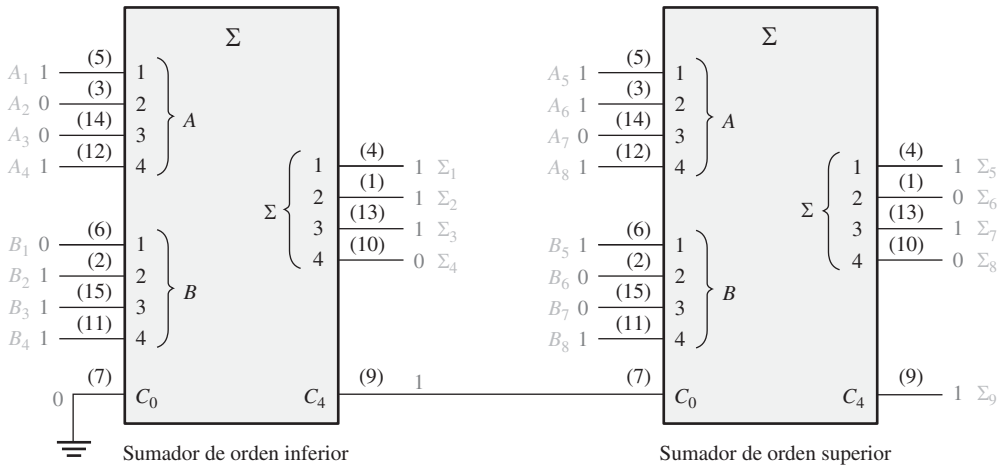


FIGURA 6.13 Dos sumadores 74LS283 conectados como un sumador en paralelo de 8 bits (la numeración de los pines se indica entre paréntesis).

Problema relacionado Utilizar sumadores 74LS283 para implementar un sumador en paralelo de 12 bits.

Aplicación

Un ejemplo de aplicación de un sumador completo y de un sumador en paralelo es un sencillo sistema de recuento de votos, que se puede utilizar simultáneamente para proporcionar el número de votos afirmativos y el de negativos. Por ejemplo, este tipo de sistema podría utilizarse en una reunión de personas donde se necesite determinar de forma inmediata el número de opiniones favorables para tomar decisiones, o votar sobre determinados temas.

En su forma más sencilla, el sistema se compone de un interruptor para seleccionar las dos posibles opciones (afirmativa o negativa) de cada persona en la reunión y un display digital para mostrar el número de votos afirmativos, y otro para los negativos. El sistema básico se muestra en la Figura 6.14, para un número de personas igual a 6, pero se puede ampliar a cualquier número de votantes, añadiendo nuevos módulos de 6 posiciones, sumadores en paralelo y circuitos de displays adicionales.

En la Figura 6.14, cada sumador completo puede generar la suma de hasta tres votos. La suma y el acarreo de salida de cada sumador completo se conectan a las dos entradas de menor orden de un sumador binario en paralelo. Las dos entradas de orden superior del sumador en paralelo se conectan a tierra (0), ya que nunca existe la posibilidad de que la entrada binaria sea mayor que 0011 (3 decimal). Para este sistema básico de 6 posiciones, las salidas del sumador en paralelo se conectan a un decodificador BCD a 7-segmentos que controla el display de 7-segmentos. Como ya se ha mencionado, se tienen que añadir circuitos adicionales si se decide ampliar el sistema.

Las resistencias entre las entradas de cada sumador completo y tierra aseguran que cada entrada se encuentra a nivel BAJO cuando el interruptor está en su posición neutra (se utiliza lógica CMOS). Cuando un inte-

rruptor se mueve a la posición “sí” o a la posición “no”, se aplica un nivel de tensión ALTO (V_{CC}) a la entrada del sumador completo asociado.

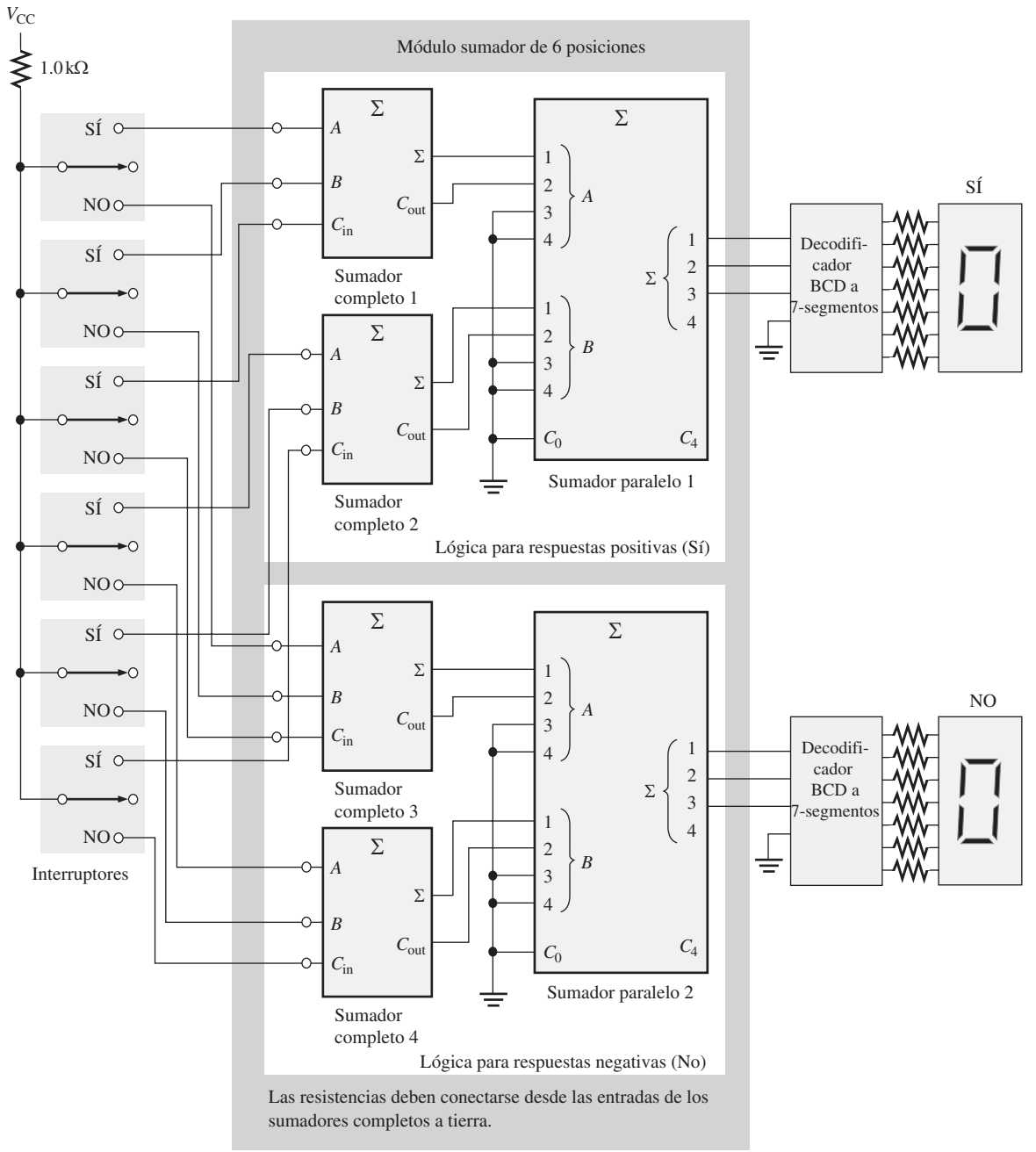


FIGURA 6.14 Sistema de votación realizado a partir de sumadores completos y sumadores binarios en paralelo.

REVISIÓN DE LA SECCIÓN 6.2

1. Se aplican dos números binarios de 4 bits (1101 y 1011) a un sumador en paralelo de 4 bits. El acarreo de entrada es 1. Determinar la suma (Σ) y el acarreo de salida.
2. ¿Cuántos sumadores 74LS283 se necesitarán para sumar dos números binarios, que representan números decimales iguales o inferiores a 1000_{10}?

6.3 SUMADORES DE ACARREO SERIE Y DE ACARREO ANTICIPADO

Como se ha mencionado en la sección anterior, los sumadores paralelo pueden clasificarse en dos categorías dependiendo de la forma en que se manejan los acarreos internos de una etapa a otra. Estas categorías son: acarreo serie y acarreo anticipado. Externamente, ambos tipos de sumador son iguales en términos de entradas y salidas. La diferencia se encuentra en la velocidad a la que se suman los números. El sumador de acarreo anticipado es mucho más rápido que el sumador de acarreo serie.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Explicar la diferencia entre un sumador de acarreo anticipado y un sumador de acarreo serie. ■ Explicar las ventajas de la operación suma utilizando acarreo anticipado. ■ Definir *generación de acarreo* y *propagación de acarreo*, y explicar la diferencia. ■ Desarrollar lógica de acarreo anticipado. ■ Explicar por qué los CI 74LS283 conectados en cascada presentan propiedades de acarreo en serie y de acarreo anticipado.

Sumador de acarreo serie

Un sumador de ***acarreo serie*** es aquel en el que la salida de acarreo de cada sumador completo se conecta a la entrada de acarreo de la siguiente etapa de orden inmediatamente superior (una etapa es un sumador completo). La suma y el acarreo de salida de cualquier etapa no se pueden generar hasta que tiene lugar el acarreo de entrada, lo que da lugar a un retardo temporal en el proceso de adición, como se muestra en la Figura 6.15. El retardo de propagación del acarreo para cada sumador completo es el tiempo transcurrido desde la apli-

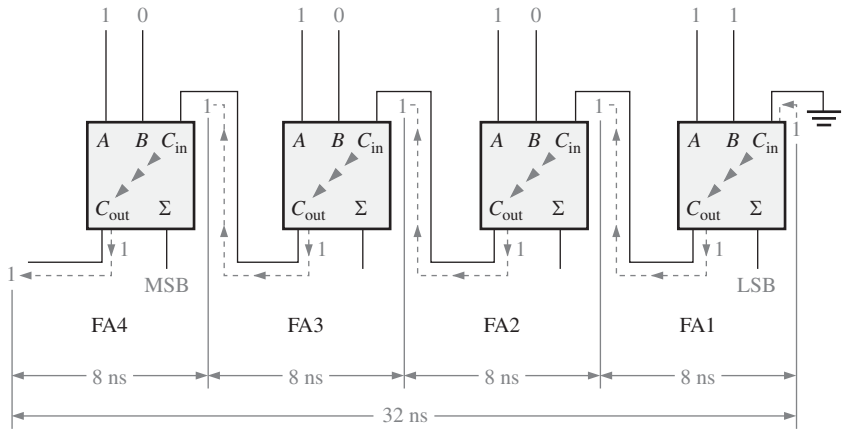


FIGURA 6.15 Un sumador paralelo de 4 bits con acarreo serie, que muestra los retardos de propagación del acarreo del caso peor.

cación del acarreo de entrada hasta que se produce el acarreo de salida, suponiendo que las entradas *A* y *B* ya existan.

El sumador completo 1 (FA1) no puede generar un acarreo de salida hasta que se aplique un acarreo de entrada. El sumador completo 2 (FA2) no puede generar un potencial acarreo de salida hasta que el sumador completo 1 produzca un acarreo de salida. El sumador completo 3 (FA3) no puede generar un potencial acarreo de salida hasta que FA1 produzca un acarreo de salida seguido de un acarreo de salida de FA2, y así sucesivamente. Como se puede ver en la Figura 6.15, el acarreo de entrada de la etapa menos significativa se transmite en serie a través de todos los sumadores antes de se produzca una suma final. El retardo acumulado a través de todas las etapas de sumador es el tiempo de suma del "caso peor". El retardo total puede variar dependiendo del bit de acarreo generado por cada sumador completo. Si se suman dos números que no generan acarreos (0) entre las etapas, el tiempo de suma es simplemente el tiempo de propagación de un solo sumador desde que se aplican los bits de datos en las entradas hasta que aparece la salida de suma.

Sumador de acarreo anticipado

La velocidad a la que se puede efectuar una suma está limitada por el tiempo necesario para que se propaguen los acarreos a través de todas las etapas de un sumador paralelo. Un método que permite acelerar el proceso de adición eliminando este retardo del acarreo serie es la adición con ***acarreo anticipado***. El sumador con acarreo anticipado anticipa el acarreo de salida de cada etapa y, en función de los bits de entrada de cada etapa, genera el acarreo de salida bien mediante la generación de acarreo o la propagación de acarreo.

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

Ecuación 6.5 $C_g = AB$

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

Ecuación 6.6 $C_p = A + B$

En la Figura 6.16 se ilustran las condiciones para la generación de acarreo y la propagación de acarreo. Las tres puntas de flecha simbolizan la propagación.

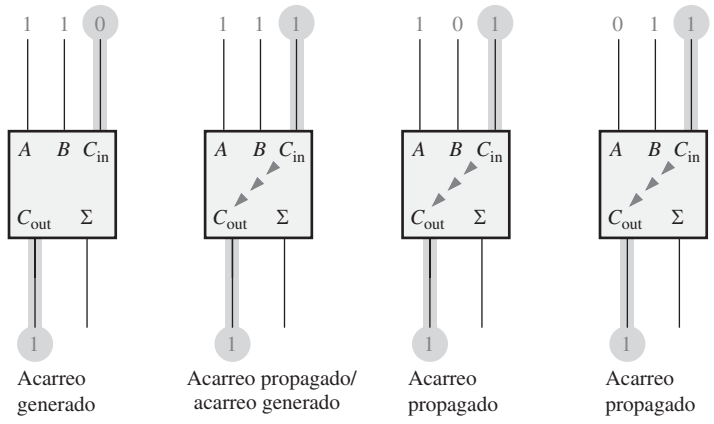


FIGURA 6.16 Ilustración de las condiciones de la generación y la propagación de acarreo.

El acarreo de salida de un sumador completo puede expresarse en función del acarreo generado (C_g) y el acarreo propagado (C_p). El acarreo de salida (C_{out}) es un 1 si el acarreo generado es 1 O si el acarreo propagado es 1 Y el acarreo de entrada (C_{in}) es 1. En otras palabras, obtenemos un acarreo de salida de 1 si el sumador completo los genera (A = 1 AND B = 1) o si el sumador propaga el acarreo de entrada (A = 1 OR B = 1) AND C_{in} = 1. Esta relación se expresa del siguiente modo:

Ecuación 6.7 $$C_{out} = C_g + C_p C_{in}$$

Veamos ahora cómo se puede aplicar este concepto a un sumador paralelo, cuyas etapas individuales se muestran en el ejemplo de 4 bits de la Figura 6.17. Para sumador completo, el acarreo de salida depende del acarreo generado (C_g), el acarreo propagado (C_p) y su acarreo de entrada (C_{in}). Las funciones C_g y C_p para cada etapa están disponibles *de forma inmediata* tan pronto como se aplican los bits de entrada A y B y el acarreo de entrada del sumador menos significativo (LSB), ya que sólo dependen de estos bits. El acarreo de entrada de cada etapa es el acarreo de salida de la etapa anterior.

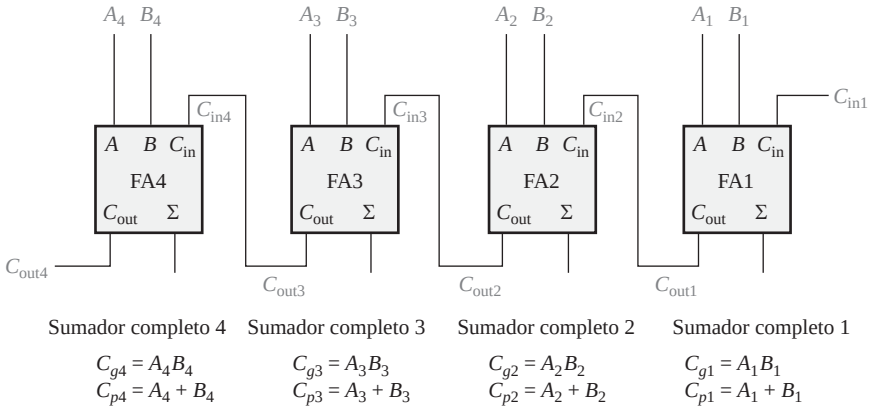


FIGURA 6.17 Generación de acarreo y propagación de acarreo en función de los bits de entrada en un sumador de 4 bits.

Basándonos en este análisis, podemos desarrollar expresiones para el acarreo de salida, C_{out} de cada etapa de sumador completo para el ejemplo de 4 bits.

Sumador completo 1:

$$C_{out1} = C_{g1} + C_{p1} C_{in1}$$

Sumador completo 2:

$$C_{in2} = C_{out1}$$
$$C_{out2} = C_{g2} + C_{p2} C_{in2} = C_{g2} + C_{p2} C_{out1} = C_{g2} + C_{p2}(C_{g1} + C_{p1} C_{in1})$$
$$= C_{g2} + C_{p2} C_{g1} + C_{p2} C_{p1} C_{in1}$$

Sumador completo 3:

$$C_{in3} = C_{out2}$$
$$C_{out3} = C_{g3} + C_{p3} C_{in3} = C_{g3} + C_{p3} C_{out2} = C_{g3} + C_{p3}(C_{g2} + C_{p2} C_{g1} + C_{p2} C_{p1} C_{in1})$$
$$= C_{g3} + C_{p3} C_{g2} + C_{p3} C_{p2} C_{g1} + C_{p3} C_{p2} C_{p1} C_{in1}$$

Sumador completo 4:

$$
\begin{aligned}
C_{\text{in4}} &= C_{\text{out3}} \\
C_{\text{out4}} &= C_{g4} + C_{p4}C_{\text{in4}} = C_{g4} + C_{p4}C_{\text{out3}} \\
&= C_{g4} + C_{p4}(C_{g3} + C_{p3}C_{g2} + C_{p3}C_{p2}C_{g1} + C_{p3}C_{p2}C_{p1}C_{\text{in1}}) \\
&= C_{g4} + C_{p4}C_{g3} + C_{p4}C_{p3}C_{g2} + C_{p4}C_{p3}C_{p2}C_{g1} + C_{p4}C_{p3}C_{p2}C_{p1}C_{\text{in1}}
\end{aligned}
$$

Observe que en cada una de estas expresiones, el acarreo de salida para cada etapa de sumador completo sólo depende del acarreo de entrada inicial (C_{in1}), las funciones C_g y C_p de dicha etapa y las funciones C_g y C_p de las etapas anteriores. Puesto que cada una de las expresiones de C_g y C_p pueden expresarse en función de las entradas A y B a los sumadores completos, todos los acarreos de salida están inmediatamente disponibles (excepto por los retardos de puerta) y no es necesario esperar a que se propague un acarreo a través de todas la etapas antes de obtener el resultado final. Por tanto, la técnica del acarreo anticipado acelera el proceso de adición.

Las ecuaciones de C_{out} se implementan con puertas lógicas y se conectan a los sumadores completos para crear un sumador de 4 bits con acarreo anticipado, como el que se muestra en la Figura 6.18.

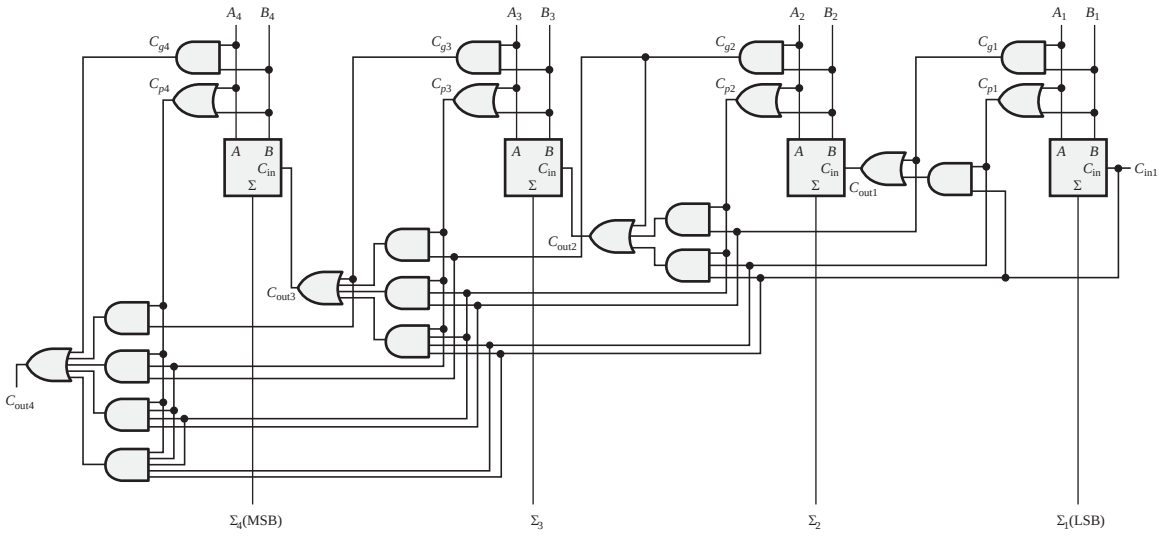


FIGURA 6.18 Diagrama lógico de un sumador de 4 etapas con acarreo anticipado.

Combinación de sumadores de acarreo serie y acarreo anticipado

En la Sección 6.2 se ha presentado el sumador de 4 bits 74LS283, que es un sumador con acarreo anticipado. Cuando estos sumadores se conectan en cascada para ampliar su capacidad de manejar números binarios de más de cuatro bits, el acarreo de salida de un sumador se conecta al acarreo de entrada del siguiente. Esto crea una condición de acarreo serie entre los sumadores de 4 bits, de modo que cuando se conectan en cascada dos o más 74LS283, el sumador resultante realmente es una combinación de un sumador con acarreo serie y acarreo anticipado. La operación de acarreo anticipado es interna en cada sumador MSI y la función de acarreo serie entra en juego cuando un acarreo pasa de un sumador al siguiente.

REVISIÓN DE LA SECCIÓN 6.3

1. Los bits de entrada de un sumador completo son $A = 1$ y $B = 0$. Determinar C_g y C_p.
2. Determinar el acarreo de salida de un sumador completo cuando $C_{in} = 1$, $C_g = 0$ y $C_p = 1$.

6.4 COMPARADORES

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

Al finalizar esta sección, el lector deberá ser capaz de:

■ Utilizar una puerta OR-exclusiva como comparador básico. ■ Analizar la lógica interna de un comparador de magnitud que posee tanto salida de igualdad como de desigualdad. ■ Utilizar el comparador 74HC85 para la comparación de dos números binarios de 4 bits. ■ Conectar en cascada comparadores 74HC85, para comparar números de ocho o más bits.

Igualdad

Como ya vimos en el Capítulo 3, la puerta OR-exclusiva se puede emplear como un comparador básico, ya que su salida es 1 si sus dos bits de entrada son diferentes y 0 si son iguales. La Figura 6.19 muestra una puerta OR-exclusiva utilizada como comparador de 2 bits.

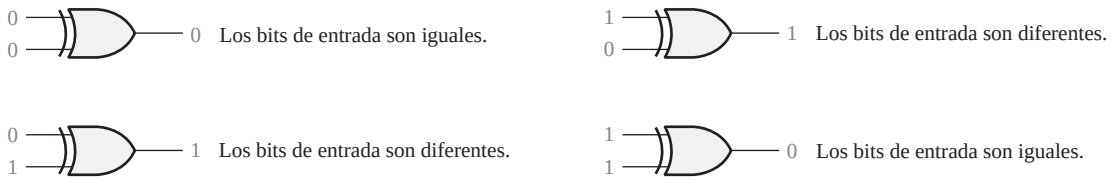
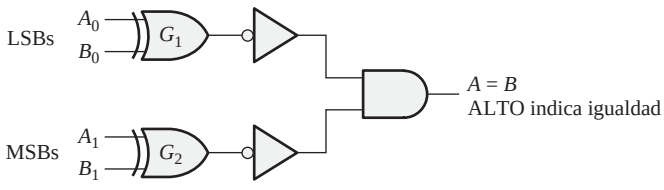


FIGURA 6.19 Funcionamiento del comparador básico.

Para comparar números binarios de dos bits, se necesita una puerta OR-exclusiva adicional. Los dos bits menos significativos (LSB) de ambos números se comparan mediante la puerta G_1 y los dos más significativos (MSB) son comparados mediante la puerta G_2, como se muestra en la Figura 6.20. Si los dos números son iguales, sus correspondientes bits también lo son, y la salida de cada puerta OR-exclusiva será 0. Si los correspondientes conjuntos de bits no son idénticos, la salida de la puerta OR-exclusiva será un 1.



Formato general: Número binario $A \rightarrow A_1A_0$
Número binario $B \rightarrow B_1B_0$

FIGURA 6.20 Diagrama lógico de la comparación de igualdad de dos números de 2 bits.

▲ *Un comparador determina si dos números binarios son iguales o distintos.*

Para obtener un único resultado de salida que indique la igualdad o desigualdad entre los dos números, se pueden usar dos inversores y una puerta AND, como muestra la Figura 6.20. La salida de cada puerta OR-exclusiva se invierte y se aplica a la entrada de la puerta AND. Cuando los bits de entrada de cada OR-exclusiva son iguales, lo que quiere decir que los bits de ambos números son iguales, las entradas de la puerta AND son 1, por lo que el resultado a su salida también será 1. Cuando los dos números no son iguales, al menos uno o ambos conjuntos de bits será distinto, lo que da lugar a, al menos, un 0 en una de las entradas de la puerta AND, y el resultado a su salida será 0. Por tanto, la salida de la puerta AND indica la igualdad (1) o desigualdad (0) entre dos números.

El Ejemplo 6.5 ilustra esta operación para dos casos específicos. La puerta OR-exclusiva y el inversor se han reemplazado por un símbolo NOR-exclusiva.

EJEMPLO 6.5

Aplicar cada uno de los siguientes conjuntos de números binarios a las entradas del comparador de la Figura 6.21 y determinar la salida, evaluando los niveles lógicos a través del circuito.

(a) 10 y 10 **(b)** 11 y 10

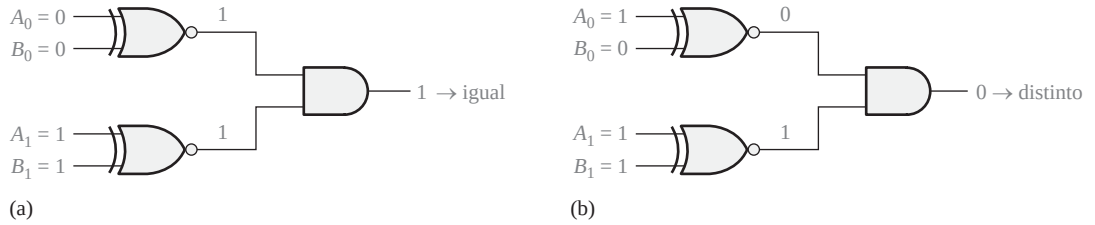


FIGURA 6.21

Solución **(a)** La salida es **1** para las entradas 10 y 10, como se muestra en la Figura 6.21(a).

(b) La salida es **0** para las entradas 11 y 10, como se muestra en la Figura 6.21(b).

Problema relacionado Repetir el proceso para las entradas binarias 01 y 10.

Como ya se ha visto en el Capítulo 3, un circuito comparador básico se puede ampliar para poder tratar cualquier número de bits. La puerta AND establece la condición de que todos los bits de los dos números que se comparan tienen que ser iguales si los números lo son.

Desigualdad

Además de disponer de una salida que indica si los dos números son iguales, muchos circuitos integrados comparadores tienen salidas adicionales que indican cuál de los dos números que se comparan es el mayor. Esto significa que existe una salida que indica cuándo el número A es mayor que el número B ($A > B$) y otra salida que indica cuándo A es menor que B ($A < B$), como se muestra en el símbolo lógico del comparador de cuatro bits de la Figura 6.22.

NOTAS INFORMÁTICAS

En una computadora, la *caché* es una memoria intermedia muy rápida entre la unidad de procesamiento central (CPU) y la memoria principal, más lenta. La CPU solicita los datos enviando la *dirección* (ubicación unívoca) en memoria. Parte de esta dirección se denomina *marcador*. El *comparador de marcadores de dirección* compara el marcador de la CPU con el marcador del directorio de la caché. Si ambos son iguales, quiere decir que los datos direccionados se encuentran ya en la caché y se recuperan de forma muy rápida. Si los marcadores son distintos, los datos deben recuperarse de la memoria principal a una velocidad mucho más lenta.

