

---

# Simplificación de las funciones booleanas

# 3

---

## 3-1 METODO DE MAPAS

La forma completa de las compuertas lógicas digitales que implementan una función booleana está relacionada en forma directa con la complejidad de las expresiones algebraicas de las cuales se implementa la función. Aunque la representación en tabla de verdad de una función es única, cuando se expresa en forma algebraica puede aparecer en muchas formas diferentes. Las funciones booleanas pueden simplificarse por medios algebraicos como se expuso en la Sección 2-4. Sin embargo, el procedimiento de minimización es difícil debido a que carece de reglas específicas para predecir cada paso sucesivo en el proceso de manipulación. El método de mapas proporciona un procedimiento simple y directo para minimizar las funciones booleanas. Este método puede considerarse ya sea como una forma gráfica de una tabla de verdad o como una extensión del diagrama de Venn. El método de mapas, que Veitch (1) fue el primero en proponer y que modificó ligeramente Karnaugh (2), también se conoce como el “diagrama de Veitch” o “mapa de Karnaugh”.

El mapa es un diagrama compuesto por cuadros. Cada cuadro representa un mintérmino. Ya que cualquier función booleana puede expresarse como una suma de mintérminos, se concluye que una función booleana se reconoce en forma gráfica en el mapa por el área encerrada en los cuadros cuyos mintérminos se incluyen en la función. De hecho, el mapa presenta un diagrama visual de todas las formas posibles en que puede expresarse una función en una manera estándar. Mediante el reconocimiento de diversos patrones, el usuario puede derivar expresiones algebraicas alternas para la misma función, de las cuales él puede seleccionar la más simple. Se supondrá que la expresión algebraica más simple es cualquiera en una suma de productos o producto de sumas que tiene un número mínimo de literales. (Esta expresión no es única necesariamente.)

## 3-2 MAPAS DE DOS Y TRES VARIABLES

En la Fig. 3-1 se muestra un mapa de dos variables. Hay cuatro mintérminos para dos variables; por tanto, el mapa consta de cuatro cuadros, uno para cada mintérmino. El

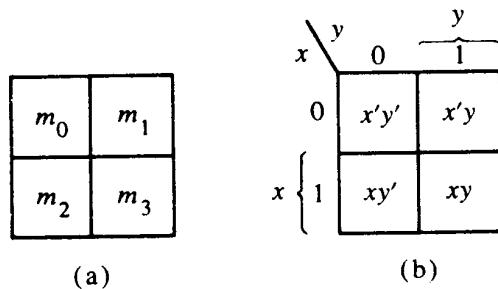


Figura 3-1 Mapa de dos variables.

mapa vuelve a dibujarse en (b) para mostrar las relaciones entre los cuadros y las dos variables. Los números 0 y 1 que se marcan para cada renglón y cada columna designan los valores de las variables  $x$  y  $y$ , respectivamente. Obsérvese que  $x$  aparece como prima en el renglón 0 y sin prima en el renglón 1. En forma similar,  $y$  aparece como prima en la columna 0 y sin prima en la columna 1.

Si se marcan los cuadros cuyos mintérminos pertenecen a una función dada, el mapa de dos variables se convierte en otra forma útil para representar cualquiera de las 16 funciones booleanas de dos variables. Como ejemplo, la función  $xy$  se muestra en la Fig. 3-2(a). Ya que  $xy$  es igual a  $m_3$ , se coloca un 1 en el interior del cuadro que pertenece a  $m_3$ . En forma semejante, la función  $x + y$  se representa en el mapa de la Fig. 3-2(b) por tres cuadros marcados por 1. Estos cuadros se encuentran mediante los mintérminos de la función:

$$x + y = x'y' + xy' + xy = m_1 + m_2 + m_3$$

Los tres cuadros pudieron haberse determinado mediante la intersección de la variable  $x$  en el segundo renglón y la variable  $y$  en la segunda columna, la cual encierra el área que pertenece a  $x$  o  $y$ .

En la Fig. 3-3 se muestra un mapa de tres variables. Hay ocho mintérminos para tres variables binarias. Por lo tanto, un mapa consta de ocho cuadros. Obsérvese que los mintérminos no están arreglados en una secuencia binaria, sino en una secuencia similar al código reflejado que se lista en la Tabla 1-4. Las características de esta secuencia es que sólo un bit cambia de 1 a 0 o de 0 a 1 en la secuencia listada. El mapa que se dibuja en la parte (b) se marca con números en cada renglón y cada columna para mostrar las relaciones entre los cuadros y las tres variables. Por ejemplo, el cuadro asignado a  $m_5$  corresponde al renglón 1 y la columna 01. Cuando estos dos

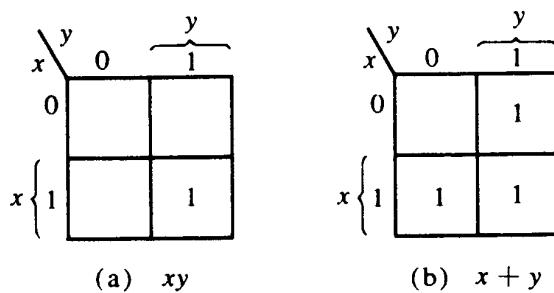
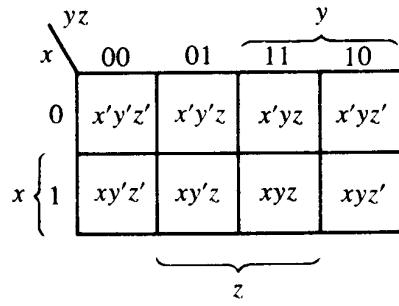


Figura 3-2 Representación de funciones en el mapa.

|       |       |       |       |
|-------|-------|-------|-------|
| $m_0$ | $m_1$ | $m_3$ | $m_2$ |
| $m_4$ | $m_5$ | $m_7$ | $m_6$ |

(a)



(b)

Figura 3-3 Mapa de tres variables.

números se concatenan, dan el número binario 101, cuyo equivalente decimal es 5. Otra forma de ver el cuadro  $m_5 = xy'z$  es considerar que está en el renglón marcado  $x$  y la columna que pertenece a  $y'z$  (columna 01). Obsérvese que hay cuatro cuadros donde cada variable es igual a 1 y cuatro donde cada 1 es igual a 0. La variable aparece sin prima en los cuatro cuadros donde es igual a 1 y con prima en los cuadros donde es igual a 0. Por motivos de comodidad, se escribe la variable con su símbolo de letra bajo los cuatro cuadros donde está sin prima.

Para entender la utilidad del mapa y simplificar funciones booleanas, debe reconocerse la propiedad básica que poseen los cuadros adyacentes. Cualesquiera dos cuadros adyacentes en el mapa difieren sólo en una variable que está con prima en un cuadro y sin prima en el otro. Por ejemplo,  $m_5$  y  $m_7$ , caen en dos cuadros adyacentes. La variable  $y$  tiene prima en  $m_5$  y no tiene prima en  $m_7$ , en tanto que las otras dos variables son las mismas en ambos cuadros. Mediante los postulados del álgebra booleana, se concluye que la suma de dos mintérminos en cuadros adyacentes puede simplificarse a un solo término AND que consta de sólo dos literales. Para aclarar esto, considérese la suma de dos cuadros adyacentes como  $m_5$  y  $m_7$ :

$$m_5 + m_7 = xy'z + xyz = xz(y' + y) = xz$$

Aquí los dos cuadros difieren por la variable  $y$ , la cual puede eliminarse cuando se forma la suma de los dos mintérminos. Por eso, cualesquiera dos mintérminos en cuadros adyacentes que se unen por el operador OR causarán una eliminación de la variable diferente. El siguiente ejemplo explica el procedimiento para minimizar una función booleana con un mapa.

**EJEMPLO 3-1:** Simplifique la función booleana:

$$F = x'yz + x'yz' + xy'z' + xy'z$$

Primero, se marca un 1 en cada cuadro como se necesite para representar la función como se muestra en la Fig. 3-4. Esto puede llevarse a cabo en dos formas: ya sea por la conversión de cada mintérmino en un número binario y marcando entonces un 1 en el cuadro correspondiente o por la obtención de la coincidencia de las variables en cada término.

