Fundamentos de diseño lógico y de computadoras

Tercera Edición

M. MORRIS MANO
California State University, Los Ángeles

CHARLES R. KIME
University of Wisconsin, Madison

Traducción
José Antonio Herrera Camacho
Profesor Titular de Escuela Universitaria
Universidad Politécnica de Madrid

Martina Eckert
Dra. Ingeniería en Telecomunicación
Universidad Politécnica de Madrid

Beatriz Valcuende Lozano
Ingeniera Técnica en Telefonía y Transmisión de Datos
Universidad Politécnica de Madrid

Revisión técnica
José Antonio Herrera Camacho
Profesor Titular de Escuela Universitaria
Universidad Politécnica de Madrid

PEARSON
Prentice Hall

Madrid • México • Santafé de Bogotá • Buenos Aires • Caracas • Lima • Montevideo
San Juan • San José • Santiago • São Paulo • White Plains

CAPÍTULO

2

CIRCUITOS LÓGICOS COMBINACIONALES

En este capítulo estudiaremos las puertas, los elementos lógicos más sencillos usados en los sistemas digitales. Además, aprenderemos las técnicas matemáticas usadas en el diseño de circuitos con esas puertas y cómo diseñar circuitos eficientes en coste. Estas técnicas, que son fundamentales para diseñar casi todos los circuitos digitales, se basan en el Álgebra de Boole. Un aspecto del diseño es evitar circuitos innecesarios y costes excesivos, una meta que se cumple mediante una técnica llamada optimización. Los Mapas de Karnaugh proporcionan un método gráfico para mejorar el entendimiento de optimización y solucionar pequeños problemas de circuitos con «dos niveles». Se introducen los métodos más generales de optimización para circuitos con más de dos niveles. Se discuten los tipos de puertas lógicas características en la realización de los circuitos integrados actuales. Se presentan las puertas OR y NOR exclusiva, junto con las técnicas algebraicas asociadas.

En términos del diagrama del principio del Capítulo 1, los conceptos de este capítulo se pueden aplicar a la mayor parte de la computadora genérica. Las excepciones a esto son circuitos que son, principalmente, memorias, como cachés y RAM, y circuitos analógicos en el monitor y el controlador del disco duro. Sin embargo, con su uso por todas partes del diseño de la mayor parte de la computadora, lo que estudiaremos en este capítulo es fundamental para un entendimiento profundo de las computadoras y los sistemas digitales, y cómo están diseñados.

2-1 LÓGICA BINARIA Y PUERTAS

Los circuitos digitales son componentes de hardware que manipulan información binaria. Los circuitos se realizan con transistores e interconexiones en complejos dispositivos de semiconductores llamados *circuitos integrados*. A cada circuito básico se le denomina *puerta lógica*. Por simplicidad en el diseño, modelamos los circuitos electrónicos basados en transistores como puertas lógicas. Así, el diseñador no tiene que preocuparse por la electrónica interna de cada una de las individuales, sino solamente por sus propiedades lógicas externas. Cada puerta realiza una operación lógica específica. Las salidas de las puertas se aplican a las entradas de otras puertas para formar un circuito digital.

Para describir las propiedades operacionales de los circuitos digitales es necesario introducir una notación matemática que especifica la operación de cada puerta y que puede ser usada para analizar y diseñar circuitos. Este sistema de lógica binaria es una clase de sistema matemático que se denomina *Álgebra de Boole*. El nombre es en honor al matemático inglés George Boole, quien publicó un libro en 1854 introduciendo la teoría matemática de la lógica. El álgebra específica de Boole que estudiaremos se usa para describir la interconexión de las puertas digitales y para diseñar circuitos lógicos a través del uso de expresiones booleanas. Primero introducimos el concepto de lógica binaria e indicamos su relación con las puertas digitales y las señales binarias. Después presentamos las propiedades del Álgebra de Boole, junto con otros conceptos y métodos útiles en el diseño de circuitos lógicos.

Lógica binaria

La lógica binaria trabaja con variables binarias, que pueden tomar dos valores discretos, y con las operaciones lógicas matemáticas aplicadas a esas variables. A los dos valores que pueden tomar las variables se les pueden llamar por diferentes nombres, como se mencionó en la Sección 1-1, pero para nuestro propósito, es conveniente pensar en términos de valores binarios y asignar 1 o 0 a cada variable. En la primera parte de este libro, se designan a las variables con las letras del alfabeto, como *A*, *B*, *C*, *X*, *Y*, y *Z*. Más tarde se extiende esta notación para incluir cadenas de letras, números y caracteres especiales. Asociados con las variables binarias hay tres operaciones lógicas llamadas AND, OR y NOT:

1. AND. Esta operación está representada por un punto o por la ausencia de un operador. Por ejemplo, $Z = X \cdot Y$ o $Z = XY$ se lee «*Z* es igual a *X* AND *Y*». La operación lógica AND se interpreta de manera que $Z = 1$ si y solamente si $X = 1$ e $Y = 1$; de lo contrario es $Z = 0$. (Recuerde que *X*, *Y* y *Z* son variables binarias y solamente pueden tener los valores 1 o 0.)
2. OR. Esta operación se representa por el símbolo «más». Por ejemplo, $Z = X + Y$ se lee «*Z* es igual a *X* OR *Y*», lo que significa que $Z = 1$ si $X = 1$ o si $Y = 1$, o si los dos $X = 1$ e $Y = 1$. $Z = 0$ si y solamente si $X = 0$ e $Y = 0$.
3. NOT. Esta operación está representada por una barra encima de la variable. Por ejemplo, $Z = \bar{X}$ se lee «*Z* es igual a NOT *X*,» lo que significa que *Z* es lo que *X* no es. En otras palabras, si $X = 1$, entonces $Z = 0$, pero si $X = 0$, entonces $Z = 1$. A la operación NOT se le denomina también como operación complementaria, ya que cambia un 1 a 0 y un 0 a 1.

Lógica binaria se parece a la aritmética binaria, y las operaciones AND y OR se parecen a la multiplicación y la suma, respectivamente. Por eso los símbolos usados para la AND y la OR

son los mismos que los que se usan para la multiplicación y la suma. Sin embargo, no se debe confundir la lógica binaria con la aritmética binaria. Uno debería darse cuenta de que una variable aritmética define a un número que se puede componer de muchos dígitos, mientras una variable lógica siempre es 1 o 0. Las siguientes ecuaciones definen la operación lógica OR:

$$0 + 0 = 0$$

$$0 + 1 = 1$$

$$1 + 0 = 1$$

$$1 + 1 = 1$$

Esto se parece a la suma binaria, excepto para la última operación. En la lógica binaria, tenemos que $1 + 1 = 1$ (léase «uno o uno es igual a uno»), pero en la aritmética binaria tenemos $1 + 1 = 10$ (léase «uno más uno es igual a dos»). Para evitar ambigüedad, a veces se usa el símbolo $\vee$ para la operación OR en vez del símbolo $+$. Pero mientras no se mezclen operaciones aritméticas y lógicas, en cada parte se puede usar el símbolo $+$ con su propio significado independiente.

Las siguientes ecuaciones definen la operación lógica AND:

$$0 \cdot 0 = 0$$

$$0 \cdot 1 = 0$$

$$1 \cdot 0 = 0$$

$$1 \cdot 1 = 1$$

Esta operación es idéntica a la multiplicación binaria, con tal de que se use solamente un único bit. Los símbolos alternativos para el $\cdot$ de la AND y el $+$ de la OR, son los símbolos $\wedge$ y $\vee$, respectivamente, que representan operaciones conjuntivas y disyuntivas en cálculos proposicionales.

Para cada combinación de los valores de variables binarias como *X* e *Y*, hay un valor de *Z* especificado por la definición de la operación lógica. Las definiciones pueden ser enumeradas de forma compacta en una tabla de verdad. Una *tabla de verdad* para una operación es una tabla de combinaciones de las variables binarias que muestran la relación entre los valores que toman las variables y los valores del resultado de la operación. Las tablas de verdad para las operaciones AND, OR y NOT se muestran en la Tabla 2-1. Las tablas enumeran todas las combinaciones posibles de valores para dos variables y el resultado de la operación. Demuestran claramente la definición de las tres operaciones.

Puertas lógicas

Las puertas lógicas son circuitos electrónicos que operan con una o más señales de entrada para producir una señal de salida. Las señales eléctricas como voltaje o corriente existen en todas las partes de un sistema digital en cada uno de los dos valores definidos. Los circuitos que operan con voltajes responden a dos rangos separados de voltajes que representan una variable binaria igual a un 1 lógico o a un 0 lógico, como se ilustra en la Figura 1-1. Los terminales de entrada

□ **TABLA 2-1**
Tablas de verdad para las tres operaciones lógicas básicas

AND

X	Y	$Z = X \cdot Y$
0	0	0
0	1	0
1	0	0
1	1	1

OR

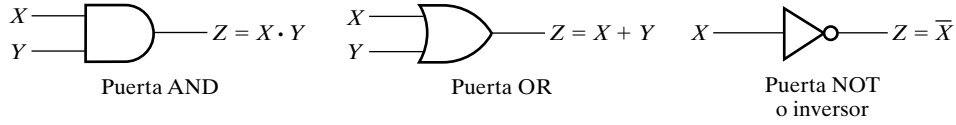
X	Y	$Z = X + Y$
0	0	0
0	1	1
1	0	1
1	1	1

NOT

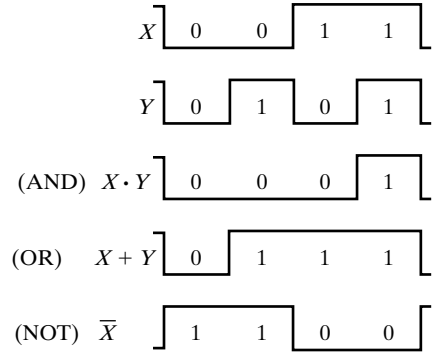
X	$Z = \overline{X}$
0	1
1	0

de las puertas lógicas aceptan señales binarias dentro del rango permitido y responden a los terminales de salida con señales binarias que caen dentro de un rango específico. Las regiones intermedias entre los rangos permitidos de la figura se cruzan solamente durante los cambios de 1 a 0 o de 0 a 1. A estos cambios se le llaman transiciones, y las regiones intermediarias se llaman regiones de tránsito.

Los símbolos gráficos usados para designar los tres tipos de puertas —AND, OR y NOT— se muestran en la Figura 2-1(a). Las puertas son circuitos electrónicos que producen los equivalentes a las señales de salida de 1 lógico y 0 lógico, de acuerdo con sus respectivas tablas de verdad, si se aplican el equivalente de las señales de entrada de 1 lógico y 0 lógico. Las dos señales de entrada X e Y de las puertas AND y OR toman una de cuatro combinaciones: 00, 01, 10, o 11. Estas señales de entrada se muestran en los diagramas de tiempos de la Figura 2-1(b), junto con los diagramas de tiempos de las señales de salida correspondientes a cada tipo de puerta. El eje horizontal de un diagrama de tiempos representa el tiempo, y el eje vertical muestra una señal cuando cambia entre los dos posibles niveles de voltaje. El nivel bajo representa

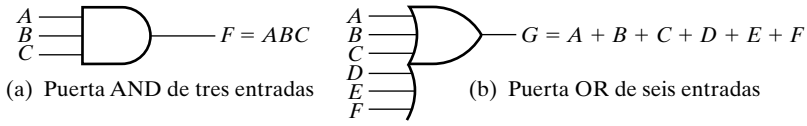


(a) Símbolo gráfico



(b) Diagrama de tiempos

□ **FIGURA 2-1**
Puertas lógicas digitales



(a) Puerta AND de tres entradas (b) Puerta OR de seis entradas

□ **FIGURA 2-2**
Puertas con más que dos entradas

el 0 lógico y el nivel alto representa el 1 lógico. La puerta AND se corresponde con una señal de salida a 1 lógico cuando las dos señales de entrada son un 1 lógico. La puerta OR responde con una señal de salida a 1 lógico si una de las señales de entrada es un 1 lógico. A la puerta NOT se le llama frecuentemente como *inversor*. La razón para este nombre es evidente por su respuesta en el diagrama de tiempos. La señal lógica de la salida es una versión invertida de la señal lógica de la entrada *X*.

Las puertas AND y OR pueden tener más de dos entradas. En la Figura 2-2 se muestra una puerta AND con tres entradas y una puerta OR con seis entradas. La puerta AND de tres entradas responde con una salida a 1 lógico si las tres entradas son 1 lógico. La salida es un 0 lógico si alguna de las entradas es un 0 lógico. La puerta OR de seis entradas responde con un lógico 1 si alguna de las entradas es un 1 lógico; su salida será un 0 lógico solamente cuando todas las entradas son 0 lógico.

2-2 ÁLGEBRA DE BOOLE

El Álgebra de Boole que presentamos es un álgebra que trata con variables binarias y operaciones lógicas. Las variables se indican con las letras del alfabeto y las operaciones básicas son AND, OR y NOT (complemento). Una expresión *booleana* es una expresión algebraica formada por variables binarias, las constantes 0 y 1, los símbolos de operación lógicos y paréntesis. Una función booleana se puede describir con una ecuación booleana que se compone de una variable binaria que identifica la función seguida por un símbolo de igualdad y una expresión booleana. Opcionalmente, al identificador le pueden seguir paréntesis que rodean a una lista de las variables de la función separadas por comas. Una *función booleana con única salida* se tabula a partir de cada combinación posible de valores 0 y 1 entre las variables de la función al valor 0 o 1. Una *función booleana con salida múltiple* se tabula a partir de cada combinación posible de valores 0 y 1 entre las variables de la función a combinaciones de 0 y 1 entre las salidas de la función. Considere un ejemplo de una ecuación booleana que representa a la función *F*:

$$F(X, Y, Z) = X + \bar{Y}Z$$

A las dos partes de la expresión, *X* y $\bar{Y}Z$, se le llaman *términos* de la expresión de *F*. La función *F* es igual a 1 si el término *X* es igual a 1 o si el término $\bar{Y}Z$ es igual a 1 (es decir, ambos $\bar{Y}$ y *Z* son iguales a 1). De otro modo, *F* es igual a 0. La operación complemento determina que si $\bar{Y} = 1$, *Y* tiene que ser igual a 0. Por tanto, podemos decir que $F = 1$ si $X = 1$ o si $Y = 0$ y $Z = 1$. Una ecuación booleana expresa la relación lógica entre variables binarias. Se evalúa determinando el valor binario de la expresión para todas las combinaciones posibles de valores para las variables.

Se puede representar una función booleana con una tabla de verdad. Una *tabla de verdad* para una función es una lista de todas las combinaciones de 1 y 0 que se pueden asignar a las variables binarias y una lista que indica el valor de la función para cada combinación binaria.

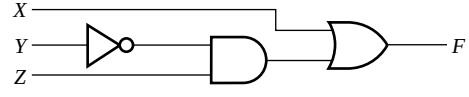
□ **TABLA 2-2**
Tabla de verdad de la función $F = X + \bar{Y}Z$

X	*Y*	*Z*	*F*
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Las tablas de verdad para las operaciones lógicas de la Tabla 2-1 son casos especiales de tablas de verdad para funciones. El número de filas en una tabla de verdad es 2^n, donde n es el número de variables de la función. Las combinaciones binarias para la tabla de verdad son los números binarios de n-bit que corresponden a la cuenta en decimal de 0 a $2^n - 1$. La Tabla 2-2 muestra la tabla de verdad de la función $F = X + \bar{Y}Z$. Hay ocho posibles combinaciones binarias que asignan bits a las tres variables *X*, *Y* y *Z*. La columna etiquetada como *F* contiene 0 o 1 para cada una de las combinaciones. La tabla indica que la función es igual a 1 si $X = 1$ y si $Y = 0$ y $Z = 1$. De otro modo, la función es igual a 0.

Una expresión algebraica para una función booleana puede transformarse en un diagrama de un circuito compuesto por puertas lógicas que efectúan la función. El diagrama lógico del circuito para la función *F* se muestra en la Figura 2-3. Un inversor en la entrada *Y* genera el complemento, $\bar{Y}$. Una puerta AND opera con $\bar{Y}$ y *Z*, y una puerta OR combina *X* y $\bar{Y}Z$. En los diagramas lógicos de los circuitos, las variables de la función *F* se toman como entradas del circuito, y la variable binaria *F* se toma como salida del circuito. Si el circuito tiene una única salida, *F* es una función de salida única. Si el circuito tiene múltiples salidas, la función *F* es una función de salida múltiple que requiere de múltiples ecuaciones para representar sus salidas. Las puertas del circuito están interconectadas por hilos que llevan las señales lógicas. A los circuitos lógicos de este tipo se les llama *circuitos lógicos combinacionales*, porque las variables están «combinadas» por las operaciones lógicas. Esto es lo contrario a la lógica secuencial, que se trata en el Capítulo 6, donde se almacenan y combinan las variables en función del tiempo.

Una función booleana se puede representar en una tabla de verdad de una sola manera. No obstante, si la función tiene forma de ecuación algebraica, puede ser expresada de diferentes maneras. La expresión particular usada para representar la función determina la interconexión de las puertas en el diagrama lógico del circuito. Manipulando una expresión booleana conforme con reglas algebraicas booleanas, muchas veces es posible obtener una expresión más simple para la misma función. Ésta función más simple reduce el número de puertas en el circuito y el número de entradas de las puertas. Para ver como se logra esto, es necesario estudiar primero las reglas básicas del Álgebra de Boole.



□ **FIGURA 2-3**
Diagrama lógico de circuito para $F = X + \bar{Y}Z$

Identidades básicas del Álgebra de Boole

En la Tabla 2-3 se enumeran las identidades básicas del Álgebra de Boole. La notación está simplificada omitiendo el símbolo de la AND siempre que no lleve a ninguna confusión. Las nueve primeras identidades indican la relación entre una única variable X, su complemento $\overline{X}$, y las constantes binarias 0 y 1. Las siguientes cinco identidades, de la 10 a la 14, son las mismas que en el álgebra ordinaria. Las tres últimas, de la 15 a la 17, no se usan en el álgebra ordinaria, pero son útiles para manipular expresiones booleanas.

Las reglas básicas enumeradas en la tabla han sido colocadas en dos columnas que demuestran la propiedad dual del Álgebra de Boole. El dual de una expresión algebraica se obtiene intercambiando las operaciones OR y AND y reemplazando los 1 por 0 y los 0 por 1. Una ecuación en una columna de la tabla se puede obtener de la ecuación correspondiente de la otra columna usando el dual de las expresiones en ambos lados del símbolo de igualdad. Por ejemplo, la relación 2 es la dual de la relación 1 porque la OR ha sido reemplazada por la AND y el 0 por el 1. Es importante darse cuenta de que, en la mayoría de las veces, el dual de la expresión no es igual que la expresión original, de manera que una expresión normalmente no puede ser reemplazada por su dual.

Las nueve identidades que involucran a una única variable se pueden verificar sencillamente sustituyendo cada uno de los posibles valores de X. Por ejemplo, para mostrar que $X + 0 = X$, sea $X = 0$ para obtener $0 + 0 = 0$, y después sea $X = 1$ para obtener $1 + 0 = 1$. Ambas ecuaciones son verdad conforme a la definición de la operación lógica OR. Cada expresión puede ser sustituida por la variable X en todas las ecuaciones booleanas enumeradas en la tabla. Así, en la identidad 3 y con $X = AB + C$, obtenemos

$$AB + C + 1 = 1$$

Véase que la identidad 9 expresa que la doble complementación restaura el valor original de la variable. Así, si X = 0, entonces $\overline{X} = 1$ y $\overline{\overline{X}} = 0 = X$.