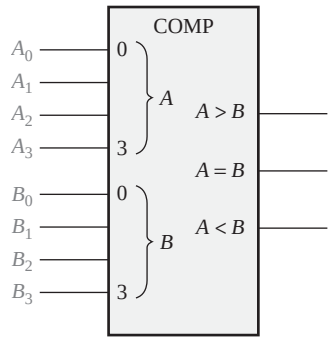


FIGURA 6.22 Símbolo lógico para un comparador de 4 bits con indicación de desigualdad.

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

EJEMPLO 6.6

Determinar las salidas $A = B$, $A > B$ y $A < B$ para los números de entrada mostrados en el comparador de la Figura 6.23.

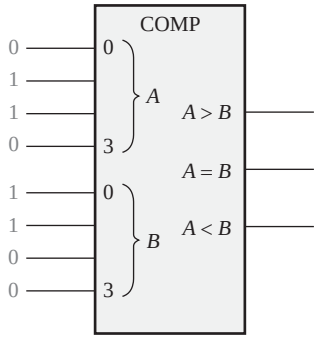


FIGURA 6.23

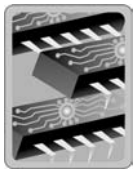
Solución El número que hay en las entradas *A* es 0110 y el número que hay en las entradas *B* es 0011. La **salida *A* > *B* está a nivel ALTO y las restantes salidas están a nivel BAJO.**

Problema relacionado ¿Cuáles serán las salidas del comparador cuando $A_3A_2A_1A_0 = 1001$ y $B_3B_2B_1B_0 = 1010$?

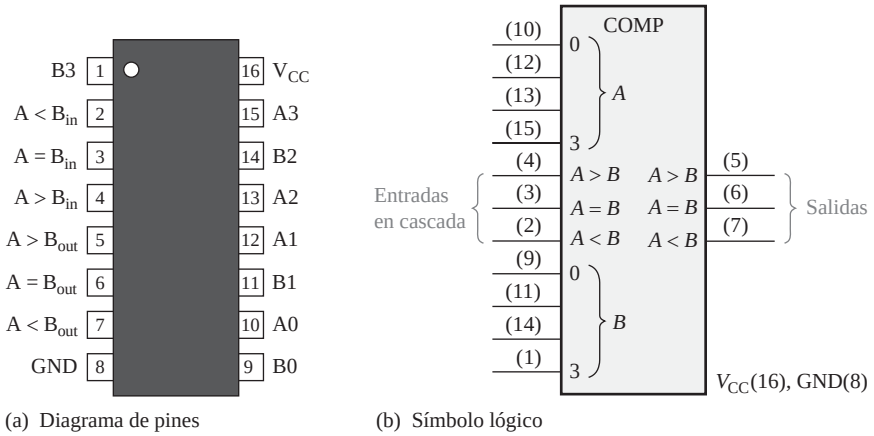
1. Si $A_3 = 1$ y $B_3 = 0$, entonces A es mayor que B.
2. Si $A_3 = 0$ y $B_3 = 1$, entonces A es menor que B.
3. Si $A_3 = B_3$, entonces tenemos que examinar los siguientes bits de orden inmediatamente inferior.

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

EL COMPARADOR DE MAGNITUD DE 4 BITS 74HC85



El 74HC85 es un comparador que también se encuentra disponible en otras familias de circuitos integrados. El diagrama de pines y el símbolo lógico se muestran en la Figura 6.24. Observe que este dispositivo tiene todas las entradas y salidas del comparador visto anteriormente y, además, tiene tres entradas en cascada: $A < B$, $A = B$ y $A > B$. Estas entradas permiten utilizar varios comparadores en cascada para la comparación de cualquier número binario con más de cuatro bits. Para ampliar el comparador, las salidas $A < B$, $A = B$ y $A > B$ del comparador de menor orden se conectan en cascada a las entradas del siguiente comparador de orden inmediatamente superior. El comparador de menor orden tiene que tener un nivel ALTO en la entrada $A = B$ y un nivel BAJO en las entradas $A < B$ y $A > B$. Este dispositivo está disponible en otras familias CMOS y TTL. Consulte el sitio web de Texas Instruments en www.ti.com.



(a) Diagrama de pines (b) Símbolo lógico

FIGURA 6.24 Diagrama de pines y símbolo lógico del comparador de magnitud de 4 bits 74HC85 (la numeración de los pines se muestra entre paréntesis).

EJEMPLO 6.7

Utilizar comparadores 74HC85 para comparar las magnitudes de dos números de 8 bits. Dibujar los comparadores con sus correspondientes interconexiones.

Solución Se necesitan dos 74HC85 para comparar dos números de 8 bits. Éstos se conectan en cascada como se muestra en la Figura 6.25, empleando una disposición en cascada.

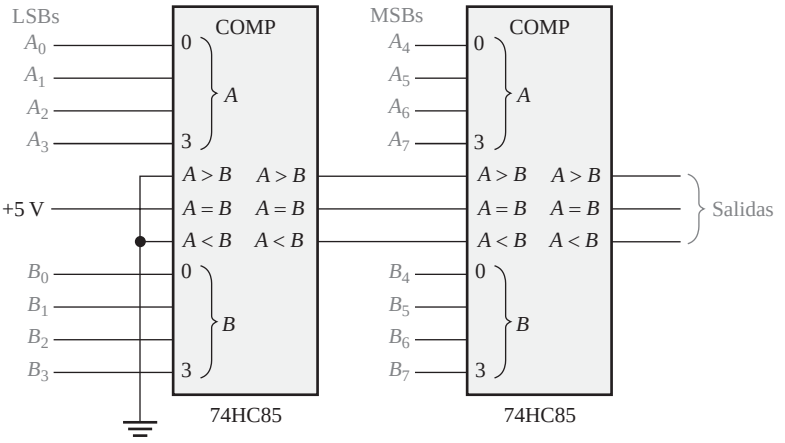


FIGURA 6.25 Comparador de 8 bits formado por dos 74HC85.

Problema relacionado Ampliar el circuito de la Figura 6.25 para realizar un comparador de 16 bits.

CONSEJOS PRÁCTICOS

La mayoría de los dispositivos CMOS contienen circuitería de protección para protegerse frente a daños generados por altas tensiones estáticas o campos eléctricos. Sin embargo, deben tomarse precauciones para evitar la aplicación de cualquier tensión mayor que la tensión máxima nominal. Para un funcionamiento correcto, las tensiones de entrada y salida deberían estar comprendidas entre tierra y V_{CC}. Recuerde también que las entradas no utilizadas deberán conectarse siempre a un nivel lógico apropiado (tierra o V_{CC}). Las salidas no utilizadas pueden dejarse en circuito abierto.

REVISIÓN DE LA SECCIÓN 6.4

1. Los números binarios A = 1011 y B = 1010 se aplican a las entradas de un 74HC85. Determinar las salidas.
2. Los números binarios A = 11001011 y B = 11010100 se aplican al comparador de 8 bits de la Figura 6.25. Determinar el estado de los pines de salida 5, 6 y 7 en cada uno de los 74HC85.

6.5 DECODIFICADORES

La función básica de un decodificador es detectar la presencia de una determinada combinación de bits (código) en sus entradas y señalar la presencia de este código mediante un cierto nivel de salida. En su forma general, un decodificador posee n líneas de entrada para gestionar n bits y en una de las 2^n líneas de salida indica la presencia de una o más combinaciones de n bits. En esta sección, se presentan varios tipos de decodificadores. Los principios básicos se pueden extender a otros decodificadores.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Definir *decodificador*. ■ Diseñar un circuito lógico para decodificar cualquier combinación de bits. ■ Describir el decodificador binario-decimal 74HC154. ■ Ampliar los decodificadores para poder tratar códigos con un número de bits superior. ■ Describir el decodificador BCD a 7-segmentos 74LS47. ■ Explicar la supresión de cero en los displays de 7-segmentos. ■ Utilizar los decodificadores en aplicaciones específicas.

NOTAS INFORMÁTICAS

Una *instrucción* indica a la computadora qué operación debe realizar. Las instrucciones se especifican en código máquina (1s y 0s) y, para que la computadora ejecute una instrucción, ésta debe ser decodificada. La decodificación de las instrucciones es uno de los pasos en la *pipeline* (secuencia de procesamiento) de las instrucciones; los pasos de dicho proceso son: la instrucción se lee desde la memoria (extracción de la instrucción), la instrucción se decodifica, se leen los operandos desde la memoria (extracción de operandos), se ejecuta la instrucción y el resultado se escribe de nuevo en memoria. Básicamente, el procesamiento *pipeline* permite que se comience a procesar la siguiente instrucción antes de haber completado la instrucción actual.

El decodificador binario básico

Supongamos que necesitamos determinar cuándo aparece el número binario 1001 en las entradas de un circuito digital. Se puede utilizar una puerta AND como elemento básico de decodificación, ya que produce una salida a nivel ALTO sólo cuando todas sus entradas están a nivel ALTO. Por tanto, debe asegurarse de que todas las entradas de la puerta AND estén a nivel ALTO cuando se introduce el número 1001, lo cual se puede conseguir invirtiendo los dos bits centrales (cuyos bits son 0), como se muestra en la Figura 6.26.

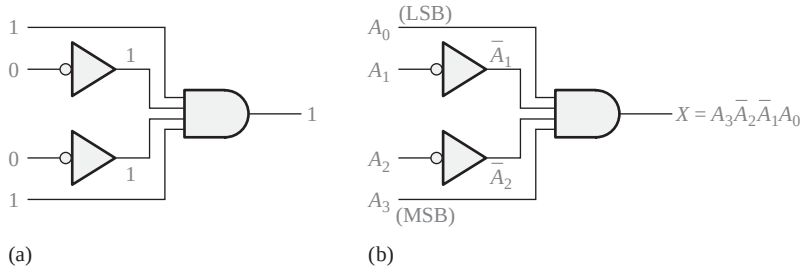


FIGURA 6.26 Lógica de decodificación del código binario 1001 con una salida activa a nivel ALTO.

La ecuación lógica para el **decodificador** de la Figura 6.26(a) se desarrolla como se ilustra en la Figura 6.26(b). Se debe comprobar que la salida es siempre 0 excepto cuando se aplican las entradas $A_0 = 1$, $A_1 = 0$, $A_2 = 0$ y $A_3 = 1$. A_0 es el bit menos significativo y A_3 el más significativo. *En este libro, cuando se representa un número binario o cualquier otro código de pesos, el bit menos significativo siempre es el situado más a la derecha cuando el número se escribe en sentido horizontal, y el de más arriba cuando se escribe en vertical, a menos que se indique lo contrario*.

Si se utiliza una puerta NAND en lugar de una AND, como se muestra en la Figura 6.26, una salida a nivel BAJO indicará la presencia del código binario adecuado, que en este caso es 1001.

EJEMPLO 6.8

Determinar la lógica requerida para decodificar el número binario 1011 de manera que produzca un nivel ALTO en la salida.

Solución

La función de decodificación la podemos realizar complementando sólo las variables cuyo valor es 0 en el número binario deseado, tal y como sigue:

$$X = A_3\overline{A}_2A_1A_0 \qquad (1011)$$

Esta función puede implementarse conectando las variables verdaderas (no complementadas) A_0, A_1 y A_3 directamente a las entradas de la puerta AND, e invirtiendo la variable A_2 antes de aplicarla a la puerta AND. La lógica de decodificación se muestra en la Figura 6.27.

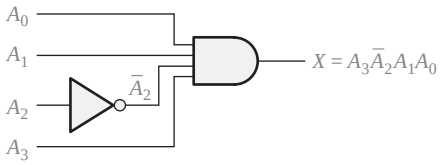


FIGURA 6.27 Lógica de decodificación para generar una salida a nivel ALTO cuando se aplica el código 1011 en las entradas.

Problema relacionado

Desarrollar la lógica requerida para detectar el código binario 10010 y generar una salida activa a nivel BAJO.

El decodificador de 4 bits

Para poder decodificar todas las posibles combinaciones de cuatro bits, se necesitan dieciséis puertas de decodificación (2^4=16). Este tipo de decodificador se denomina comúnmente *decodificador de 4 líneas a 16 líneas*, ya que existen cuatro entradas y dieciséis salidas, o también se le llama *decodificador 1 de 16*, ya que para cualquier código dado en las entradas, sólo se activa una de las dieciséis posibles salidas. En la Tabla 6.4 se muestra una lista de los dieciséis códigos binarios y sus correspondientes funciones de decodificación.

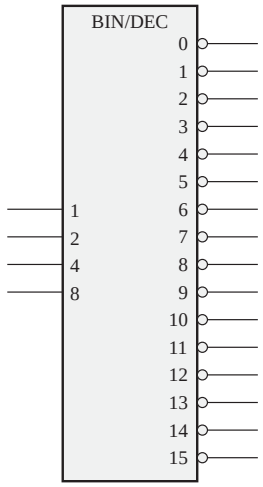


FIGURA 6.28 Símbolo lógico de un decodificador de 4-líneas a 16-líneas (1 de 16).

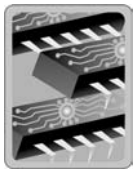
Si se necesita una salida activa a nivel BAJO para cada número decodificado, el decodificador completo se puede implementar mediante puertas NAND e inversores. Para decodificar cada uno de los dieciséis códigos binarios se requieren dieciséis puertas NAND (las puertas AND se pueden usar para producir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

El símbolo lógico de un decodificador de 4 líneas a 16 líneas (1 de 16) con salidas activas a nivel BAJO se muestra en la Figura 6.28. La etiqueta BIN/DEC indica que una entrada binaria produce su correspondiente salida decimal. Las etiquetas 8, 4, 2 y 1 en las entradas representan los pesos binarios de los bits de entrada (2^3 2^2 2^1 2^0).

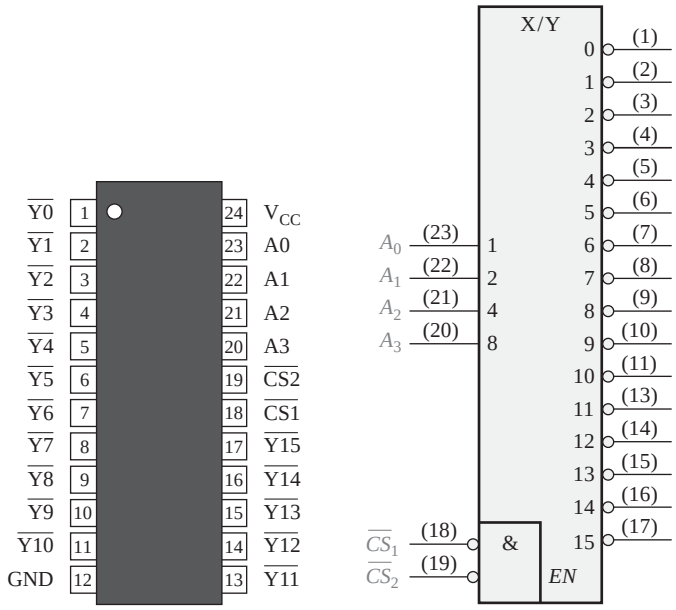
Dígito decimal	Entradas binarias A_3	A_2	A_1	A_0	Función de decodificación	Salidas 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	$\overline{A}_3\overline{A}_2\overline{A}_1\overline{A}_0$	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	0	0	0	1	$\overline{A}_3\overline{A}_2\overline{A}_1A_0$	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	0	0	1	0	$\overline{A}_3\overline{A}_2A_1\overline{A}_0$	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
3	0	0	1	1	$\overline{A}_3\overline{A}_2A_1A_0$	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1
4	0	1	0	0	$\overline{A}_3A_2\overline{A}_1\overline{A}_0$	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1
5	0	1	0	1	$\overline{A}_3A_2\overline{A}_1A_0$	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1
6	0	1	1	0	$\overline{A}_3A_2A_1\overline{A}_0$	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1
7	0	1	1	1	$\overline{A}_3A_2A_1A_0$	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1
8	1	0	0	0	$A_3\overline{A}_2\overline{A}_1\overline{A}_0$	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1
9	1	0	0	1	$A_3\overline{A}_2\overline{A}_1A_0$	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1
10	1	0	1	0	$A_3\overline{A}_2A_1\overline{A}_0$	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
11	1	0	1	1	$A_3\overline{A}_2A_1A_0$	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1
12	1	1	0	0	$A_3A_2\overline{A}_1\overline{A}_0$	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1
13	1	1	0	1	$A_3A_2\overline{A}_1A_0$	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
14	1	1	1	0	$A_3A_2A_1\overline{A}_0$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1
15	1	1	1	1	$A_3A_2A_1A_0$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

TABLA 6.4 Funciones de decodificación y tabla de verdad para un decodificador de 4-líneas a 16-líneas con salidas activas a nivel BAJO.

EL DECODIFICADOR 1 DE 16 74HC154



El 74HC154 es un buen ejemplo de un decodificador en circuito integrado. Su símbolo lógico se muestra en la Figura 6.29. En este tipo de dispositivo existe una función de activación (*enable*, *EN*), que se implementa mediante una puerta NOR utilizada como negativa-AND. En las entradas de selección del chip, $\overline{CS}_1$ y $\overline{CS}_2$, se requiere un nivel BAJO para obtener en la salida de la puerta de activación (EN, *enable*) un nivel ALTO. La salida de la puerta de activación se conecta a una entrada de *cada* puerta NAND del decodificador, por lo que debe estar a nivel ALTO para que las puertas NAND se activen. Si la puerta de activación no se activa mediante un nivel BAJO en ambas entradas, entonces las dieciséis salidas (Y) del decodificador estarán a nivel ALTO independientemente del estado de las cuatro variables de entrada A_0, A_1, A_2 y A_3. Este dispositivo puede estar disponible en otras familias CMOS o TTL. Consulte el sitio web de Texas Instruments en www.ti.com.



(a) Diagrama de pines (b) Símbolo lógico

FIGURA 6.29 Diagrama de pines y símbolo lógico para el decodificador 1 de 16 74HC154.

EJEMPLO 6.9

Una cierta aplicación requiere decodificar un número de 5 bits. Utilizar decodificadores 74HC154 para implementar el circuito lógico. El número binario se representa de la forma:

$$A_4A_3A_2A_1A_0$$

Solución

Dado que el 74HC154 puede procesar únicamente cuatro bits, habrá que usar dos decodificadores para los cinco bits. El quinto bit, A_4, está conectado a las entradas de selección del chip, $\overline{CS}_1$ y $\overline{CS}_2$, de uno de los decodificadores y $\overline{A}_4$ se conecta a las entradas de activación $\overline{CS}_1$ y $\overline{CS}_2$ del otro decodificador, como se muestra en la Figura 6.30.

Cuando el número decimal es 15 o menor, $A_4 = 0$, el decodificador de menor orden se activa y el de mayor orden se desactiva.

Cuando el número decimal es mayor que 15, $A_4 = 1$, luego $\overline{A}_4 = 0$, lo que hace que se active el decodificador de orden superior y se desactive el de orden inferior.

Problema relacionado

En la Figura 6.30, determinar la salida que se activa al introducir el código binario de entrada:

10110

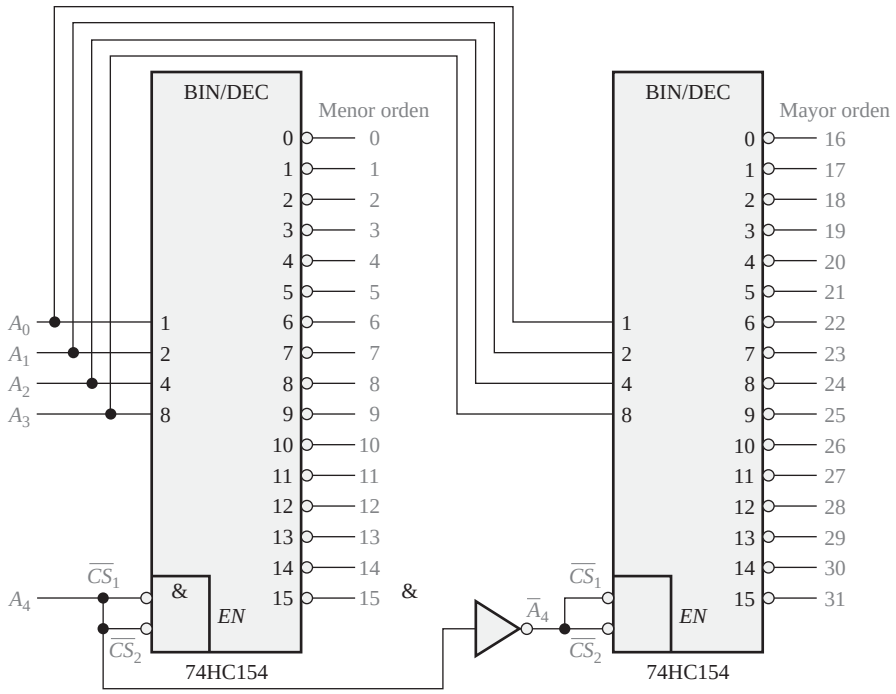


FIGURA 6.30 Decodificador de 5 bits construido con dos 74HC154.

Aplicación

Los decodificadores se utilizan en muchos tipos de aplicaciones. Un ejemplo es la selección de entradas y salidas en las computadoras, como se muestra en el diagrama general de la Figura 6.31.

Las computadoras se tienen que comunicar con una gran variedad de dispositivos externos, denominados *periféricos*, enviando y/o recibiendo datos a través de lo que se conoce como puertos de entrada/salida (E/S). Estos dispositivos externos incluyen impresoras, modems, escáneres, unidades de disco externas, teclados, monitores y otras computadoras. Como se indica en la Figura 6.31, se emplea un decodificador para seleccionar el puerto de E/S determinado por la computadora, de forma que los datos puedan ser enviados o recibidos desde algún dispositivo externo concreto.

Cada puerto de E/S tiene un número, denominado dirección, que lo identifica unívocamente. Cuando la computadora desea comunicarse con algún dispositivo en particular, envía el código de dirección apropiado del puerto de E/S al que está conectado el dispositivo en cuestión. Esta dirección binaria del puerto se decodifica, activándose la salida del decodificador apropiada que habilita el correspondiente puerto de E/S.

Como se muestra en la Figura 6.31, los datos binarios se transfieren dentro de la computadora a través de un bus de datos, que consiste en un conjunto de líneas paralelas. Por ejemplo, un bus de 8 bits consta de ocho líneas paralelas que pueden transmitir un byte de datos de una sola vez. El bus de datos está conectado a todos los puertos de E/S, pero los datos que son recibidos o transmitidos sólo pasarán a través del puerto que se encuentre activado por el decodificador de direcciones de puertos.

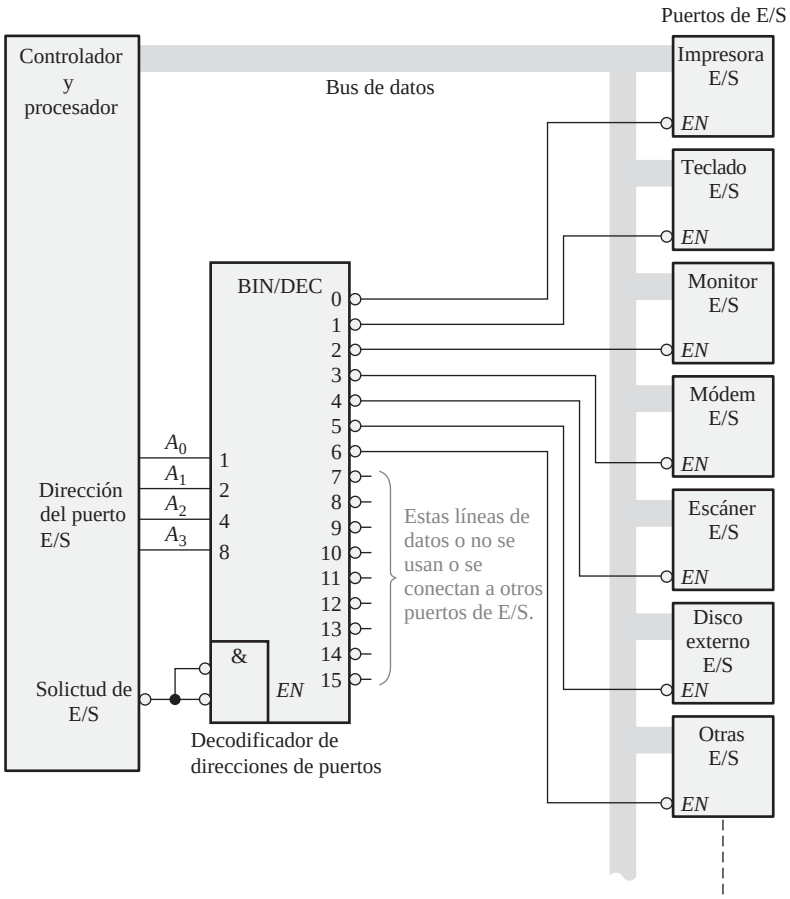


FIGURA 6.31 Un sistema simplificado de puertos de E/S con un decodificador de direcciones de puerto con sólo cuatro líneas de dirección.

El decodificador BCD a decimal

Un decodificador BCD a decimal convierte cada código BCD (código 8421) en uno de los diez posibles dígitos decimales. Frecuentemente, se le denomina *decodificador de 4-líneas a 10-líneas* o *decodificador 1 de 10*.

El método de implementación es el mismo que hemos visto anteriormente para el decodificador de 4-líneas a 16-líneas, excepto que ahora sólo se requieren diez puertas decodificadoras, dado que el código BCD sólo representa los diez dígitos decimales de 0 a 9. En la Tabla 6.5 se muestra una lista de los diez códigos BCD y sus correspondiente funciones de decodificación. Cada una de estas funciones se implementa mediante puertas NAND para proporcionar salidas activas a nivel BAJO. Si se requirieran salidas activas a nivel ALTO, se utilizarían puertas AND para la decodificación. La lógica es idéntica a la de las diez primeras puertas del decodificador de 4-líneas a 16-líneas (*véase* la Tabla 6.4).

EJEMPLO 6.10

El 74HC42 es un CI decodificador BCD-decimal. Su símbolo lógico se muestra en la Figura 6.32. Dibujar las señales de salida si se aplican las señales de entrada de la Figura 6.33(a) a las entradas del 74HC42.

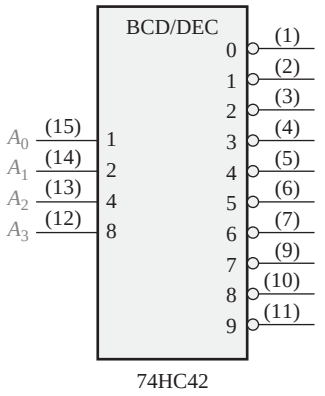


FIGURA 6.32 El decodificador BCD a decimal 74HC42.

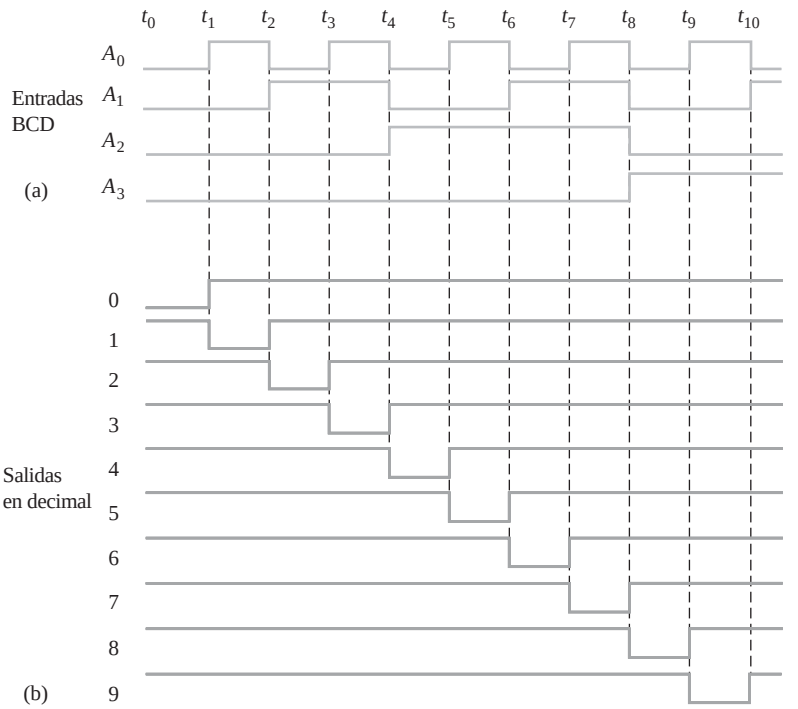


FIGURA 6.33

Solución Las formas de onda de salida se muestran en la Figura 6.33(b). Como puede ver, las entradas al decodificador constituyen una secuencia de dígitos de 0 a 9. Las formas de onda de salida del diagrama de tiempos indican dicha secuencia de valores decimales.