Por ejemplo, el término  $x'yz$  tiene el número binario correspondiente 011 y representa el mintérmino  $m_3$  en el cuadro 011. La segunda forma de reconocer el cuadro es por la coincidencia de las variables  $x'$ ,  $y$  y  $z$ , la cual se encuentra en el mapa al observar que  $x'$  pertenece a los cuatro cuadros en el primer renglón,  $y$  pertenece a los cuatro cuadros en las dos columnas de la derecha y  $z$  pertenece a los cuatro cuadros en las dos columnas centrales. El área que pertenece a todas las tres literales es el único cuadro en el primer renglón y tercera columna. En forma similar, los otros tres cuadros pertenecientes a la función  $F$  están marcados con 1 en el mapa. Por consiguiente, la función está representada por un área que contiene cuatro cuadros, cada uno marcado con un 1, como se muestra en la Fig. 3-4. El paso siguiente es subdividir el área dada en cuadros adyacentes. Esto se indica en el mapa por dos rectángulos, cada uno encierra dos 1. El rectángulo superior de la derecha representa el área encerrada por  $x'y$ ; el inferior de la izquierda, el área encerrada por  $xy'$ . La suma de estos dos términos da la respuesta:

$$F = x'y + xy'$$

A continuación considérense los dos cuadros etiquetados  $m_0$  y  $m_2$  en la Fig. 3-3(a) o  $x'y'z'$  y  $x'yz'$  en la Fig. 3-3(b). Estos dos mintérminos también difieren por una variable  $y$ , y su suma puede simplificarse a una expresión de dos literales:

$$x'y'z' + x'yz' = x'z'$$

En consecuencia, debe modificarse la definición de cuadros adyacentes para incluir éste y otros casos similares. Esto se hace considerando el mapa como si estuviera dibujado en una superficie donde las orillas derecha e izquierda se tocan una con otra para formar cuadros adyacentes.

**EJEMPLO 3-2:** Simplifique la función booleana:

$$F = x'yz + xy'z' + xyz + xyz'$$

El mapa para esta función se muestra en la Fig. 3-5. Hay cuatro cuadros marcados con 1, uno para cada mintérmino de la función. Se combinan

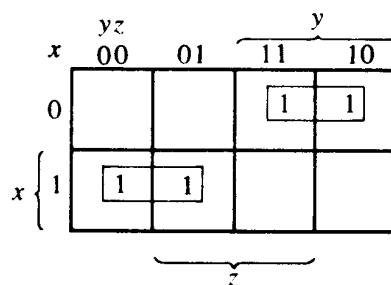


Figura 3-4 Mapa para el ejemplo 3-1;  $x'yz + xy'z' + xyz + xyz' = x'y + xy'$ .

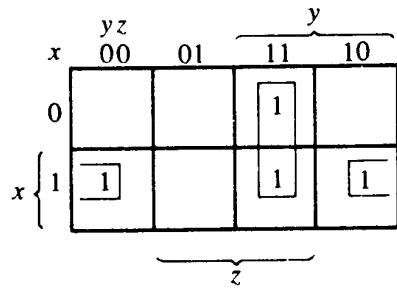


Figura 3-5 Mapa para el ejemplo 3-2;  $x'yz + xy'z' + xyz + xyz' = yz + xz'$ .

dos cuadros adyacentes en la tercera columna para dar un término de dos literales  $yz$ . Los dos cuadros restantes con 1 también son adyacentes por la nueva definición y se muestran en el diagrama dentro de medios rectángulos. Estos dos cuadros, cuando se combinan dan el término de dos literales  $xz'$ . La función simplificada llega a ser:

$$F = yz + xz'$$

Considérese ahora cualquier combinación de cuatro cuadros adyacentes en el mapa de tres variables. Cualquiera de dichas combinaciones representa la aplicación del operador OR a cuatro mintérminos adyacentes y resulta en una expresión de sólo una literal. Como un ejemplo, la suma de los cuatro mintérminos adyacentes  $m_0, m_2, m_4$  y  $m_6$  se reduce a la única literal  $z'$  como se muestra:

$$\begin{aligned} x'y'z' + x'yz' + xy'z' + xyz' &= x'z'(y' + y) + xz'(y' + y) \\ &= x'z' + xz' = z'(x' + x) = z' \end{aligned}$$

**EJEMPLO 3-3:** Simplifique la función booleana:

$$F = A'C + A'B + AB'C + BC$$

El mapa que simplifica esta función se muestra en la Fig. 3-6. Algunos de los términos de la función tienen menos de tres literales y se representan en el mapa por más de un cuadro. Por ejemplo, para encontrar los cuadros correspondientes a  $A'C$ , se forma la coincidencia de  $A'$  (primer renglón) y  $C$  (dos columnas centrales) y se obtienen los cuadros 001 y 011. Obsérvese que cuando se marcan números 1 en los cuadros, es

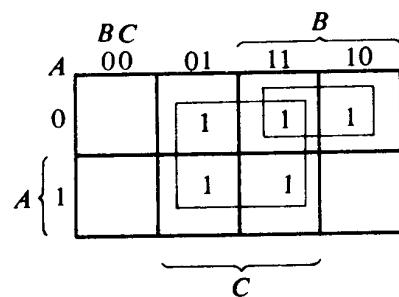


Figura 3-6 Mapa para el ejemplo 3-3;  $A'C + A'B + AB'C + BC = C + A'B$ .

possible encontrar un 1 ya colocado ahí por un término precedente. En este ejemplo, el segundo término  $A'B$  tiene números 1 en los cuadros 011 y 010, pero el cuadro 011 es común al primer término  $A'C$  y sólo un 1 está marcado en él. En este ejemplo, la función tiene cinco mintérminos, como se indica por los cinco cuadros marcados con números 1. Se simplifica por la combinación de cuatro cuadros en el centro para dar la literal  $C$ . El único cuadro restante marcado con un 1 en 010 se combina con un cuadro adyacente que ya se ha usado una vez. Esto se permite y es inclusive deseable ya que la combinación de los dos cuadros da el término  $A'B$ , en tanto que el único mintérmino representado por el cuadro da el término de tres variables  $A'BC'$ . La función simplificada es:

$$F = C + A'B$$

**EJEMPLO 3-4:** Simplifique la función booleana.

$$F(x, y, z) = \Sigma(0, 2, 4, 5, 6)$$

Aquí se dan los mintérminos por sus números decimales. Los cuadros correspondientes están marcados con números 1 como se muestra en la Fig. 3-7. Mediante el mapa se obtiene la función simplificada:

$$F = z' + xy'$$

### 3-3 MAPA DE CUATRO VARIABLES

El mapa para las funciones booleanas de cuatro variables binarias se muestra en la Fig. 3-8. En (a) se listan los 16 mintérminos y los cuadros asignados a cada uno. En (b) el mapa vuelve a dibujarse para mostrar las relaciones con las cuatro variables. Los renglones y columnas se numeran en una secuencia de código reflejado, con sólo un dígito cambiando de valor entre dos renglones adyacentes o columnas. El mintérmino que corresponde a cada cuadro puede obtenerse por la concatenación del número de renglón con el número de columna. Por ejemplo, los números del tercer renglón (11) y

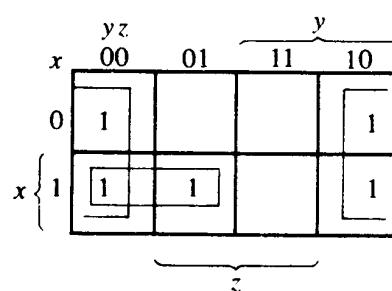
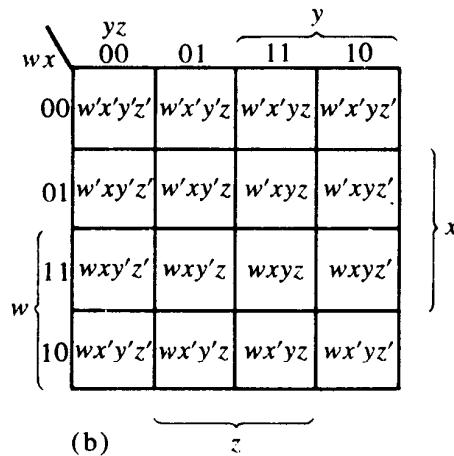


Figura 3-7  $f(x, y, z) = \Sigma(0, 2, 4, 5, 6) = z' + xy'$ .

|          |          |          |          |
|----------|----------|----------|----------|
| $m_0$    | $m_1$    | $m_3$    | $m_2$    |
| $m_4$    | $m_5$    | $m_7$    | $m_6$    |
| $m_{12}$ | $m_{13}$ | $m_{15}$ | $m_{14}$ |
| $m_8$    | $m_9$    | $m_{11}$ | $m_{10}$ |

(a)



(b)

Figura 3-8 Mapa de cuatro variables.

de la segunda columna (01), cuando se concatenan, dan el número binario 1101, el equivalente binario del decimal 13. Por tanto, el cuadro en el tercer renglón y la segunda columna representa el mintérmino  $m_{13}$ .