Las identidades 10 y 11, las leyes conmutativas, expresan que el orden en que se escriben las variables no afecta el resultado usando las operaciones OR y AND. Las identidades 12 y 13, las leyes asociativas, expresan que el resultado aplicando una operación sobre tres variables es

□ **TABLA 2-3**
Identidades básicas del Álgebra de Boole

1.	$X = 0 = X$	2.	$X \cdot 1\ X$	
3.	$X + 1 = 1$	4.	$X \cdot 0 = 0$	
5.	$X + X = X$	6.	$X \cdot X = X$	
7.	$X + \overline{X} = 1$	8.	$X \cdot \overline{X} = 0$	
9.	$\overline{\overline{X}} = X$			
10.	$X + Y = Y + X$	11.	$XY = YX$	Conmutativa
12.	$X + (Y + Z) = (X + Y) + Z$	13.	$X(YZ) = (XY)Z$	Asociativa
14.	$X(Y + Z) = XY + XZ$	15.	$X + YZ = (X + Y)(X + Z)$	Distributiva
16.	$\overline{X + Y} = \overline{X} \cdot \overline{Y}$	17.	$\overline{X \cdot Y} = \overline{X} + \overline{Y}$	De DeMorgan

independiente del orden en que se apliquen, y asimismo, los paréntesis se puede quitar como en el siguiente caso:

$$X + (Y + Z) = (X + Y) + Z = X + Y + Z$$

$$X(YZ) = (XY)Z = XYZ$$

Estas dos leyes y la primera ley distributiva, Identidad 14, son bien conocidas del álgebra ordinaria, por eso no deberían causar ninguna dificultad. La segunda ley distributiva, dada por Identidad 15, es el dual de la ley ordinaria distributiva y no se basa en el álgebra ordinaria. Como se ilustró anteriormente, se puede reemplazar cada variable de una identidad por una expresión booleana, y la identidad es todavía válida. Así, considere la expresión $(A + B)(A + CD)$. Poniendo $X = A$, $Y = B$ y $Z = CD$, y aplicando la segunda ley distributiva, obtenemos

$$(A + B)(A + CD) = A + BCD$$

A las dos últimas identidades de la Tabla 2-3,

$$\overline{X + Y} = \bar{X} \cdot \bar{Y} \text{ y } \overline{X \cdot Y} = \bar{X} + \bar{Y}$$

se les denomina como Teorema de DeMorgan. Es un teorema muy importante y se usa para obtener el complemento de una expresión y de la función correspondiente. El Teorema de DeMorgan se puede ilustrar mediante tablas de verdad que asignan todos los valores binarios posibles a X e Y. La Tabla 2-4 muestra dos tablas de verdad que verifican la primera parte del Teorema de DeMorgan. En A, evaluamos $\overline{X + Y}$ para todos los valores posibles de X e Y. Esto se hace primero evaluando $X + Y$ y después calculando el complemento. En B, evaluamos $\bar{X}$ e $\bar{Y}$ y después las conectamos con una operación AND. El resultado es el mismo para las cuatro combinaciones binarias de X e Y, que verifican la identidad de la ecuación.

Véase el orden en que se realizan las operaciones al evaluar una expresión. En la parte B de la tabla, se evalúan primero los complementos de las variables particulares, seguida de la operación AND, justo como en el álgebra ordinaria con la multiplicación y la suma. En la parte A, se evalúa primero la operación OR. Después, notando que el complemento de una expresión como $X + Y$ se considera como NOT $(X + Y)$, evaluamos la expresión de dentro de los paréntesis y tomamos el complemento del resultado. Es usual excluir los paréntesis calculando el complemento de una expresión, y a la barra encima de una expresión entera la une. Así, $(X + Y)$ se expresa como $\overline{X + Y}$ cuando se indica el complemento de $X + Y$.

El Teorema de DeMorgan puede ser extendido a tres o más variables. El Teorema General de DeMorgan se puede expresar como

$$\overline{X_1 + X_2 + ... + X_n} = \bar{X}_1\bar{X}_2...\bar{X}_n$$

$$\overline{X_1X_2...X_n} = \bar{X}_1 + \bar{X}_2 + ... + \bar{X}_n$$

□ **TABLA 2-4**
Tablas de verdad para verificar el teorema de DeMorgan

A) X	Y	$X + Y$	$\overline{X + Y}$	**B)** X	Y	$\bar{X}$	$\bar{Y}$	$\bar{X} \cdot \bar{Y}$
0	0	0	1	0	0	1	1	1
0	1	1	0	0	1	1	0	0
1	0	1	0	1	0	0	1	0
1	1	1	0	1	1	0	0	0

Observe que la operación lógica cambia de OR a AND o de AND a OR. Además, se elimina el complemento de la expresión entera y se coloca encima de cada variable. Por ejemplo,

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

Manipulación algebraica

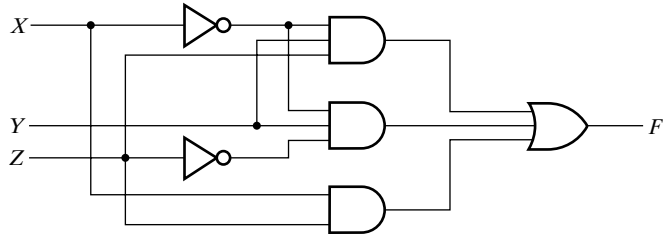
El Álgebra de Boole es un instrumento muy útil para simplificar circuitos digitales. Considere, por ejemplo, la función booleana representada por

$$F = \bar{X}YZ + \bar{X}Y\bar{Z} + XZ$$

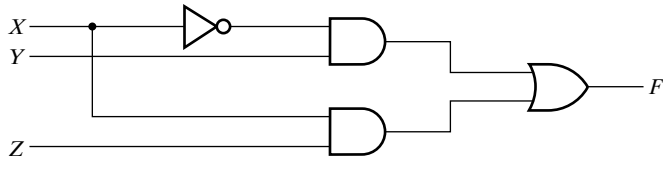
La implementación de ésta ecuación con puertas lógicas se muestra en la Figura 2-4(a). A las variables de entrada X y Z se le ha realizado el complemento con inversores para obtener $\bar{X}$ y $\bar{Z}$. Los tres términos de la expresión se realizan con tres puertas AND. La puerta OR forma la OR lógica de tres de los términos. Considere ahora una simplificación de la expresión para F aplicando algunas de las identidades listadas en la Tabla 2-3:

$$\begin{aligned} F &= \bar{X}YZ + \bar{X}Y\bar{Z} + XZ \\ &= \bar{X}Y(Z + \bar{Z}) + XZ && \text{con la identidad 14} \\ &= \bar{X}Y \cdot 1 + XZ && \text{con la identidad 7} \\ &= \bar{X}Y + XZ && \text{con la identidad 2} \end{aligned}$$

La expresión se reduce a sólo dos términos y puede ser realizada con puertas según se muestra en la Figura 2-4(b). Es obvio que el circuito de (b) es más simple que el de (a), ahora, ambos realizan la misma función. Es posible usar una tabla de verdad para verificar que dos implementaciones son equivalentes. Ésto se muestra en la Tabla 2-5. Como se expresa en la Figura 2-4(a), la función es igual a 1 si $X = 0$, $Y = 1$, y $Z = 1$; si $X = 0$, $Y = 1$, y $Z = 0$; o si X y Z son ambas 1. Esto produce los cuatro 1 para F en la parte (a) de la tabla. Como se expresa en la Figura 2-4(b), la función es igual a 1 si $X = 0$ e $Y = 1$ o si $X = 1$ y $Z = 1$. Esto produce los mismos



(a) $F = \bar{X}YZ + \bar{X}Y\bar{Z} + XZ$



(b) $F = \bar{X}Y + XZ$

□ **FIGURA 2-4**
Implementación de funciones booleanas con puertas

□ **TABLA 2-5**
Tabla de verdad para la función booleana

X	*Y*	*Z*	(a) *F*	(b) *F*
0	0	0	0	0
0	0	1	0	0
0	1	0	1	1
0	1	1	1	1
1	0	0	0	0
1	0	1	1	1
1	1	0	0	0
1	1	1	1	1

cuatro 1 en la parte (b) de la tabla. Como ambas expresiones producen las mismas tablas de verdad, se dice que son equivalentes. Por eso, los dos circuitos tienen las mismas salidas para todas las combinaciones binarias posibles de las tres variables de entrada. Cada circuito realiza la misma función, pero se prefiere la de menor número de puertas porque requiere menos componentes.

Si se implementa una ecuación booleana con puertas lógicas, cada término requiere una puerta, y cada variable dentro del término indica una entrada para la puerta. Definimos un *literal* como una variable única dentro de un término que puede estar complementado o no. La expresión para la función de la Figura 2-4(a) tiene tres términos y ocho literales; la de la Figura 2-4(b) tiene dos términos y cuatro literales. Reduciendo el número de términos, el número de literales, o ambos en una expresión booleana, muchas veces es posible obtener un circuito más sencillo. Se aplica el Álgebra de Boole para reducir una expresión con el fin de obtener un circuito más sencillo. Para funciones muy complejas es muy difícil encontrar la mejor expresión basada en sumas de términos y literales, aunque se usen programas de computadora. Ciertos métodos, sin embargo, para reducir expresiones, se incluyen frecuentemente en las herramientas por computadora para sintetizar circuitos lógicos. Estos métodos pueden obtener buenas soluciones, si no las mejores. El único método manual para el caso general es el procedimiento de intentar y probar a emplear las relaciones básicas y otras manipulaciones que uno va conociendo bien con el uso. Los siguientes ejemplos usan las identidades de la Tabla 2-3 para ilustrar algunas posibilidades:

1. $X + XY = X(1 + Y) = X$
2. $XY + X\bar{Y} = X(Y + \bar{Y}) = X$
3. $X + \bar{X}Y = (X + \bar{X})(X + Y) = X + Y$

Véase que el paso intermedio $X = X \cdot 1$ se ha omitido cuando se saca el factor X en la ecuación 1. La relación $1 + Y = 1$ es útil para eliminar términos redundantes, como se hace con el término XY en esta misma ecuación. La relación $Y + \bar{Y} = 1$ es útil para combinar dos términos, como se hace en la ecuación 2. Los dos términos combinados tienen que ser idénticos excepto en una variable, y esa variable tiene que estar complementada en un término y no complementada en el otro. La ecuación 3 está simplificada mediante la segunda ley distributiva (identidad 15 en la Tabla 2-3). A continuación hay tres ejemplos para simplificar expresiones booleanas:

4. $X(X + Y) = X + XY = X$
5. $(X + Y)(X + \bar{Y}) = X + Y\bar{Y} = X$
6. $X(\bar{X} + Y) = X\bar{X} + XY = XY$

Véase que los pasos intermediarios $XX = X = X \cdot 1$ han sido omitidos durante la manipulación de la ecuación 4. La expresión de la ecuación 5 está simplificada mediante la segunda ley distributiva. Aquí omitimos otra vez los pasos intermediarios $Y\bar{Y} = 0$ y $X + 0 = X$.

Las ecuaciones 4 a 6 son las duales de las ecuaciones 1 a 3. Recuerde que el dual de una expresión se obtiene cambiando AND por OR y OR por AND en todas partes (y 1 por 0 y 0 por 1 si aparecen en la expresión). El *principio de dualidad* del Álgebra de Boole expresa que una ecuación booleana permanece válida si tomamos el dual de la expresión en ambos lados del signo de igualdad. Por eso, las ecuaciones 4, 5 y 6 se pueden obtener tomando el dual de las ecuaciones 1, 2 y 3, respectivamente.

Junto con los resultados dados en las ecuaciones 1 a 6, el teorema siguiente, *teorema de consenso*, es útil a la hora de simplificar expresiones booleanas:

$$XY + \bar{X}Z + YZ = XY + \bar{X}Z$$

El teorema muestra que el tercer término, YZ, es redundante y se puede eliminar. Note que se asocian Y y Z con X y $\bar{X}$ en los primeros dos términos y que aparecen juntos en el término eliminado. La prueba del teorema de consenso se obtiene por la conexión AND entre YZ y $(X + \bar{X}) = 1$ y siguiendo después como se indica a continuación:

$$\begin{aligned} XY + \bar{X}Z + YZ &= XY + \bar{X}Z + YZ(X + \bar{X}) \\ &= XY + \bar{X}Z + XYZ + \bar{X}YZ \\ &= XY + XYZ + \bar{X}Z + \bar{X}YZ \\ &= XY(1 + Z) + \bar{X}Z(1 + Y) \\ &= XY + \bar{X}Z \end{aligned}$$

El dual del teorema de consenso es

$$(X + Y)(\bar{X} + Z)(Y + Z) = (X + Y)(\bar{X} + Z)$$

El ejemplo siguiente muestra cómo se puede aplicar el teorema de consenso durante la manipulación de una expresión booleana:

$$\begin{aligned} (A + B)(\bar{A} + C) &= A\bar{A} + AC + \bar{A}B + BC \\ &= AC + \bar{A}B + BC \\ &= AC + \bar{A}B \end{aligned}$$

Véase que $A\bar{A} = 0$ y $0 + AC = AC$. El término redundante eliminado por el teorema de consenso es BC.

El complemento de una función

La representación complementaria de una función F, $\bar{F}$, se obtiene de un intercambio de 1 por 0 y 0 por 1 en los valores de F en la tabla de verdad. El complemento de una función se puede derivar algebraicamente aplicando el Teorema de DeMorgan. La forma generalizada de este teorema expresa que se obtiene el complemento de una expresión mediante el intercambio de las operaciones AND y OR y complementando cada variable y cada constante, como se muestra en el Ejemplo 2-1.

EJEMPLO 2-1 Funciones de complemento

Encuentre el complemento de cada una de las funciones representadas por las ecuaciones $F_1 = \bar{X}YZ + \bar{X}\bar{Y}Z$ y $F_2 = X(\bar{Y}\bar{Z} + YZ)$. Aplicando el Teorema de DeMorgan tantas veces como sea necesario, obtenemos el complemento según lo siguiente:

$$\begin{aligned}\bar{F}_1 &= \overline{\bar{X}YZ + \bar{X}\bar{Y}Z} = \overline{(\bar{X}YZ)} \cdot \overline{(\bar{X}\bar{Y}Z)} \\ &= (X + \bar{Y} + Z)(X + Y + \bar{Z}) \\ \bar{F}_2 &= \overline{X(\bar{Y}\bar{Z} + YZ)} = \bar{X} + \overline{(\bar{Y}\bar{Z} + YZ)} \\ &= \bar{X} + (\overline{\bar{Y}\bar{Z}} \cdot \overline{YZ}) \\ &= \bar{X} + (Y + Z)(\bar{Y} + \bar{Z})\end{aligned}$$

■

Un método más simple para derivar el complemento de una función es calcular el dual de la ecuación de la función y complementar cada literal. Este método es el resultado de la generalización del Teorema de DeMorgan. Recuerde que se obtiene el dual de una expresión intercambiando las operaciones AND y OR y 1 y 0. Para evitar confusión en el manejo de funciones complejas, es útil añadir paréntesis alrededor de los términos antes de calcular el dual, según se ilustra en el siguiente ejemplo.

EJEMPLO 2-2 Complementando funciones usando dualidad

Encuentre los complementos de las funciones del Ejemplo 2-1 calculando los duales de sus ecuaciones y complementando cada literal.

Empezamos con

$$F_1 = \bar{X}YZ + \bar{X}\bar{Y}Z = (\bar{X}YZ) + (\bar{X}\bar{Y}Z)$$

El dual de F_1 es

$$(\bar{X} + Y + \bar{Z})(\bar{X} + \bar{Y} + Z)$$

Complementando cada literal, tenemos

$$(X + \bar{Y} + Z)(X + Y + \bar{Z}) = \bar{F}_1$$

Ahora,

$$F_2 = X(\bar{Y}\bar{Z} + YZ) = X((\bar{Y}\bar{Z}) + (YZ))$$

El dual de F_2 es

$$X + (\bar{Y} + \bar{Z})(Y + Z)$$

Complementando cada literal da lugar a

$$\bar{X} + (Y + Z)(\bar{Y} + \bar{Z}) = \bar{F}_2.$$

■

2-3 FORMAS CANÓNICAS

Se puede escribir una función booleana, expresada algebraicamente, de diferentes maneras. Hay, sin embargo, formas concretas de escribir las ecuaciones algebraicas que se consideran como formas canónicas. Las formas canónicas facilitan los procedimientos de simplificación para expresiones booleanas y frecuentemente da lugar a circuitos lógicos más deseables.

La forma canónica contiene *términos producto* y *términos suma*. Un ejemplo de un término producto es $X\bar{Y}Z$. Esto es un producto lógico formado por una operación AND de tres literales. Un ejemplo de un término suma es $X + Y + \bar{Z}$. Esto es una suma lógica formada por una operación OR entre los literales. Hay que darse cuenta de que las palabras «producto» y «suma» no implican operaciones aritméticas en el álgebra de Boole; en cambio, especifican las operaciones lógicas AND y OR respectivamente.

Minitérminos y maxitérminos

Se ha mostrado que la tabla de verdad define una función booleana. Una expresión algebraica que represente la función se puede derivar de la tabla buscando la suma lógica de todos los términos producto para los que la función asume el valor binario 1. A un término producto donde todas las variables aparecen exactamente una vez, sean complementadas o no complementadas, se le llama *minitérmino*. Su propiedad característica es que representa exactamente una combinación de las variables binarias en la tabla de verdad. Tiene el valor 1 para esta combinación y 0 para el resto. Hay 2^n diferentes minitérminos para n variables. Los cuatro minitérminos para las dos variables X e Y son y $\bar{X}\bar{Y}$, $\bar{X}Y$, $X\bar{Y}$ y XY. Los ocho minitérminos para las tres variables X, Y, y Z se muestran en la Tabla 2-6. Los números binarios de 000 a 111 se muestran debajo de las variables. Para cada combinación binaria hay un minitérmino asociado. Cada minitérmino es un término de producto de exactamente tres literales. Un literal es una variable complementada si el bit correspondiente de la combinación binaria asociada es 0 y es una variable no complementada si es 1. También se muestra un símbolo m_j para cada minitérmino en la tabla, donde el subíndice j denota el equivalente decimal de la combinación binaria para la que el minitérmino tiene el valor 1. Esta lista de minitérminos para cada n variables dadas se puede formar de manera similar a una lista de los números binarios de 0 a $2^n - 1$. Además, la tabla de verdad para cada minitérmino se muestra en la parte derecha de la tabla. Estas tablas de verdad muestran claramente que cada minitérmino es 1 para la combinación binaria correspondiente y 0 para todas las otras combinaciones. Más tarde, estas tablas de verdad serán útiles al usar minitérminos para formar expresiones booleanas.

A un término suma que contiene todas las variables de forma complementada o no complementada se le llama *maxitérmino*. Otra vez es posible formular 2^n maxitérminos con n variables. Los ocho maxitérminos para tres variables se muestran en la Tabla 2-7. Cada maxitérmino es una suma lógica de tres variables, donde cada variable se complementa si el bit correspondiente del número binario es 1 y no se complementa si es 0. El símbolo para un maxitérmino es M_j, donde j denota el equivalente decimal de una combinación binaria para la que el maxitérmino tiene el valor 0. En la parte derecha de la tabla, se muestra la tabla de verdad para cada maxitérmino. Véase que el valor del maxitérmino es 0 para la combinación correspondiente y 1 para el resto de combinaciones. Ahora está claro de donde salen los términos »minitérmino» y «maxitérmino»: un minitérmino es una función, no igual a 0, que tiene el menor número de 1s en su tabla de verdad; un maxitérmino es una función, no igual a 1, que tiene el mayor número de 1s en su tabla de verdad. Véase de la Tabla 2-6 y la Tabla 2-7 que los minitérminos y maxitérminos

□ **TABLA 2-6**
Minitérminos para tres variables

X	Y	Z	Término producto	Símbolo	m_0	m_1	m_2	m_3	m_4	m_5	m_6	m_7
0	0	0	$\bar{X}\bar{Y}\bar{Z}$	m_0	1	0	0	0	0	0	0	0
0	0	1	$\bar{X}\bar{Y}Z$	m_1	0	1	0	0	0	0	0	0
0	1	0	$\bar{X}Y\bar{Z}$	m_2	0	0	1	0	0	0	0	0
0	1	1	$\bar{X}YZ$	m_3	0	0	0	1	0	0	0	0
1	0	0	$X\bar{Y}\bar{Z}$	m_4	0	0	0	0	1	0	0	0
1	0	1	$X\bar{Y}Z$	m_5	0	0	0	0	0	1	0	0
1	1	0	$XY\bar{Z}$	m_6	0	0	0	0	0	0	1	0
1	1	1	XYZ	m_7	0	0	0	0	0	0	0	1

□ **TABLA 2-7**
Maxitérminos para tres variables

X	Y	Z	Término suma	Símbolo	M_0	M_1	M_2	M_3	M_4	M_5	M_6	M_7
0	0	0	$X + Y + Z$	M_0	0	1	1	1	1	1	1	1
0	0	1	$X + Y + \bar{Z}$	M_1	1	0	1	1	1	1	1	1
0	1	0	$X + \bar{Y} + Z$	M_2	1	1	0	1	1	1	1	1
0	1	1	$X + \bar{Y} + \bar{Z}$	M_3	1	1	1	0	1	1	1	1
1	0	0	$\bar{X} + Y + Z$	M_4	1	1	1	1	0	1	1	1
1	0	1	$\bar{X} + Y + \bar{Z}$	M_5	1	1	1	1	1	0	1	1
1	1	0	$\bar{X} + \bar{Y} + Z$	M_6	1	1	1	1	1	1	0	1
1	1	1	$\bar{X} + \bar{Y} + \bar{Z}$	M_7	1	1	1	1	1	1	1	0

con los mismos subíndices son los complementos entre sí; o sea, $M_j = \bar{m}_j$. Por ejemplo, para $j = 3$, tenemos

$$\bar{m}_3 = \overline{\bar{X}YZ} = X + \bar{Y} + \bar{Z} = M_3$$

Una función booleana puede ser representada algebraicamente por una tabla de verdad dada formando la suma lógica de todos los minitérminos que producen un 1 en la función. Esta expresión se llama una *suma de minitérminos*. Considere la función booleana F de la Tabla 2-8(a). La función es igual a 1 para cada una de las siguientes combinaciones binarias de las variables X, Y, y Z: 000, 010, 101 y 111. Esas combinaciones corresponden a los minitérminos 0, 2, 5 y 7. Examinando la Tabla 2-8 y las tablas de verdad para éstos minitérminos de la Tabla 2-6, es evidente que se puede expresar la función F algebraicamente como la suma lógica de los minitérminos formulados:

$$F = \bar{X}\bar{Y}\bar{Z} + \bar{X}Y\bar{Z} + X\bar{Y}Z + XYZ = m_0 + m_2 + m_5 + m_7$$

□ **TABLA 2-8**
Funciones booleanas de tres variables

(a) X	Y	Z	F	$\bar{F}$	(b) X	Y	Z	E
0	0	0	1	0	0	0	0	1
0	0	1	0	1	0	0	1	1
0	1	0	1	0	0	1	0	1
0	1	1	0	1	0	1	1	0
1	0	0	0	1	1	0	0	1
1	0	1	1	0	1	0	1	1
1	1	0	0	1	1	1	0	0
1	1	1	1	0	1	1	1	0

Esto se puede abreviar más enumerando solamente los subíndices decimales de los minitérminos:

$$F(X, Y, Z) = \Sigma m(0, 2, 5, 7)$$

El símbolo Σ significa la suma lógica (OR booleana) de los minitérminos. Los números en paréntesis representan los minitérminos de la función. Las letras entre paréntesis que van a continuación de F forman una lista de variables en el orden de conversión de los minitérminos a términos producto.