Problema relacionado Construir un diagrama de tiempos que muestre las señales de entrada y de salida para el caso en que la secuencia binaria de entrada origine los siguientes números decimales: 0, 2, 4, 6, 8, 1, 3, 5 y 9.

Dígito decimal	Código BCD A_3	A_2	A_1	A_0	Función de decodificación
0	0	0	0	0	$\overline{A}_3\overline{A}_2\overline{A}_1\overline{A}_0$
1	0	0	0	1	$\overline{A}_3\overline{A}_2\overline{A}_1A_0$
2	0	0	1	0	$\overline{A}_3\overline{A}_2A_1\overline{A}_0$
3	0	0	1	1	$\overline{A}_3\overline{A}_2A_1A_0$
4	0	1	0	0	$\overline{A}_3A_2\overline{A}_1\overline{A}_0$
5	0	1	0	1	$\overline{A}_3A_2\overline{A}_1A_0$
6	0	1	1	0	$\overline{A}_3A_2A_1\overline{A}_0$
7	0	1	1	1	$\overline{A}_3A_2A_1A_0$
8	1	0	0	0	$A_3\overline{A}_2\overline{A}_1\overline{A}_0$
9	1	0	0	1	$A_3\overline{A}_2\overline{A}_1A_0$

TABLA 6.5 Funciones de decodificación BCD.

El decodificador BCD a 7-segmentos

El decodificador BCD a 7-segmentos acepta el código BCD en sus entradas y proporciona salidas capaces de excitar un display de 7-segmentos para generar un dígito decimal. En la Figura 6.34 se muestra el diagrama lógico de un decodificador básico de 7-segmentos.

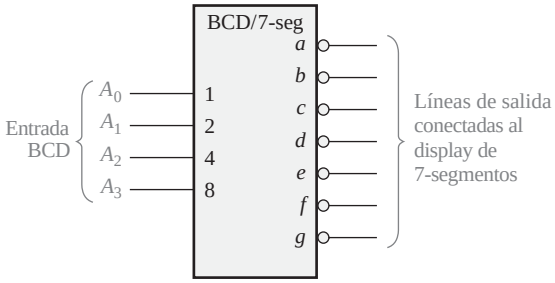
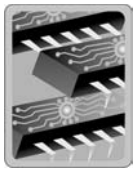


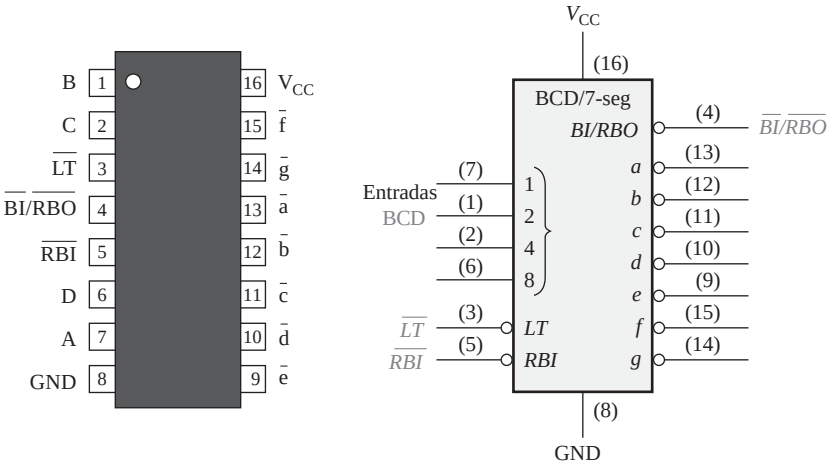
FIGURA 6.34 Símbolo lógico de un decodificador/controlador BCD a 7-segmentos con salidas activas a nivel BAJO.

EL DECODIFICADOR /CONTROLADOR BCD A 7-SEGMENTOS 74LS47



El 74LS47 es un ejemplo de circuito integrado que decodifica una entrada BCD y controla un display de 7-segmentos. Además de estas características de decodificación y control, el 74LS47 posee características adicionales, como las indicadas en el símbolo lógico de la Figura 6.35 por las funciones $\overline{LT}, \overline{RBI}, \overline{BI}$ / $\overline{RBO}$. Como indican los círculos del símbolo lógico, todas las salidas (de *a* a *g*) son activas a nivel BAJO, al igual que lo son $\overline{LT}$ (*Lamp Test*, entrada de comprobación), *RB*I (*Ripple Blanking Input*) y $\overline{BI}$ / $\overline{RBO}$.(*Blanking Input/Ripple Blanking Output*). Las salidas pueden controlar directamente un display de 7-segmentos en ánodo común. Recuerde que los displays de 7-segmentos se trataron en el Capítulo 4. Además de decodificar una entrada BCD y generar las apropiadas salidas para 7-segmentos, el 74LS47 posee las funciones de entra-

da de comprobación y de supresión de cero. Este dispositivo puede estar disponible en otras familias CMOS o TTL. Consulte el sitio web de Texas Instruments en www.ti.com.



(a) Diagrama de pines (b) Símbolo lógico

FIGURA 6.35 Diagrama de pines y símbolo lógico para el codificador/controlador BCD a 7-segmentos 74LS47.

Entrada de comprobación. Cuando se aplica un nivel BAJO a la entrada $\overline{LT}$ y la entrada $\overline{BI}$ / $\overline{RBO}$.está a nivel ALTO, se encienden los 7 segmentos del display. La entrada de comprobación se utiliza para verificar que ninguno de los segmentos está fundido.

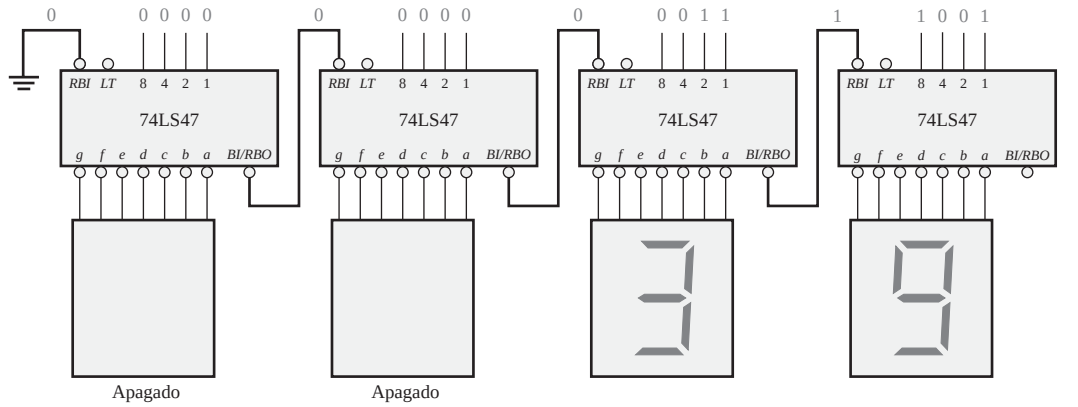
▲ *La supresión de cero da como resultado que no se muestren en un display los ceros anteriores o posteriores de un número.*

Supresión de cero. La **supresión de cero** es una característica utilizada en displays de varios dígitos para eliminar los ceros innecesarios. Por ejemplo, en un display de 6 dígitos, el número 6.4 podría mostrarse como 006.400 si no se eliminaran los ceros. La supresión de ceros al principio de un número recibe el nombre de *supresión anterior de cero*, mientras que si son los últimos los que se eliminan se denomina *supresión posterior de cero*. Tenga en cuenta que únicamente se eliminan los ceros que no son esenciales. Gracias a la supresión de cero, el número 030.080 se visualiza como 30.08 (los ceros esenciales permanecen).

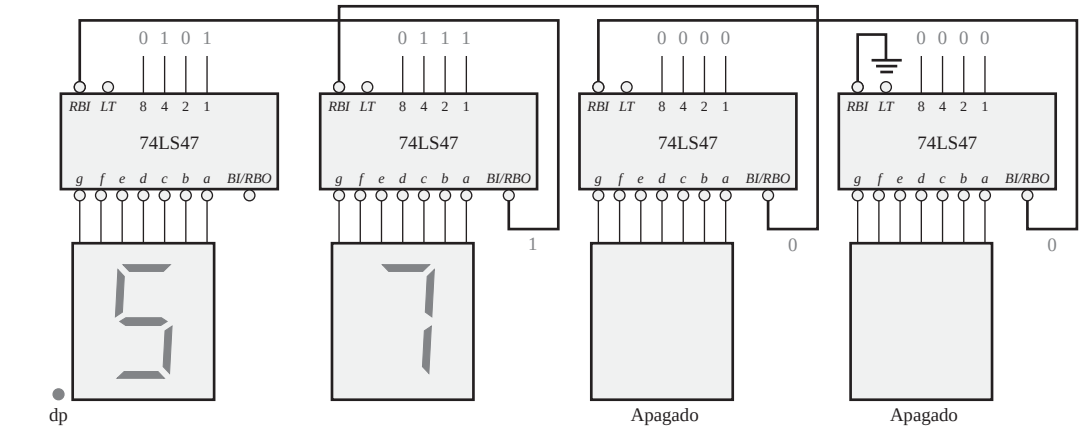
La supresión de cero en el 74LS47 se logra utilizando las funciones $\overline{RBI}$ y $\overline{BI}$ / $\overline{RBO}$. $\overline{RBI}$ es la entrada de borrado en cascada y $\overline{RBO}$ es la salida de borrado en cascada del 74LS47, las cuales se utilizan para la supresión de cero. $\overline{BI}$ es la entrada de borrado y comparte el mismo pin con $\overline{RBO}$. En otras palabras, el pin $\overline{BI}$ / $\overline{RBO}$. puede utilizarse como salida o como entrada. Cuando se emplea como $\overline{BI}$ (entrada de borrado), todas las salidas están a nivel ALTO (segmentos desactivados) cuando $\overline{BI}$ está a nivel BAJO, anulando el resto de entradas. La función $\overline{BI}$ no forma parte de las características de supresión de cero del dispositivo.

Todas las salidas del decodificador correspondientes a los segmentos se encuentran inactivas (nivel ALTO) cuando se introduce el código cero (0000) en sus entradas BCD siempre que su $\overline{RBI}$ está a nivel BAJO. Esto origina que el display no muestre nada y que la salida $\overline{RBO}$ esté a nivel BAJO.

El diagrama lógico de la Figura 6.36(a) ilustra la supresión anterior de cero para un número entero. El dígito más significativo (el de más a la izquierda) desaparece siempre que se introduzca el código del 0 en sus entradas BCD, ya que la $\overline{RBI}$ del decodificador



(a) Supresión anterior de cero



(b) Supresión posterior de cero

FIGURA 6.36 Ejemplos de supresión de cero mediante un decodificador/controlador BCD a 7-segmentos 74LS47.

de orden más alto se conecta a tierra de forma que siempre esté a nivel BAJO. La salida $\overline{RBO}$ de cada decodificador se conecta a la entrada $\overline{RBI}$ del siguiente decodificador de menor orden, de manera que se eliminan todos los ceros a la izquierda del primer dígito que no sea cero. Por ejemplo, en la parte (a) de la figura, los dos dígitos más significativos son cero y, por tanto, se eliminan. Los dos dígitos restantes, 3 y 9, se visualizan en el display.

El diagrama lógico de la Figura 6.36(b) ilustra la supresión posterior de cero para un número fraccionario. El dígito de menor orden (el de más a la derecha) se suprime siempre que se introduzca un código nulo en sus entradas BCD, ya que la entrada $\overline{RBI}$ está conectada a masa. La salida $\overline{RBO}$ de cada decodificador está conectada a la entrada $\overline{RBI}$ del decodificador siguiente de orden superior, de forma que todos los ceros a la derecha del primer dígito no nulo son eliminados. En la parte (b) de la figura, los dos dígitos menos significativos son cero, luego se eliminan. Los dos dígitos restantes, 5 y 7, se muestran en el display. Para combinar la supresión anterior de cero con la posterior en un mismo display y, además, tener posibilidad de introducir puntos decimales, necesitamos circuitos lógicos adicionales.

REVISIÓN DE LA SECCIÓN 6.5

1. Un decodificador de 3-líneas a 8-líneas se puede utilizar como decodificador octal a decimal. Cuando se introduce el número binario 101 en sus entradas, ¿qué línea de salida se activa?
2. ¿Cuántos decodificadores 1 de 16 74HC154 son necesarios para decodificar un número binario de 6 bits?
3. ¿Qué elegiríamos para controlar un display de 7-segmentos con cátodo común, un decodificador/controlador con salidas activas a nivel BAJO o con salidas activas a nivel ALTO?

6.6 CODIFICADORES

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

Al finalizar esta sección, el lector deberá ser capaz de:

■ Determinar la lógica de un codificador decimal. ■ Explicar la finalidad de la característica de prioridad en los codificadores. ■ Describir el codificador de prioridad decimal a BCD 74HC147. ■ Describir el codificador de prioridad octal a binario 74LS148. ■ Ampliar un codificador. ■ Utilizar los codificadores en aplicaciones específicas.

Codificador decimal-BCD

Este tipo de codificador tiene diez entradas, una para cada dígito decimal, y cuatro salidas que corresponden al código BCD, como se muestra en la Figura 6.37. Este es un codificador básico de 10-líneas a 4-líneas.

El código BCD (8421) se muestra en la Tabla 6.6. A partir de esta tabla podemos determinar la relación entre cada bit BCD y los dígitos decimales, con el fin de analizar la lógica. Por ejemplo, el bit más significativo del código BCD, A_3, es siempre un 1 para los dígitos decimales 8 o 9. La expresión OR para el bit A_3 en función de los dígitos decimales puede por tanto escribirse como:

$$A_3 = 8 + 9$$

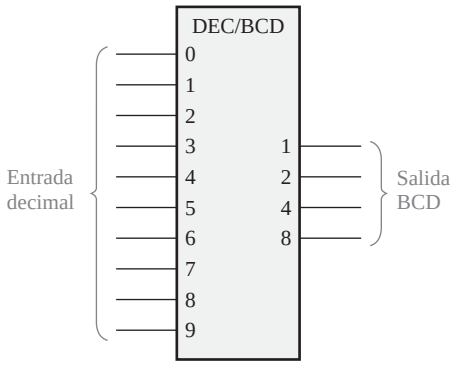


FIGURA 6.37 Símbolo lógico de un codificador decimal a BCD.

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

TABLA 6.6

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

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

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

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

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

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

Ahora vamos a implementar el circuito lógico necesario para codificar en código BCD cada dígito decimal, utilizando las expresiones lógicas que se acaban de desarrollar. Consiste simplemente en aplicar la operación OR a los dígitos decimales de entrada apropiados, para así formar cada salida BCD. La lógica del codificador que resulta de estas expresiones se muestra en la Figura 6.38.

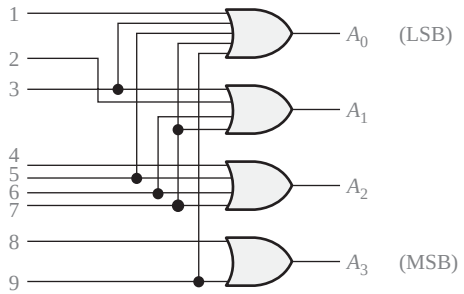


FIGURA 6.38 Diagrama lógico básico de un codificador decimal-BCD. No se necesita una entrada para el dígito 0, ya que las salidas BCD están todas a nivel BAJO cuando no hay entradas a nivel ALTO.

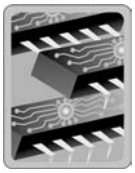
NOTAS INFORMÁTICAS

Se puede pensar en un *ensamblador* como en un codificador de software, ya que interpreta las instrucciones mnemónicas con las que se ha escrito un programa y lleva a cabo la codificación necesaria para convertir cada mnemónico en una instrucción de código máquina (series de 1s y 0s), que la computadora puede entender. Ejemplos de mnemónicos de instrucciones son ADD, MOV (*move data*, desplazamiento de datos), MUL (multiplicación), XOR, JMP (*jump*, salto) y OUT (salida a un puerto).

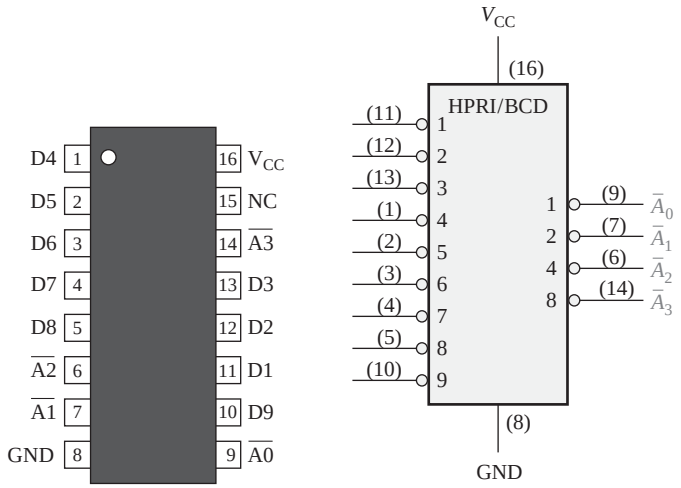
El funcionamiento básico del circuito de la Figura 6.38 es el siguiente: cuando aparece un nivel ALTO en *una* de las líneas de entrada correspondientes a los dígitos decimales, se generan los niveles apropiados en las cuatro líneas BCD de salida. Por ejemplo, si la línea de entrada 9 está a nivel ALTO (suponiendo que todas las demás entradas están a nivel BAJO), esta condición producirá un nivel ALTO en las salidas A_0 y A_3, y un nivel BAJO en A_1 y A_2, que es el código BCD (1001) del número decimal 9.

Codificador con prioridad decimal a BCD. Este tipo de codificador realiza la misma función de codificación básica que hemos visto anteriormente. Además, un ***codificador con prioridad*** ofrece una flexibilidad adicional en lo relativo a que puede utilizarse en aplicaciones que requieren detección de prioridad. La función de prioridad significa que el codificador producirá una salida BCD correspondiente al *dígito decimal de entrada de más alto orden* que se encuentre activo, e ignorará cualquier otra entrada de menor orden que esté activa. Por ejemplo, si las entradas 6 y 3 se encuentran activas, la salida BCD será 0110 (que representa al número decimal 6).

EL CODIFICADOR DECIMAL-BCD 74HC147



El 74HC147 es un codificador con prioridad con entradas activas a nivel BAJO (0) para los dígitos decimales del 1 al 9, y salidas BCD activas a nivel BAJO, como se indica en el símbolo lógico de la Figura 6.39. Una salida BCD cero se consigue cuando ninguna de las entradas está activa. La numeración de los pines del dispositivo se muestra entre paréntesis. Este dispositivo puede estar disponible en otras familias CMOS o TTL. Consulte el sitio web de Texas Instruments en www.ti.com.



(a) Diagrama de pines (b) Diagrama lógico

FIGURA 6.39 Diagrama de pines y símbolo lógico del codificador con prioridad decimal-BCD 74HC147 (HPRI, *highest value input has priority*, la entrada de mayor valor tiene prioridad).

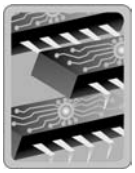
EJEMPLO 6.11

Si tenemos niveles BAJOS en los pines 1, 4 y 13 del 74HC147 que se muestra en la Figura 6.39, indicar el estado de sus cuatro salidas. Todas las demás entradas están a nivel ALTO.

Solución El pin 4 es el dígito decimal de orden más alto que tiene una entrada a nivel BAJO y representa el número decimal 7. Por tanto, los niveles de salida indican el código BCD para el decimal 7, donde $\overline{A}_0$ es el bit menos significativo (LSB) y $\overline{A}_3$ es el bit más significativo (MSB). La salida $\overline{A}_0$ es un nivel BAJO, $\overline{A}_1$ es BAJO, $\overline{A}_2$ es un nivel BAJO y $\overline{A}_3$ es un nivel ALTO.

Problema relacionado ¿Cuáles son las salidas del 74HC147 si todas sus entradas están a nivel BAJO? ¿Y si todas están a nivel ALTO?

EL CODIFICADOR 8-LÍNEAS A 3-LÍNEAS 74LS148



El 74LS148 es un codificador con prioridad que tiene ocho entradas activas a nivel BAJO y tres salidas binarias activas a nivel BAJO, como se muestra en la Figura 6.40. Este dispositivo se puede utilizar para convertir entradas octales (recuerde que los dígitos octales son del 0 al 7) en código binario de 3 bits. Para activar este dispositivo, la entrada de activación, (*Enable Input*, *EI*) tiene que estar activa a nivel BAJO. También tiene una *EO* (salida de activación, *Enable Output*) y una salida *GS* para permitir la ampliación. La salida *EO* está a nivel BAJO cuando la entrada *EI* está a nivel BAJO y ninguna de las entradas (de 0 a 7) se encuentra activada. *GS* está a nivel BAJO cuando *EI* está a nivel BAJO y cualquiera de las entradas se encuentra activada. Este dispositivo puede estar disponible en otras familias CMOS o TTL. Consulte el sitio web de Texas Instruments en www.ti.com.

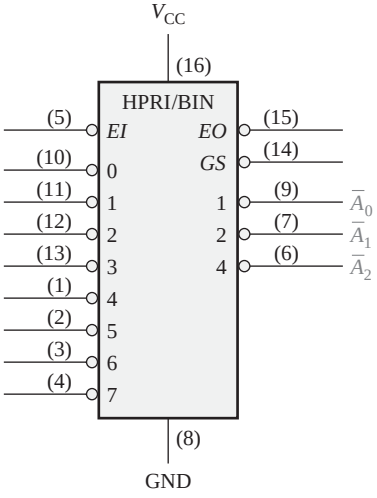


FIGURA 6.40 Símbolo lógico del codificador de 8 líneas a 3 líneas 74LS148.

El 74LS148 puede ser ampliado a un codificador de 16-líneas a 4-líneas conectando la salida *EO* del codificador de mayor orden a la entrada *EI* del codificador de menor orden, y aplicando la operación negativa-OR a las correspondientes salidas binarias, como se muestra en la Figura 6.41. La salida *EO* se utiliza como cuarto y más significa-

tivo bit. Esta configuración particular produce salidas activas a nivel ALTO para los números binarios de cuatro bits.

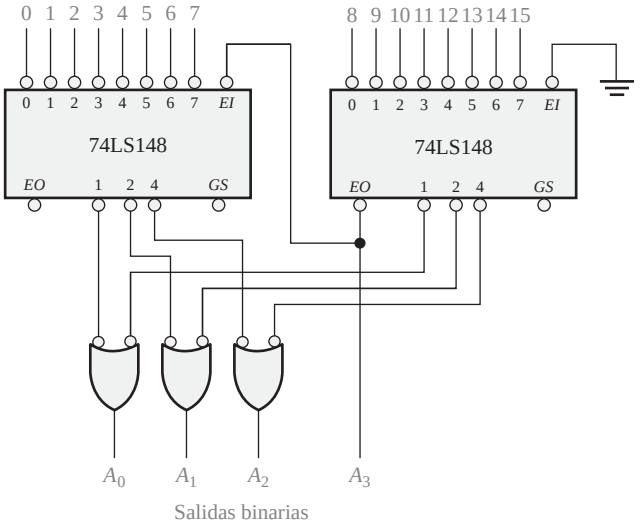


FIGURA 6.41 Un decodificador de 16 líneas a 4 líneas usando dispositivos 74LS148 y lógica externa.

Aplicación

El típico ejemplo de aplicación es un codificador de teclado. Por ejemplo, los diez dígitos decimales del teclado de una computadora tienen que codificarse para poder ser procesados por el circuito lógico. Cuando se pulsa una de las teclas, el dígito decimal se codifica a su correspondiente código BCD. La Figura 6.42 muestra la disposición de un sencillo codificador de teclado que utiliza un codificador con prioridad 74HC147. Las teclas se representan mediante diez pulsadores, conectados cada uno de ellos a una **resistencia de *pull-up*** (resistencia de conexión a la alimentación +V). Las resistencias de *pull-up* aseguran que la línea esté a nivel ALTO cuando no haya ninguna tecla pulsada. Cuando se pulsa una tecla, la línea se conecta a tierra y se aplica un nivel BAJO a la correspondiente entrada del codificador. La tecla cero no está conectada, ya que la salida BCD es cero cuando ninguna de las otras teclas está pulsada.

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

REVISIÓN DE LA SECCIÓN 6.6

1. Supongamos que, a las entradas 2 y 9 del circuito de la Figura 6.38 se les aplica un nivel ALTO.
   (a) ¿Cuáles son los estados de las líneas de salida?
   (b) ¿Representa esto un código BCD válido?
   (c) ¿Cuál es la restricción de la lógica del codificador de la Figura 6.38?
2. **(a)** ¿Cuál es la salida $\overline{A}_3\overline{A}_2\overline{A}_1\overline{A}_0$ cuando se aplican niveles BAJOS de tensión a los pines 1 y 5 del 74HC147 de la Figura 6.39?
   (b) ¿Qué representa esta salida?

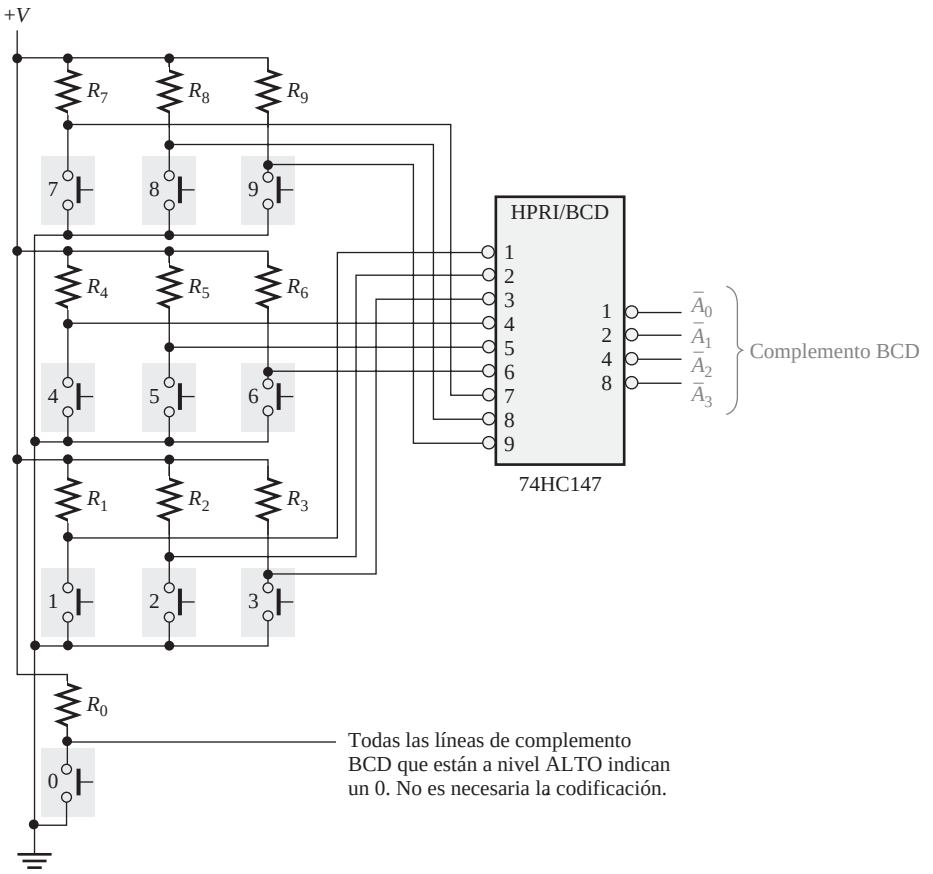


FIGURA 6.42 Codificador de un teclado simplificado.

6.7 CONVERTIDORES DE CÓDIGO

En esta sección, vamos a examinar algunos métodos que utilizan circuitos lógicos combinacionales para pasar de un código a otro.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Explicar el proceso de conversión de BCD a binario. ■ Utilizar puertas OR-exclusiva en la conversión entre código binario y código Gray.

Conversión BCD-binario

Uno de los métodos de conversión de código BCD a binario utiliza circuitos sumadores. El proceso básico de conversión consiste en lo siguiente:

1. El valor, o peso, de cada uno de los bits de un número BCD se representa por un número binario.
2. Se suman todas las representaciones binarias de los pesos de los bits del número BCD que son 1.
3. El resultado de la suma es el equivalente binario del número BCD.