La minimización por mapa de las funciones booleanas de cuatro variables es similar al método que se utiliza para minimizar las funciones de tres variables. Se definen cuadros adyacentes para que sean cuadros juntos entre sí. Además, se considera que el mapa cae en una superficie en las orillas superior e inferior, al igual que en las orillas derecha e izquierda, tocándose uno a otro para formar cuadros adyacentes. Por ejemplo,  $m_0$  y  $m_2$  forman cuadros adyacentes, como sucede con  $m_3$  y  $m_{11}$ . La combinación de cuadros adyacentes que es útil durante el proceso de simplificación se determina con facilidad por la inspección del mapa de cuatro variables:

Un cuadro representa un mintérmino, dando un término de cuatro literales.

Dos cuadros adyacentes representan un término de tres literales.

Cuatro cuadros adyacentes representan un término de dos literales.

Ocho cuadros adyacentes representan un término de una literal.

Dieciséis cuadros adyacentes representan la función igual a 1.

Ninguna otra combinación de cuadros puede simplificar la función. Los dos ejemplos siguientes muestran el procedimiento que se usa para simplificar funciones booleanas de cuatro variables.

**EJEMPLO 3-5:** Simplifique la función booleana:

$$F(w, x, y, z) = \Sigma(0, 1, 2, 4, 5, 6, 8, 9, 12, 13, 14)$$

Ya que la función tiene cuatro variables, debe usarse un mapa de cuatro variables. Los mintérminos que se listan en la suma se marcan con números 1 en el mapa de la Fig. 3-9. Ocho cuadros adyacentes marcados con números 1 pueden combinarse para formar un término de una

literal  $y'$ . Los tres 1 restantes a la derecha no pueden combinarse juntos para dar un término simplificado. Deben combinarse como dos o cuatro cuadros adyacentes. Mientras mayor sea el número de cuadros combinados, menor será el número de literales en el término. En este ejemplo, los dos 1 de la parte superior a la derecha se combinan con los dos 1 de la parte superior a la izquierda para dar el término  $w'z'$ . Obsérvese que se permite usar el mismo cuadro más de una vez. Ahora queda un cuadro marcado con 1 en el tercer renglón y cuarta columna (cuadro 1110). En lugar de tomar este cuadro sólo (lo cual daría un término de cuatro literales), se combina con cuadros que ya se han empleado para formar una área de cuatro cuadros adyacentes. Estos cuadros comprenden los dos renglones centrales y dos columnas en los extremos, dando el término  $xz'$ . La función simplificada es:

$$F = y' + w'z' + xz'$$

**EJEMPLO 3-6:** Simplifique la función booleana:

$$F = A'B'C' + B'CD' + A'BCD' + AB'C'$$

El área en el mapa cubierta por esta función consta de cuadros marcados con 1 en la Fig. 3-10. Esta función tiene cuatro variables y, como se expresa, consta de tres términos, cada uno con tres literales, y un término de cuatro literales. Cada término de tres literales se representa en el mapa por dos cuadros. Por ejemplo,  $A'B'C'$  se representa en los cuadros 0000 y 0001. La función puede simplificarse en el mapa tomando los 1 en las cuatro esquinas para dar el término  $B'D'$ . Esto es posible debido a que estos cuatro cuadros son adyacentes cuando el mapa se dibuja en una superficie, con las orillas superior e inferior o de izquierda a derecha tocándose una a otra. Los dos 1 del lado izquierdo del renglón

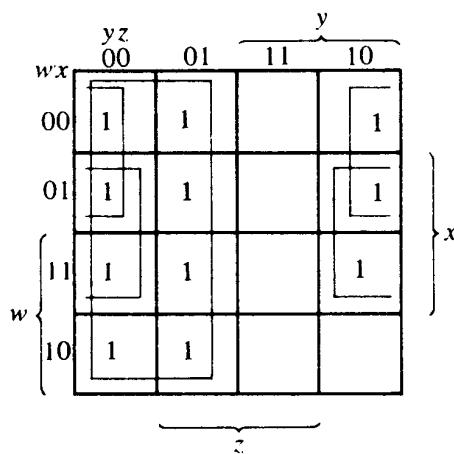
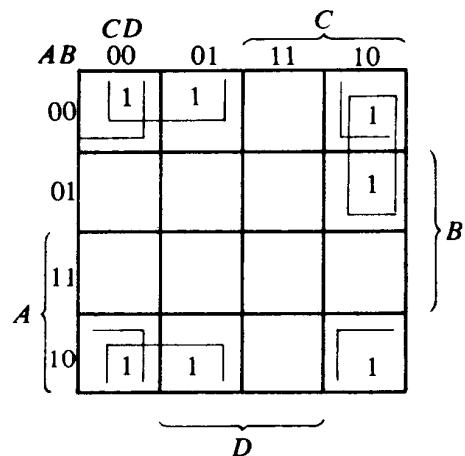


Figura 3-9 Mapa para el ejemplo 3-5;  $F(w, x, y, z) = \Sigma(0, 1, 2, 4, 5, 6, 8, 9, 12, 13, 14) = y' + w'z' + xz'$ .



**Figura 3-10** Mapa para el ejemplo 3-6;  $A'B'C' + B'CD' + A'BCD' + AB'C = B'D' + B'C' + A'CD'$ .

superior se combinan con los dos 1 en el renglón inferior para dar el término  $B'C'$ . Los restantes 1 pueden combinarse en un área de dos cuadros para dar el término  $A'CD'$ . La función simplificada es:

$$F = B'D' + B'C' + A'CD'$$

### 3-4 MAPAS DE CINCO Y SEIS VARIABLES

Los mapas de más de cuatro variables no son de uso tan simple. El número de cuadros se vuelve en exceso grande y la geometría para combinar cuadros adyacentes se vuelve más complicada. El número de cuadros siempre es igual al número de mintérminos. Para mapas de cinco variables, se necesitan 32 cuadros; para mapas de seis variables se requieren 64 cuadros. Los mapas con siete o más variables necesitan tantos cuadros que no es práctico usarlos. Los mapas de cinco o seis variables se muestran en las Figs. 3-11 y 3-12, respectivamente. Los renglones y columnas se numeran en una secuencia de código reflejado; el mintérmino asignado a cada cuadro se lee mediante esos números. En esta forma, el cuadro en el tercer renglón (11) y la segunda columna (001), en el mapa de cinco variables, es el número 11001, el equivalente del decimal 25. Por tanto, este cuadro representa el mintérmino  $m_{25}$ . El símbolo de letra de cada variable se marca junto a los cuadros donde el valor correspondiente de bit del número de código reflejado es un 1. Por ejemplo, en el mapa de cinco variables, la variable  $A$  es un 1 en los últimos dos renglones;  $B$  es un 1 en los dos renglones centrales. Los números reflejados en las columnas muestran las variables  $C$  con un 1 en las cuatro columnas más a la derecha, la variable  $D$  con un 1 en las cuatro columnas centrales y los 1 para la variable  $E$  no son físicamente adyacentes pero se dividen en dos partes. La asignación de variables en el mapa de seis variables se determina de manera semejante.