Ahora considere el complemento de una función booleana. Los valores binarios de $\bar{F}$ de la Tabla 2-8(a) se obtienen cambiando 1 a 0 y 0 a 1 en los valores de F. Partiendo de la suma lógica de los minitérminos de $\bar{F}$, obtenemos

$$\bar{F} = \bar{X}\bar{Y}Z + \bar{X}YZ + X\bar{Y}\bar{Z} + XY\bar{Z} = m_1 + m_3 + m_4 + m_6$$

o, de forma abreviada,

$$\bar{F}(X, Y, Z) = \Sigma m(1, 3, 4, 6)$$

Véase que los números de los minitérminos de $\bar{F}$ son los que faltan en la lista de números de los minitérminos de F. Ahora tomamos el complemento de para obtener F:

$$\begin{aligned} F &= \overline{m_1 + m_3 + m_4 + m_6} = \overline{m_1} \cdot \overline{m_3} \cdot \overline{m_4} \cdot \overline{m_6} \\ &= M_1 \cdot M_3 \cdot M_4 \cdot M_6 \text{ (ya que } \overline{m_j} = M_j) \\ &= (X + Y + \bar{Z})(X + \bar{Y} + \bar{Z})(\bar{X} + Y + Z)(\bar{X} + \bar{Y} + Z) \end{aligned}$$

Esto muestra el procedimiento para expresar una función booleana como *producto de maxitérminos*. La forma abreviada para ese producto es

$$F(X, Y, Z) = \Pi M(1, 3, 4, 6)$$

donde el símbolo Π denota el producto lógico (AND booleana) de los maxitérminos cuyos números se enumeran entre paréntesis. Véase que los números decimales incluidos en el producto de maxitérminos siempre serán los mismos que los de la lista de minitérminos de la función complementada, como (1, 3, 4, 6) del ejemplo anterior. Los maxitérminos se usan rara vez cuando se trata con funciones booleanas, porque es siempre posible reemplazarlos con la lista de minitérminos de $\bar{F}$.

A continuación se resumen las propiedades más importantes de los minitérminos:

1. Hay 2^n minitérminos para n variables booleanas. Estos minitérminos se pueden evaluar a partir de los números binarios de 0 a $2^n - 1$.
2. Cada función booleana se puede expresar como suma lógica de minitérminos.
3. El complemento de una función contiene los minitérminos que no están incluidos en la función original.
4. Una función que incluye todos los 2^n minitérminos es igual a un 1 lógico.

Una función que no tiene la forma de suma de minitérminos puede convertirse a esta forma mediante la tabla de verdad, mientras que la tabla de verdad especifica los minitérminos de la función. Considere, por ejemplo, la función booleana

$$E = \bar{Y} + \bar{X}\bar{Y}$$

La expresión no tiene forma de suma de minitérminos, porque cada término no contiene todas los tres variables *X*, *Y*, y *Z*. En la Tabla 2-8(b) se enumera la tabla de verdad de esta función. De la tabla, obtenemos los minitérminos de la función:

$$E(X, Y, Z) = \Sigma m(0, 1, 2, 4, 5)$$

Los minitérminos para el complemento de *E* resultan de

$$\bar{E}(X, Y, Z) = \Sigma m(3, 6, 7)$$

Véase que el número total de minitérminos en *E* y $\bar{E}$ es igual a ocho, ya que la función tiene tres variables, y tres variables producen un total de ocho minitérminos. Con cuatro variables habrá un total de 16 minitérminos, y para dos variables, habrá 4 minitérminos. Un ejemplo de una función que incluye todos los minitérminos es

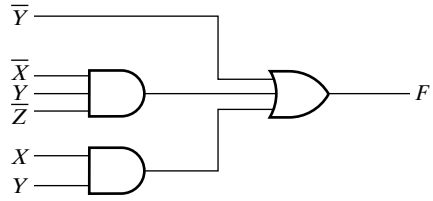
$$G(X, Y) = \Sigma m(0, 1, 2, 3) = 1$$

Como *G* es una función de dos variables y contiene todos los cuatro minitérminos, siempre es igual a un 1 lógico.

Suma de productos

La forma de suma de minitérminos es una expresión algebraica canónica que se obtiene directamente de una tabla de verdad. La expresión obtenida de esta manera contiene el número máximo de literales en cada término y tiene normalmente más productos de los que son necesarios. La razón para esto es que, por definición, cada minitérmino tiene que incluir todas las variables de la función, complementada o no complementada. Si se ha obtenido una vez la suma de minitérminos de la tabla de verdad, el siguiente paso es intentar simplificar la expresión para ver si es posible reducir el número de productos y el número de literales en los términos. El resultado es una expresión simplificada en la forma de *suma de productos*. Esto es una forma canónica alternativa de expresión que contiene productos con uno, dos, o cualquier número de literales. Un ejemplo de una función booleana expresada como suma de productos es

$$F = \bar{Y} + \bar{X}Y\bar{Z} + XY$$



□ **FIGURA 2-5**
Implementación con suma de productos

La expresión tiene tres productos, el primero con un literal, el segundo con tres literales, y el tercero con dos literales.

El diagrama lógico para una forma de suma de productos está formado por un grupo de puertas AND seguido de una única puerta OR, como se muestra en la Figura 2-5. Cada producto requiere una puerta AND, excepto para el término con un único literal. La suma lógica se forma con una puerta OR que tiene como entradas literales únicos y la salida de puertas AND. Se supone que las variables de entrada están directamente disponibles en sus formas complementadas y no complementadas, así que no se incluyen inversores en el diagrama. Las puertas AND seguidas por la puerta OR forman una configuración de circuito al que se le denomina como una *implementación de dos niveles* o como *circuito de dos niveles*.

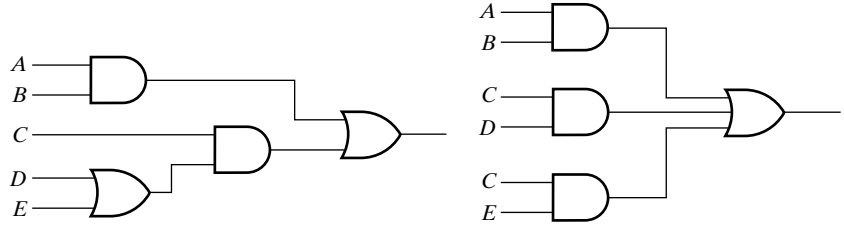
Si una expresión no está en la forma de suma de productos, se puede convertir a una forma canónica mediante las leyes distributivas. Considere la expresión

$$F = AB + C(D + E)$$

Esta no está en forma de suma de productos, porque el término $D + E$ es parte de un producto, pero no es un literal único. La expresión puede convertirse en una suma de productos aplicando la ley distributiva apropiada, como se muestra a continuación:

$$F = AB + C(D + E) = AB + CD + CE$$

En la Figura 2-6(a) se ve la función F implementada en forma no canónica. Esto requiere dos puertas AND y dos puertas OR. Hay tres niveles de puertas en el circuito. En la Figura 2-6(b), se ha implementado F en forma de suma de productos. Este circuito requiere tres puertas AND y una puerta OR y usa dos niveles de puertas. La decisión de usar una implementación de dos o de múltiples niveles (tres o más) es compleja. Los problemas aquí involucrados son el número de puertas, el número de entradas a las puertas y el retardo entre el momento en que las variables de entrada están puestas y el momento en que aparecen los valores resultantes en la



(a) $AB + C(D + E)$ (b) $AB + CD + CE$

□ **FIGURA 2-6**
Implementación de tres y dos niveles

salida. Las implementaciones de dos niveles son la forma natural para ciertas tecnologías, como veremos en el Capítulo 4.

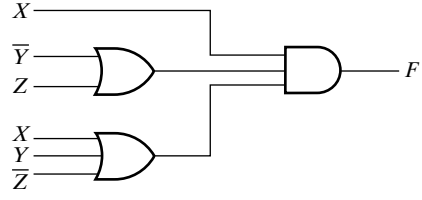
Producto de sumas

Otra forma canónica para expresar funciones booleanas algebraicamente es el *producto de sumas*. Esta forma se obtiene formando un producto lógico de sumas. Cada término de la suma lógica puede tener cualquier número de literales diferentes. Un ejemplo de una función expresada de forma de suma de productos es

$$F = X(\overline{Y} + Z)(X + Y + \overline{Z})$$

Esta expresión tiene sumas de uno, dos y tres literales. Los términos de suma realizan una operación OR, y el producto es una operación AND.

La estructura de las puertas de la expresión de productos de suma esta formada por un grupo de puertas OR para las sumas (excepto para el término con un único literal), seguido de una puerta AND. Esto se muestra en la Figura 2-7 para la anterior función *F*. Como en el caso de suma de productos, este tipo de expresión canónica está formada por una estructura de dos niveles de puertas.



□ **FIGURA 2-7**
Implementación de producto de sumas

2-4 OPTIMIZACIÓN DE CIRCUITOS DE DOS NIVELES

La complejidad de las puertas lógicas digitales que realizan una función booleana está relacionada directamente con la expresión algebraica a partir de la cual se implementa la función. Aunque la representación de una función en la tabla de verdad es única, cuando se expresa algebraicamente, la función puede aparecer en muchas formas diferentes. Las expresiones booleanas podrían simplificarse mediante manipulación algebraica como se ha discutido en la Sección 2-2. Sin embargo, este procedimiento de simplificación es malo porque carece de reglas especiales para predecir cada paso que ocurre en el proceso manipulativo y es difícil de determinar si se ha conseguido la expresión más sencilla. Por otro lado, el método del mapa provee un procedimiento directo para optimizar funciones booleanas con un máximo de cuatro variables. Se puede pintar también mapas para cinco y seis variables, pero estas son más incómodas de usar. El mapa se conoce también como *Mapa de Karnaugh*, o *mapa-K*. El mapa es un diagrama hecho con cuadrados, donde cada cuadrado representa un minitérmino de la función. Puesto que cualquier función booleana se puede expresar como suma de minitérminos, una función booleana puede ser reconocida gráficamente en el mapa por aquellos cuadrados cuyos minitérminos se incluyen en la función. De hecho, el mapa presenta un diagrama visual de todos los caminos posibles para expresar una función en forma canónica. Reconociendo diferentes patrones, el usuario puede derivar expresiones algebraicas alternativas para la misma función, de las cuales

se selecciona la más sencilla. Las expresiones optimizadas producidas por el mapa están siempre en forma de suma de productos o producto de sumas. Así, los mapas manejan la optimización para implementaciones de dos niveles, pero no se puede aplicar directamente a posibles implementaciones más sencillas para el caso general con tres o más niveles. Inicialmente, esta sección cubre la optimización de suma de productos y, más tarde, aplica la optimización de producto de sumas.

Criterios de coste

En la sección anterior, el número de literales y términos se vio como una manera de medir la simplicidad de un circuito lógico. Ahora introducimos dos criterios de coste para formalizar este concepto.

El primer criterio es el *coste por literal*, el número de veces que aparecen los literales en una expresión booleana que corresponde exactamente al diagrama lógico. Por ejemplo, para los circuitos de la Figura 2-6, Las expresiones booleanas correspondientes son

$$F = AB + C(D + E) \quad \text{y} \quad F = AB + CD + CE$$

En la primera ecuación aparecen cinco literales y seis en la segunda, de esta forma, la primera ecuación es la más simple en términos de coste por literal. El coste por literal tiene la ventaja que se puede evaluar muy sencillamente contando la aparición de los literales. Sin embargo, no representa correctamente la complejidad del circuito en todos los casos, ni siquiera para la comparación de diferentes implementaciones de la misma función lógica. Las siguientes ecuaciones booleanas, ambas de la función G, muestran esta situación:

$$G = ABCD + \bar{A}\bar{B}\bar{C}D \quad \text{y} \quad G = (\bar{A} + B)(\bar{B} + C)(\bar{C} + D)(\bar{D} + A)$$

Las implementaciones representadas por esas ecuaciones tienen ambas un coste literal de ocho. Pero, la primera ecuación tiene dos términos y la segunda tiene cuatro. Esto sugiere que la primera ecuación tiene un coste más bajo que la segunda.

Para entender la diferencia ilustrada, definimos el *coste de entradas de puerta* como el número de entradas a las puertas en la implementación que corresponde exactamente a la ecuación o las ecuaciones dadas. Este coste se puede determinar fácilmente a partir del diagrama lógico contando simplemente el número total de entradas a las puertas. Para las ecuaciones de suma de productos o producto de sumas, se puede averiguar encontrando la suma de

(1) todas las apariciones de los literales,
(2) el número de términos excluyendo términos que solamente consisten en un único literal, y, opcionalmente,
(3) el número de diferentes literales complementados.

En (1), se representan todas las entradas de las puertas desde fuera del circuito. En (2), se representa todas las entradas de las puertas dentro del circuito, excepto las que van a los inversores y en (3), los inversores necesarios para complementar las variables de entrada, que se cuentan en el momento que no se proporcionan las variables de entrada complementadas. Para las dos ecuaciones precedentes, excluyendo la suma de (3), las respectivas sumas de las entradas de las puertas son $8 + 2 = 10$ y $8 + 4 = 12$. Incluyendo la suma de (3), la de inversores de entrada, las sumas respectivas son 14 y 16. Así, la primera ecuación para G tiene un coste por entradas más bajo, aunque los costes por literales sean iguales.

El coste por entradas de puerta es ahora una buena medida para implementaciones lógicas actuales ya que es proporcional el número de transistores y conexiones usadas para implementar un circuito lógico. La representación de las entradas de las puertas va ser particularmente más importante para medir los costes de los circuitos con más que dos niveles. Típicamente, como el número de niveles incrementa, el coste literal representa una proporción más pequeña en el coste de los circuitos actuales, ya que más y más puertas no tienen entradas desde fuera del mismo circuito. Más adelante, en la Figura 2-29, introducimos otros tipos de puertas complejas para las que la evaluación del coste de las entradas de las puertas a partir de una ecuación no es válida, ya que la correspondencia entre las operaciones AND, OR y NOT de la ecuación y las puertas del circuito no se puede establecer. En estos casos, tanto como para ecuaciones más complejas que las sumas de productos y los productos de sumas, hay que determinar la suma de las entradas de las puertas directamente desde la implementación.

Sin tener en cuenta los criterios de coste usados, más adelante veremos que la expresión más sencilla no es necesariamente la única. A veces es posible encontrar dos o más expresiones que cumplen el criterio de coste aplicado. En este caso, cada solución es satisfactoria desde el punto de vista de coste.

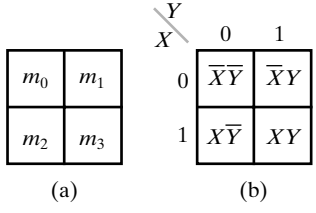
Mapa de dos variables

Hay cuatro minitérminos en una función booleana con dos variables. Así, el mapa de dos variables consiste en cuatro cuadrados, uno por cada minitérmino, según se muestra en la Figura 2-8(a). El mapa se muestra otra vez en la Figura 2-8(b) para señalar la relación entre los cuadrados y las dos variables X e Y. El 0 y 1 marcado, en el lado izquierdo y en la parte superior del mapa indican los valores de las variables. La variable X aparece complementada en la fila 0 y no complementada en la fila 1. De igual manera, Y aparece complementada en la columna 0 y no complementada en la columna 1. Véase que las cuatro combinaciones de estos valores binarios corresponden a las filas de la tabla de verdad asociadas a los cuatro minitérminos.

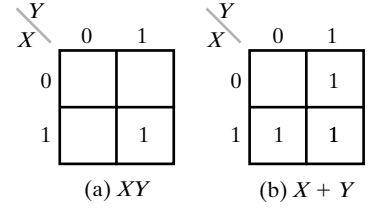
Una función de dos variables puede ser representada en un mapa marcando los cuadrados que corresponden a los minitérminos de la función. Como ejemplo, se muestra la función XY de la Figura 2-9(a). Ya que XY es igual al minitérmino m_3, se pone un 1 dentro del cuadrado que pertenece a m_3. En la Figura 2-9(b) se muestra el mapa para la suma lógica de tres minitérminos:

$$m_1 + m_2 + m_3 = \bar{X}Y + X\bar{Y} + XY = X + Y$$

La expresión optimizada $X + Y$ se determina del área de dos cuadrados para la variable X en la segunda fila y del área de dos cuadrados para Y en la segunda columna. Juntas, estas dos áreas



□ **FIGURA 2-8**
Mapa de dos variables



□ **FIGURA 2-9**
Representación de funciones en el mapa

incluyen los tres cuadrados pertenecientes a X o Y. Esta simplificación puede justificase mediante manipulación algebraica:

$$\bar{X}Y + X\bar{Y} + XY = \bar{X}Y + X(\bar{Y} + Y) = (\bar{X} + X)(Y + X) = X + Y$$

El procedimiento exacto para combinar cuadrados en el mapa se va a aclarar en los siguientes ejemplos.

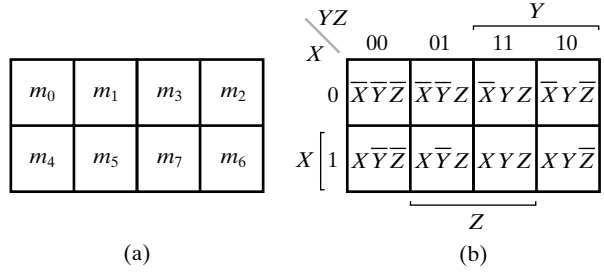
Mapa de tres variables

Hay ocho minitérminos para tres variables binarias. Por eso, un mapa de tres variables tiene ocho cuadrados, como se indica en la Figura 2-10. El mapa dibujado en la parte (b) está marcado con números binarios para cada fila y cada columna para mostrar los valores binarios de los minitérminos. Note que los números en las columnas no siguen la secuencia de la cuenta binaria. La característica de la secuencia enumerada es que sólo un bit cambia su valor de una columna hacia la adyacente, que corresponde al Código Gray introducido en el Capítulo 1.