Una manera más concisa de expresar esta operación es:

Para obtener el número binario completo hay que sumar los números binarios que representan los pesos de los bits del número BCD.

Examinemos un código BCD de 8 bits (uno que representa un número decimal de 2 dígitos) para comprender la relación entre el código binario y el BCD. Por ejemplo, ya sabemos que el número decimal 87 puede expresarse en BCD como sigue:

$$\underbrace{1000}_{8}\ \underbrace{0111}_{7}$$

El grupo de 4 bits más a la izquierda representa 80 y el grupo de 4 bits más a la derecha representa 7. Es decir, el grupo más a la izquierda tiene un peso de 10 y el grupo más a la derecha tiene un peso de 1. Dentro de cada grupo, el peso binario de cada bit es el siguiente:

	Dígito de las decenas				**Dígito de las unidades**			
Peso	80	40	20	10	8	4	2	1
Designación de bit:	B_3	B_2	B_1	B_0	A_3	A_2	A_1	A_0

El equivalente binario de cada bit BCD es un número binario que representa el peso de cada bit dentro del número BCD completo. Esta representación se muestra en la Tabla 6.7.

Bit BCD	**Peso BCD**	**(MSB) Representación binaria (LSB)**						
		64	**32**	**16**	**8**	**4**	**2**	**1**
A_0	1	0	0	0	0	0	0	1
A_1	2	0	0	0	0	0	1	0
A_2	4	0	0	0	0	1	0	0
A_3	8	0	0	0	1	0	0	0
B_0	10	0	0	0	1	0	1	0
B_1	20	0	0	1	0	1	0	0
B_2	40	0	1	0	1	0	0	0
B_3	80	1	0	1	0	0	0	0

TABLA 6.7 Representación binaria de los pesos de los bits BCD.

Si las representaciones binarias de los pesos de todos los 1s del número BCD se suman, el resultado es el número binario que corresponde al número BCD. El Ejemplo 6.12 de la página siguiente ilustra esto.

Teniendo este procedimiento básico en mente, vamos a determinar cómo podemos implementar este proceso mediante circuitos lógicos. Una vez que se determina la representación binaria de cada 1 del número BCD, se pueden emplear circuitos sumadores para sumar los 1s de cada columna de la representación binaria. Los 1s aparecen en una determinada columna, sólo cuando los correspondientes bits BCD valen 1. Por tanto, la aparición de un 1 BCD puede utilizarse para generar el 1 binario apropiado en la columna correspondiente de la estructura del sumador. Para manejar un número decimal de dos cifras (dos potencias de diez), en código BCD, necesitamos ocho líneas BCD de entrada y siete líneas binarias de salida. Se necesitan siete bits binarios para representar números de 0 a 99.

EJEMPLO 6.12

Convertir a binario los números BCD 00100111 (27 decimal) y 10011000 (98 decimal).

Solución Escribir la representación binaria de los pesos de todos los 1s que aparecen en los números y, a continuación, sumarlos todos.

80	**40**	**20**	**10**	**8**	**4**	**2**	**1**		
0	0	1	0	0	1	1	1		
								0000001	1
								0000010	2
								0000100	4
								+0010100	20
								0011011	Número binario para el decimal 27

80	**40**	**20**	**10**	**8**	**4**	**2**	**1**		
1	0	0	1	1	0	0	0		
								0001000	8
								0001010	10
								+1010000	80
								1100010	Número binario para el decimal 98

Problema relacionado Explicar el proceso de conversión del número BCD 01000001 a binario.

Conversión binario-Gray y Gray-binario

El proceso básico de conversión de código Gray a binario se ha tratado en el Capítulo 2. Ahora vamos a ver cómo se pueden utilizar puertas OR-exclusiva en estas conversiones. Los dispositivos lógicos programables (PLD) también se pueden utilizar para realizar estas conversiones de código. La Figura 6.43 muestra un convertidor de 4 bits binarios a código Gray, y la Figura 6.44 ilustra un convertidor de 4 bits Gray a binario.

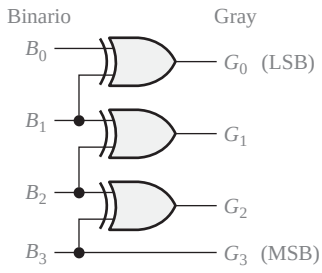


FIGURA 6.43 Lógica de conversión de 4 bits binarios a Gray.

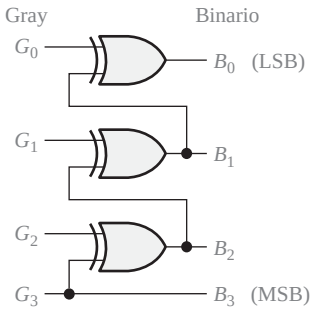


FIGURA 6.44 Lógica de conversión de 4 bits Gray a binario.

EJEMPLO 6.13

(a) Convertir el número binario 0101 a código Gray utilizando puertas OR-exclusiva.
(b) Convertir el código Gray 1011 a binario utilizando puertas OR-exclusiva.

Solución **(a)** 0101_2 es 0111 en código Gray. *Véase* la Figura 6.45(a).
(b) 1011 en código Gray es 1101_2. *Véase* la Figura 6.45(b).

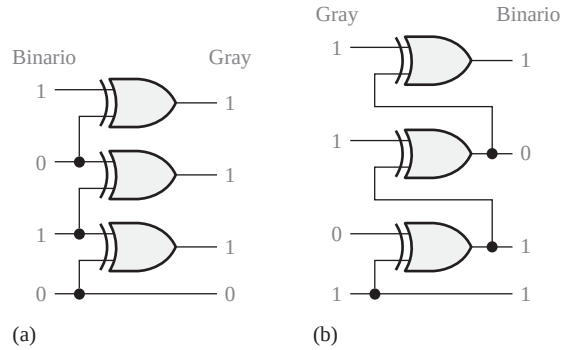


FIGURA 6.45

Problema relacionado ¿Cuántas puertas OR-exclusiva son necesarias para convertir 8 bits binarios a código Gray?

REVISIÓN DE LA SECCIÓN 6.7

1. Convertir el número BCD 10000101 a binario.
2. Dibujar el diagrama lógico para convertir a código Gray un número binario de 8 bits.

6.8 MULTIPLEXORES (SELECTORES DE DATOS)

Un ***multiplexor*** (MUX) es un dispositivo que permite dirigir la información digital procedente de diversas fuentes a una única línea para ser transmitida a través de dicha línea a un destino común. El

multiplexor básico posee varias líneas de entrada de datos y una única línea de salida. También posee entradas de selección de datos, que permiten conmutar los datos digitales provenientes de cualquier entrada hacia la línea de salida. A los multiplexores también se les conoce como selectores de datos.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Explicar el funcionamiento básico de un multiplexor. ■ Describir los multiplexores 74LS151 y 74HC157. ■ Ampliar un multiplexor para poder manejar mayor cantidad de entradas de datos. ■ Utilizar un multiplexor como generador de funciones lógicas.

▲ *En un multiplexor, los datos procedentes de varias líneas pasan a una sola línea.*

El símbolo lógico de un multiplexor (MUX) de cuatro entradas se muestra en la Figura 6.46. Observe que dispone de dos líneas de selección de datos, dado que con dos bits se puede seleccionar cualquiera de las cuatro líneas de entrada de datos.

En la Figura 6.46, un código binario de dos bits en las entradas de selección de datos (S) va a permitir que los datos de la entrada seleccionada pasen a la salida de datos. Si aplicamos un 0 binario ($S_1 = 0$ y $S_0 = 0$) a las líneas de selección de datos, los datos de la entrada D_0 aparecerán en la línea de datos de salida. Si aplicamos un 1 binario ($S_1 = 0$ y $S_0 = 1$), los datos de la entrada D_1 aparecerán en la salida de datos. Si se aplica un 2 binario ($S_1 = 1$ y $S_0 = 0$), obtendremos en la salida los datos de D_2. Si aplicamos un 3 binario ($S_1 = 1$ y $S_0 = 1$), los datos de D_3 serán conmutados a la línea de salida. El resumen del funcionamiento se puede ver en la Tabla 6.8.

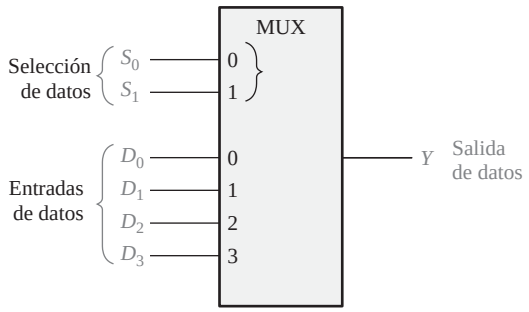


FIGURA 6.46 Símbolo lógico de un selector/multiplexor de datos de una salida y cuatro entradas.

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

TABLA 6.8 Selección de datos de un multiplexor de 1 salida y 4 entradas.

Ahora veamos la circuitería lógica necesaria para implementar esta operación de multiplexación. La salida de datos es igual al estado de la entrada de datos *seleccionada*. Por tanto, podemos deducir una expresión lógica para la salida en función de las entradas de datos y de las entradas de selección.

La salida de datos es igual a D_0 sólo si $S_1 = 0$ y $S_0 = 0 : Y = D_0\overline{S}_1\overline{S}_0$.

La salida de datos es igual a D_1 sólo si $S_1 = 0$ y $S_0 = 1 : \boldsymbol{Y} = D_1\overline{S}_1S_0$.

La salida de datos es igual a D_2 sólo si $S_1 = 1$ y $\boldsymbol{S}_0 = 0 : Y = D_2S_1\overline{S}_0$.

La salida de datos es igual a D_3 sólo si $S_1 = 1$ y $S_0 = 1 : Y = D_3S_1S_0$.

Si se aplica la operación OR a estos términos, la expresión total para la salida de datos es:

$$Y = D_0\overline{S}_1\overline{S}_0 + D_1\overline{S}_1S_0 + D_2S_1\overline{S}_0 + D_3S_1S_0$$

La implementación de esta ecuación requiere cuatro puertas AND de tres entradas, una puerta OR de cuatro entradas y dos inversores para generar los complementos de S_1 y S_0, como se muestra en la Figura 6.47. Dado que los datos pueden ser seleccionados desde cualquier línea de entrada, se conoce también a este circuito con el nombre de **selector de datos**.

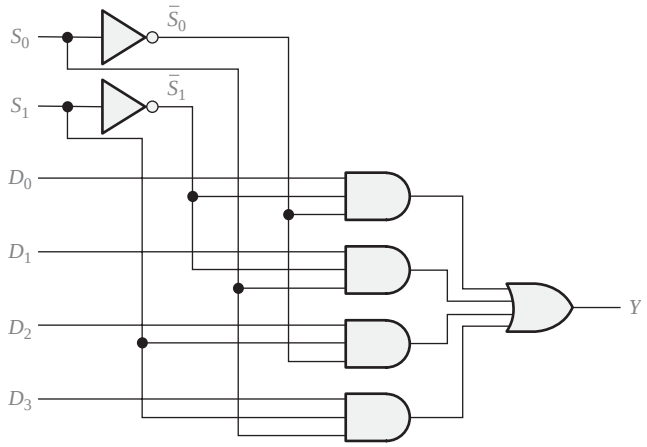


FIGURA 6.47 Diagrama lógico de un multiplexor de cuatro entradas.

NOTAS INFORMÁTICAS

Un *bus* es una ruta interna por la que se envían señales eléctricas desde una parte a otra de una computadora. En las redes de computadoras, un *bus compartido* es aquél que está conectado a todos los microprocesadores del sistema, con el fin de intercambiar datos. Un bus compartido puede contener dispositivos de memoria y de entrada/salida a los que pueden acceder todos los microprocesadores del sistema. El acceso al bus compartido se controla mediante un *árbitro de bus* (una especie de multiplexor), el cual hace que sólo un procesador utilice cada vez el bus compartido del sistema.

EJEMPLO 6.14

Se aplican las formas de onda de la Figura 6.47(a) a la entrada de datos y a la entrada de selección del multiplexor de la Figura 6.46. Determinar la señal de salida en relación a las entradas.

Solución El estado binario de las entradas de selección de datos durante cada intervalo determina cuál es la entrada de datos seleccionada. Observe que las entradas

selección de datos siguen la secuencia binaria repetitiva 00, 01, 10, 11, 00, 01, 10, 11, etc. La forma de onda de salida resultante se muestra en la Figura 6.48(b).

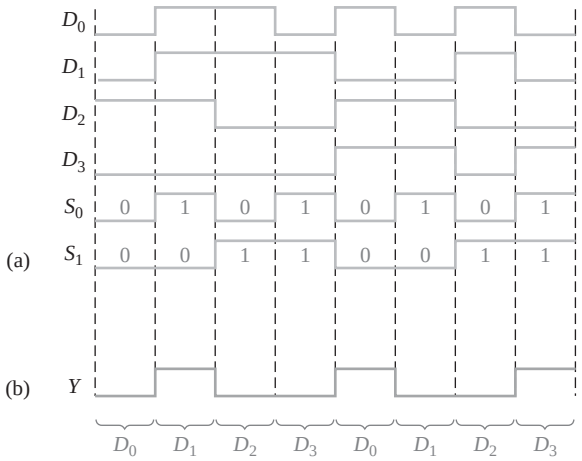
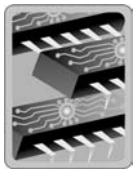


FIGURA 6.48

Problema relacionado Construir un diagrama de tiempos que muestre todas las entradas y la salida, si las formas de onda S_0 y S_1 de la Figura 6.48 se intercambian.

EL CUÁDRUPLE MULTIPLEXOR/SELECTOR DE DATOS DE 2 ENTRADAS 74HC157



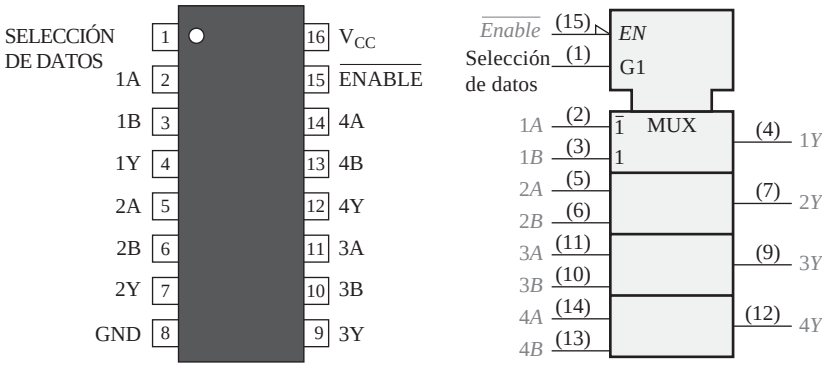
El 74HC157, al igual que su versión LS, está formado por cuatro multiplexores de dos entradas. Cada uno de los cuatro multiplexores comparten una misma línea de selección de datos y una de *habilitación* (*enable*). Ya que sólo existen dos entradas de datos que puedan ser seleccionadas en cada multiplexor, es suficiente con tener una única entrada de selección.

Un nivel BAJO en la entrada de *habilitación* ($\overline{Enable}$) permite al dato de entrada seleccionado pasar a la salida. Un nivel ALTO en la entrada $\overline{Enable}$ evita que los datos pasen a la salida, es decir, inhabilita los multiplexores. Este dispositivo puede estar disponible en otras familias CMOS o TTL. Consulte el sitio web de Texas Instruments en www.ti.com.

Símbolos lógicos ANSI/IEEE. En la Figura 6.49(a) se muestra el diagrama de pines del 74HC157 y su símbolo lógico ANSI/IEEE en la Figura 6.49(b). Observe que los cuatro multiplexores se representan mediante divisiones del bloque y que las entradas comunes a los cuatro multiplexores se indican como entradas al bloque recortado de la parte superior, que recibe el nombre de *bloque común de control*. Todas las etiquetas dentro del bloque superior del MUX se aplican a los bloques que haya por debajo.

Observe las etiquetas 1 y $\overline{1}$ de los bloques del MUX y la etiqueta G1 en el bloque común de control. Estas etiquetas son un ejemplo del sistema de **notación de dependencia** especificado en el estándar ANSI/IEEE 91-1984. En este caso, G1 indica una relación AND entre la entrada de selección de datos y las entradas de datos designadas por

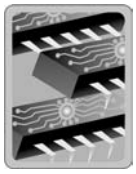
1 ó $\overline{1}$. El $\overline{1}$ indica que la relación AND se aplica al complemento de la entrada G1. En otras palabras, cuando la entrada de selección está a nivel ALTO, se seleccionan las entradas *B* de los multiplexores y, cuando la entrada de selección está a nivel BAJO, se seleccionan las entradas *A*. Para indicar dependencia AND siempre se usa una "G". Otros aspectos de la notación de dependencia serán tratados a lo largo del libro.



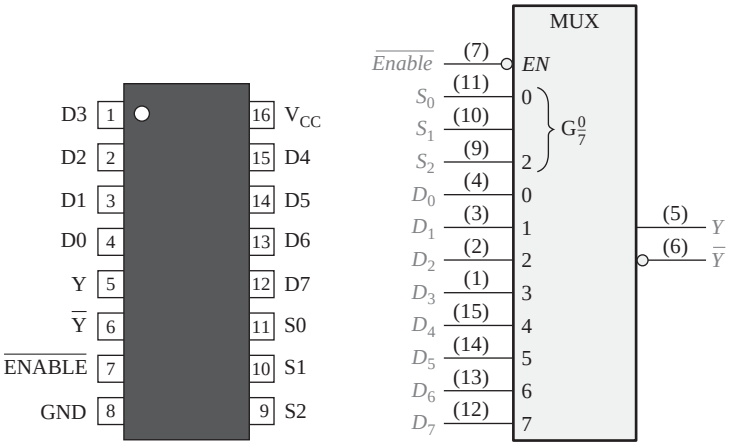
(a) Diagrama de pines (b) Símbolo lógico

FIGURA 6.49 Diagrama de pines y símbolo lógico para el cuádruple selector de datos/multiplexor de dos entradas 74HC157.

EL MULTIPLEXOR/SELECTOR DE DATOS DE 8 ENTRADAS 74LS151



El 74LS151 tiene ocho entradas de datos (D_0 –D_7) y, por tanto, tres líneas de entrada de dirección o de selección de datos (S_0-S_2). Se necesitan tres bits para seleccionar cualquiera de las ocho entradas de datos ($2^3 = 8$). Un nivel BAJO en la entrada de habilitación ($\overline{Enable}$) permite que los datos de entrada seleccionados pasen a la salida. Observe que se encuentran disponibles tanto la salida de datos como su complemento. En la Figura 6.50(a) se muestra el diagrama de pines y en la parte (b) el símbolo lógico ANSI/IEEE.



(a) Diagrama de pines (b) Símbolo lógico

FIGURA 6.50 Diagrama de pines y símbolo lógico para el multiplexor/selector de datos de 8 entradas 74LS151.

En este caso no hay necesidad de tener un bloque de control común en el símbolo lógico, ya que sólo hay que controlar un único multiplexor, y no cuatro como en el 74HC157. La etiqueta $G\frac{0}{7}$ dentro del símbolo lógico indica la relación AND entre las entradas de selección de datos y cada una de las entradas de datos, de la 0 a la 7. Este dispositivo puede estar disponible en otras familias CMOS o TTL. Consulte el sitio web de Texas Instruments en www.ti.com.

EJEMPLO 6.15

Utilizar multiplexores 74LS151 y cualquier otra lógica necesaria para multiplexar 16 líneas de datos en una única línea de salida de datos.

Solución

En la Figura 6.51 se muestra una implementación de este sistema. Se necesitan cuatro bits para seleccionar cualquiera de las 16 líneas de entrada de datos ($2^4 = 16$). En esta aplicación, la entrada de habilitación ($\overline{Enable}$) se utiliza como el bit más significativo de selección de datos. Cuando el MSB del código de selección de datos está a nivel BAJO, se habilita el 74LS151 de la izquierda y se selecciona una de las entradas de datos (D_0 a la D_7) mediante los otros tres bits de selección de datos. Cuando el MSB de selección de datos está a nivel ALTO, se habilita el 74LS151 de la derecha y se selecciona una de las entradas de datos (D_8 a la D_{15}). Los datos de entrada seleccionados pasan luego a través de la puerta negativa-OR y van a dar a la única línea de salida.

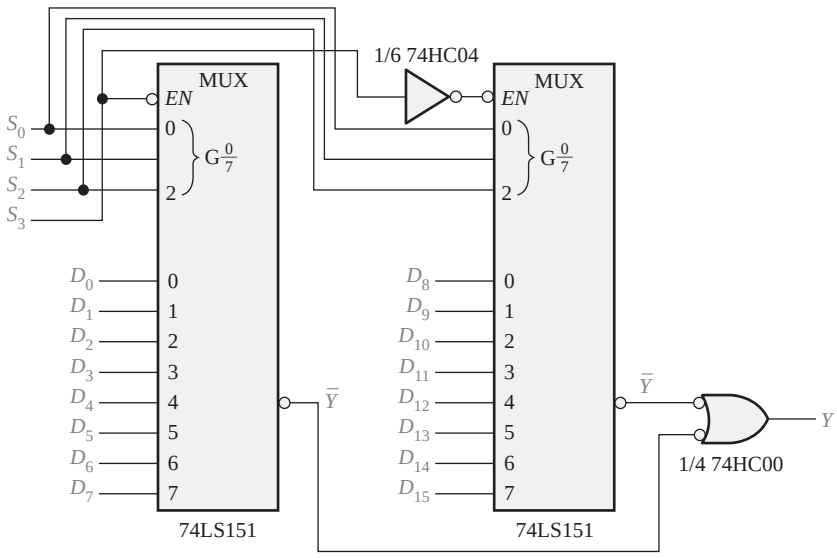


FIGURA 6.51 Multiplexor de 16 entradas.

Problema relacionado

Determinar los códigos necesarios en las entradas de selección de datos para seleccionar cada una de las siguientes entradas de datos: D_0, D_4, D_8 y D_{13}.

Aplicación

Display multiplexor de 7-segmentos. La Figura 6.52 muestra un método simplificado de multiplexación de números BCD para un display de 7-segmentos. En este ejemplo, se visualizan en el display de 7-segmentos números de dos dígitos, mediante el uso de un único decodificador BCD a 7-segmentos. Este método básico de multiplexación puede ampliarse para visualizar números con cualquier cantidad de dígitos.

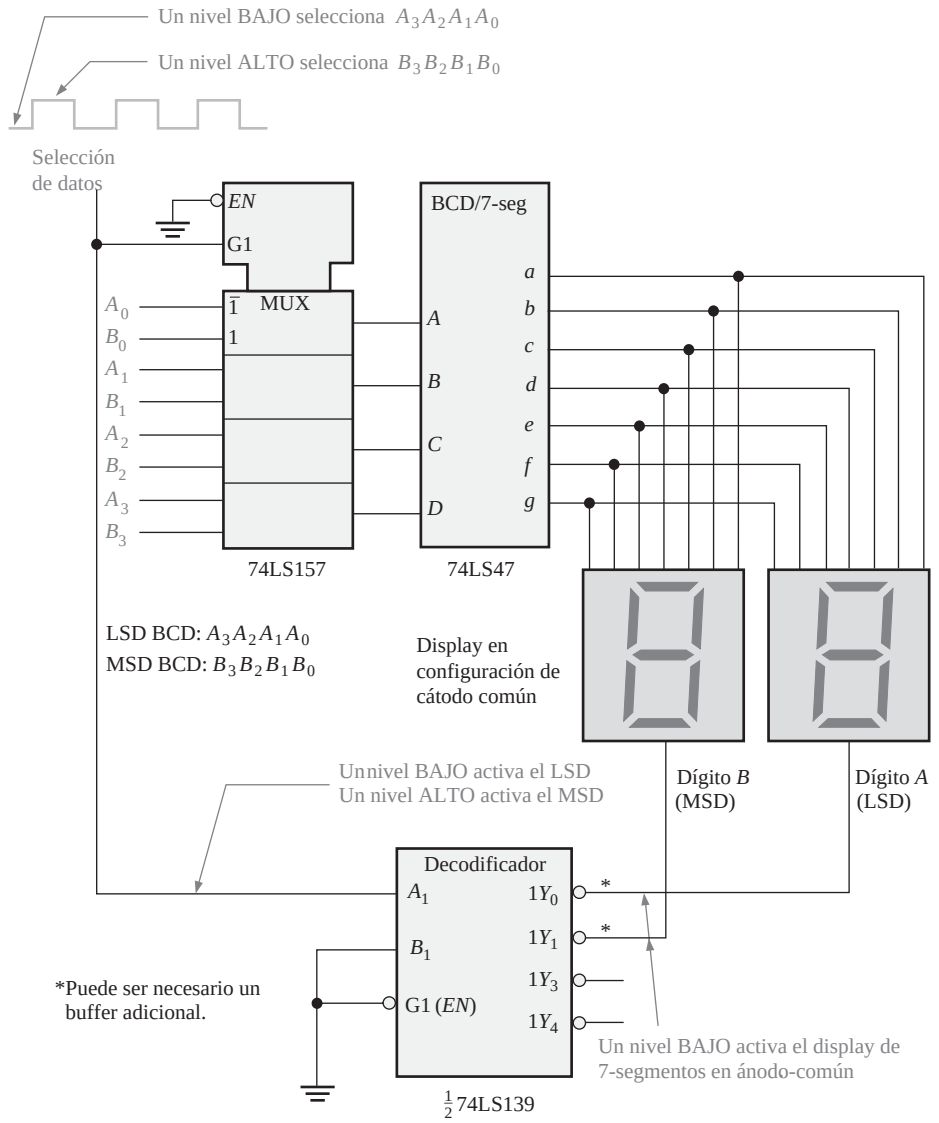


FIGURA 6.52 Lógica de multiplexación simplificada de un display de 7-segmentos.

Su funcionamiento básico es el siguiente. Se aplican dos dígitos BCD ($A_3A_2A_1A_0$ y $B_3B_2B_1B_0$) a las entradas de un multiplexor. Se aplica una señal cuadrada a la línea de selección de datos de forma que, cuando está a nivel BAJO, los bits de A ($A_3A_2A_1A_0$) pasan a las entradas del decodificador BCD a 7-segmentos74LS47. El

nivel BAJO en la entrada de selección de datos genera un nivel BAJO en la entrada A_1 del decodificador de 2-líneas a 4-líneas74LS139, activando su salida 0 y habilitando el display del dígito *A*, al conectar su terminal común a masa. El dígito *A* se encuentra ahora encendido, mientras que el *B* está apagado.

Cuando la línea de selección de datos pasa a nivel ALTO, los bits de *B* ($B_3B_2B_1B_0$) pasan a las entradas del decodificador BCD a 7-segmentos. Ahora se activa la salida 1 del decodificador 74LS139, encendiendo el display del dígito *B*, que pasa a visualizarse, mientras que el *A* se encuentra apagado. El ciclo se repite a la frecuencia de la señal cuadrada que se aplica a la entrada de selección de datos. Esta frecuencia tiene que ser lo suficientemente alta (unos 30 Hz) para evitar el parpadeo en los displays cuando se multiplexa la presentación de los dígitos.

Generador de funciones lógicas. Una aplicación muy útil de los multiplexores/selectores de datos consiste en la generación de funciones lógicas combinacionales en forma de suma de productos. Cuando se emplea de esta manera, este dispositivo puede reemplazar puertas lógicas discretas, puede reducir significativamente el número de circuitos integrados y permite que los cambios en el diseño sean mucho más sencillos.

Con el fin de ilustrar esto, se ha utilizado un multiplexor/selector de datos de 8 entradas 74LS151, para implementar cualquier función lógica de 3 variables, conectando las variables a las entradas de selección de datos y asignando a cada entrada de datos el nivel lógico requerido por la tabla de verdad para dicha función. Por ejemplo, si la función es 1 cuando la combinación de variables es $\overline{A}_2\overline{A}_1\overline{A}_0$, la entrada 2 (seleccionada por 010) se conecta a un nivel ALTO. Este nivel ALTO pasa a la salida cuando esta combinación particular de variables ocurre en las líneas de selección de datos. Un ejemplo nos servirá para clarificar esta aplicación.

EJEMPLO 6.16

Implementar la función lógica especificada en la Tabla 6.9, utilizando un multiplexor/selector de datos de 8 entradas 74LS151. Comparar este método con una implementación discreta con puertas lógicas.