La definición de cuadros adyacentes para los mapas en las Figs. 3-11 y 3-12 debe modificarse de nuevo para tomar en cuenta el hecho de que algunas variables están divididas en dos partes. Debe considerarse que el mapa de cinco variables consta de

| AB                          |    | CDE |     |     |     |                             | C  |     |     |
|-----------------------------|----|-----|-----|-----|-----|-----------------------------|----|-----|-----|
|                             |    | 000 | 001 | 011 | 010 | 110                         |    | 101 | 100 |
| A                           | 00 | 0   | 1   | 3   | 2   | 6                           | 7  | 5   | 4   |
|                             | 01 | 8   | 9   | 11  | 10  | 14                          | 15 | 13  | 12  |
|                             | 11 | 24  | 25  | 27  | 26  | 30                          | 31 | 29  | 28  |
|                             | 10 | 16  | 17  | 19  | 18  | 22                          | 23 | 21  | 20  |
| $\overbrace{\hspace{10em}}$ |    |     |     |     | D   | $\overbrace{\hspace{10em}}$ |    |     |     |
| $\overbrace{\hspace{10em}}$ |    |     |     |     | E   | $\overbrace{\hspace{10em}}$ |    |     |     |
| B                           |    |     |     |     |     |                             |    |     |     |

Figura 3-11 Mapa de cinco variables.

dos mapas de cuatro variables y, que el mapa de seis variables consta de cuatro mapas de cuatro variables. Cada uno de estos mapas de cuatro variables se reconoce por las líneas dobles en el centro del mapa; cada uno tiene la adyacencia previamente definida cuando se toma de manera individual. Además, la doble línea del centro debe considerarse como el centro de un libro, como si cada mitad del mapa fuera una

| ABC                         |     | DEF |     |     |     |                             | D  |     |     |  |  |  |  |
|-----------------------------|-----|-----|-----|-----|-----|-----------------------------|----|-----|-----|--|--|--|--|
|                             |     | 000 | 001 | 011 | 010 | 110                         |    | 101 | 100 |  |  |  |  |
| A                           | 000 | 0   | 1   | 3   | 2   | 6                           | 7  | 5   | 4   |  |  |  |  |
|                             | 001 | 8   | 9   | 11  | 10  | 14                          | 15 | 13  | 12  |  |  |  |  |
|                             | 011 | 24  | 25  | 27  | 26  | 30                          | 31 | 29  | 28  |  |  |  |  |
|                             | 010 | 16  | 17  | 19  | 18  | 22                          | 23 | 21  | 20  |  |  |  |  |
|                             | 110 | 48  | 49  | 51  | 50  | 54                          | 55 | 53  | 52  |  |  |  |  |
|                             | 111 | 56  | 57  | 59  | 58  | 62                          | 63 | 61  | 60  |  |  |  |  |
|                             | 101 | 40  | 41  | 43  | 42  | 46                          | 47 | 45  | 44  |  |  |  |  |
|                             | 100 | 32  | 33  | 35  | 34  | 38                          | 39 | 37  | 36  |  |  |  |  |
| $\overbrace{\hspace{10em}}$ |     |     |     |     | E   | $\overbrace{\hspace{10em}}$ |    |     |     |  |  |  |  |
| $\overbrace{\hspace{10em}}$ |     |     |     |     | F   | $\overbrace{\hspace{10em}}$ |    |     |     |  |  |  |  |
| C                           |     |     |     |     |     |                             |    |     |     |  |  |  |  |
| B                           |     |     |     |     |     |                             |    |     |     |  |  |  |  |
| C                           |     |     |     |     |     |                             |    |     |     |  |  |  |  |

Figura 3-12 Mapa de seis variables.

página. Cuando se cierra el libro, dos cuadros adyacentes caen uno sobre otro. En otras palabras, la línea doble del centro es como un espejo con cada cuadro que es adyacente, no sólo a sus cuatro cuadros vecinos, sino también a su imagen de espejo. Por ejemplo, el mintérmino 31 en el mapa de cinco variables es adyacente a los mintérminos 30, 15, 29, 23 y 27. El mismo mintérmino en el mapa de seis variables es adyacente a todos esos mintérminos más el mintérmino 63.

Mediante inspección, y tomando en cuenta la nueva definición de cuadros adyacentes, es posible mostrar que cualesquiera cuadros adyacentes  $2^k$  para  $k = 0, 1, 2, \dots, n$ , en un mapa de  $n$  variables, representarán un área que da un término de  $n - k$  literales. Para que la enunciación anterior tenga algún significado,  $n$  debe ser mayor que  $k$ . Cuando  $n = k$ , el área entera del mapa está combinada para dar la función de identidad. La Tabla 3-1 muestra la relación entre el número de cuadros adyacentes con el número de literales en el término. Por ejemplo, ocho cuadros adyacentes se combinan en un área en el mapa de cinco variables para dar un término de dos literales.

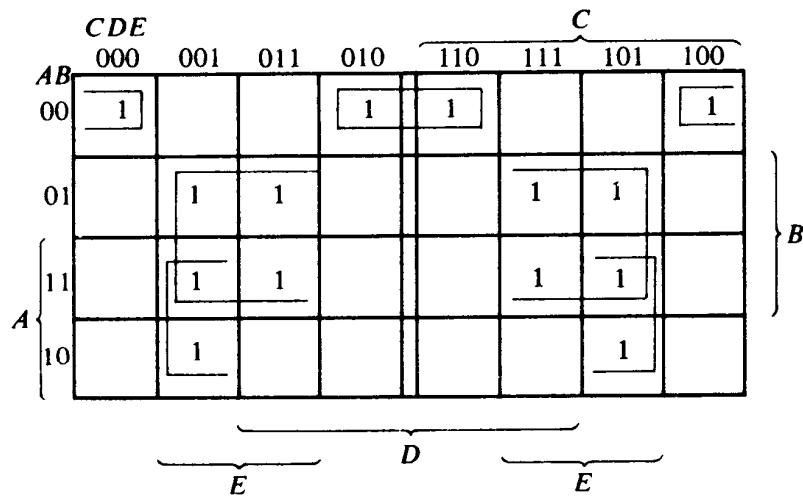
**EJEMPLO 3-7:** Simplifique la función booleana:

$$F(A, B, C, D, E) = \Sigma(0, 2, 4, 6, 9, 11, 13, 15, 17, 21, 25, 27, 29, 31)$$

El mapa de cinco variables de esta función se muestra en la Fig. 3-13. Cada mintérmino se convierte en su número binario equivalente y los 1 se marcan en sus cuadros correspondientes. Ahora es necesario encontrar combinaciones de cuadros adyacentes que resulten en el área más grande posible. Los cuatro cuadros en el centro de la mitad del mapa a la derecha se reflejan a través de la línea doble y se combinan con los cuatro cuadros en el centro del mapa de la mitad izquierda, para dar ocho cuadros adyacentes disponibles equivalentes al término  $BE$ . Los

**TABLA 3-1** La relación entre el número de cuadros adyacentes y el número de literales en el término

| Número<br>de<br>cuadros<br>adyacentes | Número de literales de un término en un mapa de $n$ variables |       |         |         |         |         |         |
|---------------------------------------|---------------------------------------------------------------|-------|---------|---------|---------|---------|---------|
|                                       | $k$                                                           | $2^k$ | $n = 2$ | $n = 3$ | $n = 4$ | $n = 5$ | $n = 6$ |
| 0                                     | 1                                                             | 2     | 3       | 4       | 5       | 6       | 7       |
| 1                                     | 2                                                             | 1     | 2       | 3       | 4       | 5       | 6       |
| 2                                     | 4                                                             | 0     | 1       | 2       | 3       | 4       | 5       |
| 3                                     | 8                                                             |       | 0       | 1       | 2       | 3       | 4       |
| 4                                     | 16                                                            |       |         | 0       | 1       | 2       | 3       |
| 5                                     | 32                                                            |       |         |         | 0       | 1       | 2       |
| 6                                     | 64                                                            |       |         |         |         | 0       | 1       |



**Figura 3-13** Mapa para el ejemplo 3-7;  $F(A, B, C, D, E) = \Sigma(0, 2, 4, 6, 9, 11, 13, 15, 17, 21, 25, 27, 29, 31) = BE + AD'E + A'B'E'$ .

dos 1 en el renglón inferior son reflejo uno de otro sobre la doble línea del centro. Por la combinación de ellos con los otros dos cuadros adyacentes, se obtiene el término  $AD'E$ . Los cuatro 1 en el renglón superior son todos adyacentes y pueden combinarse para dar el término  $A'B'E'$ . Todos los 1 están incluidos ahora. La función simplificada es:

$$F = BE + AD'E + A'B'E'$$

### 3-5 SIMPLIFICACION DE PRODUCTOS DE SUMA

Las funciones booleanas minimizadas derivadas del mapa en todos los ejemplos previos se expresaron en la forma de suma de productos. Con una pequeña modificación, puede obtenerse la forma de producto de sumas.

El procedimiento para obtener una función minimizada en producto de sumas es una consecuencia de las propiedades básicas de las funciones booleanas. Los 1 ubicados en los cuadros del mapa representan los mintérminos de la función. Los mintérminos que no se incluyen en la función denotan el complemento de la función. De esto puede verse que el complemento de una función está representado en el mapa por los cuadros no marcados por 1. Si se marcan los cuadros vacíos con 0 y se combinan en cuadros válidos adyacentes, se obtiene una expresión simplificada del complemento de la función, esto es, de  $F'$ . El complemento de  $F'$  resulta de nuevo en la función  $F$ . Debido al teorema generalizado de De Morgan, la función así obtenida automáticamente está en la forma de producto de sumas. La mejor manera para mostrar esto es con un ejemplo.