Un cuadrado perteneciente a un minitérmino se puede localizar en el mapa de dos maneras. Primero, podemos memorizar los números enumerados en la Figura 2-10(a) para cada lugar del minitérmino, o podemos referirnos a los números binarios dentro de las filas y columnas de la Figura 2-10(b). Por ejemplo, el cuadrado asignado a m_5 corresponde a la fila 1 y a la columna 01. Cuando se combinan estos dos números, dan lugar al número binario 101, cuyo equivalente decimal es 5.

Otra posibilidad de localizar el cuadrado $m_5 = X\bar{Y}Z$ es verlo situado en la fila marcada con X y la columna que pertenece a $\bar{Y}Z$ (columna 01). Véase que hay cuatro cuadrados donde cada variable es igual a 1 y cuatro donde cada una es igual a 0. La variable aparece sin complementar en los cuatro cuadrados donde es igual a 1 y de forma complementada en los cuatro cuadrados donde es igual a 0. Es más conveniente escribir el nombre de la variable al lado de los cuatro cuadrados donde no aparece complementada. Después de familiarizarse con los mapas, el uso de los nombres de los variables es suficiente para identificar las regiones del mapa. Para esto, es importante localizar estas etiquetas para obtener todos los minitérminos del mapa.

En el mapa de dos variables, la función XY mostraba que una función o un término de una función pueden estar formados por un único cuadrado del mapa. Pero para conseguir una simplificación, hay que considerar varios cuadrados que corresponden a términos productos. Para entender cómo la combinación de cuadrados simplifica las funciones booleanas, tenemos que conocer la propiedad básica de cuadrados adyacentes: cada pareja de cuadrados adyacentes, horizontales o verticales (pero no diagonales), que forman un rectángulo, corresponden a minitér-



□ **FIGURA 2-10**
Mapa de tres variables

minos que varían en una sola variable. Esta variable aparece sin complementar en un cuadrado y complementada en el otro. Por ejemplo, m_5 y m_7 están situadas en dos cuadrados adyacentes. Se encuentra el complemento de Y en m_5 y la misma variable, sin complementar, en m_7, mientras las otras dos variables son iguales en los dos cuadrados. La suma lógica de dos de estos minitérminos adyacentes se puede simplificar en un único término producto de dos variables:

$$m_5 + m_7 = X\bar{Y}Z + XYZ = XZ(\bar{Y} + Y) = XZ$$

Aquí, los dos cuadrados son diferentes en cuanto a la variable Y, que se puede quitar cuando se calcula la suma lógica (OR) de los minitérminos. Así, en un mapa de 3 variables, cada dos minitérminos en cuadrados adyacentes que se combinan con una OR producen un término producto de dos variables. Esto se muestra en el Ejemplo 2-3.

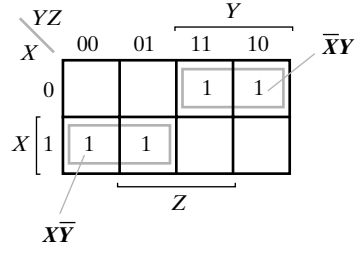
EJEMPLO 2-3 Simplificación de una función booleana usando un mapa

Simplifique la función booleana

$$F(X, Y, Z) = \Sigma m(2, 3, 4, 5)$$

Primero, se pone un 1 en cada minitérmino que representa la función. Esto se muestra en la Figura 2-11, donde los cuadrados para los minitérminos 010, 011, 100, y 101 se han rellenado con 1s. Es mejor dejar todos los cuadrados en los que la función tiene el valor 0 en blanco en vez de poner los 0s. El paso siguiente es encontrar grupos de cuadrados en el mapa que representan términos producto considerados para la expresión simplificada. Llamamos a estos objetos *rectángulos*, ya que su forma es un rectángulo (incluyendo, por supuesto, un cuadrado). Sin embargo, los rectángulos que corresponden a términos producto están restringidos al contener un número de cuadrados que son potencias de 2, como 1, 2, 4, 8, ... Así, nuestro objetivo es encontrar el menor número de rectángulos que incluyan a todos los minitérminos marcados con 1s. Esto va a dar lugar a un mínimo de términos producto. En el mapa de la figura, los dos rectángulos acogen a los cuatro cuadrados que contienen 1. El rectángulo superior derecho representa al término producto $\bar{X}Y$. Esto se determina observando que el rectángulo está en la fila 0, que corresponde a $\bar{X}$, y las últimas dos columnas, correspondiendo a Y. De igual manera, el rectángulo inferior izquierdo representa el término de producto $X\bar{Y}$. (La segunda fila representa X y las dos columnas a la izquierda representan $\bar{Y}$.) Ya que estos dos rectángulos incluyen a todos de los 1s del mapa, la suma lógica de los dos términos producto correspondientes dan como resultado una expresión optimizada de F:

$$F = \bar{X}Y + X\bar{Y}$$



□ **FIGURA 2-11**
Mapa para Ejemplo 2-3: $F(X, Y, Z) = \Sigma m(2, 3, 4, 5) = \bar{X}Y + X\bar{Y}$ ■

En algunos casos, dos cuadrados del mapa son adyacentes y forman un rectángulo de tamaño dos, aunque no se tocan. Por ejemplo, en la Figura 2-10, m_0 es adyacente a m_2 y m_4 es adyacente a m_6 porque los minitérminos se distinguen en una variable. Esto se puede verificar algebraicamente:

$$m_0 + m_2 = \bar{X}\bar{Y}\bar{Z} + \bar{X}Y\bar{Z} = \bar{X}\bar{Z}(\bar{Y} + Y) = \bar{X}\bar{Z}$$
$$m_4 + m_6 = X\bar{Y}\bar{Z} + XY\bar{Z} = X\bar{Z}(\bar{Y} + Y) = X\bar{Z}$$

Los rectángulos que corresponden a estos dos términos producto, $\bar{X}\bar{Z}$ y $X\bar{Z}$, se muestran en el mapa de la Figura 2-12(a). Basándose en la ubicación de estos rectángulos, tenemos que modificar la definición de los cuadrados adyacentes para incluir este y otros casos similares. Lo hacemos considerando el mapa como si estuviese dibujado en forma de cilindro, como se muestra en la Figura 2-12(b), donde los bordes derechos e izquierdos se tocan para establecer correctamente las vecindades de los minitérminos y formar los rectángulos. En los mapas de la Figura 2-12, hemos usado simplemente números en vez de *m* para representar los minitérminos. Cualquiera de estas notaciones se usará libremente.

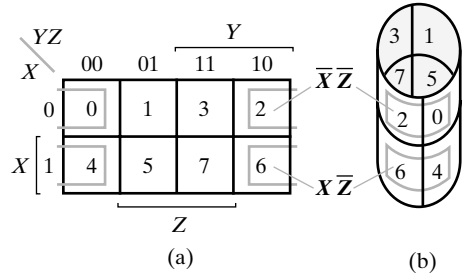
Un rectángulo de cuatro cuadrados representa un término producto que es la suma lógica de cuatro minitérminos. Para el caso de tres variables, un término producto de estos está formado por un solo literal. Como ejemplo, la suma lógica de cuatro minitérminos adyacentes 0, 2, 4, y 6 se reducen a un único término literal $\bar{Z}$:

$$\begin{aligned} m_0 + m_2 + m_4 + m_6 &= \bar{X}\bar{Y}\bar{Z} + \bar{X}Y\bar{Z} + X\bar{Y}\bar{Z} + XY\bar{Z} \\ &= \bar{X}\bar{Z}(\bar{Y} + Y) + X\bar{Z}(\bar{Y} + Y) \\ &= \bar{X}\bar{Z} + X\bar{Z} = \bar{Z}(\bar{X} + X) = \bar{Z} \end{aligned}$$

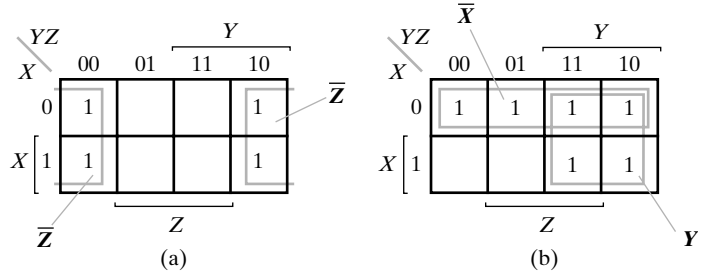
El rectángulo para este término producto se muestra en la Figura 2-13(a). Véase que el término producto se basa en que los bordes izquierdos y derechos del mapa son adyacentes de manera que forman un rectángulo. Los otros dos ejemplos de rectángulos que se corresponden con términos producto derivados de cuatro minitérminos se muestran en la Figura 2-13(b).

En general, ya que se combinan más cuadrados, obtenemos un término producto con menos literales. Los mapas de tres variables requieren las siguientes características:

Un cuadrado representa un minitérmino de tres literales.
Un rectángulo de dos cuadrados representa un término de producto de dos literales.
Un rectángulo de cuatro cuadrados representa un término de producto de un literal.
Un rectángulo de ocho cuadrados abarca el mapa entero y produce una función que es siempre igual a 1 lógico.
Éstas características se muestran en el Ejemplo 2-4.



□ **FIGURA 2-12**
Mapa de tres variables: Plano y en cilindro para mostrar los cuadrados adyacentes



□ **FIGURA 2-13**
Términos producto usando cuatro minitérminos

EJEMPLO 2-4 Simplificación de funciones de tres variables con mapas

Simplifique las dos funciones booleanas siguientes:

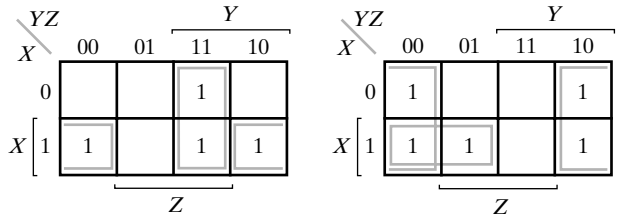
$$F_1(X, Y, Z) = \Sigma m(3, 4, 6, 7)$$
$$F_2(X, Y, Z) = \Sigma m(0, 2, 4, 5, 6)$$

El mapa de F_1 se muestra en la Figura 2-14(a). Hay cuatro cuadrados marcados con 1s, uno para cada minitérmino de la función. Se combinan dos cuadrados adyacentes en la tercera columna para llegar al término de dos literales YZ. Los dos cuadrados sobrantes, también con 1, son adyacentes por la definición basada en el cilindro y se muestran en el diagrama con sus valores incluidos en medios rectángulos. Cuando se combinan, estos dos cuadrados construyen el término de dos literales $X\bar{Z}$. Así la función optimizada pasa a ser

$$F_1 = YZ + X\bar{Z}$$

El mapa para F_2 se muestra en la Figura 2-14(b). Primero, combinamos los cuatro cuadrados adyacentes en las primeras y últimas columnas, basándonos en lo que hemos aprendido de la Figura 2-13, para llegar al término de un literal $\bar{Z}$. El último cuadrado sobrante que representa el minitérmino 5 se combina con un cuadrado adyacente que ya ha sido usado una vez. Esto no sólo se permite, sino que es deseable, ya que los dos cuadrados adyacentes llegan al término de dos literales $X\bar{Y}$, con el cuadrado simple representando el minitérmino de tres literales $XY\bar{Z}$. La función optimizada es

$$F_2 = \bar{Z} + X\bar{Y}$$



(a) $F_1(X, Y, Z) = \Sigma m(3, 4, 6, 7) = YZ + X\bar{Z}$ (b) $F_2(X, Y, Z) = \Sigma m(0, 2, 4, 5, 6) = \bar{Z} + X\bar{Y}$

□ **FIGURA 2-14**
Mapas para el Ejemplo 2-4

■

Existen en ocasiones formas alternativas de combinar cuadrados para producir expresiones igualmente optimizadas. Un ejemplo de esto se muestra en el mapa de la Figura 2-15. Los minitérminos 1 y 3 se combinan para llegar al término $\bar{X}Z$, y los minitérminos 4 y 6 producen el término $X\bar{Z}$. Sin embargo, hay dos formas de combinar el cuadrado del minitérmino 5 con otro cuadrado adyacente para producir un tercer término de dos literales. Combinándolo con el minitérmino 4 se llega al término $X\bar{Y}$; pero combinándolo con el minitérmino 1 se llega al término $\bar{Y}Z$. Cada una de las dos expresiones, enumeradas en la Figura 2-15, tienen tres términos de dos literales cada uno, de esta forma, hay dos posibles soluciones optimizadas para esta función.

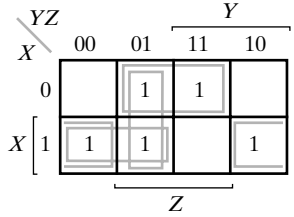
Si una función no se expresa como suma de minitérminos, podemos usar el mapa para obtener los minitérminos de la función y después simplificarla. Sin embargo, es necesario tener la expresión algebraica en forma de suma de productos, a partir de la cual cada término producto se dibuja en el mapa. Como ejemplo considere la función booleana

$$F = \bar{X}Z + \bar{X}Y + X\bar{Y}Z + YZ$$

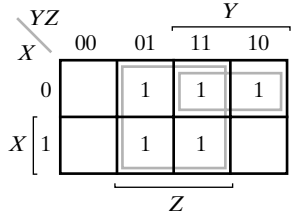
Los tres términos producto de la expresión tienen dos literales y están representados en un mapa de tres variables por dos cuadrados cada uno. Los dos cuadrados correspondientes al primer término, $\bar{X}Z$, se encuentran en la Figura 2-16 donde coinciden $\bar{X}$ (primera columna) y Z (dos columnas medianas), dando lugar a 1 en los cuadrados 001 y 011. Véase que cuando se marcan los 1 en los cuadrados, es posible encontrar un 1 ya puesto por el término precedente. Esto pasa con el segundo término, $\bar{X}Y$, que tiene 1 en los cuadrados 011 y 010; pero el cuadrado 011 lo tiene en común con el primer término, $\bar{X}Z$, así se marca solamente un 1 en él. Continuando de esta manera, encontramos que la función tiene cinco minitérminos, como se indica por los cinco 1 de la figura. Los minitérminos se leen directamente a partir del mapa para ser 1, 2, 3, 5, y 7. La función originalmente dada tiene cuatro términos producto. Puede ser optimizada en el mapa con solamente dos únicos términos

$$F = Z + \bar{X}Y$$

con lo que resulta una reducción significativa del coste de la implementación.



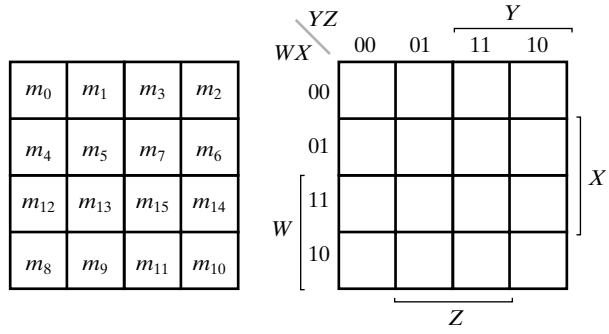
□ **FIGURA 2-15**
$F(X, Y, Z) = \Sigma m(1, 3, 4, 5, 6)$
$= \bar{X}Z + X\bar{Z} + X\bar{Y}$
$= \bar{X}Z + X\bar{Z} + \bar{Y}Z$



□ **FIGURA 2-16**
$F(X, Y, Z) = \Sigma m(1, 2, 3, 5, 7) = Z + \bar{X}Y$

Mapa de cuatro variables

Hay 16 minitérminos para cuatro variables binarias, y por esto, un mapa de cuatro variables está formado por 16 cuadrados, como se indica en la Figura 2-17. La asignación de minitérminos en cada cuadrado se indica en la parte (a) del diagrama. Se dibuja el mapa en (b) otra vez para mostrar la relación de las cuatro variables. Las filas y columnas se enumeran de manera que



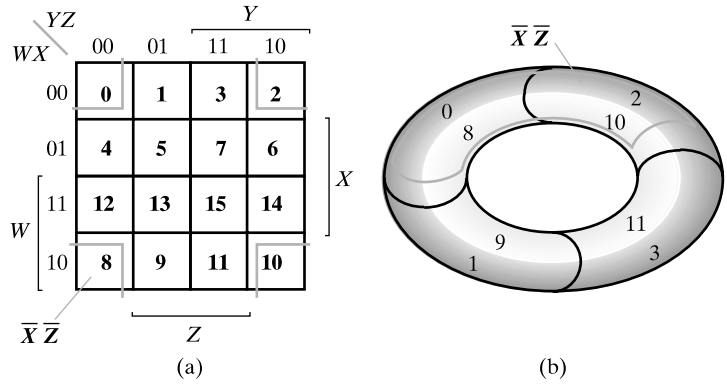
□ **FIGURA 2-17**
Mapa de cuatro variables

sólo un bit del número binario cambia su valor entre cada dos columnas o filas adyacentes, garantizando la misma propiedad para los cuadrados adyacentes. Los números de las filas y las columnas corresponden a un Código Gray de dos bits, como se introdujo en Capítulo 1. Los minitérminos correspondientes a cada cuadrado se pueden obtener combinando los números de la fila y la columna. Por ejemplo, combinando los números de la tercera fila (11) y la segunda columna (01) se obtiene el número binario 1101, el equivalente binario de 13. Así, el cuadrado en la tercera fila, segunda columna, representa el minitérmino m_{13}. Además, se marca cada variable en el mapa para mostrar los ocho cuadrados donde aparecen sin complementar. Los otros ocho cuadrados, donde no se indica ningúna etiqueta, corresponden a la variable que se complementa. Así, *W* aparece complementado en las dos primeras líneas y sin complementar en las dos segundas filas.

El método usado para simplificar funciones de cuatro variables es similar al que se usa para simplificar funciones de tres variables. Como cuadrados adyacentes se definen los que se encuentran uno al lado de otro, como para los mapas de dos y tres variables. Para mostrar las vecindades entre cuadrados, se ha dibujado el mapa de la Figura 2-18(a) como un toro en la Figura 2-18(b), con los bordes superior e inferior, como también los bordes derecho e izquierdo, tocándose mutuamente para mostrar cuadrados adyacentes. Por ejemplo, m_0 y m_2 son dos cuadrados adyacentes, como lo son también m_0 y m_8. Las combinaciones elegibles posibles durante el proceso de optimización en el mapa de cuatro variables son las siguientes:

Un cuadrado representa un minitérmino de cuatro literales.
Un rectángulo de 2 cuadrados representa un término producto de tres literales.
Un rectángulo de 4 cuadrados representa un término producto de dos literales.
Un rectángulo de 8 cuadrados representa un término producto de un literal.
Un rectángulo de 16 cuadrados produce una función que es siempre igual a 1 lógico.

No se puede usar ninguna otra combinación de cuadrados. Un término producto interesante de dos literales, $\overline{X} \cdot \overline{Z}$, se muestra en la Figura 2-18. En (b), donde se ve el mapa como un toro, las vecindades de los cuadrados que representan este término producto quedan claras, pero en (a) estos cuadrados están en las cuatro esquinas del mapa y así aparecen muy lejanos uno de otro. Es importante recordar este término producto, ya que se olvida muchas veces. También sirve como recordatorio de que los bordes izquierdo y derecho del mapa son adyacentes, tanto como los bordes superior e inferior. Así, los rectángulos del mapa cruzan los bordes derecho e izquierdo, superior e inferior, o ambos.



□ **FIGURA 2-18**
Mapa de cuatro variables: Plano y toro para mostrar las vecindades

Los siguientes ejemplos muestran el procedimiento de simplificar funciones booleanas de cuatro variables.