Entradas			Salida
A_2	A_1	A_0	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

TABLA 6.9

Solución

Observe en la tabla de verdad que *Y* vale 1 para las siguientes combinaciones de variables de entrada: 001, 011, 101 y 110. Para el resto de las combinaciones, *Y* vale 0. Para poder implementar esta función mediante el selector de datos, la entrada de datos seleccionada por cada una de las combinaciones mencionadas anteriormente tiene que conectarse a un nivel ALTO (5V). El resto de las entradas de datos debe conectarse a un nivel BAJO (tierra), como se muestra en la Figura 6.53.

La implementación de esta función mediante puertas lógicas requeriría cuatro puertas AND de 3 entradas, una puerta OR de 4 entradas y tres inversores, a menos que la expresión pudiera simplificarse.

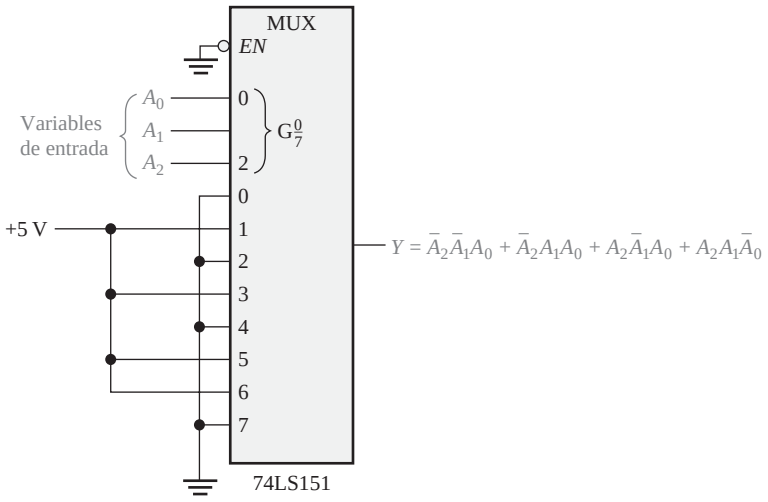


FIGURA 6.53 Multiplexor/selector de datos conectado como generador de funciones lógicas de 3 variables.

Problema relacionado Utilizar un 74LS151 para implementar la siguiente expresión:

$$Y = \bar{A}_2\bar{A}_1\bar{A}_0 + A_2\bar{A}_1\bar{A}_0 + \bar{A}_2A_1\bar{A}_0$$

El Ejemplo 6.16 ilustra cómo se puede utilizar el selector de datos de 8 entradas como generador de funciones lógicas de tres variables. En realidad, este dispositivo puede usarse también como generador de funciones lógicas de 4 variables utilizando uno de los bits (A_0) junto con las entradas de datos.

Una tabla de verdad de 4 variables da lugar a dieciséis combinaciones de las variables de entrada. Cuando se emplea un selector de datos de 8 bits, cada entrada se selecciona dos veces: la primera vez cuando A_0 es 0 y la segunda cuando A_0 es 1. Teniendo esto en cuenta. podemos aplicar las siguientes reglas (Y es la salida y A_0 es el bit menos significativo):

1. Si $Y = 0$ las dos veces que una entrada de datos dada se selecciona mediante una determinada combinación de las variables de entrada $A_3\,A_2\,A_1$, dicha entrada de datos se conecta a tierra (0).
2. Si $Y = 1$ las dos veces que una entrada de datos dada se selecciona mediante una determinada combinación de las variables de entrada $A_3\,A_2\,A_1$, dicha entrada de datos se conecta a $+V$ (1).
3. Si Y es diferente las dos veces que una entrada de datos dada se selecciona mediante una determinada combinación de las variables de entrada $A_3\,A_2\,A_1$, y si $Y = A_0$, la entrada de datos se conecta a A_0.
4. Si Y es diferente las dos veces que una entrada de datos dada se selecciona mediante una determinada combinación de las variables de entrada $A_3\,A_2\,A_1$, y si $Y = \bar{A}_0$ la entrada de datos se conecta a $\bar{A}_0$.

EJEMPLO 6.17

Implementar la función lógica de la Tabla 6.10 utilizando el multiplexor/selector de datos de 8 entradas 74LS151. Comparar este método con una implementación realizada con puertas lógicas discretas.

Dígito decimal	Entradas A_3	A_2	A_1	A_0	Salida Y
0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	1
3	0	0	1	1	0
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	1
7	0	1	1	1	1
8	1	0	0	0	1
9	1	0	0	1	0
10	1	0	1	0	1
11	1	0	1	1	0
12	1	1	0	0	1
13	1	1	0	1	1
14	1	1	1	0	0
15	1	1	1	1	1

TABLA 6.10

Solución Las entradas de selección de datos son $A_3A_2A_1$. En la primera fila de la tabla, $A_3A_2A_1 = 000$ e $Y = A_0$. En la segunda fila, de nuevo $A_3A_2A_1$ es 000, $Y = A_0$. Por

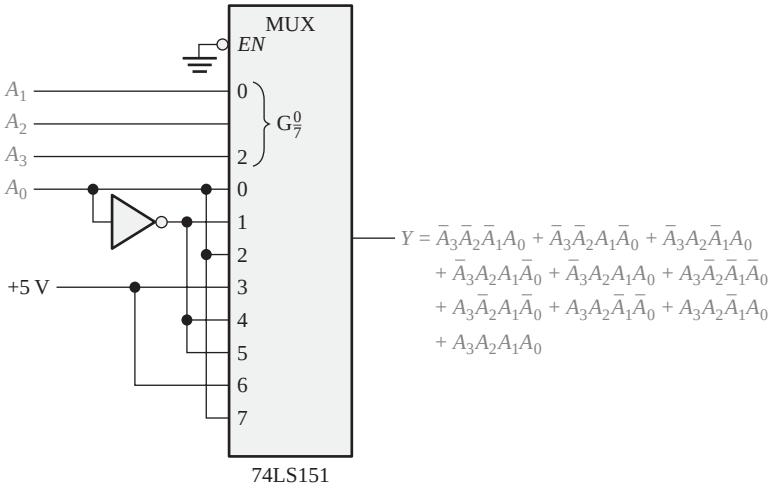


FIGURA 6.54 Multiplexor/selector de datos conectado como generador de funciones lógicas de 4 variables.

tanto, A_0 se conecta a la entrada 0. En la tercera fila de la tabla $A_3A_2A_1 = 001$, $Y = \overline{A}_0$. También, en la cuarta columna, $A_3A_2A_1$ es de nuevo 001 e $Y = \overline{A}_0$. Por tanto, A_0 se invierte y se conecta a la entrada 1. Este análisis continúa hasta que cada entrada se conecta apropiadamente de acuerdo con las reglas especificadas. La implementación se muestra en la Figura 6.54.

Si se implementara con puertas lógicas, la función requeriría al menos diez puertas AND de 4 entradas, una puerta OR de 10 entradas y cuatro inversores, aunque una posible simplificación reduciría este requisito.

Problema relacionado. Si, en la Tabla 6.10, $Y = 0$ cuando las entradas son todas cero y, alternativamente, 1 y 0 para las restantes filas de la tabla, implementar la función lógica resultante utilizando un 74LS151.

REVISIÓN DE LA SECCIÓN 6.8

1. En la Figura 6.47, $D_0 = 1$, $D_1 = 0$, $D_2 = 1$, $D_3 = 0$, $S_0 = 1$ y $S_1 = 0$. ¿Cuál es la salida?
2. Identificar cada dispositivo:
   (a) 74LS157 **(b)** 74LS151
3. En las entradas de datos de un 74LS151 se aplican alternativamente niveles BAJOS y ALTOS, comenzando por $D_0 = 0$. Las líneas de selección de datos se secuencian mediante un contador binario (000, 001, 010, etc.) a una frecuencia de 1 kHz. La entrada de habilitación está a nivel BAJO. Describir la forma de onda de salida.
4. Describir brevemente el propósito de cada uno de los siguientes dispositivos de la Figura 6.52.
   (a) 74LS157 **(b)** 74LS47 **(c)** 74LS139

6.9 DEMULTIPLEXORES

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

Al finalizar esta sección, el lector deberá ser capaz de:

■ Explicar el funcionamiento básico de un demultiplexor. ■ Describir cómo el decodificador de 4-líneas a 16-líneas 74HC154 puede utilizarse como demultiplexor. ■ Desarrollar el diagrama de tiempos de un demultiplexor con un número determinado de entradas de datos y de selección de datos.

▲ *En un demultiplexor, los datos pasan de una línea a varias líneas.*

La Figura 6.55 muestra un circuito demultiplexor (DEMUX) de 1-línea a 4-líneas. La línea de entrada de datos está conectada a todas las puertas AND. Las dos líneas de selección de datos activan únicamente una puerta cada vez y los datos que aparecen en la línea de entrada de datos pasarán a través de la puerta seleccionada hasta la línea de salida de datos asociada.

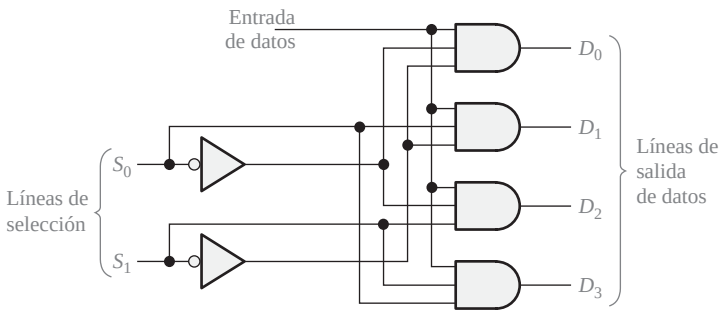


FIGURA 6.55 Demultiplexor de 1-línea a 4-líneas

EJEMPLO 6.18

En la Figura 6.56 se muestra una forma de onda de entrada de datos serie y las entradas de selección de datos (S_0 y S_1). Determinar las formas de onda de datos de salida que obtendríamos en las salidas D_0 hasta la D_3 para el demultiplexor de la Figura 6.55.

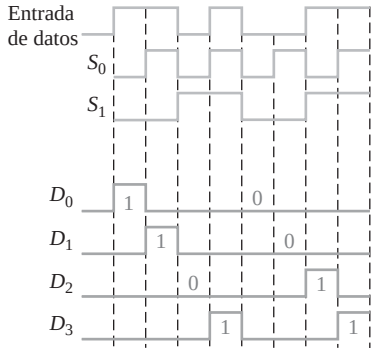
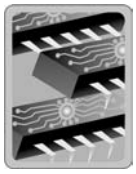


FIGURA 6.56

Solución Las líneas de selección de datos reciben una secuencia binaria que hace que cada bit de entrada sucesivo sea redirigido hacia D_0, D_1, D_2 y D_3 secuencialmente, como se puede ver en las formas de onda de salida de la Figura 6.56.

Problema relacionado Desarrollar el diagrama de tiempos del demultiplexor si se invierten las señales S_0 y S_1.

EL DEMULTIPLEXOR 74HC154



Hasta ahora hemos visto el 74HC154 como decodificador de 4-líneas a 16-líneas (Sección 6.5). Este dispositivo, así como otros decodificadores, se utiliza también en diversas aplicaciones como demultiplexor. El símbolo lógico de este circuito, cuando se emplea como demultiplexor, se muestra en la Figura 6.57. Cuando se utiliza con este fin, se usan las líneas de entrada como líneas de selección de datos, una de las entradas de

activación del chip se usa como línea de entrada de datos y la otra se mantiene a nivel BAJO, para activar la puerta interna negativa-AND que se encuentra en la parte inferior del diagrama. Este dispositivo puede estar disponible en otras familias CMOS o TTL. Consulte el sitio web de Texas Instruments en www.ti.com.

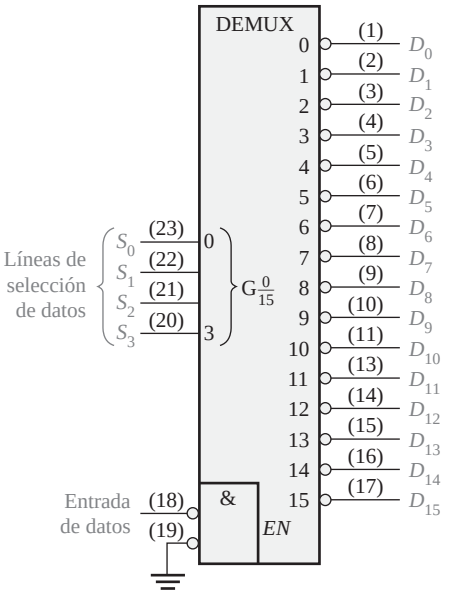


FIGURA 6.57 El decodificador 74HC154 utilizado como demultiplexor.

REVISIÓN DE LA SECCIÓN 6.9

1. En general, ¿cómo puede utilizarse un decodificador como demultiplexor?
2. El demultiplexor 74HC154 de la Figura 6.57 tiene en las líneas de selección de datos el código binario 1010 y la línea de entrada de datos está a nivel BAJO. ¿Cuáles son los estados de las líneas de salida?

6.10 GENERADORES / COMPROBADORES DE PARIDAD

Cuando se transfieren datos digitales de un punto a otro dentro de un sistema digital o cuando se transmiten códigos desde un sistema a otro, se pueden producir errores. Estos errores se manifiestan mediante cambios indeseados en los bits que conforman la información codificada; es decir, un 1 puede cambiar a 0 o un 0 a 1, debido a un mal funcionamiento de los componentes o al ruido eléctrico. En la mayoría de los sistemas digitales, la probabilidad de que haya un bit erróneo es muy pequeña, y la de que haya más de uno es todavía menor. En cualquier caso, cuando no se detecta un error, pueden originarse serios problemas en un sistema digital.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Explicar el concepto de paridad. ■ Implementar un circuito de paridad básico con puertas OR-exclusiva. ■ Describir el funcionamiento de la lógica de un generador/comprobador de paridad básico. ■ Explicar el generador/comprobador de paridad de 9 bits 74LS280. ■ Explicar cómo se puede implementar la detección de errores en una transmisión de datos.

En el Capítulo 2 se ha estudiado el método de paridad de detección de errores, en el que se añade un ***bit de paridad*** a un grupo de bits de información para conseguir que el número total de 1s sea par o impar (dependiendo del sistema que se trate). Además de los bits de paridad, hay otros códigos específicos que también permiten realizar la detección de errores.

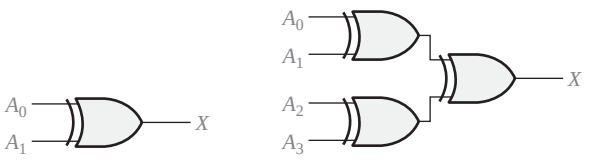
Lógica básica de la paridad

▲ *Un bit de paridad indica si el número de 1s en un código es par o impar con el fin de detectar errores.*

Para poder comprobar o generar la paridad adecuada dentro de un determinado código, se puede aplicar un principio muy sencillo:

La suma (descartando los acarreos) de un número par de 1s siempre es 0 y la suma de un número impar de 1s siempre es 1.

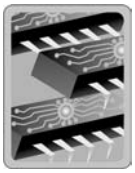
Por tanto, para determinar si un cierto código tiene **paridad par** o **paridad impar**, se suman todos los bits de ese código. Como sabemos, la suma de dos bits se puede generar mediante una puerta OR-exclusiva, como se muestra en la Figura 6.58(a); la suma de cuatro bits se puede realizar a partir de tres puertas OR-exclusiva conectadas como se indica en la Figura 6.58(b), y así sucesivamente. Cuando el número de 1s en las entradas es par, la salida X es 0 (nivel BAJO). Cuando el número de 1s es impar, la salida X es 1 (nivel ALTO).



(a) Suma de dos bits (b) Suma de cuatro bits

FIGURA 6.58

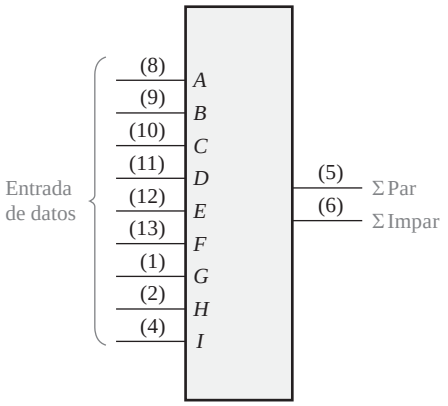
EL GENERADOR/COMPROBADOR DE PARIDAD DE 9 BITS 74LS280



El símbolo lógico y la tabla de funciones de un 74LS280 se representa en la Figura 6.59. Este dispositivo se puede utilizar para comprobar la paridad par o impar en un código de 9 bits (ocho bits de datos y un bit de paridad), o puede también emplearse para generar un bit de paridad para un código binario de hasta 9 bits. Sus entradas son desde A hasta I; cuando en las entradas hay un número par de 1s, la salida ΣPar es un nivel ALTO y la salida ΣImpar es un nivel BAJO. Este dispositivo puede estar disponible en otras familias CMOS o TTL. Consulte el sitio web de Texas Instruments en www.ti.com.

Comprobador de paridad. Cuando este dispositivo se utiliza como un comprobador de paridad par, el número de bits de entrada deberá ser siempre par; y cuando se produzca un error, la salida ΣPar pasará a nivel BAJO (L) y la salida ΣImpar será un nivel ALTO (H). Cuando se emplea como comprobador de paridad impar, el número de bits de entrada deberá ser siempre impar, y cuando se produzca un error, la salida ΣImpar será un nivel BAJO (L) y la salida ΣPar será un nivel ALTO (H).

Generador de paridad. Si este dispositivo se utiliza como generador de paridad par, el bit de paridad se toma en la salida ΣImpar, ya que esta salida es 0 cuando hay un número par de bits de entrada y 1 cuando hay un número impar. Cuando se emplea como generador de paridad impar, el bit de paridad se toma en la salida ΣPar, dado que ésta es 0 cuando el número de bits de entrada es impar.



(a) Símbolo lógico tradicional

Número de entradas A–I a nivel ALTO (H)	Salidas Σ Par	Salidas Σ Impar
0, 2, 4, 6, 8	H	L
1, 3, 5, 7, 9	L	H

(b) Tabla de funciones

FIGURA 6.59 El generador/comprobador de paridad de 9 bits 74LS280

Sistema de transmisión de datos con detección de errores

En la Figura 6.60 se muestra un sistema simplificado de transmisión de datos para ilustrar una aplicación de los generadores/comprobadores de paridad, así como de los multiplexores y demultiplexores, y para ilustrar la necesidad de dispositivos de almacenamiento de datos en algunas aplicaciones.

En esta aplicación, los datos digitales procedentes de siete fuentes se multiplexan en una única línea para ser transmitidos a un punto distante. Se aplican los siete bits de datos (D_0 hasta D_6) a las entradas de datos del multiplexor y, al mismo tiempo, a las entradas del generador de paridad par. La salida ΣImpar del generador de paridad se utiliza como bit de paridad par. Este bit es 0 si el número de 1s en las entradas de la *A* a la *I* es par, y es 1 si el número de 1s en las mismas entradas es impar. Éste es el bit D_7 del código transmitido.

Las entradas de selección de datos van pasando cíclicamente por una secuencia binaria y cada bit de datos, comenzando en D_0, se transmite en serie por la línea de transmisión ($\overline{Y}$). En este ejemplo, la línea de transmisión está formada por cuatro conductores: uno para los datos serie y los otros tres para las señales de temporización (selección de datos). Existen maneras más sofisticadas de enviar información de temporización, pero estamos usando este método directo para ilustrar un principio básico.

NOTAS INFORMÁTICAS

El microprocesador Pentium realiza comprobaciones internas de paridad, así como comprobaciones de paridad de los buses externos de direcciones y datos. En una operación de lectura, el sistema externo puede transferir la información de paridad junto con los bytes de datos. El microprocesador Pentium comprueba si la paridad resultante es par y envía la señal correspondiente. Cuando se envía un código de dirección, este microprocesador no lleva a cabo una comprobación de paridad de la dirección, pero sí que genera un bit de paridad para la dirección.

En el extremo demultiplexor del sistema, las señales de selección de datos y la cadena de datos serie se aplican al demultiplexor. Los bits de datos se distribuyen mediante el demultiplexor a las líneas de salida en el orden en que llegaron a las entradas del multiplexor. Es decir, D_0 llega a la salida D_0, D_1 llega a la salida D_1, etc. El bit de paridad llega a la salida D_7. Estos ocho bits se almacenan temporalmente y se aplican al comprobador de paridad par. No todos estos bits se encuentran presentes en las entradas del comprobador de paridad hasta que el bit de paridad D_7 aparece y se almacena. En este instante, la puerta de error es activada por el código de selección de datos 111. Si la paridad es correcta, aparece un 0 en la salida ΣPar, manteniendo la

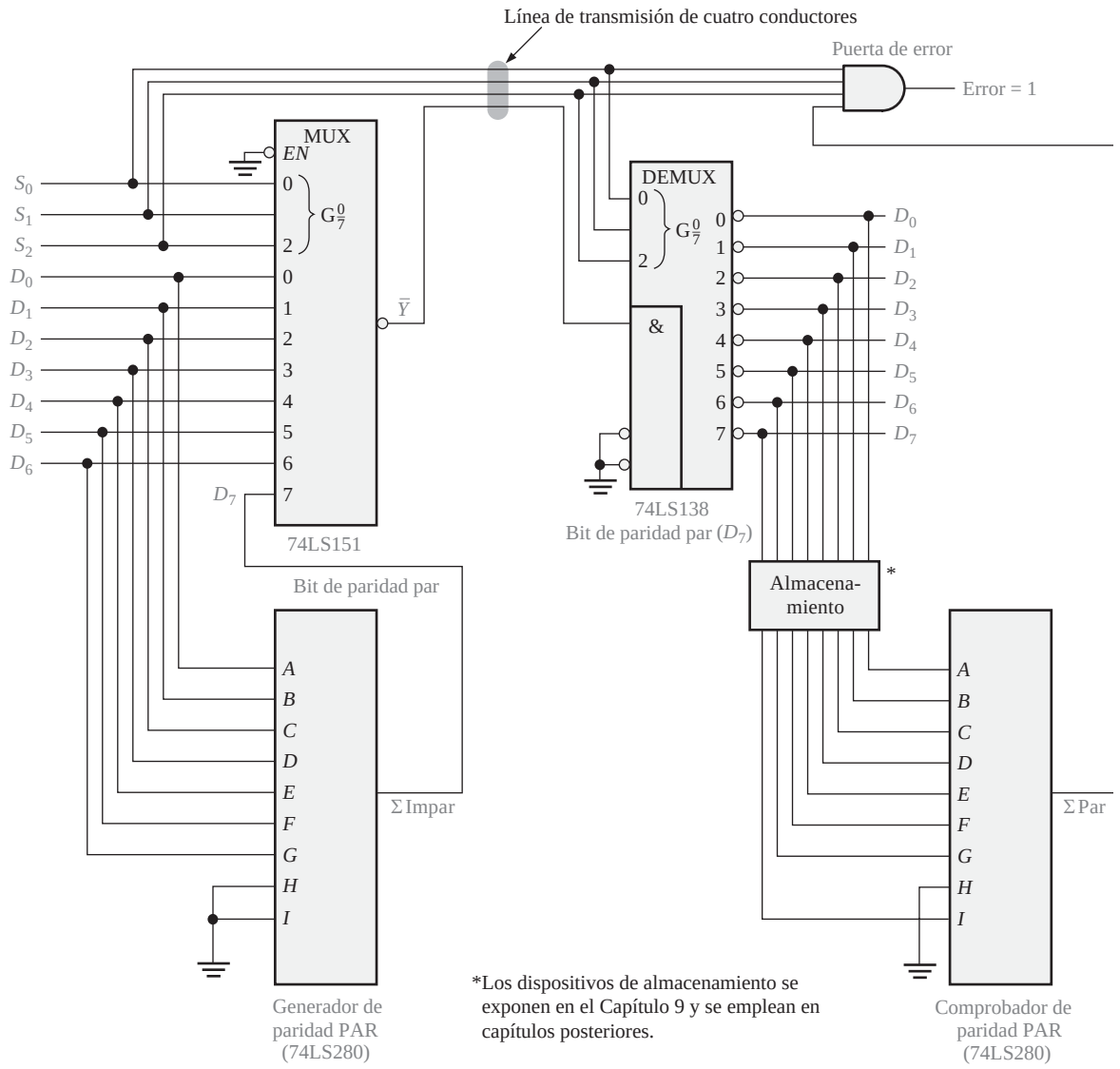


FIGURA 6.60 Sistema simplificado de transmisión de datos con detección de errores.

salida ERROR a nivel 0. Si la paridad es incorrecta, todos los 1s aparecerán en las entradas de la puerta de error, lo que da lugar a un 1 en la salida ERROR.

Esta aplicación particular demuestra que es necesario disponer de algún dispositivo de almacenamiento de datos, por lo que, cuando en el Capítulo 7 estudiemos los dispositivos de almacenamiento y los utilicemos en capítulos posteriores, seremos capaces de apreciar mejor su utilidad.

El diagrama de tiempos que se muestra en la Figura 6.61 ilustra el caso específico de transmisión de dos palabras de 8 bits, una de las cuales tiene paridad correcta y la otra se transmite con un error.

REVISIÓN DE LA SECCIÓN 6.10

1. Añadir un bit de paridad par a cada uno de los siguientes códigos:
(a) 110100 **(b)** 01100011

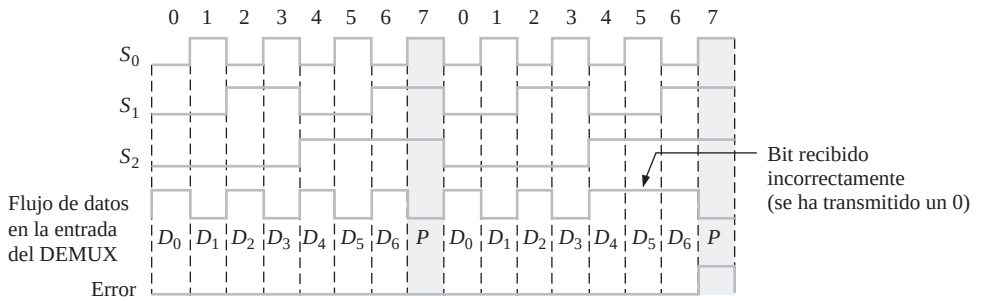


FIGURA 6.61 Ejemplo de transmisión de datos con y sin error en el sistema de la Figura 6.60.

2. Añadir un bit de paridad impar a cada uno de los siguientes códigos:
(a) 1010101 **(b)** 1000001

3. Comprobar cada uno de los siguientes códigos con paridad par e indicar si contienen errores.
(a) 100010101 **(b)** 1110111001

6.11 LOCALIZACIÓN DE AVERÍAS

En esta sección, se introduce y examina el problema de los impulsos de muy corta duración (*glitches*) en los decodificadores desde un punto de vista práctico. Un ***glitch*** es un pico de tensión o de corriente (impulso) no deseado de muy corta duración. Los circuitos lógicos pueden interpretar estos impulsos como una señal válida, originando fallos en el funcionamiento del circuito.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Explicar qué es un glitch. ■ Determinar la causa de los *glitches* en una aplicación de un decodificador. ■ Utilizar el método de la validación de salida para eliminar *glitches*.

El circuito 74LS138 se ha utilizado como demultiplexor en el sistema de transmisión de datos de la Figura 6.60. Ahora, en la Figura 6.62, se utiliza como un decodificador de 3-líneas a 8-líneas (binario-octal), para ilustrar cómo pueden ocurrir los *glitches* y cómo identificar su origen. Las entradas $A_2A_1A_0$ del decodificador se secuencian mediante un contador binario y las señales resultantes de entrada y salida se pueden visualizar en la pantalla de un analizador lógico, como se muestra en la Figura 6.62. Las transiciones de la señal A_2 están retrasadas con respecto a las transiciones de A_1, y las de A_1 respecto a las transiciones de A_0. Esto es lo que suele ocurrir cuando se emplea un contador binario para generar las señales, como veremos en el Capítulo 8.

Las señales de salida son correctas excepto por los *glitches* que aparecen en algunas de ellas. Se puede utilizar un osciloscopio o un analizador lógico para visualizar los *glitches*, que normalmente son difíciles de ver. Generalmente, es preferible el analizador lógico, especialmente para detectar las velocidades de repetición bajas (menores de 10 kHz) y/o su ocurrencia irregular, ya que la mayoría de los analizadores lógicos disponen de la función de *captura de glitches*. Los osciloscopios se pueden emplear para observar los *glitches* con cierta seguridad, especialmente si éstos se producen con una velocidad de repetición alta y constante (mayor que 10 kHz).