**EJEMPLO 3-8:** Simplifique la siguiente función booleana en (a) suma de productos y (b) producto de sumas.

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

Los 1 marcados en el mapa de la Fig. 3-14 representan todos los mintérminos de la función. Los cuadros marcados con 0 representan los mintérminos no incluidos en  $F$  y, en consecuencia, denotan el complemento de  $F$ . La combinación de los cuadros con 1 da la función simplificada en suma de productos:

$$(a) \quad F = B'D' + B'C' + A'C'D$$

Si los cuadros marcados con 0 se combinan, como se muestra en el diagrama, se obtiene la función complementada en forma simple:

$$F' = AB + CD + BD'$$

Mediante la aplicación del teorema de De Morgan (se toma la dual y se complementa cada literal como se describió en la Sección 2-4), se obtiene la función simplificada en producto de sumas:

$$(b) \quad F = (A' + B')(C' + D')(B' + D)$$

La implementación de las expresiones simplificadas obtenidas en el Ejemplo 3-8 se muestra en la Fig. 3-15. La expresión de la suma de productos se implementa en (a) con un grupo de compuertas AND, una para cada término AND. Las salidas de las compuertas AND se conectan a la entrada de una sola compuerta OR. La misma función se implementa en (b) en la forma de su producto de sumas con un grupo de compuertas OR, una para cada término OR. Las salidas de las compuertas OR se conectan con las entradas de una sola compuerta AND. En cada caso, se supone que las variables de entrada están disponibles en forma directa en su complemento, de

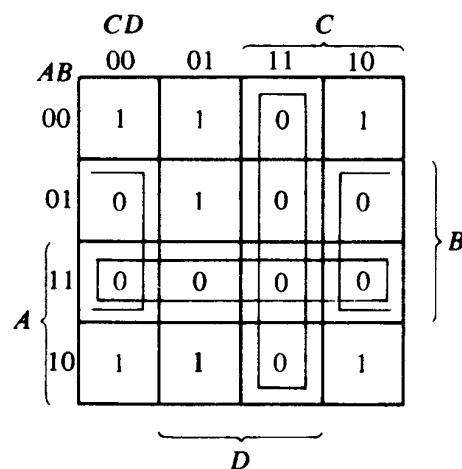
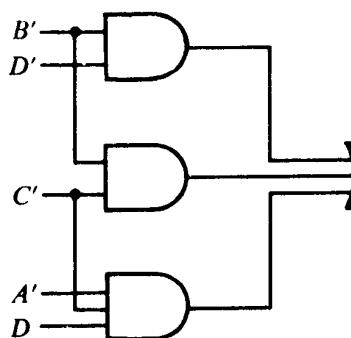
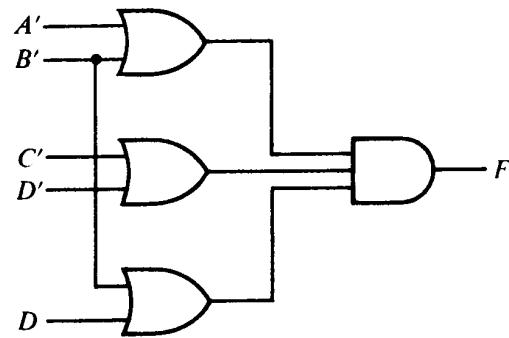


Figura 3-14 Mapa para el ejemplo 3-8;  $F(A, B, C, D) = \Sigma(0, 1, 2, 5, 8, 9, 10) = B'D' + B'C' + A'C'D = (A' + B')(C' + D')(B' + D)$ .



$$(a) \quad F = B'D' + B'C' + A'C'D$$



$$(b) \quad F = (A' + B')(C' + D')(B' + D)$$

Figura 3-15 Implementación con compuertas de la función del ejemplo 3-8.

modo que no son necesarios inversores. El patrón de configuración establecido en la Fig. 3-15 es la forma general mediante la cual cualquier función booleana se implementan cuando se expresa en una de las formas estándar. Las compuertas AND se conectan a una sola compuerta OR cuando se requiere la suma de productos; las compuertas OR se conectan a una sola compuerta AND cuando se necesita un producto de sumas. Cualquier configuración forma dos niveles de compuertas. Así que, la implementación de una función en una forma estándar se dice que es una implementación de dos niveles.

En el ejemplo 3-8 se mostró el procedimiento para obtener la simplificación cuando la función original estaba expresada en la forma canónica de suma de mintérminos. El procedimiento también es válido cuando la función original está expresada en la forma canónica de producto de maxtérminos. Considérese, por ejemplo, la tabla de verdad que define la función  $F$  en la Tabla 3-2. Esta función se expresa en suma de mintérminos como:

$$F(x, y, z) = \Sigma(1, 3, 4, 6)$$

En producto de maxtérminos, se expresa como:

$$F(x, y, z) = \Pi(0, 2, 5, 7)$$

TABLA 3-2 Tabla de verdad para la función  $F$

| $x$ | $y$ | $z$ | $F$ |
|-----|-----|-----|-----|
| 0   | 0   | 0   | 0   |
| 0   | 0   | 1   | 1   |
| 0   | 1   | 0   | 0   |
| 0   | 1   | 1   | 1   |
| 1   | 0   | 0   | 1   |
| 1   | 0   | 1   | 0   |
| 1   | 1   | 0   | 1   |
| 1   | 1   | 1   | 0   |

En otras palabras, los 1 de la función representan los mintérminos, y los 0 representan los maxtérminos. El mapa para esta función se dibuja en la Fig. 3-16. Puede iniciarse la simplificación de esta función marcando primero los 1 para cada mintérmino en los que la función es un 1. Los cuadros restantes se marcan con 0. Por otra parte, si al inicio está dado el producto de maxtérminos, puede principiarse marcando 0 en los cuadros que se listan en la función; los cuadros restantes se marcan entonces con 1. Una vez que se han marcado los 1 y 0, la función puede simplificarse en cualquiera de las formas estándar. Para la suma de productos, se combinan los 1 para obtener:

$$F = x'z + xz'$$

Para el producto de sumas, se combinan los 0 para obtener la función complementada simplificada:

$$F' = xz + x'z'$$

la cual muestra que la función excluyente OR es el complemento de la función de equivalencia (Sección 2-6). Al tomar el complemento de  $F'$ , se obtiene la función simplificada en producto de sumas:

$$F = (x' + z')(x + z)$$

Para darle entrada a una función expresada en producto de sumas en el mapa, se toma el complemento de la función y, por este medio, se encuentran los cuadros que se marcan con números 0. Por ejemplo, la función:

$$F = (A' + B' + C)(B + D)$$

puede introducirse en el mapa al tomar primero su complemento:

$$F' = ABC' + B'D'$$

y entonces se marcan números 0 en los cuadros que representan los mintérminos de  $F'$ . Los cuadros restantes se marcan con números 1.

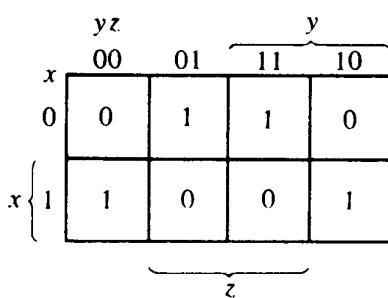


Figura 3-16 Mapa para la función de la Tabla 3-2.

### 3-6 IMPLEMENTACION CON NOR Y NAND

Los circuitos digitales con más frecuencia se construyen mediante compuertas NAND y NOR que con compuertas AND u OR. Las compuertas NAND y NOR son más fáciles de fabricar con componentes electrónicos y son las compuertas básicas que se utilizan en todas las familias IC de lógica digital. Debido a la preeminencia de las compuertas NAND y NOR en el diseño de circuitos digitales, se han desarrollado reglas y procedimientos para la conversión de las funciones booleanas dadas en términos de AND, OR y NOT en diagramas lógicos equivalentes NAND o NOR. El procedimiento para la implementación de dos niveles se presenta en esta sección. La implementación en niveles múltiples se expone en la Sección 4-7.

Para facilitar la conversión a las lógicas NAND y NOR, es conveniente definir otros dos símbolos gráficos para esas compuertas. En la Fig. 3-17(a) se muestran dos símbolos equivalentes para la compuerta NAND. El símbolo AND invertido se ha definido con anterioridad y consta de un símbolo gráfico AND seguido por un círculo pequeño. En lugar de esto, es posible representar una compuerta NOR con un símbolo gráfico OR precedido por círculos pequeños en todas las entradas. El símbolo OR invertido para la compuerta NAND es consecuencia del teorema de De Morgan y de la convención de que los círculos pequeños denotan la complementación.