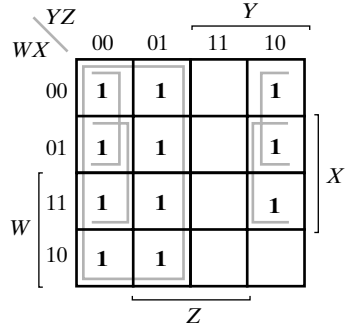
EJEMPLO 2-5 Simplificación de una función de 4 variables mediante un mapa

Simplifique la función booleana

$$F(W, X, Y, Z) = \Sigma m(0, 1, 2, 4, 5, 6, 8, 9, 12, 13, 14)$$

Los minitérminos de la función se han marcado con 1 en el mapa de la Figura 2-19. Los ocho cuadrados de las dos columnas a la izquierda se combinan para formar un rectángulo para el término literal único, $\bar{Y}$. Los tres 1s sobrantes no se pueden combinar para llegar a un término simplificado; más bien, se tienen que combinar como rectángulos de dos o cuatro cuadrados. Los dos 1s superiores de la derecha se combinan con los dos 1s superiores de la izquierda para dar lugar al término $\bar{W}\bar{Z}$. Véase otra vez que está permitido usar el mismo cuadrado más de una vez. Ahora nos queda un cuadrado marcado con 1 en la tercera fila y cuarta columna (minitérmino 1110). En vez de tomar este cuadrado sólo, lo que da lugar a un término de cuatro literales, lo combinamos con cuadrados ya usados para formar un rectángulo de cuatro cuadrados en las dos filas intermedias y las dos columnas finales, resultando el término $X\bar{Z}$. La expresión optimizada es la suma lógica de los tres términos:

$$F = \bar{Y} + \bar{W}\bar{Z} + X\bar{Z}$$



□ **FIGURA 2-19**
Mapa para el Ejemplo 2-5: $F = \bar{Y} + \bar{W}\bar{Z} + X\bar{Z}$

■

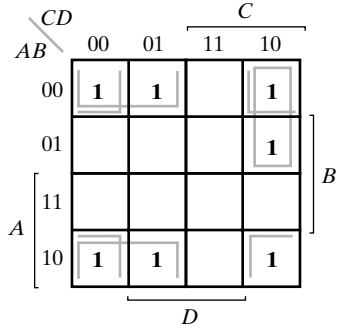
EJEMPLO 2-6 Simplificación de una función de cuatro variables mediante un mapa

Simplifique la función booleana

$$F = \bar{A}\bar{B}\bar{C} + \bar{B}C\bar{D} + A\bar{B}\bar{C} + \bar{A}BC\bar{D}$$

Esta función tiene cuatro variables: *A*, *B*, *C*, y *D*. Se expresa en forma de suma de productos con tres términos de tres literales cada y un término de cuatro literales. El área del mapa cubierto por la función se muestra en la Figura 2-20. Cada término de tres literales se representa en el mapa por dos cuadrados. $\bar{A}\bar{B}\bar{C}$ se representa por los cuadrados 0000 y 0001, $\bar{B}C\bar{D}$ por los cuadrados 0010 y 1010, y $A\bar{B}\bar{C}$ por los cuadrados 1000 y 1001. El término con cuatro literales es el minitérmino 0110. La función está simplificada en el mapa tomando los 1s de las cuatro esquinas, para llegar al término $\bar{B}\bar{D}$. Este término producto está en los mismos sitios del mapa que $\bar{X}\bar{Z}$ de la Figura 2-18. Los dos 1s de la línea superior se combinan con los dos 1s de la fila inferior llevando al término $\bar{B}\bar{C}$. El 1 sobrante, en el cuadrado 0110, se combina con su cuadrado adyacente, 0010, llevando al término $\bar{A}C\bar{D}$. La función optimizada es por tanto

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



□ **FIGURA 2-20**
Mapa para el Ejemplo 2-6: $F = \bar{B}\bar{D} + \bar{B}\bar{C} + \bar{A}C\bar{D}$ ■

2-5 MANIPULACIÓN DEL MAPA

Cuando se combinan los cuadrados de un mapa, es necesario asegurar que se incluyen todos los minitérminos de la función. Al mismo tiempo, es necesario minimizar el número de términos de la función optimizada evitando todos los términos redundantes cuyos minitérminos ya están incluidos en otros términos. En esta sección consideramos un procedimiento que ayuda al reconocimiento de patrones útiles en el mapa. Otros temas a tratar son la optimización de productos de sumas y la optimización de funciones incompletas.

Implicantes primos esenciales

El procedimiento para combinar cuadrados en un mapa se podría hacer de forma más sistemática si presentamos los términos «implicante», «implicante primo» e «implicante primo esencial». Un término producto es un *implicante* de una función si la función vale 1 para todos los minitérminos del término producto. Claramente, todos los rectángulos en un mapa compuestos por cuadrados que contienen 1 corresponden a implicantes. Si se elimina cualquier literal de un implicante *P*, resulta un término producto que no es un implicante de la función, entonces *P* es un

implicante primo. En un mapa para una función de *n*-variables, el conjunto de implicantes primos corresponde al conjunto de todos los rectángulos hechos con 2^m cuadrados que contienen 1 ($m = 0, 1, ..., n$), donde cada rectángulo contiene tantos cuadrados como le sea posible.

Si un minitérmino de una función está incluido en un único implicante primo, este implicante primo se llama *esencial*. Así, si un cuadrado que contiene un 1 está en un sólo rectángulo que representa un implicante primo, entonces este implicante primo es esencial. En la Figura 2-15, los términos $\bar{X}Z$ y $X\bar{Z}$ son implicantes primos esenciales, y los términos $X\bar{Y}$ e $\bar{Y}Z$ son implicantes primos no esenciales.

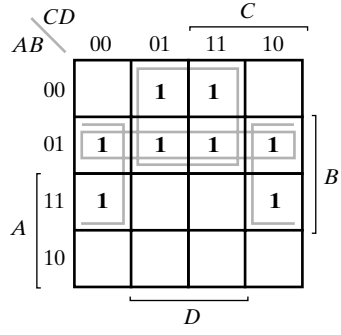
Los implicantes primos de una función se pueden obtener de un mapa de la función como todas las colecciones máximas de 2^m cuadrados que contienen 1 ($m = 0, 1, ..., n$) que constituyen rectángulos. Esto quiere decir que un único 1 en un mapa representa un implicante primo si no es adyacente a cualquier otro 1. Dos 1s adyacentes forman un rectángulo representando un implicante primo, con tal de que no estén dentro de un rectángulo de cuatro o más cuadrados que contengan 1. Cuatro 1s forman un rectángulo que representa un implicante primo si no están dentro de un rectángulo de ocho o más cuadrados que contengan 1, y así sucesivamente. Cada implicante primo esencial contiene al menos un cuadrado que no está dentro de ningún otro implicante primo.

El procedimiento sistemático para encontrar la expresión optimizada del mapa requiere que primero determinemos todos los implicantes primos. Después, se obtiene la expresión optimizada de la suma lógica de todos los implicantes primos esenciales, más otros implicantes primos necesarios para incluir los minitérminos sobrantes que no están incluidos en los implicantes primos esenciales. Este procedimiento se aclarará con ejemplos.

EJEMPLO 2-7 Simplificación usando implicantes primos

Considere el mapa de la Figura 2-21. Hay tres caminos para combinar cuatro cuadrados en rectángulos. Los términos producto obtenidos a partir de estas combinaciones son los implicantes primos de la función, $A\bar{D}$, $B\bar{D}$ y $\bar{A}B$. Los términos $A\bar{D}$ y $B\bar{D}$ son implicantes primos esenciales, pero $\bar{A}B$ no es esencial. Esto es porque los minitérminos 1 y 3 se pueden incluir solamente en el término $\bar{A}D$ y los minitérminos 12 y 14 sólo se puede incluir en el término $B\bar{D}$. Pero los minitérminos 4, 5, 6, y 7 están todos incluidos en dos implicantes primos, uno de ellos es $\bar{A}B$, así el término $\bar{A}B$ no es un implicante primo esencial. De hecho, si se han elegido los implicantes primos esenciales, el tercer término no es necesario porque todos los minitérminos están ya incluidos en los implicantes primos esenciales. La expresión optimizada para la función de la Figura 2-21 es

$$F = \bar{A}D + B\bar{D}$$

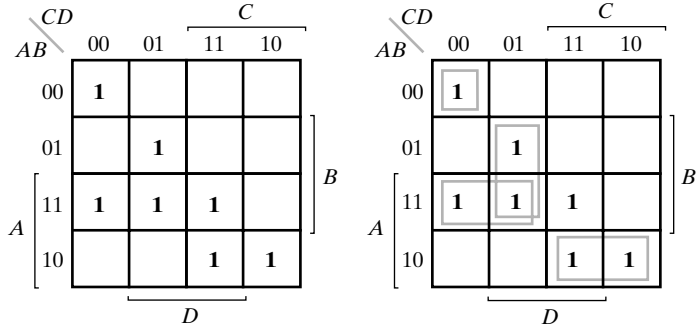


□ **FIGURA 2-21**
Implicantes primos para el Ejemplo 2-7: $\bar{A}D$, $B\bar{D}$, y $A\bar{B}$ ■

EJEMPLO 2-8 Simplificación mediante implicantes primos esenciales y no esenciales

Un segundo ejemplo se muestra en la Figura 2-22. La función dibujada en la parte (a) tiene siete minitérminos. Si intentamos combinar los cuadrados, nos encontramos con seis implicantes primos. Para obtener un número mínimo de términos de la función, tenemos que determinar primero los implicantes primos que son esenciales. Como se muestra en la parte (b) de la figura, la función tiene cuatro implicantes primos esenciales. El término producto $\bar{A}\bar{B}CD$ es esencial porque es el único implicante primo que incluye el minitérmino 0. De igual manera, los términos producto $B\bar{C}D$, $AB\bar{C}$ y $A\bar{B}C$ son implicantes primos esenciales porque son los únicos implicantes primos que incluyen a los minitérminos 5, 12 y 10, respectivamente. El minitérmino 15 está incluido en dos implicantes primos no esenciales. La expresión optimizada para la función consiste en la suma lógica de los cuatro implicantes primos esenciales y un implicante primo que incluye el minitérmino 15:

$$F = \bar{A}\bar{B}\bar{C}\bar{D} + B\bar{C}D + AB\bar{C} + A\bar{B}C + \begin{pmatrix} ACD \\ \text{o} \\ ABD \end{pmatrix}$$



(a) Dibujando los minitérminos (b) Implicantes primos esenciales

□ **FIGURA 2-22**
Simplificación con los implicantes primos del Ejemplo 2-8 ■

La identificación de los implicantes primos esenciales en el mapa proporciona una herramienta adicional que muestra los términos que tienen que aparecer necesariamente en cada expresión de suma de productos de una función y proporciona una estructura parcial para un método más sistemático de elegir patrones de cuadrados.

Implicantes primos no esenciales

Más allá de usar todos los implicantes primos esenciales, se puede aplicar la siguiente regla para incluir los minitérminos restantes de la función en implicantes primos no esenciales:

Regla de selección: minimice el solapamiento entre implicantes primos cuanto sea posible. En particular, en la solución final, asegúrese de que cada implicante primo seleccionado incluye al menos un minitérmino que no está incluido en algún otro implicante primo seleccionado.

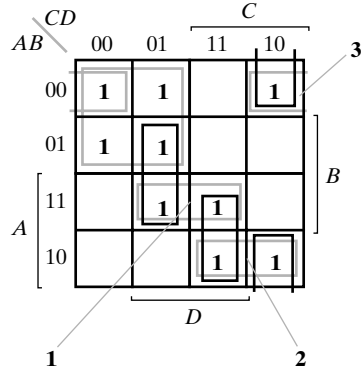
En la mayoría de los casos, esto da lugar a una expresión de suma de producto simplificada, aunque no necesariamente de coste mínimo. El uso de la regla de selección se ilustra en el siguiente ejemplo.

EJEMPLO 2-9 Simplificación de una función usando la regla de selección

Encuentre una forma simplificada de suma de productos para $F(A, B, C, D) = \Sigma m(0, 1, 2, 4, 5, 10, 11, 13, 15)$.

El mapa de F se presenta en la Figura 2-23, mostrando todos los implicantes primos. $\bar{A}\bar{C}$ es el único implicante primo esencial. Usando la anterior regla de selección, podemos elegir los implicantes primos sobrantes para la forma de suma de productos en el orden indicado por los números. Véase como se seleccionan los implicantes primos 1 y 2 para incluir minitérminos sin solapamiento. El implicante primo 3 ($\bar{A}\bar{B}\bar{D}$) y el implicante primo $\bar{B}C\bar{D}$, ambos incluyen el minitérmino 0010 sobrante, el implicante primo 3 se selecciona arbitrariamente para incluir el minitérmino y completar la expresión de suma de productos:

$$F(A, B, C, D) = \bar{A}\bar{C} + ABD + A\bar{B}C + \bar{A}\bar{B}\bar{D}$$



□ **FIGURA 2-23**
Mapa para el Ejemplo 2-9 ■

Optimización de producto de sumas

Las funciones booleanas optimizadas derivadas de los mapas en todos los ejemplos previos han sido expresadas en forma de suma de productos. Con pequeñas modificaciones se puede obtener la forma de producto de sumas.

El procedimiento para obtener una expresión optimizada en forma de producto de suma sale de las propiedades básicas de las funciones booleanas. Los 1s colocados en los cuadrados del mapa representan los minitérminos de la función. Los minitérminos que no están incluidos en la función pertenecen al complemento de la función. De esto vemos que el complemento de una función se representa en el mapa por los cuadrados no marcados con 1. Si marcamos los cuadrados vacíos con 0 y los combinamos en rectángulos válidos, obtenemos una expresión optimizada del complemento de la función. Entonces, tomamos el complemento de $\bar{F}$ para obtener la función F como producto de sumas. Esto se hace tomando el dual y complementando cada literal, como se ha descrito en el Ejemplo 2-2.

EJEMPLO 2-10 Simplificación de una forma de productos de sumas

Simplifique la siguiente función booleana en forma de productos de sumas:

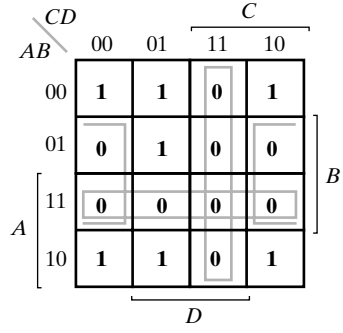
$$F(A, B, C, D) = \Sigma m(0, 1, 2, 5, 8, 9, 10)$$

Los 1s marcados en el mapa de la Figura 2-24 representan los minitérminos de la función. Los cuadrados marcados con 0 representan los minitérminos no incluidos en F y así denotan el complemento de F. Combinando los cuadrados marcados con 0, obtenemos la función complementada optimizada

$$\bar{F} = AB + CD + B\bar{D}$$

Tomando el dual y complementando cada literal se obtiene el complemento de $\bar{F}$. Así es F en forma de producto de sumas:

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



□ **FIGURA 2-24**
Mapa para el Ejemplo 2-10: $F = (\bar{A} + \bar{B})(\bar{C} + \bar{D})(\bar{B} + D)$ ■

El ejemplo anterior muestra el procedimiento para obtener la optimización en producto de sumas cuando la función originalmente se expresa como suma de minitérminos. El procedimiento también es válido si la función se expresa originalmente como producto de maxitérminos o como producto de sumas. Recuerde que los números del maxitérmino son los mismos que los números del minitérmino de la función complementada, de esta forma se introducen 0s en el mapa para los maxitérminos o para el complemento de la función. Para introducir una función expresada como producto de sumas en el mapa, tomamos el complemento de la función y, de allí, encontramos los cuadrados que hay que marcar con 0. Por ejemplo, se puede dibujar la función

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

en el mapa obteniendo primero el complemento,

$$\bar{F} = AB\bar{C} + \bar{B}\bar{D}$$

y después marcando los 0s en los cuadrados que representan los minitérminos de $\bar{F}$. Los cuadrados sobrantes se marcan con 1. Después, combinando los 1s se llega a la expresión optimizada en forma de suma de productos. Combinando los 0s y después calculando el complemento resulta la expresión optimizada en forma de producto de sumas. Así, para cada función dibujada en el mapa, podemos derivar la función optimizada en cualquiera de las dos formas canónicas.

Condiciones de indiferencia

Los minitérminos de una función booleana especifican todas las combinaciones de valores de variables para los que la función es igual a 1. Se supone que la función es igual a 0 para el resto de los minitérminos. Sin embargo, esta suposición no es siempre válida, ya que hay aplicaciones en las que la función no está especificada para ciertas combinaciones de valores de las variables. Hay dos casos donde ocurre esto. En el primer caso, las combinaciones de entrada no ocurren nunca. Como ejemplo, el código binario de cuatro bits para los dígitos decimales tiene seis combinaciones que no se usan y que no se espera que ocurran. En el segundo caso, las combinaciones de entrada se espera que ocurran, pero el valor de la salida como respuesta a estas combinaciones no importa. En ambos casos se dice que no se han especificado las salidas para estas combinaciones. Las funciones que tienen salidas sin especificar para algunas combinaciones de entradas se llaman *funciones incompletamente especificadas*. En la mayoría de las aplicaciones, simplemente nos da igual qué valor asume la función para los minitérminos no especificados. Por esta razón, es usual llamar los minitérminos no especificados *condiciones de indiferencia*. Estas condiciones se pueden usar en un mapa para proporcionar la función más simplificada.

Habría que darse cuenta de que un minitérmino indiferente no se puede marcar con un 1 en el mapa, porque esto implicaría que la función sería siempre 1 para uno de estos minitérminos. Asimismo, poniendo un 0 en el cuadrado implica que la función es 0. Para distinguir la condición de indiferencia de 1 y 0, se usa una *X*. Así, una *X* dentro de un cuadrado en el mapa indica que no nos importa si está asignado el valor de 0 o 1 a la función para un minitérmino en particular.

Se pueden usar los minitérminos de indiferencia para simplificar la función en un mapa. Cuando se simplifica la función *F*, usando los 1s podemos elegir si incluimos los minitérminos de indiferencia que resultan del implicante primo más simple de *F*. Cuando se simplifica la función $\bar{F}$ usando los 0s, podemos elegir si incluimos los minitérminos de indiferencia que resultan de los implicantes primos más sencillos de $\bar{F}$, independientemente de los que están incluidos en los implicantes primos de *F*. En ambos casos, es irrelevante si los minitérminos de indiferencia están incluidos o no en los términos de la expresión final. El manejo de condiciones de indiferencia se demuestra en el siguiente ejemplo.

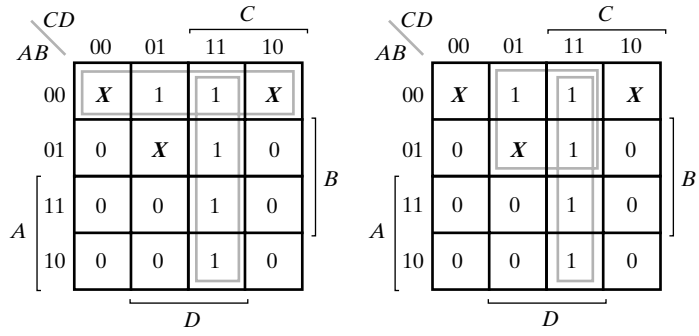
EJEMPLO 2-11 Simplificación con condiciones de indiferencia

Para clarificar el procedimiento en el manejo de condiciones de indiferencia, considere la siguiente función *F* que no está completamente especificada, que tiene tres minitérminos de indiferencia *d*:

$$F(A, B, C, D) = \Sigma m(1, 3, 7, 11, 15)$$

$$d(A, B, C, D) = \Sigma m(0, 2, 5)$$

Los minitérminos de *F* son las combinaciones de variables que igualan la función a 1. Los minitérminos de *d* son minitérminos de indiferencia. La optimización del mapa se muestra en la Figura 2-25. Los minitérminos de *F* están marcados con 1, los de *d* están marcados con *X*, y los cuadrados sobrantes se han rellenado con 0. Para conseguir la función simplificada en forma de suma de productos, tenemos que incluir los cinco 1s en el mapa, pero podemos o no incluir alguna de las *X*, dependiendo de que se produzca la expresión más sencilla de la función. El término *CD* incluye los cuatro minitérminos en la tercera columna. Los minitérminos sobrantes



□ **FIGURA 2-25**
Ejemplo con condiciones de indiferencia

en el cuadrado 0001 se puede combinar con el cuadrado 0011 para dar lugar a un término de tres literales. Sin embargo, incluyendo una o dos *X* adyacentes, podemos combinar cuatro cuadrados en un rectángulo para llegar a un término de dos literales. En la parte (a) de la figura, los minitérminos de indiferencia 0 y 2 están incluidos con los 1s, lo cual da lugar a la función simplificada

$$F = CD + \bar{A}\bar{B}$$

En la parte (b), el minitérmino de indiferencia 5 está incluido con los 1s, y la función simplificada es ahora

$$F = CD + \bar{A}D$$

Las dos expresiones representan dos funciones que son algebraicamente diferentes. Ambas incluyen los minitérminos especificados en la función original incompletamente especificada, pero cada uno incluye diferentes minitérminos de indiferencia. Por lo que respecta a la función incompletamente especificada, ambas expresiones son aceptables. La única diferencia está en el valor de *F* para los minitérminos no especificados.