Los puntos de interés, que son las zonas marcadas en las señales de entrada de la Figura 6.62, se visualizan como se muestra en la Figura 6.63. En el punto 1, se produce una transición por el estado 000 debido a las diferencias de los retardos de las señales. Esto origina el primer *glitch* en la salida $\overline{0}$ del decodificador. En el punto 2, aparecen dos estados de transición, 010 y 000. Estos originan un *glitch* en la salida $\overline{2}$ del decodi-

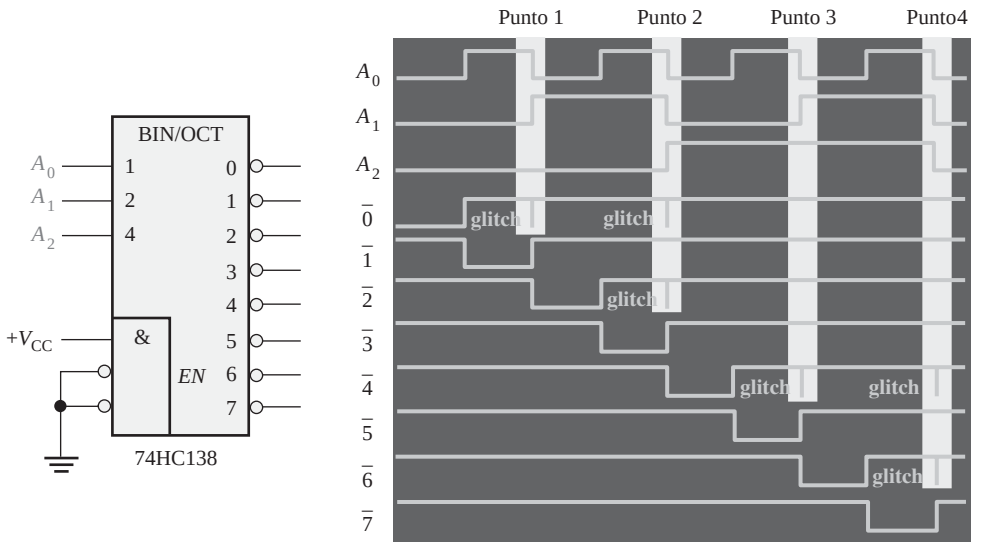


FIGURA 6.62 Formas de onda del decodificador con *glitches* en la salida.

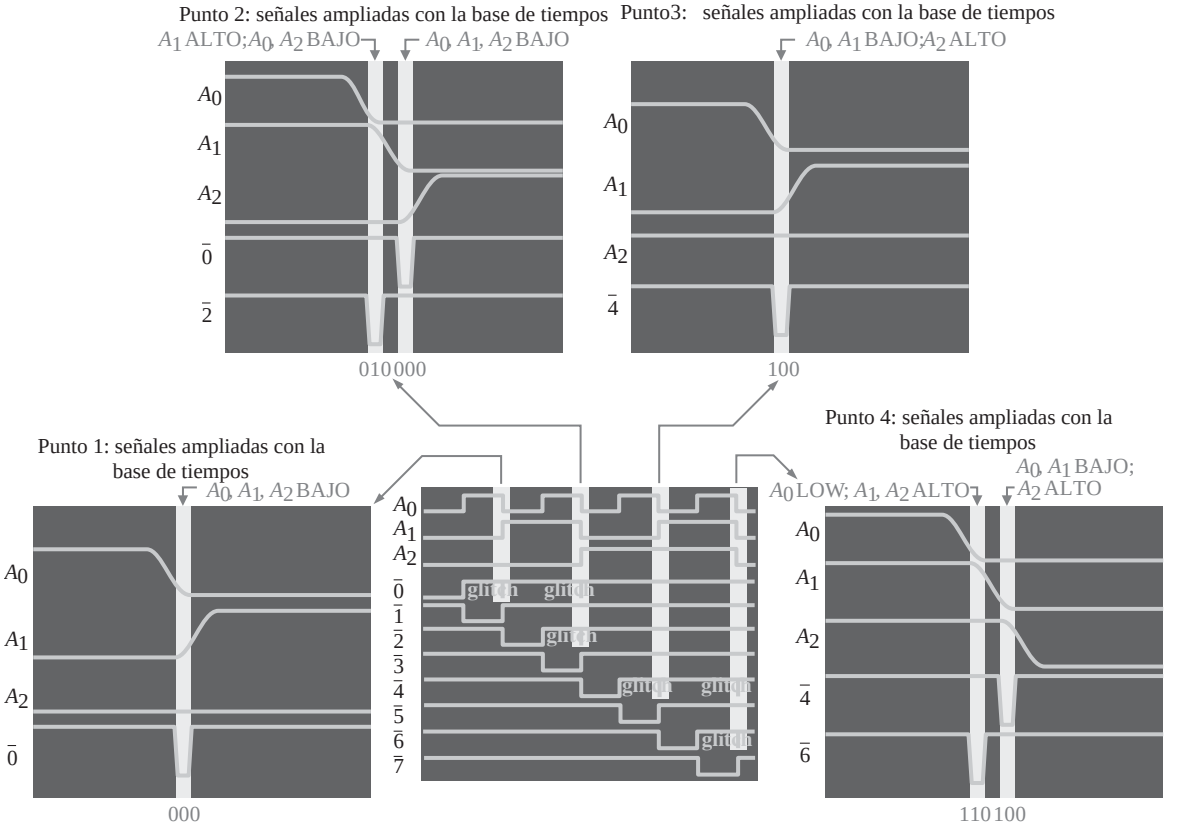


FIGURA 6.63 Formas de onda del decodificador que muestran cómo los estados de transición de entrada producen *glitches* en las señales de salida.

ficador y un segundo *glitch* en la salida $\overline{0}$, respectivamente. En el punto 3, el estado de transición es 100, lo que origina el primer *glitch* en la salida $\overline{4}$ del decodificador. En el punto 4, los dos estados de transición son 110 y 100, los cuales originan el *glitch* en la salida $\overline{6}$ del decodificador y en la salida $\overline{4}$, respectivamente.

Una manera de eliminar este problema es aplicar impulsos de **validación** (*strobing*), lo que consiste en activar el decodificador mediante un impulso de validación (*strobe*) únicamente durante los intervalos de tiempo en que las señales no se encuentran en un estado de transición. Este método se ilustra en la Figura 6.64.

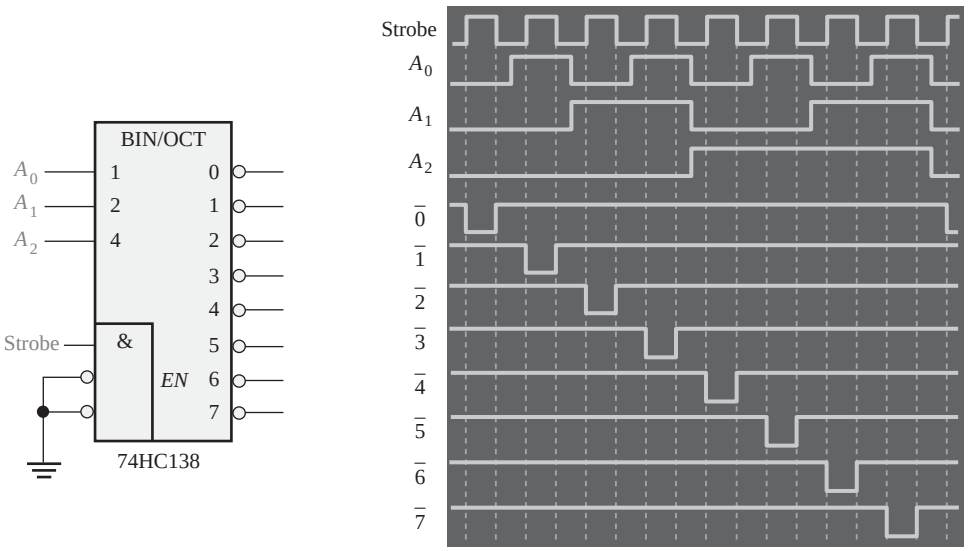


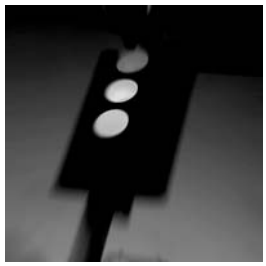
FIGURA 6.6 Aplicación de una señal de validación (*strobe*) para eliminar *glitches* en las salidas del decodificador.

CONSEJOS PRÁCTICOS

Además de los *glitches* que son el resultado de los retardos de propagación, como hemos visto en el caso de un decodificador, existen otros tipos de impulsos de ruido no deseados que pueden constituir también un problema. Los impulsos de corriente y tensión en las líneas de masa y alimentación (V_{CC}) son debidos a las señales de conmutación rápida en los circuitos digitales. Este problema se puede minimizar realizando un apropiado diseño de la placa de circuito impreso. Los impulsos de conmutación pueden ser absorbidos mediante el desacoplo de la tarjeta de circuito impreso con condensadores de 1 μF entre V_{CC} y masa. También deberían distribuirse condensadores más pequeños de desacoplo (0,022 μF a 0,1 μF) en distintos puntos de la placa de circuito impreso entre V_{CC} y masa. El desacoplo debería realizarse muy próximo a los dispositivos que conmutan a altas velocidades o que excitan más cargas, como por ejemplo, osciladores, contadores, buffers y controladores de bus.

REVISIÓN DE LA SECCIÓN 6.11

1. Definir el término *glitch*.
2. Explicar la principal causa de los *glitches* en los decodificadores.
3. Definir el impulso de *validación* (*strobe*).



APLICACIÓN A LOS SISTEMAS DIGITALES

En esta aplicación, vamos a comenzar a trabajar con el sistema de control de semáforos. En esta sección se establecen los requisitos del sistema, se desarrolla un diagrama de bloques, así como un diagrama de estados para ayudar a establecer la secuencia de funcionamiento. Diseñaremos la parte del sistema que involucra lógica combinacional y se propondrán los métodos de prueba. En los Capítulos 7 y 8 se tratarán los circuitos de lógica secuencial y de temporización del sistema.

Requerimientos generales del sistema

Se requiere un controlador digital para controlar un semáforo en la intersección de una calle de tráfico muy denso con una calle de tráfico moderado. La calle principal va a tener una luz verde durante un mínimo de 25 seg. o mientras no haya ningún vehículo en la calle perpendicular. Esta calle lateral tiene que tener la luz verde hasta que no circule ningún coche por ella, o durante un máximo de 25 seg. La luz ámbar de precaución tiene que durar 4 seg. en los cambios de luz verde a roja en ambas calles, principal y lateral. Estos requisitos se muestran en el diagrama de la Figura 6.65.

Desarrollo de un diagrama de bloques del sistema

A partir de los requisitos, se puede desarrollar un diagrama de bloques del sistema. En primer lugar, sabemos que el sistema tiene que controlar seis pares de luces diferentes. Estas son las luces roja, ámbar y verde para ambos sentidos, tanto en la calle principal como en la lateral. También sabemos que existe una entrada externa (además de la alimentación) que proviene de un sensor de vehículos situado en la calle lateral. En la Figura 6.66, puede ver un diagrama de bloques mínimo que ilustra estos requisitos.

A partir del diagrama de bloques mínimo vamos a ir entrando en los detalles. El sistema tiene cuatro estados, como se indica en la Figura 6.65, por lo que se necesita un circuito lógico para controlar la secuencia de estados (lógica secuencial). Además, se necesitan circuitos para generar los intervalos de tiempo adecuados de 25 seg. y 4 seg., que se requieren en el sistema y para generar una señal de reloj cíclica en el sistema (circuitos de temporización). Los intervalos de tiempo (largo y corto) y el sensor de vehículos son entradas de la lógica secuencial, dado que la secuenciación de estados es una función de estas variables. Se necesitan también circuitos lógicos para determinar cuál de los cuatro estados del sistema está activo en un determinado instante de tiempo, para así generar las sali-

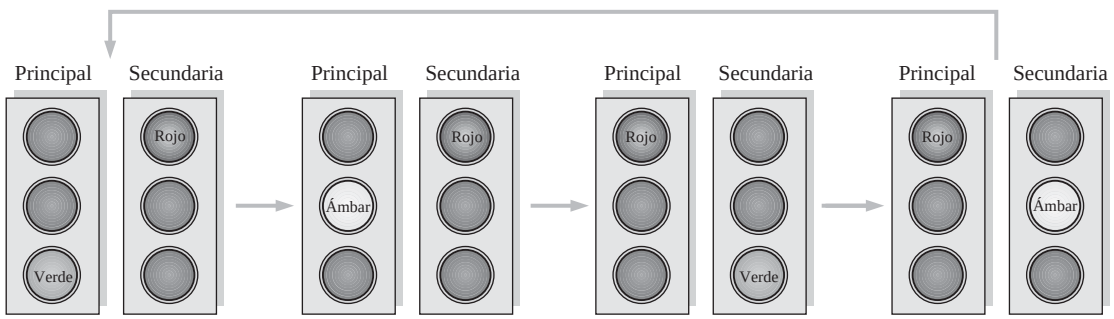


FIGURA 6.65 Requisitos para la secuencia de luces de los semáforos.

das adecuadas en las luces (decodificación de estados y lógica de salida), y para iniciar los intervalos de tiempo largo y corto. Finalmente, se necesita un circuito de interfaz para convertir los niveles lógicos de la decodificación y del circuito de salida en las tensiones y corrientes requeridas para encender cada una de las luces. La Figura 6.67 representa un diagrama de bloques más detallado que muestra estos elementos esenciales.

El diagrama de estados

Un diagrama de estados nos muestra gráficamente la secuencia de estados en un sistema y las condiciones de cada estado y de las transiciones entre cada uno de ellos. En realidad, la Figura 6.65 es, en cierta medida, un diagrama de estados, ya que muestra la secuencia de estados y las distintas condiciones.

Definición de las variables. Antes de poder desarrollar un diagrama de estados tradicional, es necesario definir las variables que determinan cómo pasa el sistema a través de los diferentes estados. A continuación se enumeran estas variables y sus símbolos:

- Presencia de vehículos en la calle lateral = V_s
- El temporizador de 25 s. (largo) está *activado* = T_L
- El temporizador de 4 s. (corto) está *activado* = T_S

El uso de variables complementadas indica la condición contraria. Por ejemplo, $\overline{V}_s$ indica que no hay ningún vehículo en la calle lateral; $\overline{T}_L$ indica que el temporizador de larga duración está desactivado y $\overline{T}_S$ indica que el temporizador de corta duración está desactivado.

Descripción del diagrama de estados. En la Figura 6.68 se muestra un diagrama de estados. Cada uno de los cuatro estados se etiqueta de acuerdo a la secuencia de 2 bits en código Gray, como se indica mediante los círculos. La flecha circular en cada estado indica que el sistema permanece en dicho estado bajo la condición definida por la variable o expresión asociada. Cada una de las flechas que van de un estado al siguiente indican un cambio de estado cuando se produce la condición definida por la variable o expresión asociada.

Primer estado El código Gray para este estado es 00. El semáforo de la calle principal está en verde y el de la calle lateral está en rojo. El sistema permanece en este estado al menos 25 segundos cuando el temporizador largo se encuentra activado o mientras que no haya ningún vehículo en la calle lateral $(T_L + \overline{V}_s)$. El sistema pasa al siguiente estado cuando el temporizador de 25 segundos está desactivado o cuando aparece algún vehículo en la calle secundaria $(\overline{T}_L V_s)$.

Segundo estado El código Gray para este estado es 01. El semáforo de la calle principal está en ámbar (precaución) y el de la calle lateral está en rojo. El sistema permanece en este estado durante 4 segundos mientras el temporizador corto está activado (T_S) y pasa al siguiente estado cuando este temporizador se desactiva $(\overline{T}_S)$.

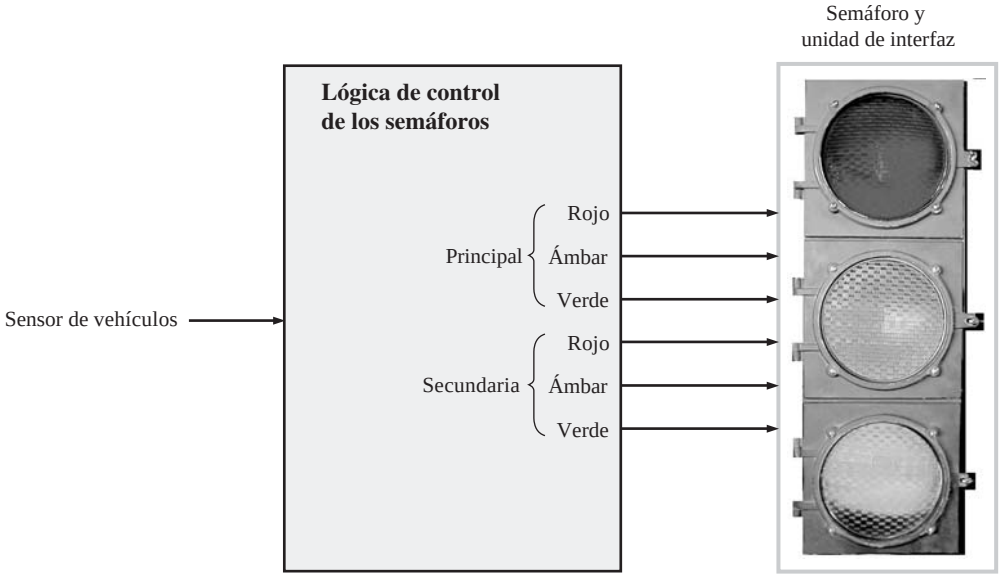


FIGURA 6.66 Diagrama de bloques mínimo del sistema.

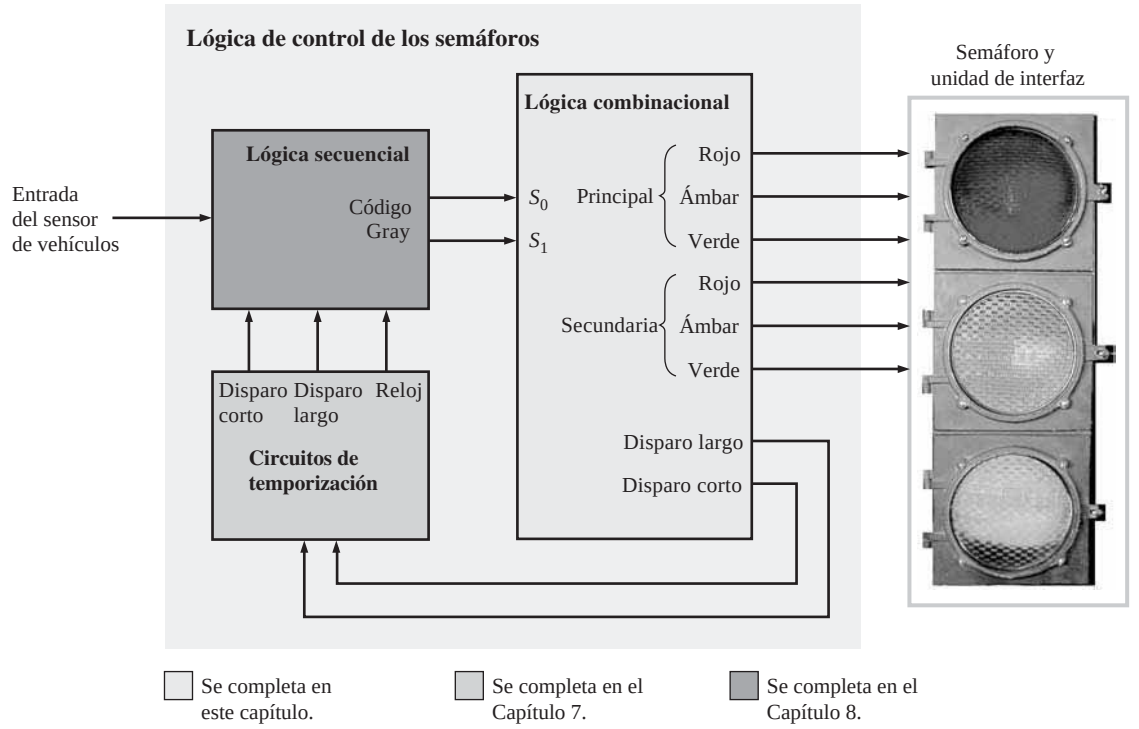


FIGURA 6.67 Diagrama de bloques del sistema en el que se indican los elementos esenciales.

Tercer estado El código Gray para este estado es 11. El semáforo de la calle principal está en rojo y el de la calle lateral está en verde. El sistema permanece en este estado cuando el temporizador largo se encuentra activado y hay un vehículo en la calle lateral ($T_L V_S$). El sistema pasa al siguiente estado cuando han transcurrido los 25 segundos o cuando no hay ningún vehículo en la calle secundaria, lo primero que ocurra ($\overline{T}_L + \overline{V}_S$).

Cuarto estado El código Gray para este estado es 10. El semáforo de la calle principal está en rojo y el de la calle lateral está en ámbar. El sistema permanece en este estado 4 segundos cuando el temporizador corto se encuentra activado (T_S) y vuelve al primer estado cuando el temporizador corto se desactiva ($\overline{T}_S$).

La lógica combinacional

En esta sección dedicada a las aplicaciones de sistemas nos vamos a centrar en la parte correspondiente a la lógica combinacional del diagrama de bloques de la Figura 6.67. Los circuitos de temporización y de la lógica secuencial serán el tema del que tratemos en las secciones dedicadas a las aplicaciones de sistemas de los Capítulos 7 y 8.

El primer paso en el diseño de la lógica consiste en desarrollar un diagrama de bloques para la parte de la lógica combinacional del sistema. Las tres funciones que esta lógica debe realizar se definen a continuación y el diagrama resultante, con un bloque para cada una de las tres funciones, se muestra en la Figura 6.69:

- **Decodificador de estados.** Decodificar el código Gray de 2 bits de la lógica secuencial, para determinar en cuál de los cuatro estados se encuentra el sistema.
- **Lógica de salida de las luces.** Utilizar el estado decodificado para activar (mediante los circuitos de interfaz) las dos luces de los semáforos apropiadas, de las seis luces existentes.
- **Lógica del circuito de disparo.** Utilizar los estados decodificados para generar las señales que inicialicen (disparen) adecuadamente los temporizadores largo y corto.

Implementación de la lógica combinacional

Implementación de la lógica del decodificador El decodificador de estados tiene dos entradas (código Gray de 2

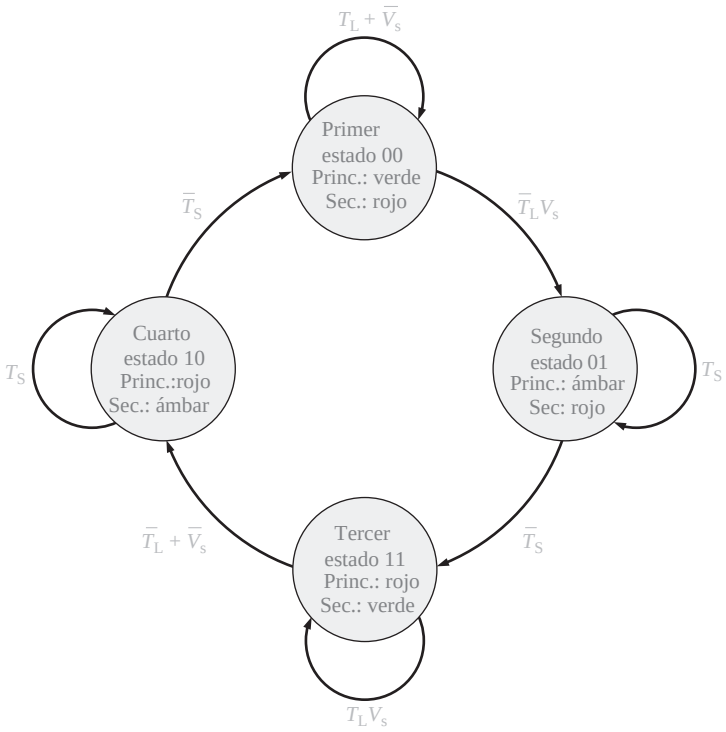


FIGURA 6.68 Diagrama de estados del sistema de control de semáforos, indicando la secuencia de código Gray.

bits) y tiene que tener una salida para cada uno de los cuatro estados, como se muestra en la Figura 6.70. Las dos entradas en código Gray se designan por S_0 y S_1 y las cuatro salidas de estado se etiquetan como SO_1, SO_2, SO_3 y SO_4. Las expresiones booleanas para las salidas de los estados son las siguientes:

$$SO_1 = \overline{S}_1\overline{S}_0$$
$$SO_2 = \overline{S}_1 S_0$$
$$SO_3 = S_1 S_0$$
$$SO_4 = S_1\overline{S}_0$$

La tabla de verdad para la lógica del decodificador de estados se muestra en la Tabla 6.11.

Implementación de la lógica de salida de las luces Esta lógica tiene que tomar las cuatro salidas de estado y generar seis salidas para activar las luces de los semáforos. Estas salidas se designan por *MR, MY, MG* (para la luces roja, ámbar y verde del semáforo de la calle principal) y por *SR, SY, SG* (para la luces roja, ámbar y verde del semáforo de la calle secundaria). Usando como referencia la Tabla 6.11, vemos que las salidas de las luces de los semáforos puede expresarse como:

$$MR = SO_3 + SO_4 \qquad SR = SO_1 + SO_2$$
$$MY = SO_2 \qquad SY = SO_4$$
$$MG = SO_1 \qquad SG = SO_3$$

La lógica de salida se implementa como se muestra en la Figura 6.71.

Implementación de la lógica del circuito de disparo La lógica de disparo produce dos salidas. La salida para el circuito de temporización de 25 s. (temporizador largo) da lugar a una transición de nivel BAJO a nivel ALTO, cuando el sistema pasa a los estados primero (00) o tercero (11). La salida para disparar el circuito temporizador de 4 s. (temporizador corto) da lugar a una transición de nivel BAJO a nivel ALTO cuando el sistema pasa a los estados segundo (01) o cuarto (10). Las salidas del circuito de disparo se muestran en la Tabla 6.11 y en forma de ecuación son:

$$Temporizador\ largo = SO_1 + SO_3$$
$$Temporizador\ corto = SO_2 + SO_4$$

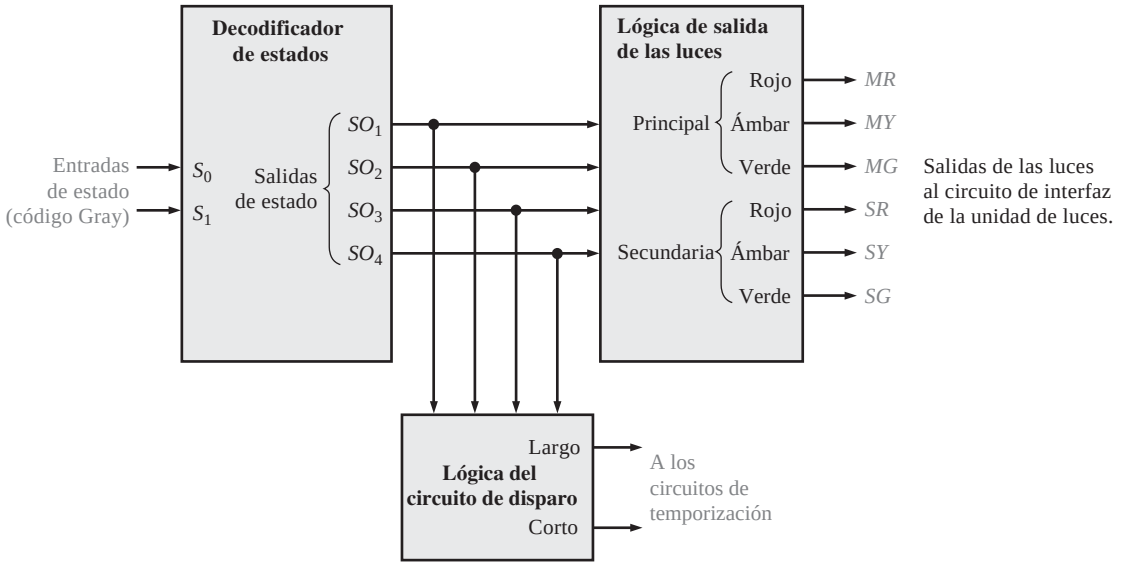


FIGURA 6.69 Diagrama de bloques para la lógica combinacional.