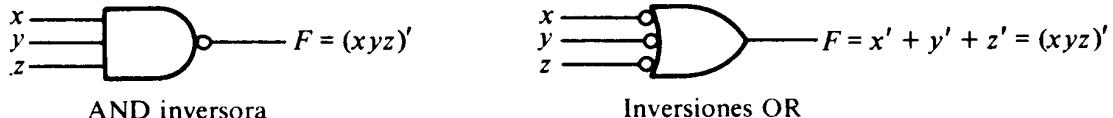
De manera semejante, hay dos símbolos gráficos para la compuerta NOR como se muestra en la Fig. 3-17(b). El símbolo OR invertido es el convencional. La compuerta AND invertida es una alternativa conveniente que utiliza el teorema de De Morgan y la convención de que los círculos pequeños en las entradas denotan la complementación.

Una compuerta de una entrada NAND o NOR se comporta como una inversora. En consecuencia, una compuerta inversora puede dibujarse en tres formas diferentes como se muestra en la Fig. 3-17(c). Los círculos pequeños en todos los símbolos inversores pueden transferirse a la terminal de entrada sin cambiar la lógica de la compuerta.

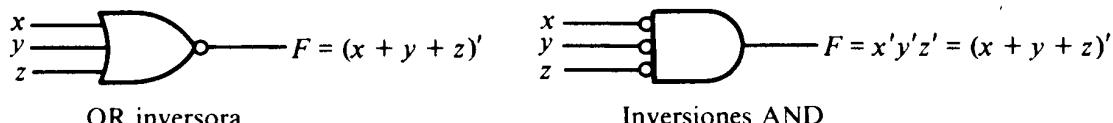
Debe puntualizarse que los símbolos alternos para las compuertas NAND y NOR pueden dibujarse con triángulos pequeños en todas las terminales de entrada en lugar de círculos. Un triángulo pequeño es un indicador de polaridad de lógica negativa (véase la Sección 2-8 y la Fig. 2-11). Con triángulos pequeños en las terminales de entrada, el símbolo gráfico denota una polaridad de lógica negativa para las entradas, pero la salida de la compuerta (que no tiene un triángulo) debe tener asignada una lógica positiva. A lo largo de este libro se prefiere utilizar la lógica positiva y emplear círculos pequeños cuando sea necesario denotar complementación.

#### Implementación con NAND

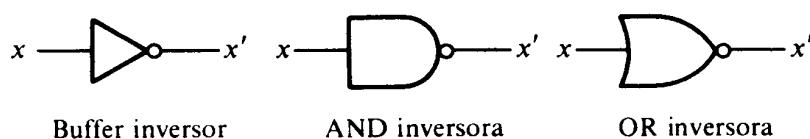
La implementación de una función booleana con compuertas NAND requiere que la función se simplifique en la forma de suma de productos. Para ver la relación entre una expresión de suma de productos y su implementación equivalente NAND, considérense los diagramas lógicos que se dibujan en la Fig. 3-18. Todos los tres diagramas son equivalentes e implementan la función:



(a) Dos símbolos gráficos para la compuerta NAND.



(b) Dos símbolos gráficos para la compuerta NOR.



(c) Tres símbolos gráficos para inversores

**Figura 3-17** Símbolos gráficos para las compuertas NAND y OR.

$$F = AB + CD + E$$

La función está implantada en la Fig. 3-18(a) en la forma de suma de productos con compuertas NAND y OR. En (b) las compuertas AND se reemplazan por compuertas NAND y la compuerta OR se sustituye por una compuerta NAND con un símbolo OR invertido. La variable única  $E$  se complementa y se aplica a la compuerta OR invertida del segundo nivel. Recuérdese que un círculo pequeño denota la complementación. Por tanto, dos círculos en la misma línea representan complementación doble y ambos pueden eliminarse. El complemento de  $E$  pasa a través de un círculo pequeño que complementa la variable otra vez para producir el valor normal de  $E$ . La eliminación de los círculos pequeños en las compuertas de la Fig. 3-18(b) produce el circuito en (a). Por tanto, los dos diagramas implementan la misma función y son equivalentes.

En la Fig. 3-18(c), la compuerta NAND de salida se vuelve a dibujar con el símbolo ordinario. La compuerta NAND de una entrada complementa la variable  $E$ . Es posible eliminar esta inversora y aplicar  $E'$  directamente a la entrada de la compuerta NAND en el segundo nivel. El diagrama en (c) es equivalente al que se muestra en (b), el cual a su vez es equivalente al diagrama en (a). Obsérvese la similitud entre los diagramas en (a) y (c). Las compuertas AND y OR se han cambiado a compuertas NAND, pero se ha incluido una compuerta NAND adicional con la variable única  $E$ . Cuando se dibujan diagramas lógicos NAND, el circuito que se muestra ya sea en (b) o (c) es aceptable. Sin embargo, el que se ilustra en (b) representa una relación más directa con la expresión booleana que implementa.

La implementación NAND en la Fig. 3-18(c) puede verificarse en forma algebraica. La función NAND que implanta puede convertirse con facilidad en una forma de suma de productos por la aplicación del teorema de De Morgan:

$$F = [(AB)' \cdot (CD)' \cdot E']' = AB + CD + E$$

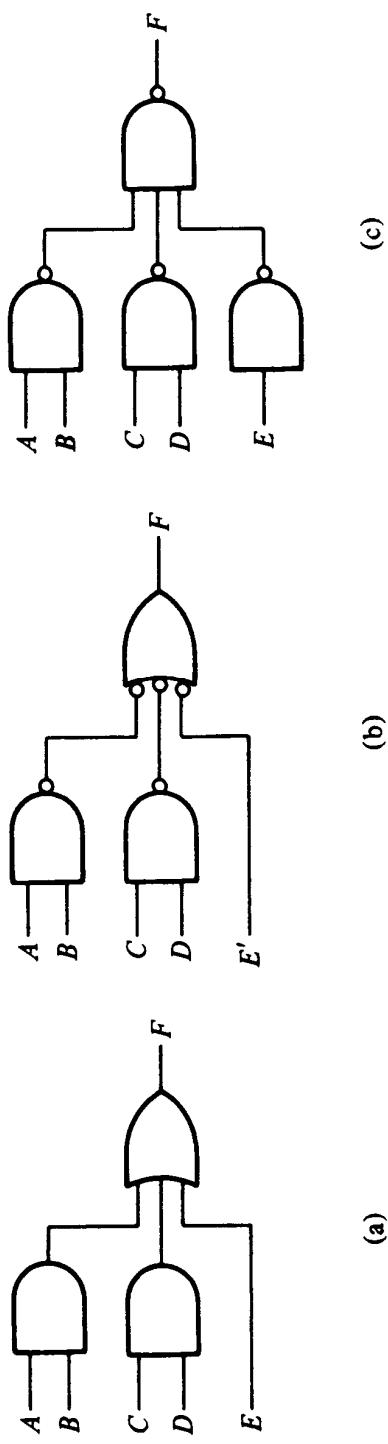


Figura 3-18 Tres formas de implementar  $F = AB + CD + E$ .

Mediante la transformación que se muestra en la Fig. 3-18, se concluye que una función booleana puede implementarse con dos niveles de compuertas NAND. La regla para obtener el diagrama lógico NAND mediante una función booleana es la siguiente:

1. Simplifíquese la función y exprésese en una suma de productos.
2. Dibújese una compuerta NAND para cada término producto de la función que tiene cuando menos dos literales. Las entradas a cada compuerta NAND son las literales del término. Esto constituye un grupo de compuertas de primer nivel.
3. Dibújese una sola compuerta NAND (utilizando el símbolo gráfico AND invertido o bien OR invertido) en el segundo nivel, con entradas que vienen de las salidas de las compuertas del primer nivel.
4. Un término con una sola literal requiere un inversor en el primer nivel o puede complementarse y aplicarse como una entrada a la compuerta NAND en el segundo nivel.

Antes de aplicar estas reglas a un ejemplo específico, debe mencionarse que hay una segunda forma de implementar una función booleana con compuertas NAND. Recuérdese que si se combinan los 0 en un mapa, se obtiene la expresión simplificada del *complemento* de la función en suma de productos. Entonces, el complemento de la función puede implementarse con dos niveles de compuertas NAND por el uso de las reglas que antes se establecieron. Si se desea la salida normal, es necesario insertar una compuerta NAND de una entrada o inversora para generar el valor verdadero de la variable de salida. Hay ocasiones en que es posible que el diseñador quiera generar el complemento de la función; de modo que puede ser preferible emplear este segundo método.