También es posible obtener una expresión optimizada de producto de sumas para la función de la Figura 2-25. En este caso, la manera de combinar los 0s es incluir los minitérminos de indiferencia 0 y 2 con los 0s, resultando la función optimizada complementada

$$\bar{F} = \bar{D} + A\bar{C}$$

Tomando el complemento de $\bar{F}$ resulta la expresión optimizada en forma de producto de sumas:

$$F = D(\bar{A} + C)$$ ■

El ejemplo anterior muestra que inicialmente se consideran los minitérminos de indiferencia en el mapa representando ambos 0 y 1. El valor 0 o 1 finalmente asignado depende del proceso de optimización. Debido a este proceso, la función optimizada tendrá el valor 0 o 1 para cada minitérmino de la función original, incluyendo los que inicialmente eran indiferentes. Así, aunque las salidas de la especificación inicial podrían contener *X*, las salidas en una implementación particular de la especificación son solamente 0 y 1.

2-6 OPTIMIZACIÓN DE CIRCUITOS MULTINIVEL

Aunque hemos averiguado que la optimización de circuitos de dos niveles puede reducir el coste de los circuitos lógicos combinacionales, muchas veces se puede ahorrar más costes usando circuitos con más de dos niveles. A estos circuitos se les llama circuitos multinivel. Este ahorro se demuestra mediante la implementación de la función:

$$G = ABC + ABD + E + ACF + ADF$$

La Figura 2-26(a) muestra la implementación de dos niveles de G que tiene un coste de 17 entradas de puerta. Ahora suponemos que aplicamos la ley distributiva del Álgebra de Boole a G para conseguir:

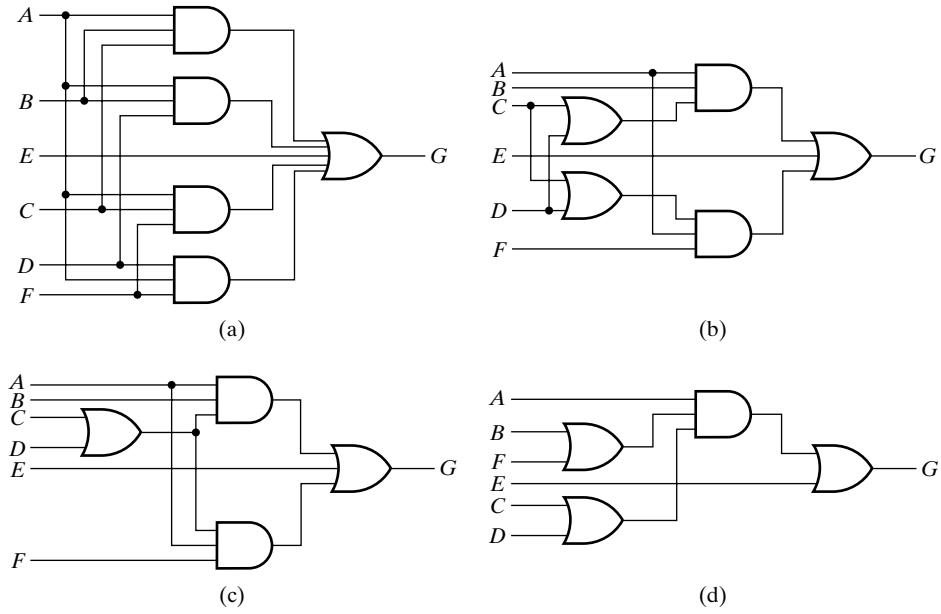
$$G = AB(C + D) + E + A(C + D)F$$

Esta ecuación da lugar a la implementación en varios niveles de G de la Figura 2-26(b) que tiene un coste de 13 entradas de puerta, con una mejora de 4 entradas de puerta. En la Figura 2-26(b), $C + D$ se ha implementado dos veces. En cambio, una implementación de esta subfunción puede compartirse para dar lugar al circuito de la Figura 2-26(c) con un coste de 11 entradas de puerta, con una mejora de 2. Este uso de $(C + D)$ sugiere que se puede escribir G como

$$G = (AB + AF)(C + D) + E$$

Esto incrementa el coste a 12. Pero sacando el factor A de $AB + AF$, obtenemos

$$G = A(B + F)(C + D) + E$$



□ **FIGURA 2-26**
Ejemplo de un circuito multinivel

La Figura 2-26(d) muestra la implementación en varios niveles de *G* usando esta ecuación que tiene un coste de entradas de puerta de solo nueve, que es ligeramente más que la mitad del coste original.

Esta reducción se ha obtenido mediante una secuencia de aplicaciones de identidades algebraicas, observando en cada paso el efecto en el coste de las entradas de puerta. Sólo usando el álgebra de Boole para obtener circuitos simplificados de dos niveles, el procedimiento usado aquí no es realmente sistemático. Además, no existe ningún procedimiento algorítmico para usar los Mapas de Karnaugh para la optimización de circuitos de dos niveles que tengan un coste óptimo, debido al amplio rango de acciones posibles y al número de soluciones posibles. Así la optimización en múltiples niveles se basa en el uso de un conjunto de las transformaciones aplicadas junto con la evaluación del coste para encontrar una solución buena pero no necesariamente óptima. Para el resto de esta sección, tenemos en cuenta estas transformaciones e ilustramos su aplicación para reducir el coste del circuito. Las transformaciones, mostradas en el siguiente ejemplo, se definen como:

1. *Factorización*: es encontrar una forma factorizada de una expresión de suma de productos o de producto de sumas para una función.
2. *Descomposición*: es la expresión de una función como un conjunto de funciones nuevas.
3. *Extracción*: es la expresión de varias funciones como un conjunto de nuevas funciones.
4. *Substitución* de una función *G* por una función *F* es expresar *F* como función de *G* y de algunas o todas las variables originales de *F*.
5. *Eliminación*: es la inversa de la sustitución donde la función *G* dentro de una expresión de función *F* se sustituye por la expresión de *G*. A la eliminación también se le llama *flattening* (aplanar) o *collapsing* (colapsar).

EJEMPLO 2-12 Transformaciones de optimización de múltiples niveles

Se usarán las siguientes funciones para ilustrar las transformaciones:

$$G = A\bar{C}E + A\bar{C}F + A\bar{D}E + A\bar{D}F + BCD\bar{E}\bar{F}$$
$$H = \bar{A}BCD + ACE + ACF + BCE + BCF$$

La primera transformación a mostrar es la factorización usando la función *G*. Inicialmente, miramos a la *factorización algebraica*, que evita axiomas que son únicos en el Álgebra de Boole, tal como los que incluyen el complemento y la idempotencia. Se pueden encontrar los factores no solamente para la expresión entera de *G*, sino también para sus subexpresiones. Por ejemplo, ya que los primeros cuatro términos de *G* contienen la variable *A*, se puede sacar fuera de estos términos dando lugar a en:

$$G = A(\bar{C}E + \bar{C}F + \bar{D}E + \bar{D}F) + BCD\bar{E}\bar{F}$$

En este caso, véase que A y $\bar{C}E + \bar{C}F + \bar{D}E + \bar{D}F$ son factores, y $BCD\bar{E}\bar{F}$ no está involucrado en la operación de factorización. Sacando los factores $\bar{C}$ y $\bar{D}$, $\bar{C}E + \bar{C}F + \bar{D}E + \bar{D}F$ se puede escribir como $\bar{C}(E + F) + \bar{D}(E + F)$ lo que, además, se puede escribir como $(\bar{C} + \bar{D})(E + F)$. La integración de esta expresión en *G* da como resultado:

$$G = A(\bar{C} + \bar{D})(E + F) + BCD\bar{E}\bar{F}$$

El término $BCD\bar{E}\bar{F}$ se podría factorizar en términos producto, pero esta factorización no reduciría el número de entradas de puertas y por tanto no se tiene en cuenta. El número de entradas

por puerta en la expresión original de la suma de productos para G es 26 y en la forma factorizada de G es 18, ahorrándose 8 entradas de puerta. Debido a esta factorización, hay más puertas en serie desde las entradas hasta las salidas, un máximo de cuatro niveles en vez de tres niveles incluyendo los inversores de entrada. Esto daría lugar a un incremento del retardo del circuito después de aplicar un mapeo tecnológico.

La segunda transformación que se muestra es la descomposición que permite operaciones más allá de la factorización algebraica. La forma factorizada de G se puede escribir como una descomposición, según sigue a continuación:

$$G = A(\bar{C} + \bar{D})X_2 + BX_1\bar{E}\bar{F}$$
$$X_1 = CD$$
$$X_2 = E + F$$

Una vez que X_1 y X_2 se han definido, se puede calcular el complemento, y los complementos pueden reemplazar $\bar{C} + \bar{D}$ y $\bar{E}\bar{F}$, respectivamente, en G. Una ilustración de la transformación de sustitución es

$$G = A\bar{X}_1X_2 + BX_1\bar{X}_2$$
$$X_1 = CD$$
$$X_2 = E + F$$

El número de entradas de puertas para esta descomposición es 14, dando lugar a un ahorro de 12 entradas de puerta de la expresión original de suma de productos para G, y de 4 puertas de entrada de la forma factorizada de G.

Para ilustrar la extracción, necesitamos realizar la descomposición en H y extraer subexpresiones comunes en G y H. Sacando el factor B de H, tenemos

$$H = B(\bar{A}CD + AE + A + CE + CF)$$

Determinando factores adicionales en H, podemos escribir

$$H = B(\bar{A}(CD) + (A + C)(E + F))$$

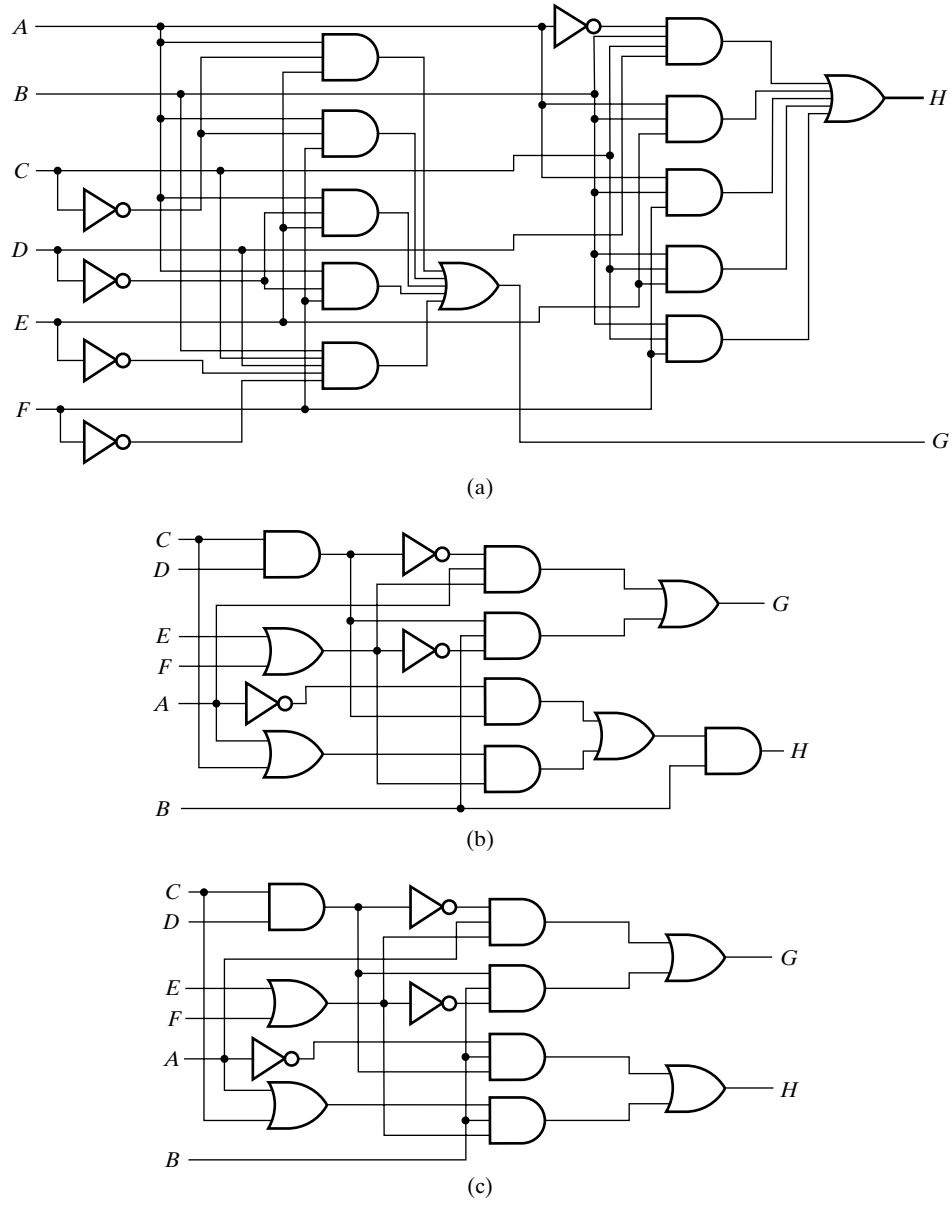
Ahora los factores X_1, X_2, y X_3 se puede extraer para obtener

$$X_1 = CD$$
$$X_2 = E + F$$
$$X_3 = A + C$$

y los factores X_1 y X_2 se puede compartir entre G y H. Realizando la sustitución, podemos escribir G y H como

$$G = A\bar{X}_1X_2 + BX_1\bar{X}_2$$
$$H = B(\bar{A}X_1 + X_3X_2)$$

Se da un diagrama lógico para la suma de productos original en la Figura 2-27(a) y para la forma extraída en la Figura 2-27(b). El coste de entradas de puerta para las funciones originales G y H, sin términos compartidos, excepto para los inversores de entrada, es de 48. Para G y H



□ **FIGURA 2-27**
Ejemplo de optimización de un circuito de múltiples niveles

en forma decompuesta, sin términos compartidos entre *G* y *H*, es de 31. Con términos compartidos es de 25, dividiendo el coste de entradas de puertas de entrada a la mitad. ■

Este ejemplo muestra el valor de las transformaciones reduciendo el coste de entradas. En general, debido al amplio rango de soluciones alternativas y la complejidad de determinar divisores para usar en la descomposición y la extracción, la obtención de soluciones realmente óptimas, en cuanto al coste de entradas, no es factible normalmente, así que solamente se buscan soluciones buenas.

La clave para realizar transformaciones con éxito es la determinación de los factores que se usan en la descomposición o extracción y la elección de la secuencia aplicada en la transformación. Esas decisiones son complejas y fuera del ámbito de nuestros estudios, pero normalmente se incorporan en las herramientas de síntesis lógica.

Nuestra discusión, hasta ahora, trataba solamente de la optimización en varios niveles en términos de reducir el número de entradas de puertas. En muchos diseños, la longitud de la ruta o rutas más largas por el circuito se restringe por el retardo en la ruta, el tiempo que tarda en propagarse el cambio de una señal por un camino a través de las puertas. En estos casos podría ser necesario reducir el número de puertas en serie. Una reducción así, usando la transformación final, eliminación, se muestra en el siguiente ejemplo.

EJEMPLO 2-13 Ejemplo de transformación para la reducción del retardo

En el circuito de la Figura 2-27(b), las rutas de *C, D, E, F* y *A* a *H* pasan por cuatro puertas de 2 entradas. Suponiendo que todas las puertas, con diferente número de entradas, contribuyen con el mismo retardo en la ruta (retardo mayor que el de un inversor) estas son las rutas más lentas del circuito. Debido a una especificación de un retardo de ruta máximo para un circuito, hay que acortar estos caminos en al menos tres puertas de varias entradas o su equivalente en retardos de puertas de varias entradas o de inversores. Este acortamiento de las rutas se debería hacer con un incremento mínimo del número de entradas de puerta.

La transformación de eliminación que reemplaza variables intermediarias, X_i, con las expresiones a su lado derecho o elimina otra factorización como la de la variable *B*, es el mecanismo para reducir el número de puertas en serie. Para determinar qué factor o combinación de factores se debería eliminar, tenemos que contemplar el efecto en el número de entradas de puerta. El incremento en el número de entradas de puertas para las combinaciones de eliminaciones que reducen las longitudes de rutas problemáticas en al menos una puerta de entrada son interesantes. Hay solamente tres de estas combinaciones: eliminación de la factorización de *B*, eliminación de variables intermediarias X_1, X_2, y X_3, y eliminación del factor *B* y las tres variables intermediarias X_1, X_2, y X_3. Los incrementos en los números de entradas de puerta respectivos para estas acciones son 0, 12, y 12, respectivamente. Claramente, la eliminación del factor *B* es la mejor elección ya que el número de entradas de puerta no se incrementa. Esto también demuestra que, debido a la descomposición adicional de *H*, la ganancia del coste de entradas de puerta de 3, que ocurrió sacando el factor *B* al principio, ha desaparecido. El diagrama lógico resultante de la eliminación del factor *B* se muestra en la Figura 2-27(c). ■

Mientras que la reducción necesaria del retardo se ha obtenido usando la eliminación para reducir el número de puertas a lo largo de la ruta en el Ejemplo 2-13, en general, tal reducción de puerta podría no reducir el retardo, o incluso podría incrementarlo debido a las diferencias en las características de retardo de las puertas que se discutirá más adelante en el Capítulo 3.

2-7 OTROS TIPOS DE PUERTAS

Ya que las funciones booleanas se expresan en términos de operaciones AND, OR, y NOT, éstas constituyen un procedimiento directo para implementar una función booleana con puertas AND, OR, y NOT. Sin embargo, encontramos que la posibilidad de considerar puertas con otras operaciones lógicas es de un considerable interés práctico. Los factores que hay que considerar al construir otros tipos de puertas son la viabilidad y economía de implementar la puerta con

componentes electrónicos, la capacidad de la puerta para implementar funciones booleanas por si solas o en combinación con otras puertas, y la utilidad de representar puertas funcionales que se usan frecuentemente. En esta sección introducimos estos otros tipos de puertas que se usan a lo largo del resto del texto. Las técnicas específicas para la incorporación de estos tipos de puertas en los circuitos se muestran en la Sección 3-5.

Los símbolos gráficos y las tablas de verdad de las seis puertas lógicas se muestra en la Figura 2-28, con seis tipos de puertas adicionales mostradas en la Figura 2-29. A las puertas de la Figura 2-28 se las denomina como puertas *primitivas*, y a las de la Figura 2-29 como puertas *complejas*.

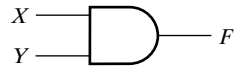
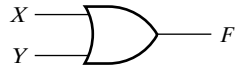
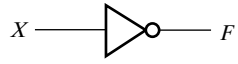
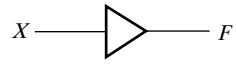
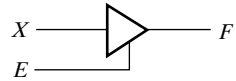
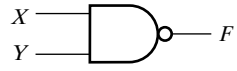
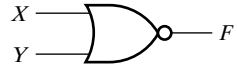
Aunque las puertas de la Figura 2-28 se muestra solamente con dos variables binarias de entrada, *X* e *Y*, y una variable binaria de salida, *F*, con la excepción del inversor y el *buffer*, todos podrían tener más de dos entradas. Las diferentes formas de los símbolos mostrados, tanto como los símbolos rectangulares no mostrados, se especifica en detalle en el «*Institute of Electrical y Electronics Engineers*» (IEEE) *Standard Graphic Symbols for Logic Funcions* (IEEE Standard 91-1984). Las puertas AND, OR, y NOT se han definido anteriormente. El circuito NOT invierte el sentido lógico de una señal binaria para producir una operación de complemento. Recuerde que a este circuito se le llama típicamente *inversor* en vez de puerta NOT. El círculo pequeño a la salida del símbolo gráfico de un inversor se llama formalmente un *indicador de negación* y designa el complemento lógico. Informalmente nos referimos al indicador de negación como a una «burbuja». El mismo símbolo triangular designa un circuito *buffer*. Un *buffer* produce la función lógica $Z = X$, ya que el valor binario de la salida es igual al valor binario de la entrada. Este circuito se usa principalmente para amplificar una señal eléctrica para permitir que más puertas sean conectadas a la salida y se reduce el tiempo de propagación de las señales por el circuito.

El *buffer* triestado es único en el sentido de que se pueden conectar sus salidas entre ellos, con tal de que una sola de las señales de sus entradas *E* sea 1 en un momento dado. Este tipo de *buffer* y su uso básico se discuten en detalle más tarde en esta sección.