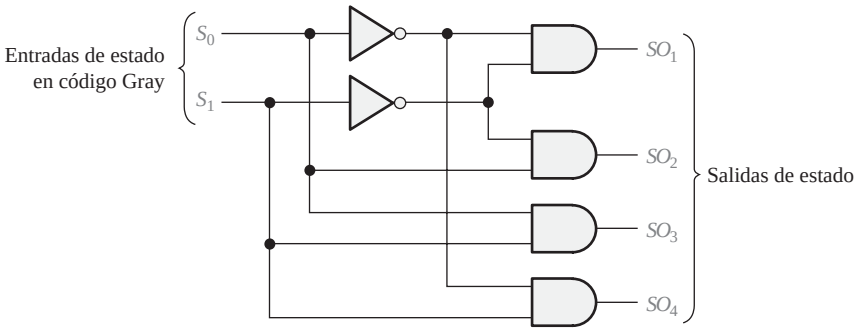


FIGURA 6.70 Lógica del decodificador de estados.

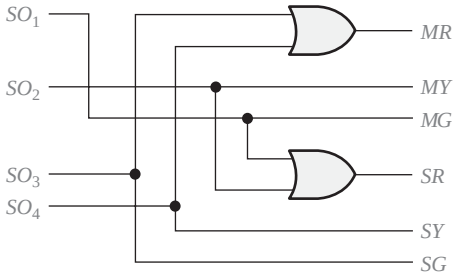


FIGURA 6.71 Lógica de salida de las luces.

Entradas de estado		Salidas de estado				Salidas de las luces						Salidas del circuito de disparo	
S_1	S_0	SO_1	SO_2	SO_3	SO_4	*MR*	*MY*	*MG*	*SR*	*SY*	*SG*	*LARGO*	*CORTO*
0	0	1	0	0	0	0	0	1	1	0	0	1	0
0	1	0	1	0	0	0	1	0	1	0	0	0	1
1	1	0	0	1	0	1	0	0	0	0	1	1	0
1	0	0	0	0	1	1	0	0	0	1	0	0	1

Las salidas de estado y las salidas de las luces son activas a nivel ALTO. *MR* corresponde al semáforo de la calle principal (*M)* cuando está en rojo (*R*), *SG* corresponde al semáforo de la calle secundaria (*S*) cuando está en verde (*G*).

TABLA 6.11 Tabla de verdad de la lógica combinacional.

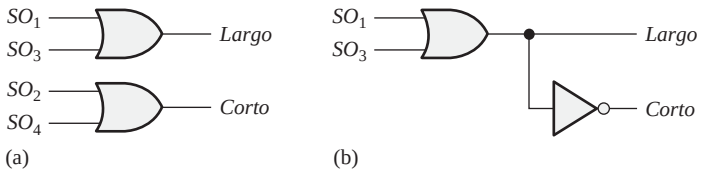


FIGURA 6.72 Lógica del circuito de disparo.

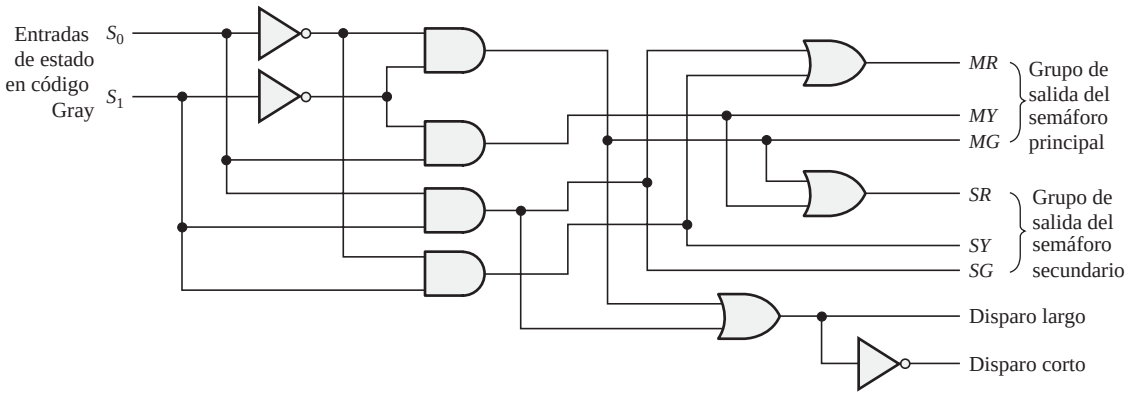


FIGURA 6.73 Lógica combinacional completa.

En la Figura 6.72(a) se muestra la lógica del circuito de disparo. La Tabla 6.11 también muestra que las salidas *LARGO* y *CORTO* son complementarias, por lo que la lógica puede implementarse también con una puerta OR y un inversor, como se ilustra en la parte (b).

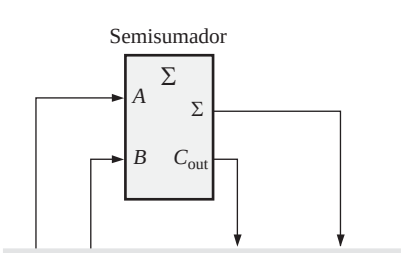
La Figura 6.73 muestra la lógica combinacional completa que combina el decodificador de estados, la lógica de salida de las luces y la lógica de disparo.

Prácticas de sistemas

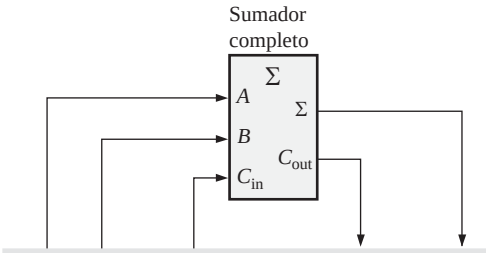
- ***Actividad 1***. Aplicar las formas de onda del código Gray de 2 bits en las entradas S_0 y S_1 de la lógica combinacional y desarrollar todas las señales de salida.
- ***Actividad 2***. Demostrar cómo podría implementarse la lógica combinacional usando funciones 74XX.
- ***Actividad opcional***. Escribir un programa VHDL que describa la lógica combinacional.

RESUMEN

- El funcionamiento del semi-sumador y del sumador completo se muestran en la Figura 6.74.
- En la Figura 6.75 se muestran los símbolos lógicos y la numeración de pines de los CI empleados en este capítulo. La designación de pines puede variar dependiendo del fabricante.

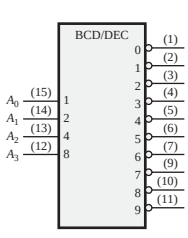


Entradas A	Entradas B	Salida de acarreo C_{out}	Suma Σ
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

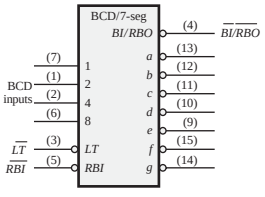


Entradas A	Entradas B	Entrada de acarreo C_{in}	Salida de acarreo C_{out}	Suma Σ
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

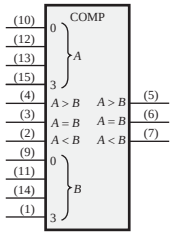
FIGURA 6.74



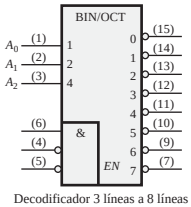
Decodificador BCD-decimal
74HC42



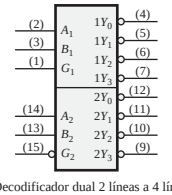
Controlador/decodificador BCD a 7-segmentos
74LS47



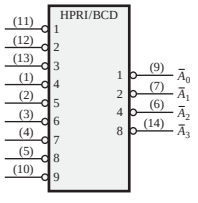
Comparador de magnitud de 4 bits
74HC85



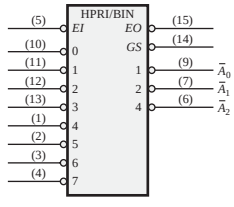
Decodificador 3 líneas a 8 líneas
74LS138



Decodificador dual 2 líneas a 4 líneas
74LS139



Codificador con prioridad decimal-BCD
74HC147



Codificador octal-binario
74LS148

FIGURA 6.75 *(Continúa)*

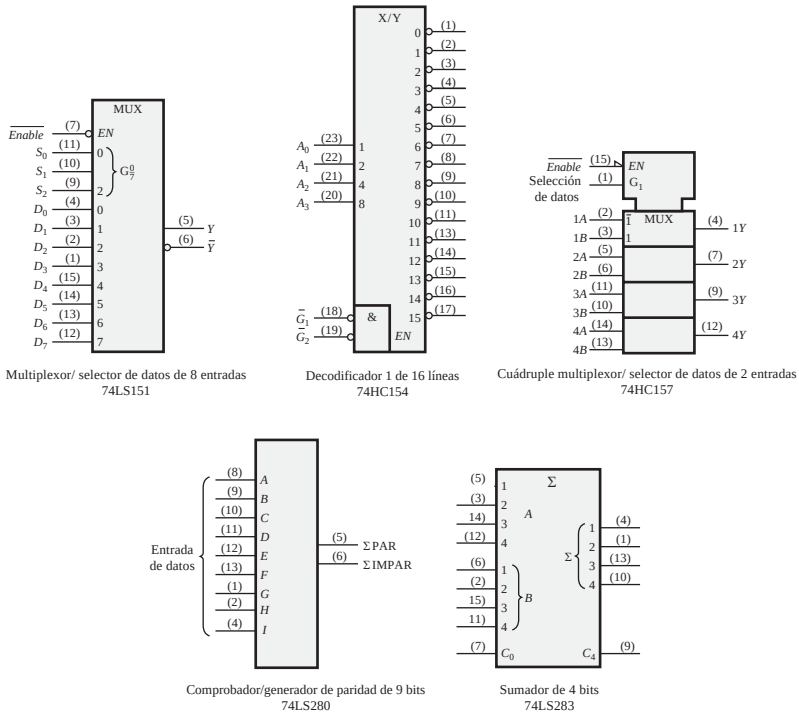


FIGURA 6.75 *(Continuación)*

PALABRAS CLAVE

Las palabras clave y otros términos que se han resaltado en negrita se encuentran en el glosario final del libro.

Acarreo anticipado Método de suma binaria en que los acarreos de las etapas sumadoras precedentes se anticipan, eliminando de esta manera los retardos en la propagación de los acarreos.

Acarreo serie Método de suma binaria en el que el acarreo de salida de cada sumador se convierte en el acarreo de entrada del sumador siguiente.

Bit de paridad Bit que se añade a cada grupo de bits de información para hacer que el número de unos sea par o impar en dicho grupo de bits.

Codificador Circuito digital que convierte la información a un formato codificado.

Codificador con prioridad Codificador en el que sólo se codifica el dígito de entrada de valor más alto, ignorándose cualquier otra entrada activa.

Conexión en cascada Conectar la salida de un dispositivo a la entrada de un dispositivo similar, permitiendo a uno de los dispositivos excitar a otro, para aumentar la capacidad de operación.

Decodificador Circuito digital que convierte la información codificada en un formato más familiar o no codificado.

Demultiplexor (DEMUX) Circuito que conmuta los datos digitales desde una línea de entrada a varias líneas de salida según una secuencia temporal especificada.

Glitch Un pico de corriente o de tensión de corta duración, generalmente indeseado y que se produce de forma no intencionada.

Multiplexor (MUX) Circuito que conmuta los datos digitales de distintas líneas de entrada a una única línea de salida según una secuencia temporal especificada.

Semisumador Circuito digital que suma dos bits y genera una suma y un acarreo de salida. No puede manipular acarreos de entrada.

Sumador completo Circuito digital que suma dos bits y un acarreo de entrada para producir una suma y un acarreo de salida.

AUTOTEST

Las respuestas se encuentran al final del capítulo.

1. Un semi-sumador se caracteriza por tener:

(a) dos entradas y dos salidas **(b)** tres entradas y dos salidas.

(c) dos entradas y tres salidas **(d)** dos entradas y una salida.

2. Un sumador completo se caracteriza por tener:

(a) dos entradas y dos salidas **(b)** tres entradas y dos salidas.

(c) dos entradas y tres salidas **(d)** dos entradas y una salida.

3. Las entradas de un sumador completo son $A = 1$, $B = 1$, $C_{in} = 0$. Sus salidas son:

(a) $\Sigma = 1$, $C_{out} = 1$ **(b)** $\Sigma = 1$, $C_{out} = 0$

(c) $\Sigma = 0$, $C_{out} = 1$ **(d)** $\Sigma = 0$, $C_{out} = 0$

4. Un sumador en paralelo de 4 bits puede sumar:

(a) dos números binarios de 4 bits **(b)** dos números binarios de 2 bits

(c) cuatro bits a la vez **(d)** una secuencia de cuatro bits

5. Para ampliar un sumador en paralelo de 4 bits a un sumador en paralelo de 8 bits, hay que:

(a) usar cuatro sumadores de 4 bits sin interconexiones

(b) usar dos sumadores de 4 bits y conectar las salidas de la suma de uno de ellos a las entradas de datos del otro.

(c) usar ocho sumadores de 4 bits sin interconexiones

(d) usar dos sumadores de 4 bits y conectar la salida de acarreo de uno de ellos a la entrada de acarreo del otro.

6. Si un comparador de magnitud 74HC85 tiene $A = 1011$ y $B = 1001$ en su entrada, las salidas son:

(a) $A > B = 0$, $A < B = 1$, $A = B = 0$ **(b)** $A > B = 1$, $A < B = 0$, $A = B = 0$

(c) $A > B = 1$, $A < B = 1$, $A = B = 0$ **(d)** $A > B = 0$, $A < B = 0$, $A = B = 1$

7. Si un decodificador de 4 líneas a 16 líneas con salidas activas a nivel BAJO persenta un nivel BAJO en la salida decimal 12, ¿cuáles son las entradas?

(a) $A_3A_2A_1A_0 = 1010$ **(b)** $A_3A_2A_1A_0 = 1110$

(c) $A_3A_2A_1A_0 = 1100$ **(d)** $A_3A_2A_1A_0 = 0100$

8. Un decodificador BCD a 7 segmentos tiene 0100 en sus entradas. Las salida activas serán:

(a) *a, c, f, g* **(b)** *b, c, f, g* **(c)** *b, c, e, f* **(d)** *b, d, e, g*

9. Si un codificador con prioridad octal-binario tiene en sus entradas 0, 2, 5 y 6 en un nivel activo, la salida binaria activa a nivel ALTO será:

(a) 110 **(b)** 010 **(c)** 101 **(d)** 000

10. En general, un multiplexor tiene

(a) una entrada de datos, varias salidas de datos y entradas de selección.

(b) una entrada de datos, una salida de datos y una entrada de selección.

(c) varias entradas de datos, varias salidas de datos y entradas de selección.

(d) varias entradas de datos, una salida de datos y entradas de selección.

11. Básicamente, los selectores de datos son lo mismo que:

(a) decodificadores **(b)** demultiplexores

(c) multiplexores **(d)** codificadores

12. ¿Cuáles de los códigos siguientes tienen paridad par?

(a) 10011000 **(b)** 01111000 **(c)** 11111111

(d) 11010101 **(e)** todos **(f)** las respuestas (b) y (c)

PROBLEMAS

Las respuestas a los problemas impares se encuentran al final del libro.

SECCIÓN 6.1 Sumadores básicos

1. Para el sumador completo de la Figura 6.4, determinar el estado lógico (1 o 0) a la salida de cada puerta para las siguientes entradas:

(a) $A = 1, B = 1, C_{in} = 1$ **(b)** $A = 0, B = 1, C_{in} = 1$ **(c)** $A = 0, B = 1, C_{in} = 0$

2. ¿Cuáles serían las entradas que producirían en un sumador completo las siguientes salidas?

(a) $\Sigma = 0, C_{out} = 0$ **(b)** $\Sigma = 1, C_{out} = 0$ **(c)** $\Sigma = 1, C_{out} = 1$ **(d)** $\Sigma = 0, C_{out} = 1$

3. Determinar las salidas de un sumador completo para cada una de las siguientes entradas:

(a) $A = 1, B = 0, C_{in} = 0$ **(b)** $A = 0, B = 0, C_{in} = 1$

(a) $A = 0, B = 1, C_{in} = 1$ **(d)** $A = 1, B = 1, C_{in} = 1$

SECCIÓN 6.2 Sumadores binarios en paralelo

4. Para el sumador en paralelo de la Figura 6.76, determinar la suma completa mediante el análisis del funcionamiento lógico del circuito. Comprobar el resultado sumando manualmente los dos números de entrada.

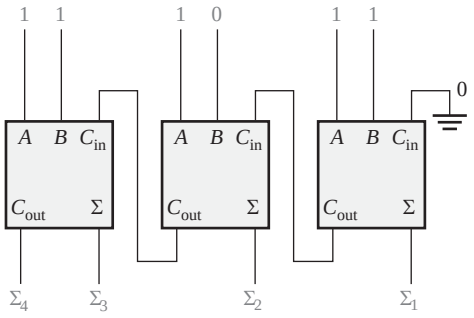


FIGURA 6.76

5. Repetir el Problema 4 para el circuito y las condiciones de entrada de la Figura 6.77.

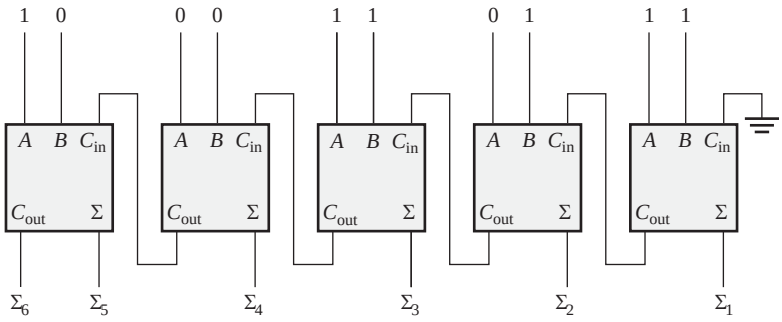


FIGURA 6.77

6. Las formas de onda de entrada de la Figura 6.78 se aplican a un sumador de 2 bits. Determinar, mediante un diagrama de tiempo, las señales correspondientes a la suma y a la salida de acarreo, en función de las entradas.

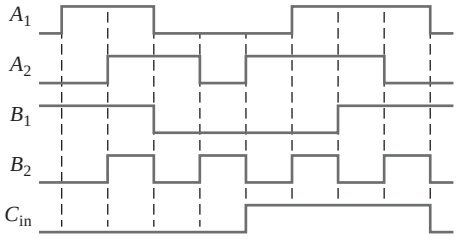


FIGURA 6.78

7. Las siguientes secuencias de bits (el bit de la derecha es el primero) se aplican a las entradas de un sumador en paralelo de 4 bits. Determinar la secuencia de bits resultante en cada salida.

A_1	1001
A_2	1110
A_3	0000
A_4	1011
B_1	1111
B_2	1100
B_3	1010
B_4	0010

8. En las pruebas de un sumador completo de 4 bits 74LS83, se observan los siguientes niveles de tensión en sus pines: 1-ALTO, 2-ALTO, 3-ALTO, 4-ALTO, 5-BAJO, 6-BAJO, 7-BAJO, 9-ALTO, 10-BAJO, 11-ALTO, 12-BAJO, 13-ALTO, 14-ALTO y 15-ALTO. Determinar si el circuito integrado funciona correctamente.

SECCIÓN 6.3 Sumadores de acarreo serie y acarreo anticipado

9. Cada uno de los ocho sumadores completos de un sumador de 8 bits con acarreo anticipado presenta los siguientes retardos de propagación:

A a Σ y C_{out}:	40 ns
B a Σ y C_{out}:	40 ns
C_{in}:	35 ns
C_{in} a C_{out}:	25 ns

Determinar el tiempo total máximo necesario para sumar dos números de 8 bits.

10. Indicar qué circuitería adicional es necesaria para convertir en sumador de 4 bits de acarreo anticipado de la Figura 6.18 en un sumador de 5 bits.

SECCIÓN 6.4 Comparadores

11. Se aplican las formas de onda mostradas en la Figura 6.79 a las entradas del comparador. Determinar la señal de salida (A=B).

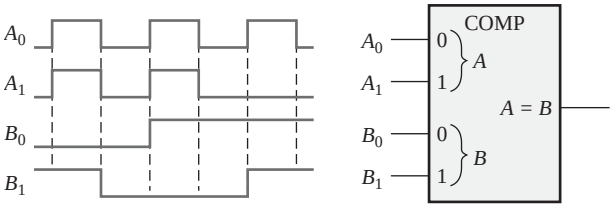


FIGURA 6.79

12. Para el comparador de 4 bits de la Figura 6.80, dibujar cada forma de onda de salida para las entradas que se muestran. Las salidas son activas a nivel ALTO.

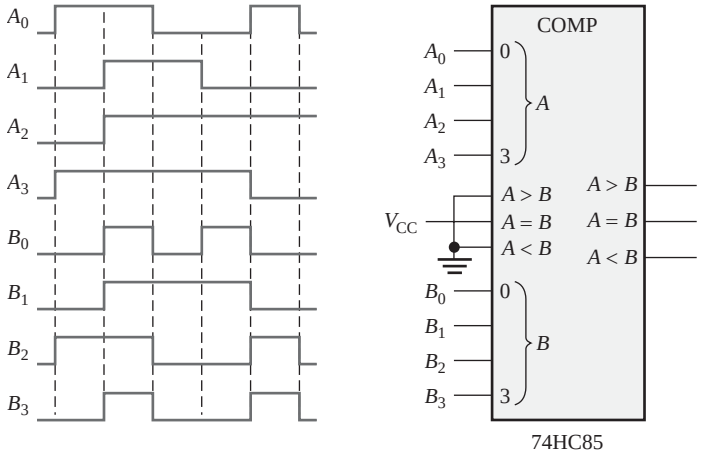


FIGURA 6.80

13. Para los siguientes grupos de números binarios, determinar los estados de salida para el comparador de la Figura 6.22.

(a) $A_3A_2A_1A_0 = 1100$; $B_3B_2B_1B_0 = 1001$

(b) $A_3A_2A_1A_0 = 1000$; $B_3B_2B_1B_0 = 1011$

(c) $A_3A_2A_1A_0 = 0100$; $B_3B_2B_1B_0 = 0100$

SECCIÓN 6.5 Decodificadores

14. Cuando en la salida de cada puerta de decodificación de la Figura 6.81 hay un nivel ALTO, ¿cuál es el código binario que aparece en sus entradas? El bit más significativo (MSB) es A_3.

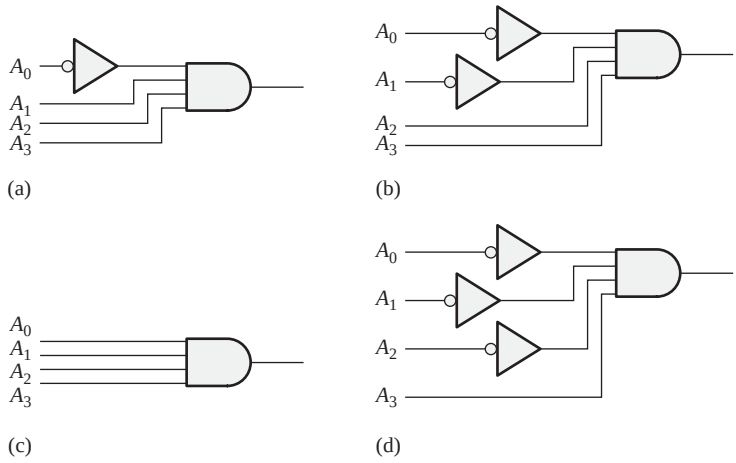


FIGURA 6.81

15. ¿Cuál es la lógica de decodificación para cada uno de los siguientes códigos, si se requiere una salida activa a nivel ALTO (1)?

(a) 1101 **(b)** 1000 **(c)** 11011 **(d)** 11100

(e) 101010 **(f)** 111110 **(g)** 000101 **(h)** 1110110

16. Resolver el Problema 13, suponiendo que se requiere una salida activa a nivel BAJO (0).

17. Se desea detectar únicamente la presencia de los códigos 1010, 1100, 0001 y 1011. Para indicar la presencia de dichos códigos se requiere una salida activa a nivel ALTO. Desarrollar la lógica de decodificación mínima necesaria que tenga una única salida que indique cuándo cualquiera de estos códigos se encuentra en las entradas. Para cualquier otro código, la salida ha de ser un nivel BAJO.

18. Si se aplican las formas de onda de entrada a la lógica de decodificación de la Figura 6.82, dibujar las formas de onda de salida en función de dichas entradas.

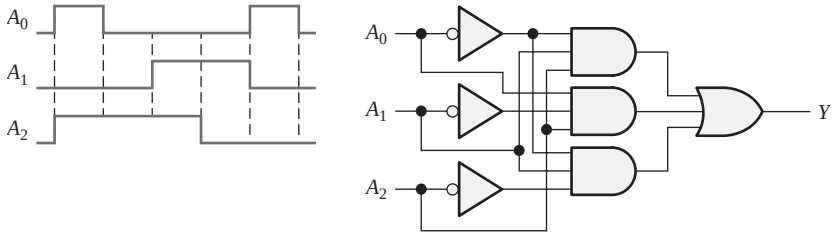


FIGURA 6.82

19. Se aplican secuencialmente números BCD al decodificador BCD-decimal de la Figura 6.83. Dibujar un diagrama de tiempos que muestre cada salida en relación con el resto de las señales de salida y con las de entrada.

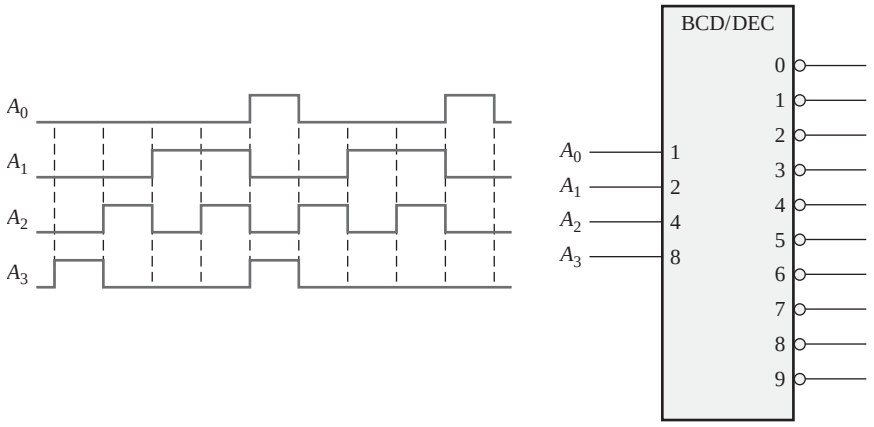


FIGURA 6.83

20. Un decodificador/excitador de 7-segmentos controla el display de la Figura 6.84. Si se aplican las formas de onda de entrada que se muestran, determinar la secuencia de dígitos que aparece en el display.

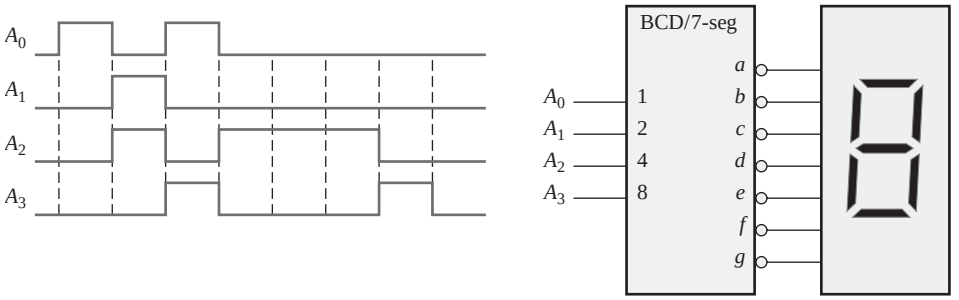


FIGURA 6.84

SECCIÓN 6.6 Codificadores

21. Suponer que el codificador lógico decimal-BCD de la Figura 6.38 tiene las entradas 3 y 9 a nivel ALTO. ¿Cuál es el código de salida? ¿Es éste un código BCD (8421) válido?

22. Un decodificador 74HC147 tiene niveles BAJOS de tensión en sus pines 2, 5 y 12. ¿Qué código BCD aparece en las salidas si todas las demás entradas están a nivel ALTO?