**EJEMPLO 3-9:** Impleméntese la siguiente función con compuertas NAND:

$$F(x, y, z) = \Sigma(0, 6)$$

El primer paso es simplificar la función en la forma de suma de productos. Esto se intenta con el mapa que se muestra en la Fig. 3-19(a). Hay sólo dos 1 en el mapa y no puede combinarse. La función simplificada en suma de productos para este ejemplo es:

$$F = x'y'z' + xyz'$$

La implementación NAND de dos niveles se muestra en la Fig. 3-19(b). A continuación se tratará de simplificar el complemento de la función en suma de productos. Esto se hace por la combinación de los 0 en el mapa:

$$F' = x'y + xy' + z$$

La compuerta NAND de dos niveles para generar  $F'$  se muestra en la Fig. 3-19(c). Si se requiere la salida  $F$ , es necesario agregar una compuerta NAND de una entrada para invertir la función. Esto da una implementación de tres niveles. En cada caso, se supone que están disponibles las variables de entrada tanto en forma normal como complementaria. Si están disponibles sólo en una forma, puede ser necesario insertar inversores en las entradas, lo que agregaría otro nivel a los circuitos. La compuerta NAND de una entrada asociada con la variable única  $z$  puede eliminarse siempre que la entrada se cambie a  $z'$ .

### Implementación NOR

La función NOR es la dual de la función NAND. Por esta razón, todos los procedimientos y las reglas para la lógica NOR son la dual de los procedimientos y reglas correspondientes que se desarrollaron para la lógica NAND.

La implementación de una función booleana con compuertas NOR requiere que la función se simplifique en la forma de producto de sumas. Una expresión en producto de sumas especifica un grupo de compuertas OR para los términos suma, seguidos por una compuerta AND para obtener el producto. La transformación del diagrama OR-AND al NOR-NOR se indica en la Fig. 3-20. Es similar a la transformación NAND expuesta, excepto que ahora se utiliza la expresión de producto de sumas:

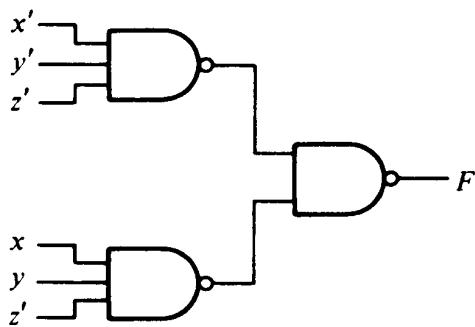
$$F = (A + B)(C + D)E$$

|   |   | yz |    | y  |    |
|---|---|----|----|----|----|
|   |   | 00 | 01 | 11 | 10 |
| x |   | 0  | 1  | 0  | 0  |
| x | 0 | 1  | 0  | 0  | 0  |
|   | 1 | 0  | 0  | 0  | 1  |

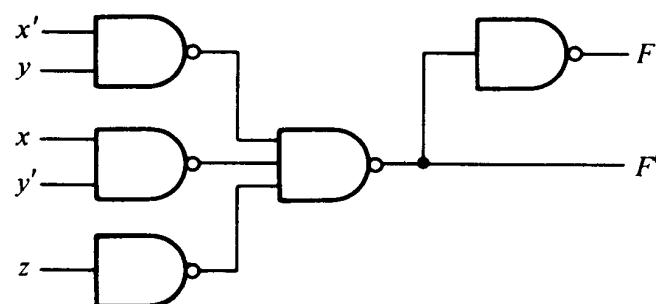
$\underbrace{\phantom{0}}_{z}$

$F = x'y'z' + xyz'$   
 $F' = x'y + xy' + z$

(a) para la simplificación en suma de productos



$$(b) F = x'y'z' + xyz'$$



$$(c) F' = x'y + xy' + z$$

Figura 3-19 Implementación de la función del ejemplo 3-9 con compuertas NAND.

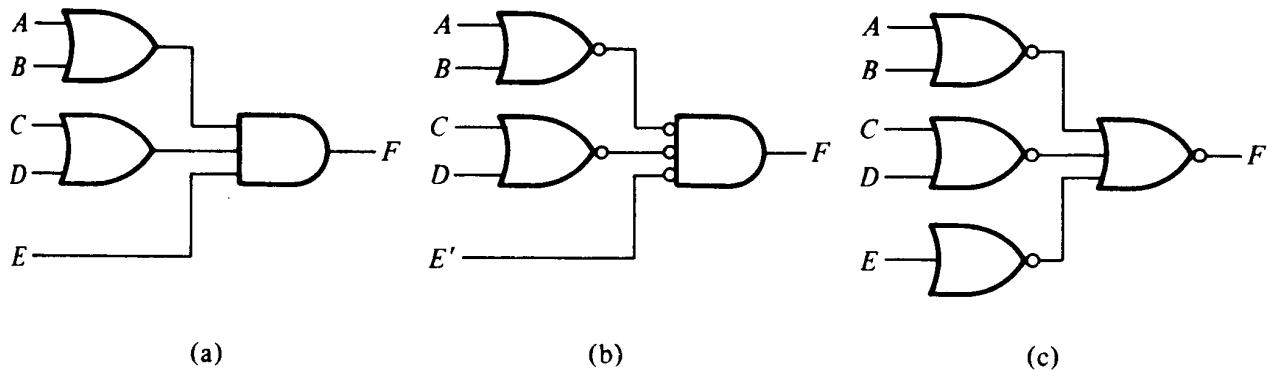


Figura 3-20 Tres formas de implementar  $F = (A + B)(C + D)E$ .

La regla para obtener el diagrama lógico NOR de una función booleana puede derivarse de esta transformación. Es similar a la regla NAND de tres pasos, excepto que la expresión simplificada debe ser el producto de sumas y los términos para el primer nivel de compuertas NOR son los términos suma. Un término con una sola literal requiere una compuerta de una entrada NOR o inversora, o bien puede complementarse y aplicarse directamente al segundo nivel de compuerta NOR.

Un segundo modo para implementar una función con compuertas NOR sería usar la expresión para el complemento de la función en producto de sumas. Esto dará una implementación de nivel dos para  $F'$  y una implementación de nivel tres si se requiere la salida normal de  $F$ .

Para obtener el producto simplificado de sumas de un mapa, es necesario combinar los 0 en el mapa y complementar entonces la función. Para obtener la expresión simplificada de producto de sumas para el complemento de la función, es necesario combinar los 1 en el mapa y complementar entonces la función. El siguiente ejemplo demuestra el procedimiento para la implementación NOR.

**EJEMPLO 3-10:** Implemente la función del Ejemplo 3-9 con compuertas NOR.

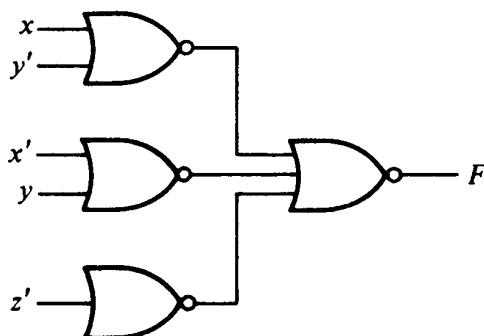
El mapa de esta función se dibuja en la Fig. 3-19(a). Primero, se combinan los 0 en el mapa para obtener dos puntos.

$$F' = x'y + xy' + z$$

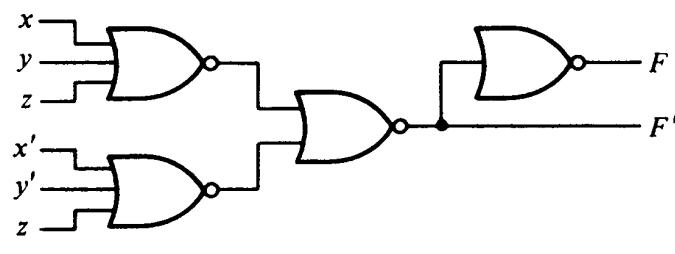
Esto es el complemento de la función en suma de productos. Se complementa  $F'$  para obtener la función simplificada en producto de sumas como se requiere para la implementación NOR:

$$F = (x + y')(x' + y)z'$$

La implementación de dos niveles con compuertas NOR se muestra en la Fig. 3-21(a). El término con una sola literal  $z'$  requiere una compuerta de una entrada NOR o inversora. Esta se elimina y la entrada  $z$  se aplica directamente a la entrada de segundo nivel NOR.



$$(a) F = (x + y')(x' + y)z'$$



$$(b) F' = (x + y + z)(x' + y' + z)$$

Figura 3-21 Implementación con compuertas NOR.