La puerta NAND representa el complemento de la operación AND, y la puerta NOR representa el complemento de la operación OR. Los nombres respectivos son abreviaturas de NOT-AND y NOT-OR, respectivamente. Los símbolos gráficos para la puerta NAND y la puerta NOR están compuestos por el símbolo de la puerta AND y el símbolo de la puerta OR, respectivamente, con una burbuja en la salida, que indica el complemento de la operación. En las tecnologías actuales de circuitos integrados, las puertas NAND y NOR son las funciones primitivas naturales para los circuitos más simples y más rápidos. Si consideramos el inversor como una versión degenerada de las puertas NAND y NOR con solamente una entrada, las puertas NAND o puertas NOR, por si mismas, pueden implementar cualquier función booleana. Así, estos tipos de puertas se usan mucho más que las puertas AND y OR en los circuitos lógicos actuales. Como consecuencia, las implementaciones de los circuitos actuales se realiza muchas veces en térinos de este tipo de puertas.

Un tipo de puerta que se puede usar únicamente para implementar todas las funciones booleanas se llama una *puerta universal*. Para mostrar que la puerta NAND es una puerta universal, solo tenemos que demostrar que se pueden obtener las operaciones lógicas de AND, OR, y NOT usando solamente puertas NAND. Esto se ha realizado en la Figura 2-30. La operación complemento obtenida de una puerta NAND con una entrada corresponde a una puerta NOT. De hecho, la puerta NAND de una entrada es un símbolo no válido y se sustituye por un símbolo NOT, como se muestra en la figura. La operación AND requiere una puerta NAND seguida de una puerta NOT. El NOT invierte la salida de la NAND, resultando una operación AND. La operación OR se logra usando una puerta NAND con un NOT en cada entrada. Si se aplica el

Símbolos gráficos

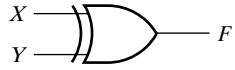
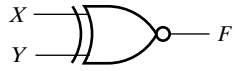
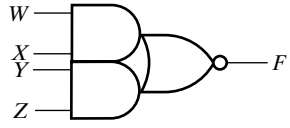
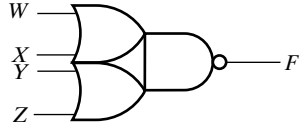
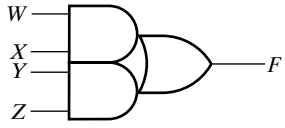
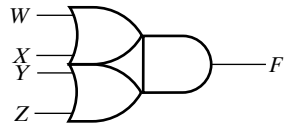
Nombre	Símbolo	Ecuación algebraica	Tabla de verdad
AND		$F = XY$	$X\ Y \mid F$ $0\ 0 \mid 0$ $0\ 1 \mid 0$ $1\ 0 \mid 0$ $1\ 1 \mid 1$
OR		$F = X + Y$	$X\ Y \mid F$ $0\ 0 \mid 0$ $0\ 1 \mid 1$ $1\ 0 \mid 1$ $1\ 1 \mid 1$
NOT (inversor)		$F = \overline{X}$	$X \mid F$ $0 \mid 1$ $1 \mid 0$
Buffer		$F = X$	$X \mid F$ $0 \mid 0$ $1 \mid 1$
Buffer triestado			$E\ X \mid F$ $0\ 0 \mid$ Hi-Z $0\ 1 \mid$ Hi-Z $1\ 0 \mid 0$ $1\ 1 \mid 1$
NAND		$F = \overline{X \cdot Y}$	$X\ Y \mid F$ $0\ 0 \mid 1$ $0\ 1 \mid 1$ $1\ 0 \mid 1$ $1\ 1 \mid 0$
NOR		$F = \overline{X + Y}$	$X\ Y \mid F$ $0\ 0 \mid 1$ $0\ 1 \mid 0$ $1\ 0 \mid 0$ $1\ 1 \mid 0$

□ **FIGURA 2-28**
Puertas lógicas digitales primitivas

Teorema de DeMorgan, como se muestra en la Figura 2-30, las inversiones se anulan y resulta una función OR.

La puerta OR exclusiva (XOR) mostrada en la Figura 2-29 es similar a la puerta OR, pero excluye (tiene el valor 0 para) la combinación con ambas entradas X e Y iguales a 1. El símbolo gráfico para la puerta XOR es similar al de la puerta OR, excepto en la línea curvada adicional en las entradas. La OR exclusiva tiene el símbolo especial $\oplus$ para designar esta operación. La

Símbolos gráficos

Nombre	Símbolo	Ecuación algebraica	Tabla de verdad
OR exclusiva (XOR)		$F = X\overline{Y} + \overline{X}Y$ $= X \oplus Y$	$X\ Y \mid F$ $0\ 0 \mid 0$ $0\ 1 \mid 1$ $1\ 0 \mid 1$ $1\ 1 \mid 0$
NOR exclusiva (XNOR)		$F = XY + \overline{X}\overline{Y}$ $= \overline{X \oplus Y}$	$X\ Y \mid F$ $0\ 0 \mid 1$ $0\ 1 \mid 0$ $1\ 0 \mid 0$ $1\ 1 \mid 1$
AND-OR-INVERSOR (AOI)		$F = \overline{WX + YZ}$	
OR-AND -INVERSOR (OAI)		$F = \overline{(W + X)(Y + Z)}$	
AND-OR (AO)		$F = WX + YZ$	
OR-AND (OA)		$F = (W + X)(Y + Z)$	

□ **FIGURA 2-29**
Puertas lógicas digitales complejas primitivas

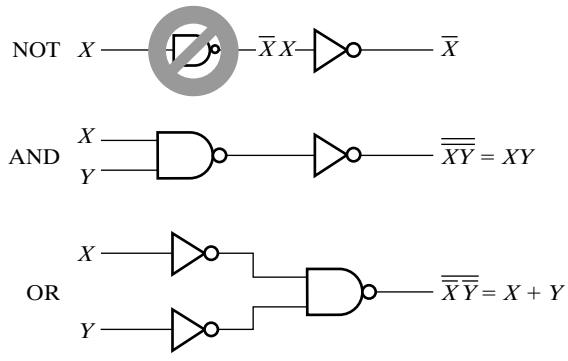
NOR exclusiva es el complemento de la OR exclusiva, como se indica con la burbuja a la salida de su símbolo gráfico.

La puerta AND-OR-INVERT (AOI) forma el complemento de una suma de productos. Hay muchas puertas diferentes AND-OR-INVERT dependiendo del número de puertas AND y el número de entradas en cada AND y sus salidas van conectadas directamente a la puerta OR. Por ejemplo, suponga que la función implementada por un AOI es

$$F = \overline{XY + Z}$$

A esta AOI se le denomina como una AOI 2-1 ya que consiste en una AND de 2 entradas y una entrada directa a la puerta OR. Si la función implementada es

$$F = \overline{TUV + WX + YZ}$$



□ **FIGURA 2-30**
Operaciones lógicas con puertas NAND

entonces a la AOI se le llama AOI 3-2-2. LA OR-AND-INVERT (OAI) es la dual de la AOI e implementa el complemento en forma de producto de sumas. La AND-OR (AO) y OR-AND (OA) son versiones de AOI y OAI sin el complemento.

En general, las puertas complejas se usan para reducir la complejidad del circuito necesario para la implementación de funciones específicas de Boole con el fin de reducir costes del circuito integrado. Además, reducen el tiempo necesario para la propagación de señales por el circuito.

CIRCUITOS CMOS Este suplemento, que discute la implementación de puertas primitivas y complejas en tecnología CMOS, está disponible en la página web del texto: www.librosite.net/Mano.

2-8 OPERADOR Y PUERTAS OR EXCLUSIVA

Además de la puerta de OR exclusiva mostrada en la Figura 2-29, hay un operador de OR exclusivo con sus identidades algebraicas propias. El operador OR exclusivo, denotado por $\oplus$, es una operación lógica que ejecuta la función

$$X \oplus Y = X\bar{Y} + \bar{X}Y$$

Es igual a 1 si sólo una variable de entrada es igual a 1. El operador NOR exclusivo, también conocido como *equivalencia*, es el complemento del OR exclusivo y se expresa mediante la función

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

Es igual a 1 si ambas entradas, X e Y, son iguales a 1 o si ambas entradas son iguales a 0. Se puede demostrar que las dos funciones son complementos una de la otra, tanto por medio de la tabla de verdad o, como sigue a continuación, por manipulación algebraica:

$$\overline{X \oplus Y} = \overline{X\bar{Y} + \bar{X}Y} = (\bar{X} + Y)(X + \bar{Y}) = XY + \bar{X}\bar{Y}$$

Las siguientes identidades se pueden aplicar a la operación OR exclusiva:

$$X \oplus 0 = X \qquad X \oplus 1 = \bar{X}$$
$$X \oplus X = 0 \qquad X \oplus \bar{X} = 1$$
$$X \oplus \bar{Y} = \overline{X \oplus Y} \qquad \bar{X} \oplus Y = \overline{X \oplus Y}$$

Se puede verificar cada una de estas identidades usando una tabla de verdad o reemplazando la operación $\oplus$ por su expresión booleana equivalente. También se puede mostrar que la operación OR exclusiva es tanto conmutativa como asociativa; o sea,

$$A \oplus B = B \oplus A$$

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

Esto significa que las dos entradas a una puerta de OR exclusiva pueden ser intercambiadas sin tener efecto en la operación. También significa que podemos evaluar una operación OR exclusiva de tres variables en cualquier orden y por esa razón, se pueden expresar las ORs exclusivas con tres o más variables sin paréntesis.

Una función OR exclusiva de dos entradas se puede construir con puertas convencionales. Se usan dos puertas NOT, dos puertas AND, y una puerta OR. La asociatividad del operador OR exclusivo sugiere la posibilidad de puertas de OR exclusivas con más que dos entradas. Sin embargo, el concepto del OR exclusivo para más de dos variables se reemplaza por la función impar discutida a continuación. Por esto, no hay ningún símbolo para la OR exclusiva de más de dos entradas. Por dualidad, se reemplaza la NOR exclusiva por la función par y no tiene ningún símbolo para más de dos entradas.

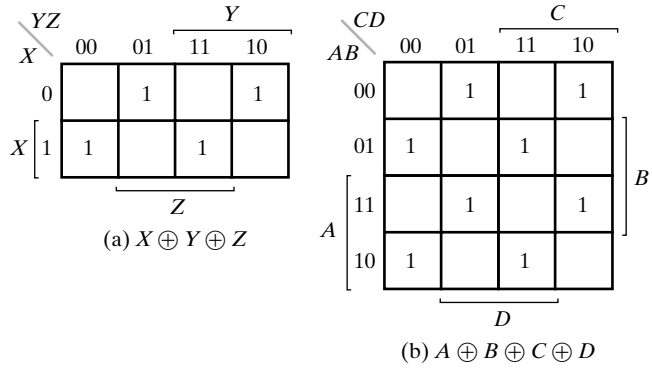
Función impar

La operación OR exclusiva con tres o más variables se puede convertir en una función booleana ordinaria reemplazando el símbolo $\oplus$ con su expresión booleana equivalente. En concreto, el caso de tres variables puede convertirse en una expresión booleana como la siguiente:

$$\begin{aligned} X \oplus Y \oplus Z &= (X\bar{Y} + \bar{X}Y)\bar{Z} + (XY + \bar{X}\bar{Y})Z \\ &= X\bar{Y}\bar{Z} + \bar{X}Y\bar{Z} + \bar{X}\bar{Y}Z + XYZ \end{aligned}$$

La expresión booleana indica claramente que la OR exclusiva de tres variables es igual a 1 si solamente una variable es igual a 1 o si las tres variables son iguales a 1. Por esto, mientras que en la función de dos variables sólo una variable tiene que ser igual a 1, con tres o más variables tiene que ser un número impar de variables iguales a 1. Como consecuencia, se define la operación OR exclusiva de múltiples variables como *función impar*. De hecho, estrictamente hablando, este es el nombre correcto para la operación $\oplus$ con tres o más variables; el nombre «OR exclusiva» es aplicable al caso con solamente dos variables.

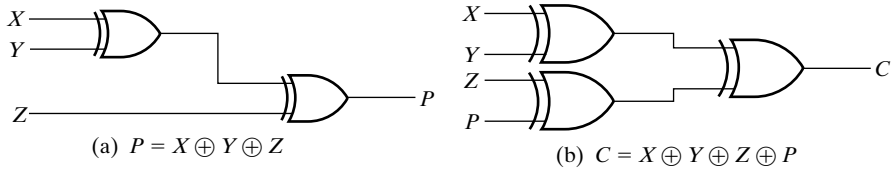
La definición de la función impar se puede clarificar dibujando la función en un mapa. La Figura 2-31(a) muestra el mapa para la función impar de tres variables. Los cuatro minitérminos de la función son diferentes entre si en al menos dos literales y por esto no pueden estar adyacentes en el mapa. Se dice que estos minitérminos tienen una *distancia* de dos uno al otro. La función impar se identifica por los cuatro minitérminos cuyas variables binarias tienen un número impar de 1. El caso de cuatro variables se muestra en la Figura 2-31(b). Los ocho minitérminos marcados con 1 en el mapa constituyen la función impar. Véase el patrón característico de la distancia entre los 1 en el mapa. Se debería mencionar que los minitérminos no marcados con 1 en el mapa tienen un número impar de 1 y constituyen el complemento de la función impar, llamada *función par*. La función impar se implementa mediante puertas OR exclusiva de dos entradas, como se muestra en la Figura 2-32. La función par se puede obtener reemplazando la puerta de salida con una puerta NOR exclusiva.



(a) $X \oplus Y \oplus Z$

(b) $A \oplus B \oplus C \oplus D$

□ **FIGURA 2-31**
Mapas para funciones impares de múltiples variables



(a) $P = X \oplus Y \oplus Z$

(b) $C = X \oplus Y \oplus Z \oplus P$

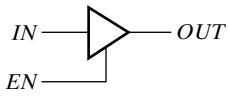
□ **FIGURA 2-32**
Funciones impares de múltiples entradas

2-9 SALIDAS EN ALTA IMPEDANCIA

Hasta ahora, hemos considerado solamente puertas que tienen los valores de salida 0 lógico y 1 lógico. En esta sección introducimos dos estructuras importantes, los *buffers* triestado y las puertas de transmisión, que proporcionan un tercer valor de salida a la que se llama como *estado de alta impedancia* y que se denota como Hi-*Z* o simplemente *Z* o *z*. El valor Hi-*Z* se comporta como un circuito abierto, que quiere decir que, mirando hacia atrás del circuito, encontramos que la salida aparece como desconectada. Las salidas de alta impedancia podrían aparecer en cualquier puerta, pero aquí nos restringimos a dos estructuras de puertas con entradas de datos simples. Las puertas con valores de salida Hi-*Z* pueden tener sus salidas conectadas entre sí, con tal de que no pueda haber dos puertas que conduzcan al mismo tiempo con valores opuestos 0 y 1. Por contra, las puertas con salidas lógicas de 0 y 1 no pueden tener sus salidas conectadas.

Buffers triestado El *buffer* triestado se ha presentado anteriormente como una de las puertas primitivas. Como indica su nombre, una salida lógica de tres estados muestra tres estados diferentes. Dos de los estados son el 1 y el 0 lógico de la lógica convencional. El tercer estado es el valor *Hi-Z*, al cual, para la lógica triestado, se le denomina como *estado Hi-Z* o estado de alta impedancia.

El símbolo gráfico y la tabla de verdad para un *buffer* triestado se presenta en la Figura 2-33. El símbolo de la Figura 2-33(a) se distingue del símbolo de un *buffer* normal por su entrada de habilitación, *EN*, que entra por debajo del símbolo. Según la tabla de verdad de la



(a) Símbolo lógico

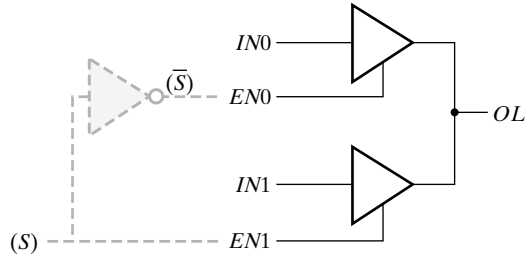
EN	*IN*	*OUT*
0	*X*	*Hi-Z*
1	0	0
1	1	1

(b) Tabla de verdad

□ **FIGURA 2-33**
Buffer de tres estados

Figura 2-33(b), si $EN = 1$, *OUT* es igual a *IN*, comportándose como un buffer normal. Pero para $EN = 0$, el valor de salida es de alta impedancia (*Hi-Z*), independiente del valor de *IN*.

Las salidas del buffer triestado se pueden conectar para formar una salida multiplexada. La Figura 2-34(a) muestra dos buffers triestados con sus salidas conectadas para formar la salida *OL*. Centramos el estudio en la salida de esta estructura en función de las cuatro entradas *EN*1, *EN*0, *IN*1, y *IN*0. El comportamiento de la salida se muestra en la tabla de verdad de la Figura 2-34(b). Para *EN*1 y *EN*0 igual a 0, ambas salidas del buffer son *Hi-Z*. Ya que ambas aparecen como circuitos abiertos, *OL* también es un circuito abierto, representado por un valor *Hi-Z*. Para $EN1 = 0$ y $EN0 = 1$, la salida del buffer superior es *IN*0 y la salida del buffer inferior es *Hi-Z*. Ya que el valor de *IN*0 combinado con un circuito abierto es justamente *IN*0, *OL* tiene el valor *IN*0, dando lugar a la segunda y tercera fila de la tabla de verdad. El caso contrario ocurre para $EN1 = 1$ y $EN0 = 0$, así *OL* tiene el valor *IN*1, dando lugar a la cuarta y quinta fila de la tabla



(a) Diagrama lógico

*EN*1	*EN*0	*IN*1	*IN*0	*OL*
0	0	*X*	*X*	*Hi-Z*
(*S*) 0	**($\overline{S}$)** 1	*X*	0	0
0	1	*X*	1	1
1	0	0	*X*	0
1	0	1	*X*	1
1	1	0	0	0
1	1	1	1	1
1	1	0	1	
1	1	1	0	

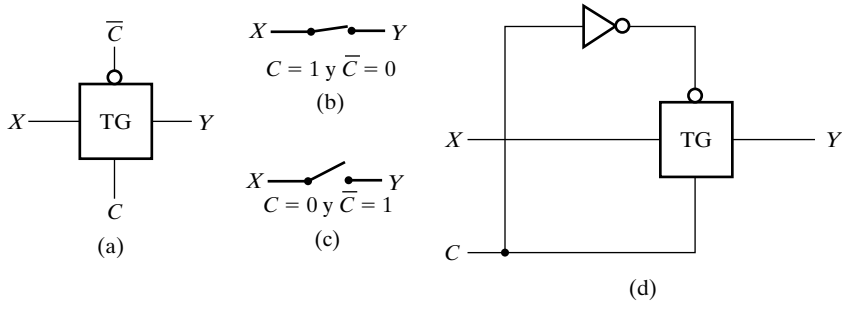
(b) Tabla de verdad

□ **FIGURA 2-34**
Buffers de tres estados formando una línea multiplexada OL

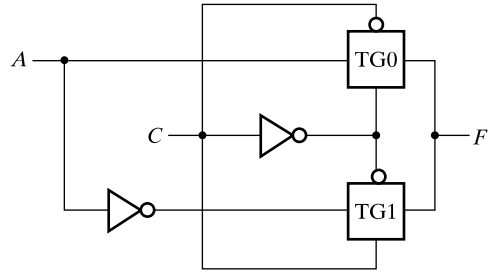
de verdad. Para *EN*1 y *EN*0, ambas 1, la situación es más complicada. Si *IN*1 = *IN*0, su valor aparece en *OL*. Pero si *IN* ≠ *IN*0, sus valores tienen un conflicto en la salida. El conflicto resulta en un flujo de corriente de la salida del buffer que está en 1 hacia la salida del buffer que está en 0. Esta corriente muchas veces es lo suficientemente alta para producir calentamiento y incluso podría destruir el circuito, como se simboliza por los iconos de «humo» en la tabla de verdad. Claramente hay que evitar semejante situación. El diseñador tiene que asegurar que *EN*0 y *EN*1 no son nunca iguales a 1 al mismo tiempo. En el caso general, para buffers triestados vinculados a una línea de bus, *EN* puede ser igual a 1 para solamente uno de los buffers y tiene que ser 0 para el resto. Una posibilidad para asegurar esto es usar un decodificador para generar las señales de *EN*. Para el caso de dos buffers, el decodificador es solamente un inversor con entrada seleccionable *S*, como se muestra en las líneas punteadas de la Figura 2-34(a). Es interesante examinar la tabla de verdad con el inversor puesto. Observe el área sombreada de la tabla de la Figura 2-34(b). Claramente, el valor en *S* selecciona entre las entradas *IN*0 y *IN*1. Además, la salida del circuito OL no está nunca en el estado Hi-*Z*.