SECCIÓN 6.7 Convertidores de código

23. Convertir a BCD los siguientes números decimales y luego a binario.

(a) 2 **(b)** 8 **(c)** 13

(d) 26 **(e)** 33

24. Explicar la lógica requerida para convertir a código Gray un número binario de 10 bits, y utilizar esta lógica para convertir los siguientes números binarios:

(a) 1010101010 **(b)** 1111100000

(c) 0000001110 **(d)** 1111111111

25. Explicar la lógica requerida para convertir a binario un código Gray de 10 bits y utilizar esta lógica para convertir a binario los siguientes códigos Gray:

(a) 1010000000 **(b)** 0011001100

(c) 1111000111 **(d)** 0000000001

SECCIÓN 6.8 Multiplexores (selectores de datos)

26. En el demultiplexor de la Figura 6.85, determinar la salida para los siguientes estados de entrada: $D_0 = 0$, $D_1 = 1$, $D_2 = 1$, $D_3 = 0$, $S_0 = 1$, $S_1 = 0$

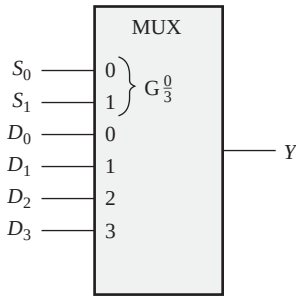


FIGURA 6.85

27. Si las entradas de selección de datos del multiplexor de la Figura 6.85 se secuencian tal y como se muestra en las formas de onda de la Figura 6.86, determinar la forma de onda de salida para los datos de entrada del Problema 26.

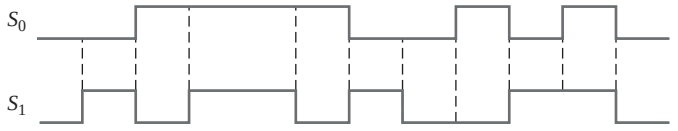


FIGURA 6.86

28. Las formas de onda mostradas en la Figura 6.87 se aplican a las entradas de un multiplexor de ocho entradas 74LS151. Dibujar la señal de salida *Y*.

SECCIÓN 6.9 Demultiplexores

29. Desarrollar el diagrama de tiempos completo (entradas y salidas) de un 74HC154 utilizado en una aplicación de demultiplexación en el que las entradas son las siguientes: las entradas de selección de datos toman, de forma repetitiva y secuencialmente, los valores generados por un

contador binario que comienza en 0000, y la entrada de datos es una cadena de datos serie, en BCD, que representan al número decimal 2468. El dígito menos significativo (8) es el primero de la secuencia, con el bit menos significativo en primer lugar, y deberá aparecer en los cuatro primeros bits de la salida.

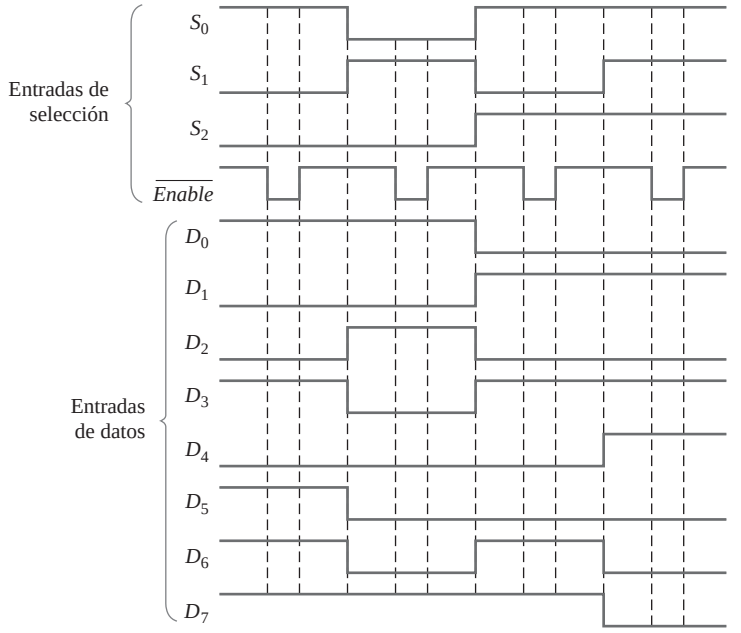


FIGURA 6.87

SECCIÓN 6.10 Generadores / Comprobadores de paridad

30. Se aplican las formas de onda de la Figura 6.88 al circuito de paridad de 4 bits. Determinar las señales de salida en función de las entradas. ¿Durante cuántos periodos de bit ocurre la paridad par y cómo se indica? El diagrama de tiempos incluye ocho periodos de bit.

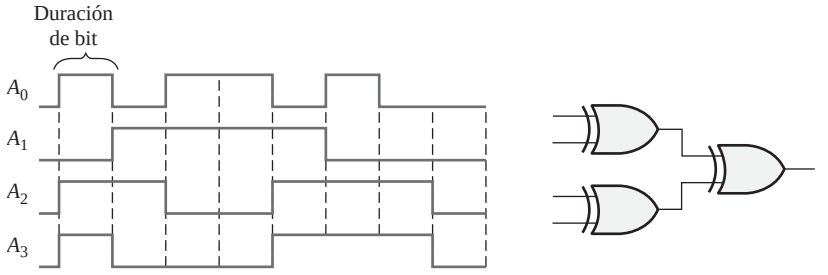


FIGURA 6.88

31. Determinar las salidas ΣImpar y ΣPar de un generador/comprobador de paridad 74LS280 de 9 bits, para las entradas de la Figura 6.89. Utilice la tabla de verdad de la Figura 6.59.

SECCIÓN 6.11 Localización de averías

32. El sumador completo de la Figura 6.90 se prueba bajo todas las condiciones de entrada posibles, con las señales de entrada indicadas. A partir de la observación de las señales Σ y C_{out}, ¿funciona correctamente? Si la respuesta es no, ¿cuál es la causa más probable de fallo?

33. Enumerar los posibles fallos de cada codificador/display de la Figura 6.91.

34. Desarrollar un procedimiento de pruebas sistemático para verificar el funcionamiento completo del codificador de teclado de la Figura 6.42.

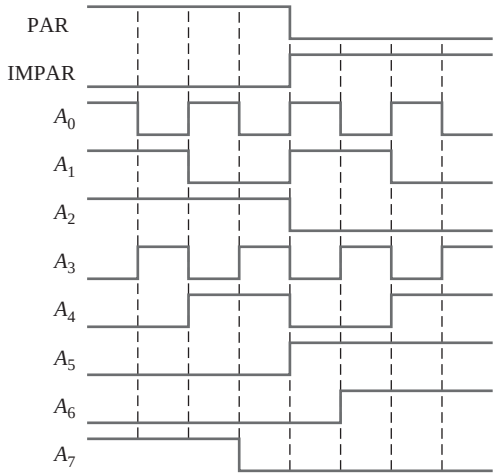


FIGURA 6.89

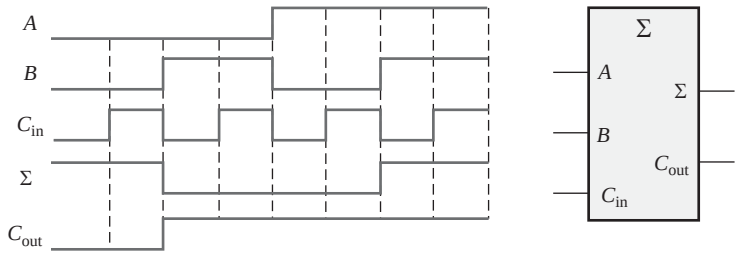


FIGURA 6.90

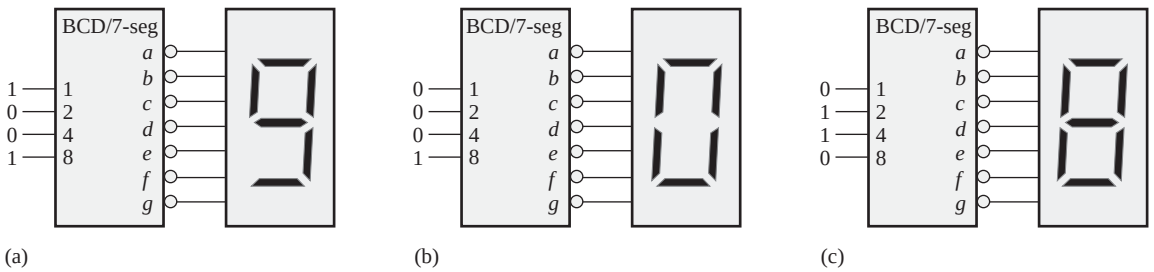


FIGURA 6.91

35. Hay que probar el convertidor BCD-binario formado por cuatro sumadores que se muestra en la Figura 6.92. En primer lugar, hay que verificar que el circuito convierte de BCD a binario. El procedimiento de prueba requiere la aplicación secuencial de números BCD, comenzando por 0_{10}, para comprobar que la salida binaria es la correcta. ¿Qué síntoma o síntomas aparecerían en las salidas binarias si ocurrieran cada uno de los siguientes fallos? ¿Cuál es el número BCD para el que se detecta por *primera vez* cada error?

(a) La entrada A_1 está en circuito abierto (sumador superior).

(b) C_{out} está en circuito abierto (sumador superior).

(c) La salida Σ_4 está cortocircuitada a masa (sumador superior).

(d) La salida 32 está cortocircuitada a masa (sumador inferior).

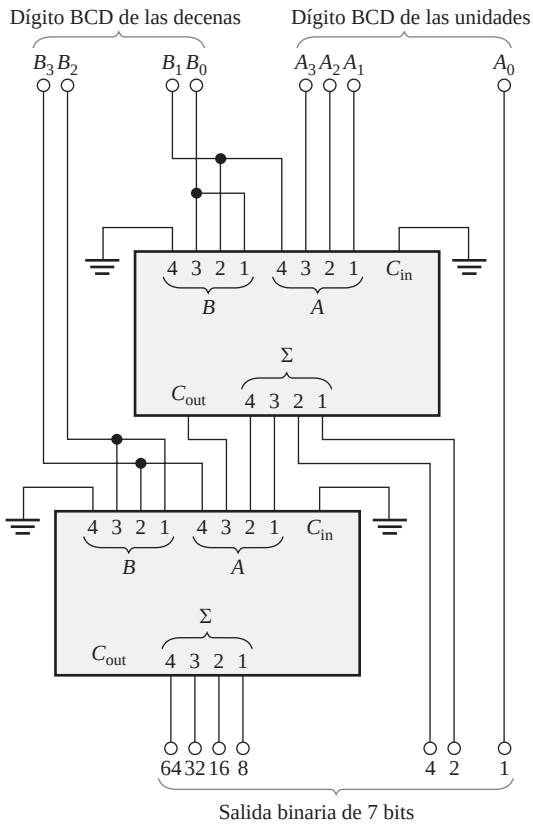


FIGURA 6.92

36. En el display con multiplexación de la Figura 6.52, determinar la causa (o las causas) más probable para cada uno de los siguientes síntomas:

(a) El display del dígito *B* (el más significativo) no se enciende.

(b) Ninguno de los displays de 7-segmentos se enciende.

(c) El segmento *f* de ambos displays aparece encendido siempre.

(d) Hay un parpadeo visible en los displays.

37. Desarrollar un procedimiento sistemático para probar exhaustivamente el CI selector de datos 74LS151.

38. Durante las pruebas del sistema de transmisión de datos de la Figura 6.60, se aplica un código a las entradas D_0 a D_6 que contiene un número impar de 1s. Se introduce deliberadamente un único bit erróneo en la línea de transmisión serie entre el multiplexor y el demultiplexor, pero el sistema no detecta el error (salida de error = 0). Tras algún tiempo de investigación, se verifican las entradas con el comprobador de paridad par y se encuentra que en D_0 a D_6 hay un número par de 1s, como se esperaba, y también se comprueba que el bit de paridad D_7 es 1. ¿Cuáles son las posibles razones de que el sistema no indique el error?

39. Describir de forma general cómo probaríamos el sistema de transmisión de datos de la Figura 6.60 y especificar un método de introducción de errores de paridad.

Aplicación a los sistemas digitales

40. El bloque de la lógica de salida del semáforo se implementa en el sistema usando lógica de función fija mediante un 74LS08 con puertas AND operando como puertas negativa-NOR.

Utilizar un 74LS00 (puertas NAND cuádruples) y cualquier otro dispositivo que sea necesario para generar salidas activas a nivel BAJO para las entradas dadas.

41. Implementar la lógica de salida del semáforo con el 74LS00 si se necesitan salidas activas a nivel BAJO.

Problemas especiales de diseño

42. Modificar el diseño del sistema de multiplexación del display de 7-segmentos de la Figura 6.52 para permitir visualizar dos dígitos adicionales.

43. Utilizando la Tabla 6.2, escribir las expresiones de suma de productos para Σ y C_{out} de un sumador completo. Utilizar un mapa de Karnaugh para minimizar las expresiones y luego implementarlas empleando inversores y lógica AND-OR. Indicar cómo se puede reemplazar la lógica AND-OR con selectores de datos 74LS151.

44. Implementar la función lógica especificada en la Tabla 6.12 utilizando un selector de datos 74LS151.

Entradas				Salida
A_3	A_2	A_1	A_0	Y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	1

TABLA 6.12

45. Utilizando dos de los módulos sumadores de 6 posiciones de la Figura 6.14, diseñar un sistema de votación de 12 posiciones.

46. El bloque sumador del sistema de control y recuento de pastillas de la Figura 6.93 realiza la suma del número binario de 8 bits del contador y del número binario de 16 bits del registro B. El resultado de la suma se almacena en el registro B. Utilizar circuitos 74LS283 para implementar esta función y dibujar un diagrama lógico completo que incluya la numeración de los pines. Revise el funcionamiento del sistema en el Capítulo 1.

47. Utilizar circuitos 74HC85 para implementar el bloque comparador del sistema de control y recuento de pastillas de la Figura 6.93 y dibujar un diagrama lógico completo que incluya la numeración de pines. El comparador compara el número binario de 8 bits (en realidad sólo se requieren siete bits) del convertidor BCD-binario con el número binario de 8 bits del contador.

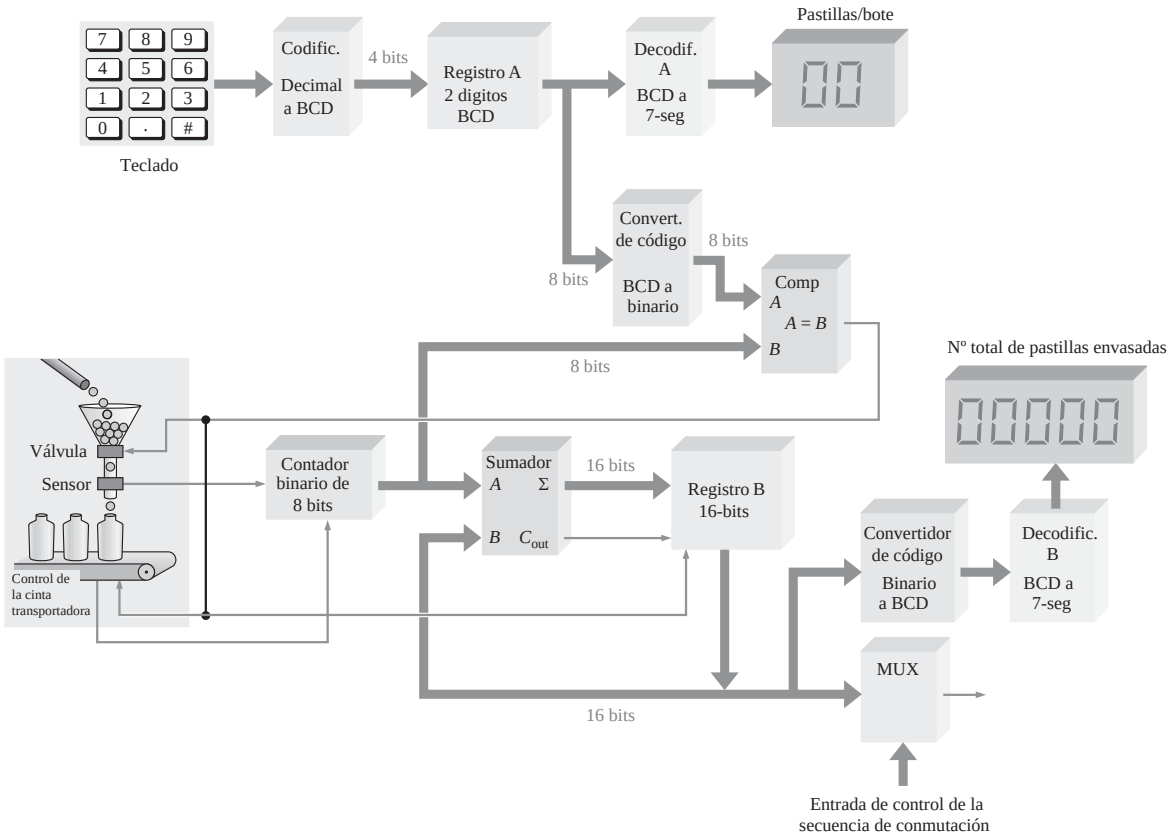


FIGURA 6.93

48. Se utilizan dos decodificadores BCD-7-segmentos en el sistema de control y recuento de la Figura 6.93. Uno de ellos se requiere para controlar el display de dos dígitos *pastillas/bote*, y el otro para controlar el display de 5 dígitos *número total de pastillas envasadas*. Utilizar circuitos 74LS47 para implementar cada decodificador y dibujar un diagrama lógico completo que incluya la numeración de pines.

49. El codificador que se muestra en el diagrama de bloques de la Figura 6.93 codifica cada pulsación de una tecla decimal y la convierte en BCD. Utilizar un 74HC147 para implementar esta función y dibujar un diagrama lógico completo que incluya la numeración de pines.

50. El sistema de la Figura 6.93 requiere dos convertidores de código. El convertidor BCD-binario convierte los dos números BCD de dos dígitos del registro *A* en un código binario de 8 bits (en realidad sólo se necesitan 7 bits dado que el bit más significativo siempre es 0). Utilizar los convertidores de código circuito integrado apropiados para implementar la función del convertidor BCD-binario, y dibujar un diagrama lógico completo que incluya la numeración de pines.

RESPUESTAS

REVISIONES DE CADA SECCIÓN

SECCIÓN 6.1 Sumadores básicos

1. **(a)** $\Sigma = 1$, $C_{out} = 0$ **(b)** $\Sigma = 0$, $C_{out} = 0$

(c) $\Sigma = 1$, $C_{out} = 0$ **(d)** $\Sigma = 0$, $C_{out} = 1$

2. $\Sigma = 1$, $C_{out} = 1$

SECCIÓN 6.2 Sumadores binarios en paralelo

1. $C_{out}\Sigma_4\Sigma_3\Sigma_2\Sigma_1 = 11001$

2. Se requieren tres 74LS283 para sumar dos números de 10 bits.

SECCIÓN 6.3 Sumadores de acarreo serie y acarreo anticipado

1. $C_g = 0$, $C_p = 1$

2. $C_{out} = 1$

SECCIÓN 6.4 Comparadores

1. $A > B = 1$, $A < B = 0$, $A = B = 0$ cuando $A = 1011$ y $B = 1010$

2. Comparador de la derecha: pin 7: $A < B = 1$; pin 6: $A = B = 0$; pin 5: $A > B = 0$

Comparador de la izquierda: pin 7: $A < B = 0$; pin 6: $A = B = 0$; pin 5: $A > B = 1$

SECCIÓN 6.5 Decodificadores

1. La salida 5 está activa cuando en las entradas se aplica 101.

2. Se utilizan cuatro 74HC154 para decodificar un número binario de 6 bits.

3. La salida activa a nivel BAJO controla el display de diodos LED en cátodo común.

SECCIÓN 6.6 Codificadores

1. (a) $A_0 = 1$, $A_1 = 1$, $A_2 = 0$, $A_3 = 1$

(b) No, no es un código BCD válido.

(c) Sólo puede estar activada una entrada para obtener una salida válida.

2. **(a)** $\overline{A}_3 = 0, \overline{A}_2 = 1, \overline{A}_1 = 1, \overline{A}_0 = 1$

(b) La salida es 0111, que es el complemento de 1000 (8).

SECCIÓN 6.7 Convertidores de código

1. 10000101 (BCD) = 1010101_2

2. Un convertidor binario-código Gray de ocho bits está formado por siete puertas OR-exclusiva en una disposición como la de la Figura 6.43.

SECCIÓN 6.8 Multiplexores (selectores de datos)

1. La salida es 0.

2. **(a)** 74LS157: Selector de datos cuádruple de 2 entradas.

(b) 74LS151: Selector de datos de 8 entradas.

3. La salida de datos alterna entre un nivel BAJO y un nivel ALTO a medida que las entradas de selección de datos cambian, secuencialmente, entre los distintos estados binarios.

4. **(a)** El 74HC157A multiplexa los dos códigos BCD al decodificador de 7-segmentos.

(b) El 74LS47 decodifica el código BCD para excitar el display.

(c) El 74LS139 activa los displays de 7-segmentos alternativamente.

SECCIÓN 6.9 Demultiplexores

1. Se puede utilizar un decodificador como multiplexor, utilizando las líneas de entrada como entradas de selección de datos y una línea de activación como entrada de datos.

2. Las salidas están todas a nivel ALTO excepto D_{10}, que es un nivel BAJO.

SECCIÓN 6.10 Comprobadores/generadores de paridad

1. **(a)** Paridad par: 1110100
   (b) Paridad par: 001100011
2. **(a)** Paridad impar: 11010101
   (b) 11000001
3. **(a)** El código es correcto, cuatro 1s.
   (b) El código es erróneo, siete 1s.

SECCIÓN 6.11 Localización de averías

1. Un *glitch* es un pico de tensión de muy corta duración (generalmente indeseado).
2. Los *glitches* los originan los estados de transición.
3. Validar (*strobing*) consiste en la activación de un dispositivo durante un período de tiempo especificado, mientras el dispositivo no se encuentra en un estado de transición

PROBLEMAS RELACIONADOS

6.1 $\Sigma = 1$, $C_{out} = 1$

6.2 $\Sigma_1 = 0$, $\Sigma_2 = 0$, $\Sigma_3 = 1$, $\Sigma_4 = 1$

6.3 1011 + 1010 = 10101

6.4. *Véase* la Figura 6.94.

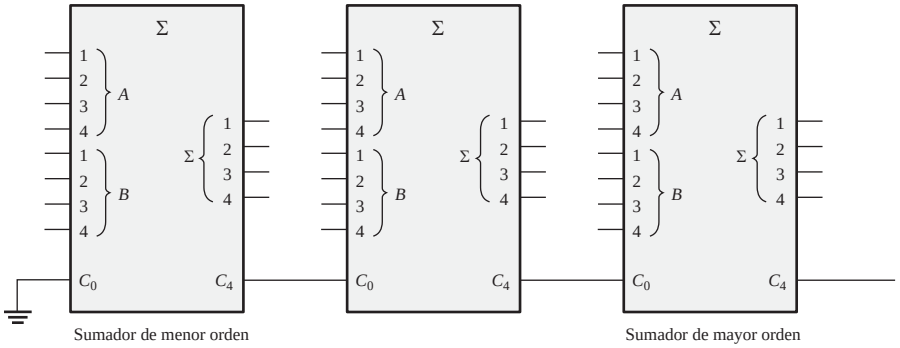


FIGURA 6.94

6.5. *Véase* la Figura 6.95.

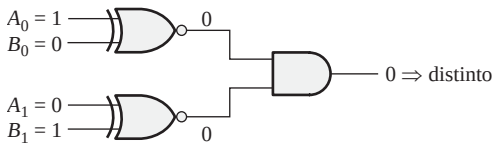


FIGURA 6.95

6.6 $A > B = 0$, $A = B = 0$, $A < B = 1$

6.7. *Véase* la Figura 6.96.

6.8. *Véase* la Figura 6.97.

6.9. Salida 22

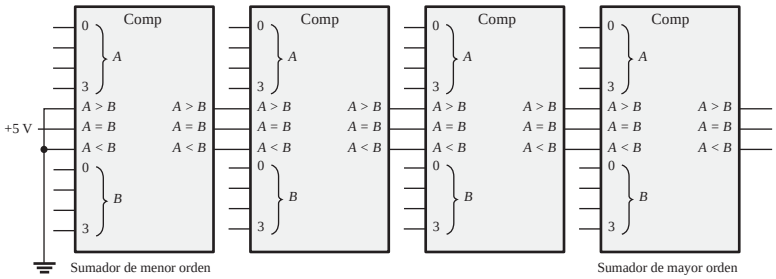


FIGURA 6.96

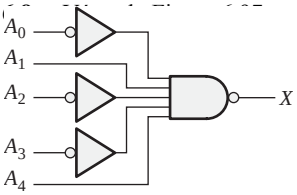


FIGURA 6.97

6.10. *Véase* la Figura 6.98.

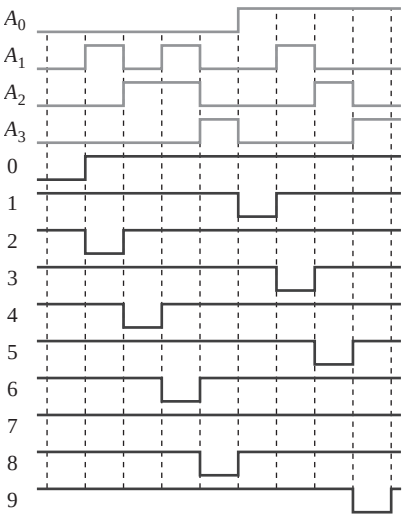


FIGURA 6.98

6.11. Todas las entradas a nivel BAJO: $\bar{A}_0 = 0, \bar{A}_1 = 1, \bar{A}_2 = 1, \bar{A}_3 = 0$
Todas las entradas a nivel ALTO: todas las salidas a nivel ALTO.

6.12 BCD 01000001

	00000001	1
	00101000	40
Binario	00101001	41

6.13 Siete puertas OR-exclusiva.

6.14. *Véase* la Figura 6.99.

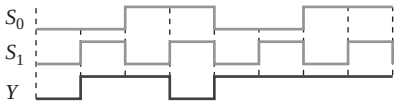


FIGURA 6.99

6.15. D_0: $S_3 = 0$, $S_2 = 0$, $S_1 = 0$, $S_0 = 0$
D_4: $S_3 = 0$, $S_2 = 1$, $S_1 = 0$, $S_0 = 0$
D_8: $S_3 = 1$, $S_2 = 0$, $S_1 = 0$, $S_0 = 0$
D_{13}: $S_3 = 1$, $S_2 = 1$, $S_1 = 0$, $S_0 = 1$

6.16. *Véase* la Figura 6.100.

6.17. *Véase* la Figura 6.101.

6.18. *Véase* la Figura 6.102.

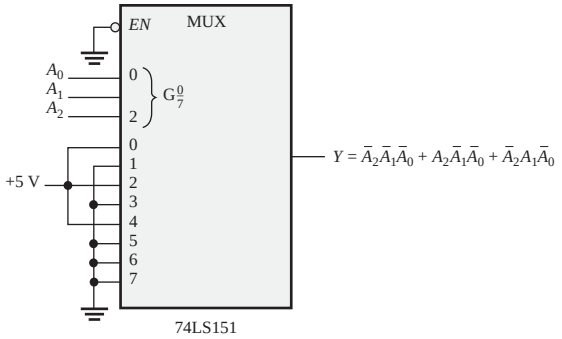


FIGURA 6.100

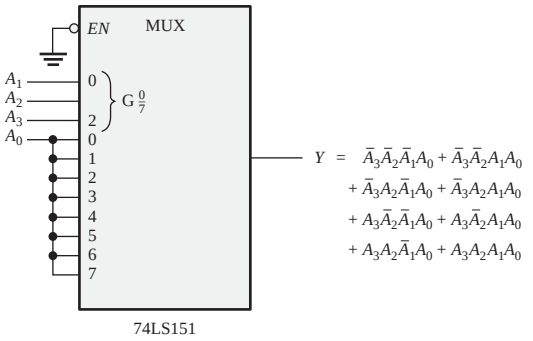


FIGURA 6.101

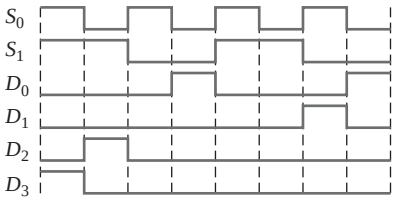


FIGURA 6.102

AUTOTEST

1. (a) **2.** (b) **3.** (c) **4.** (a) **5.** (d) **6.** (b) **7.** (c)
8. (b) **9.** (a) **10.** (d) **11.** (c) **12.** (f)