Puertas de transmisión En los circuitos integrados lógicos, hay un circuito lógico con transistores CMOS que es suficientemente importante para ser presentado por separado, a nivel de puertas. Este circuito, llamado puerta de transmisión (*transmission gate* (TG)), es una especie de interruptor electrónico para conectar y desconectar dos puntos en un circuito. La Figura 2-35(a) muestra el símbolo IEEE para la puerta de transmisión. Tiene cuatro conexiones externas o puertos. *C* y $\overline{C}$ son las entradas de control y *X* e *Y* son las señales que van a ser conectadas o desconectadas por la TG. En la Figura 2-35(*b*) y (*c*), aparece el modelo con interruptores para la puerta de transmisión. Si $C = 1$ y $\overline{C} = 0$, *X* e *Y* se conectan según el modelo, por un interruptor «cerrado» y las señales pueden pasar de *X* a *Y* o de *Y* a *X*. Si $C = 0$ y $\overline{C} = 1$, *X* e *Y* están desconectados, como se representa en el modelo, por un interruptor «abierto» y las señales no pueden pasar entre *X* e *Y*. En uso normal, las entradas de control están conectadas por un inverso, como se muestra en la Figura 2-35(d), de manera que *C* y $\overline{C}$ son uno el complemento del otro.

Para ilustrar el uso de una puerta de transmisión, se muestra en la Figura 2-36(a) una puerta OR exclusiva construida con dos puertas de transmisión y dos inversores. La entrada *C* controla las rutas por las puertas de transmisión, y la entrada *A* proporciona la salida para *F*. Si la entrada *C* es igual a 1, existe un camino por la puerta de transmisión TG1 conectando *F* con $\overline{A}$, y no existe ninguna ruta por TG0. Si la entrada *C* es igual a 0, existe una ruta por TG0 conectando *F* con *A*, y no existe ninguna ruta por TG1. Así, la salida *F* está conectada con *A*. Esto da lugar a la tabla de verdad de la OR exclusiva, como se indica en la Figura 2-36(b).



□ **FIGURA 2-35**
Puerta de transmisión (TG)



(a)

A	*C*	TG1	TG0	*F*
0	0	Abierta	Cerrada	0
0	1	Cerrada	Abierta	1
1	0	Abierta	Cerrada	1
1	1	Cerrada	Abierta	0

(b)

□ **FIGURA 2-36**
Puerta de transmisión OR exclusiva

2-10 RESUMEN DEL CAPÍTULO

Las operaciones lógicas primitivas AND, OR, y NOT definen tres componentes lógicos llamados puertas, usadas para implementar sistemas digitales. El Álgebra de Boole, definido en términos de estas operaciones, proporciona una herramienta para manipular funciones booleanas en el diseño de circuitos lógicos. Las formas canónicas de minitérminos y maxitérminos se corresponden directamente con las tablas de verdad de las funciones. Estas formas canónicas se pueden manipular en forma de suma de productos o producto de sumas, que corresponden a circuitos con dos niveles puertas. Dos medidas de coste para minimizar, optimizando un circuito, son el número de literales de entrada al circuito y el número total de puertas del circuito. Los mapas-K, desde dos hasta cuatro variables, son una alternativa efectiva a la manipulación algebraica en la optimización de circuitos pequeños. Estos mapas se pueden usar para optimizar las formas de suma de productos, producto de sumas, y funciones especificadas incompletamente con condiciones de indiferencia. Se han ilustrado las transformaciones para optimizar circuitos de múltiples niveles con tres o más niveles de puertas.

Las operaciones primitivas AND y OR no están directamente implementadas por elementos lógicos primitivos en la familia lógica más popular. Así, se han presentado las primitivas NAND y NOR tanto así como las puertas complejas que implementan esas familias. Se ha presentado una primitiva más compleja, la OR exclusiva, tanto como su complemento, la NOR exclusiva, junto con sus propiedades matemáticas.

REFERENCIAS

1. BOOLE, G.: *An Investigation of the Laws of Thought*. New York: Dover, 1854.
2. KARNAUGH, M.: *«A Map Method for Synthesis of Combinational Logic Circuitos», Transactions of AIEE, Communication y Electronics*, 72, part I (Nov. 1953), 593-99.
3. DIETMEYER, D. L.: *Logic Design of Digital Systems*, 3rd ed. Boston: Allyn Bacon, 1988.
4. MANO, M. M.: *Digital Design*, 3rd ed. Upper Saddle River, NJ: Prentice Hall, 2002.
5. ROTH, C. H.: *Fundamentals of Logic Design*, 4th ed. St. Paul: West, 1992.
6. HAYES, J. P.: *Introduction to Digital Logic Design*. Reading, MA: Addison-Wesley, 1993.
7. WAKERLY, J. F.: *Digital Design*: *Principles y Practices*, 3rd ed. Upper Saddle River, NJ: Prentice Hall, 2000.
8. GAJSKI, D. D.: *Principles of Digital Design*. Upper Saddle River, NJ: Prentice Hall, 1997.

9. *IEEE Standard Graphic Symbols for Logic Funcións*. (Includes IEEE Std 91a-1991 Supplement y IEEE Std 91-1984.) New York: The Institute of Electrical y Electronics Engineers, 1991.

PROBLEMAS

El signo (+) indica problemas más avanzados y el asterisco (*) indica que hay una solución disponible en la dirección de Internet: http://www.librosite.net/Mano.

2-1. *Demuestre por medio de tablas de verdad la validez de las siguientes identidades:

(a) Teorema de DeMorgan para tres variables: $\overline{XYZ} = \bar{X} + \bar{Y} + \bar{Z}$
(b) La segunda ley distributiva: $X + YZ = (X + Y)(X + Z)$
(c) $\bar{X}Y + \bar{Y}Z + X\bar{Z} = X\bar{Y} + Y\bar{Z} + \bar{X}Z$

2-2. *Demuestre la identidad de cada una de las siguientes ecuaciones booleanas, usando la manipulación algebraica:

(a) $\bar{X}\bar{Y} + \bar{X}Y + XY = \bar{X} + Y$
(b) $\bar{A}B + \bar{B}\bar{C} + AB + \bar{B}C = 1$
(c) $Y + \bar{X}Z + X\bar{Y} = X + Y + Z$
(d) $\bar{X}\bar{Y} + \bar{Y}Z + XZ + XY + Y\bar{Z} = \bar{X}\bar{Y} + XZ + Y\bar{Z}$

2-3. +Demuestre la identidad de cada una de las siguientes ecuaciones booleanas, usando la manipulación algebraica:

(a) $AB + B\bar{C}\bar{D} + \bar{A}BC + \bar{C}D = B + \bar{C}D$
(b) $WY + \bar{W}YZ + WXZ + \bar{W}X\bar{Y} = WY + \bar{W}X\bar{Z} + \bar{X}Y\bar{Z} + X\bar{Y}Z$
(c) $A\bar{C} + \bar{A}B + \bar{B}C + \bar{D} = (\bar{A} + \bar{B} + \bar{C} + \bar{D})(A + B + C + \bar{D})$

2-4. +Dado que $A \cdot B = 0$ y $A + B = 1$, use la manipulación algebraica para demostrar que

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

2-5. +En este capítulo se ha usado un Álgebra específica de Boole con solamente dos elementos 0 y 1. Se pueden definir otras álgebras booleanas con más que dos elementos usando elementos que corresponden a cadenas binarias. Estas álgebras forman la base matemática para las operaciones lógicas de bit a bit que vamos a estudiar en el Capítulo 7. Suponga que las cadenas son cada una un nibble (medio byte) de cuatro bits. Entonces hay 2^4, o 16, elementos en el álgebra, donde un elemento *I* es el nibble de 4-bit en binario correspondiente a *I* en decimal. Basándose en la aplicación bit a bit del Álgebra de Boole de dos elementos, defina cada uno de los siguientes puntos para la nueva álgebra de manera que las identidades booleanas sean correctas:

(a) La operación OR $A + B$ para cada dos elementos *A y B*
(b) La operación AND $A \cdot B$ para cada dos elementos *A y B*
(c) El elemento que actúa como el 0 para el álgebra
(d) El elemento que actúa como el 1 para el álgebra
(e) Para cada elemento *A*, el elemento $\bar{A}$.

2-6. Simplifique las siguientes expresiones booleanas a las expresiones conteniendo un número mínimo de literales:

(a) $\bar{A}\bar{C} + \bar{A}BC + \bar{B}C$

(b) $\overline{(A + B)}(\bar{A} + \bar{B})$
(c) $ABC + \bar{A}C$
(d) $BC + B(AD + \bar{C}D)$
(e) $(B + \bar{C} + B\bar{C})(BC + A\bar{B} + AC)$

2-7. *Reduzca las siguientes expresiones booleanas al número de literales indicado:
(a) $\bar{X}\bar{Y} + XYZ + \bar{X}Y$ a tres literales
(b) $X + Y(Z + \overline{X + Z})$ a dos literales
(c) $\bar{W}X(\bar{Z} + \bar{Y}Z) + X(W + \bar{W}YZ)$ a un literal
(d) $(AB + \bar{A}\bar{B})(\bar{C}\bar{D} + CD) + \bar{A}\bar{C}$ a cuatro literales

2-8. Usando el Teorema de DeMorgan, exprese la función

$$F = \bar{A}BC + \bar{B}\bar{C} + A\bar{B}$$

(a) Solamente con operaciones de OR y de complemento.
(b) Solamente con operaciones AND y de complemento.

2-9. *Encuentre el complemento de las siguientes expresiones:
(a) $A\bar{B} + \bar{A}B$
(b) $(\bar{V}W + X)Y + \bar{Z}$
(c) $WX(\bar{Y}Z + Y\bar{Z}) + \bar{W}\bar{X}(\bar{Y} + Z)(Y + \bar{Z})$
(d) $(A + \bar{B} + C)(\bar{A}\bar{B} + C)(A + \bar{B}\bar{C})$

2-10. *Obtenga la tabla de verdad para las siguientes funciones, y exprese cada función en forma de suma de minitérminos y de producto de maxitérminos:
(a) $(XY + Z)(Y + XZ)$
(b) $(\bar{A} + B)(\bar{B} + C)$
(c) $WX\bar{Y} + WX\bar{Z} + WXZ + Y\bar{Z}$

2-11. Para las Funciones de Boole E y F, según la siguiente tabla de verdad:

X	*Y*	*Z*	*E*	*F*
0	0	0	1	0
0	0	1	0	0
0	1	0	1	1
0	1	1	0	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	1
1	1	1	0	1

(a) Enumere los minitérminos y maxitérminos de cada función.
(b) Enumere los minitérminos de $\bar{E}$ y $\bar{F}$.
(c) Enumere los minitérminos de $E + F$ y $E \cdot F$.
(d) Exprese E y F en forma algebraica de suma de minitérminos.
(e) Simplifique E y F a expresiones con un número mínimo de literales.

2-12. *Convierta las siguientes expresiones en formas de suma de productos y de producto de sumas:

(a) $(AB + C)(B + \bar{C}D)$
(b) $\bar{X} + X(X + \bar{Y})(Y + \bar{Z})$
(c) $(A + B\bar{C} + CD)(\bar{B} + EF)$

2-13. Dibuje el diagrama lógico para las siguientes expresiones booleanas. El diagrama debería corresponder exactamente a la ecuación. Suponga que los complementos de las entradas no están disponibles.

(a) $W\bar{X}\bar{Y} + \bar{W}Z + YZ$
(b) $A(B\bar{D} + \bar{B}D) + D(BC + \bar{B}\bar{C})$
(c) $W\bar{Y}(X + Z) + \bar{X}Z(W + Y) + W\bar{X}(Y + Z)$

2-14. Optimice las siguientes ecuaciones booleanas mediante un mapa de tres variables:

(a) $F(X, Y, Z) = \Sigma m(1, 3, 6, 7)$
(b) $F(X, Y, Z) = \Sigma m(3, 5, 6, 7)$
(c) $F(A, B, C) = \Sigma m(0, 1, 2, 4, 6)$
(d) $F(A, B, C) = \Sigma m(0, 3, 4, 5, 7)$

2-15. *Optimice la siguiente expresión booleana usando un mapa:

(a) $\bar{X}\bar{Z} + Y\bar{Z} + XYZ$
(b) $\bar{A}B + \bar{B}C + \bar{A}\bar{B}\bar{C}$
(c) $\bar{A}\bar{B} + A\bar{C} + \bar{B}C + \bar{A}B\bar{C}$

2-16. Optimice las siguientes funciones booleanas mediante un mapa de cuatro variables:

(a) $F(A, B, C, D) = \Sigma m(2, 3, 8, 9, 10, 12, 13, 14)$
(b) $F(W, X, Y, Z) = \Sigma m(0, 2, 5, 6, 8, 10, 13, 14, 15)$
(c) $F(A, B, C, D) = \Sigma m(0, 2, 3, 7, 8, 10, 12, 13)$

2-17. Optimice las siguientes funciones booleanas, usando un mapa:

(a) $F(W, X, Y, Z) = \Sigma m(0, 2, 5, 8, 9, 11, 12, 13)$
(b) $F(A, B, C, D) = \Sigma m(3, 4, 6, 7, 9, 12, 13, 14, 15)$

2-18. *Encuentre los minitérminos de las siguientes expresiones dibujando primero cada expresión en un mapa:

(a) $XY + XZ + \bar{X}YZ$
(b) $XZ + \bar{W}X\bar{Y} + WXY + \bar{W}YZ + W\bar{Y}Z$
(c) $\bar{B}\bar{D} + ABD + \bar{A}BC$

2-19. *Encuentre todos los implicantes primos para las siguientes funciones booleanas, y determine cuáles son esenciales:

(a) $F(W, X, Y, Z) = \Sigma m(0, 2, 5, 7, 8, 10, 12, 13, 14, 15)$
(b) $F(A, B, C, D) = \Sigma m(0, 2, 3, 5, 7, 8, 10, 11, 14, 15)$
(c) $F(A, B, C, D) = \Sigma m(1, 3, 4, 5, 9, 10, 11, 12, 13, 14, 15)$

2-20. Optimice las siguientes funciones booleanas encontrando todos los implicantes primos y los implicantes primos esenciales aplicando la regla de selección:

(a) $F(W, X, Y, Z) = \Sigma m(0, 1, 4, 5, 7, 8, 9, 12, 14, 15)$

(b) $F(A, B, C, D) = \Sigma m(1, 5, 6, 7, 11, 12, 13, 15)$
(c) $F(W, X, Y, Z) = \Sigma m(0, 2, 3, 4, 5, 7, 8, 10, 11, 12, 13, 15)$

2-21. Optimice las siguientes funciones booleanas en forma de producto de sumas:

(a) $F(W, X, Y, Z) = \Sigma m(0, 2, 3, 4, 8, 10, 11, 15)$
(b) $F(A, B, C, D) = \Pi M(0, 2, 4, 5, 8, 10, 11, 12, 13, 14)$

2-22. Optimice las siguientes expresiones en (1) forma de suma de productos y (2) forma de producto de sumas:

(a) $A\bar{C} + \bar{B}D + \bar{A}CD + ABCD$
(b) $(\bar{A} + \bar{B} + \bar{D})(A + \bar{B} + \bar{C})(\bar{A} + B + \bar{D})(B + \bar{C} + \bar{D})$
(c) $(\bar{A} + \bar{B} + D)(\bar{A} + \bar{D})(A + B + \bar{D})(A + \bar{B} + C + D)$

2-23. Optimice las siguientes funciones en forma de (1) suma de productos y (2) forma de producto de sumas:

(a) $F(A, B, C, D) = \Sigma m(2, 3, 5, 7, 8, 10, 12, 13)$
(b) $F(W, X, Y, Z) = \Pi M(2, 10, 13)$

2-24. Optimice las siguientes funciones booleanas F junto con las condiciones de indiferencia d:

(a) $F(A, B, C, D) = \Sigma m(0, 3, 5, 7, 11, 13)$, $d(A, B, C, D) = \Sigma m(4, 6, 14, 15)$
(b) $F(W, X, Y, Z) = \Sigma m(0, 6, 8, 13, 14)$, $d(W, X, Y, Z) = \Sigma m(2, 4, 7, 10, 12)$
(c) $F(A, B, C) = \Sigma m(0, 1, 2, 4, 5)$, $d(A, B, C) = \Sigma m(3, 6, 7)$

2-25. *Optimice las siguientes funciones booleanas F junto con las condiciones de indiferencia d. Encuentre todos los implicantes primos y los implicantes primos esenciales, y aplique la regla de selección.

(a) $F(A, B, C) = \Sigma m(3, 5, 6)$, $d(A, B, C) = \Sigma m(0, 7)$
(b) $F(W, X, Y, Z) = \Sigma m(0, 2, 4, 5, 8, 14, 15)$, $d(W, X, Y, Z) = \Sigma m(7, 10, 13)$
(c) $F(A, B, C, D) = \Sigma m(4, 6, 7, 8, 12, 15)$, $d(A, B, C, D) = \Sigma m(2, 3, 5, 10, 11, 14)$

2-26. Optimice las siguientes funciones booleanas F junto con las condiciones de indiferencia d en forma de (1) suma de productos y (2) producto de sumas:

(a) $F(A, B, C, D) = \Pi M(1, 3, 4, 6, 9, 11)$, $d(A, B, C, D) = \Sigma m(0, 2, 5, 10, 12, 14)$
(b) $F(W, X, Y, Z) = \Sigma m(3, 4, 9, 15)$, $d(W, X, Y, Z) = \Sigma m(0, 2, 5, 10, 12, 14)$

2-27. Use la descomposición para encontrar el menor número de entradas de puerta, implementaciones de múltiples niveles, para las funciones usando puertas AND y OR y inversores.

(a) $F(A, B, C, D) = A\bar{B}C + \bar{A}BC + A\bar{B}D + \bar{A}BD$
(b) $F(W, X, Y, Z) = WY + XY + \bar{W}XZ + W\bar{X}Z$

2-28. Use extracción para encontrar el menor número de entradas de puertas compartidas, implementación de múltiple nivel para el par de funciones dadas usando puertas AND y OR y inversores.

(a) $F(A, B, C, D) = \Sigma m(0, 5, 11, 14, 15)$, $d(A, B, C, D) = \Sigma m(10)$
(b) $G(A, B, C, D) = \Sigma m(2, 7, 10, 11, 14)$, $d(A, B, C, D) = \Sigma m(15)$

2-29. Use eliminación para aplanar (flatten) cada uno de los conjuntos de funciones dados en una forma de suma de productos de dos niveles.

(a) $F(A, B, G, H) = AB\bar{G} + \bar{B}G + \bar{A}\bar{H}$, $G(C, D) = C\bar{D} + \bar{C}D$, $H(B, C, D) = B + CD$

(b) $T(U, V, Y, Z) = YZU + \bar{Y}\bar{Z}V$, $U(W, X) = W + \bar{X}$, $V(W, X, Y) = \bar{W}Y + X$

2-30. *Demuestre que el dual del OR exclusiva es igual a su complemento.

2-31. Implemente la siguiente función booleana con puertas de OR exclusiva y AND, usando el menor número de entradas de puerta:

$$F(A, B, C, D) = AB\bar{C}D + A\bar{D} + \bar{A}D$$

2-32. **(a)** Implemente la función $H = \bar{X}Y + XZ$ usando dos *buffers* triestados y un inversor.

(b) Construya una puerta de OR exclusiva interconectando dos *buffers* triestados y dos inversores.

2-33. **(a)** Conecte las salidas de tres *buffers* triestados, y añada la lógica adicional para implementar la función

$$F = \bar{A}BC + ABD + A\bar{B}\bar{D}$$

Suponga que C, D y $\bar{D}$ son entradas de datos a los buffers y A y B pasan por una lógica que genera las entradas de habilitación.

(b) ¿Está su diseño de la parte (a) libre de conflictos en la salida triestado? Si no, cambie el diseño si es necesario para estar libre de estos conflictos.

2-34. Use solamente puertas de transmisión e inversores para implementar la función del Problema 2-32.

2-35. Dependiendo del diseño y de la familia lógica usada, en general no es una buena idea dejar la salida de un circuito triestado o de puertas de transmisión en un estado de alta impedancia (Hi-Z).

(a) Para el circuito de puertas de transmisión diseñado en el Problema 2-33, presente todas las combinaciones de entrada para las que la salida F está en un estado de alta impedancia.

(b) Modifique la lógica de habilitación cambiando las entradas de habilitación de manera que la salida sea 0 o 1 (en vez de Hi-Z).