Es posible una segunda implementación mediante el complemento de la función en producto de sumas. Para este caso, se combinan primero los 1 en el mapa para obtener:

$$F = x'y'z' + xyz'$$

Esta es la expresión simplificada en suma de productos. Se complementa esta función para obtener el complemento de la función en producto de sumas como se requiere para la implementación NOR:

$$F' = (x + y + z)(x' + y' + z)$$

La implementación de dos niveles para  $F'$  se muestra en la Fig. 3-21(b). La salida  $F$ , puede generarse con una inversora en el tercer nivel.

En la Tabla 3-3 se resumen los procedimientos para la implementación NAND o NOR. No debe olvidarse simplificar siempre la función con objeto de reducir el número de compuertas en la implementación. Las formas estándar obtenidas por los procedimientos de simplificación de mapa se aplican de manera directa y son muy útiles cuando se trata con las lógicas NAND o NOR.

TABLA 3-3 Reglas para las implementaciones NAND y NOR

| Caso | Función que se simplifica | Forma estándar a usar | Cómo derivarla              | Implementada con | Número de niveles para $F$ |
|------|---------------------------|-----------------------|-----------------------------|------------------|----------------------------|
| (a)  | $F$                       | Suma de productos     | Combínense los 1 en el mapa | NAND             | 2                          |
| (b)  | $F'$                      | Suma de productos     | Combínense los 0 en el mapa | NAND             | 3                          |
| (c)  | $F$                       | Producto de sumas     | Compleméntese $F'$ en (b)   | NOR              | 2                          |
| (d)  | $F'$                      | Producto de sumas     | Compleméntese $F$ en (a)    | NOR              | 3                          |

### 3-7 OTRAS IMPLEMENTACIONES DE NIVEL DOS

Los tipos de compuertas que con más frecuencia se encuentran en los circuitos integrados son NAND y NOR. Por esta razón, las implementaciones de las lógicas NAND y NOR son las más importantes desde el punto de vista práctico. Algunas compuertas NAND o NOR (pero no todas) permiten la posibilidad de una conexión alambrada entre las salidas de dos compuertas para proporcionar una función lógica específica. Este tipo de lógica se conoce como *lógica alambrada*. Por ejemplo, las compuertas de colector abierto TTL NAND, cuando se ligan, realizan la lógica alambrada AND. (La compuerta de colector abierto TTL se muestra en el Capítulo 10, Fig. 10-11.) La lógica alambrada AND realizada con dos compuertas NAND se muestra en la Fig. 3-22(a). La compuerta AND se dibuja con las líneas pasando a través del centro de la compuerta para distinguirla de una compuerta convencional. La compuerta alambrada AND no es una compuerta física, sino sólo un símbolo para designar la función que se obtiene por la conexión alambrada indicada. La función lógica que implanta el circuito en la Fig. 3-22(a) es:

$$F = (AB)' \cdot (CD)' = (AB + CD)'$$

y se denomina función AND-OR-INVERTIDA.

De manera semejante, la salida NOR de las compuertas ECL pueden ligarse para realizar una función alambrada OR. La función lógica que implementa el circuito en la Fig. 3-22(b) es:

$$F = (A + B)' + (C + D)' = [(A + B)(C + D)]'$$

y se denomina función OR-AND-INVERTIDA.

Una compuerta lógica alambrada no produce una compuerta física de segundo nivel, ya que simplemente es una conexión alambrada. Sin embargo, para propósitos de exposición se considerarán los circuitos de la Fig. 3-22 como una implementación de nivel dos. El primer nivel consta de compuertas NAND (o NOR) y el segundo nivel tiene una sola compuerta AND (u OR). La conexión alambrada en el símbolo gráfico se omitirá en las exposiciones subsecuentes.

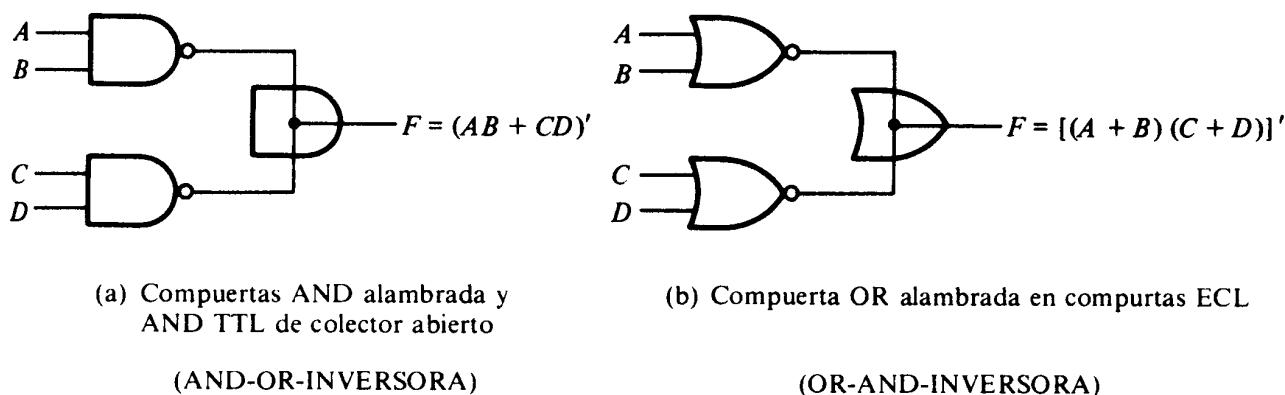


Figura 3-22 Lógica alambrada.

## Formas no degeneradas

Desde un punto de vista teórico será instructivo encontrar cuántas combinaciones de compuertas en el nivel dos son posibles. Se consideran cuatro tipos de compuertas: AND, OR, NAND y NOR. Si se asigna un tipo de compuerta para el primer nivel y un tipo para el segundo nivel, se encuentra que hay 16 combinaciones posibles de formas de nivel dos. (El mismo tipo de compuerta puede estar en el primero y segundo niveles, como en la implementación NAND-NAND.) Ocho de estas combinaciones se dice que son formas *degeneradas* ya que degeneran en una sola operación. Esto puede verse mediante un circuito con compuertas AND en el primer nivel y una compuerta AND en el segundo nivel. La salida del circuito es simplemente la función AND de todas las variables de entrada. Las otras ocho formas no degeneradas producen una implementación en suma de productos o producto de sumas. Las ocho formas *no degeneradas* son:

|           |          |
|-----------|----------|
| AND-OR    | OR-AND   |
| NAND-NAND | NOR-NOR  |
| NOR-OR    | NAND-AND |
| OR-NAND   | AND-NOR  |

La primera compuerta que se lista en cada una de las formas constituye un primer nivel en la implementación. La segunda compuerta que se lista es una sola compuerta colocada en el segundo nivel. Obsérvese que cualesquiera de las dos formas listadas en el mismo renglón son las duales de las otras.

Las formas AND-OR y OR-AND son las formas básicas de nivel dos que se exponen en la Sección 3-5. Las NAND-NAND y NOR-NOR se introdujeron en la Sección 3-6. Las cuatro formas restantes se investigan en esta sección.

## Implementación con AND-OR-INVERTIDA

Las dos formas NAND-AND y AND-NOR son formas equivalentes y pueden tratarse juntas. Ambas realizan la función AND-OR-INVERTIDA, como se muestra en la Fig. 3-23. La forma AND-NOR es semejante a la forma AND-OR con una inversión hecha por el círculo pequeño en la salida de la compuerta NOR. Implantá la función:

$$F = (AB + CD + E)'$$

Por el uso del símbolo gráfico alterno para la compuerta NOR, se obtiene el diagrama de la Fig. 3-23(b). Obsérvese que la única variable *E* no está complementada debido a que el único cambio hecho es en el símbolo gráfico de la compuerta NOR. Ahora se mueven los círculos de la terminal de entrada de la compuerta de segundo nivel a las terminales de salida de las compuertas de primer nivel. Se necesita una inversora para la variable única para mantener el círculo. En forma alterna, la inversora puede quitarse siempre que la entrada *E* se complemente. El circuito de la Fig. 3-23(c) es una forma NAND-AND y se mostró en la Fig. 3-22 para implementar la función AND-OR-INVERTIDA.

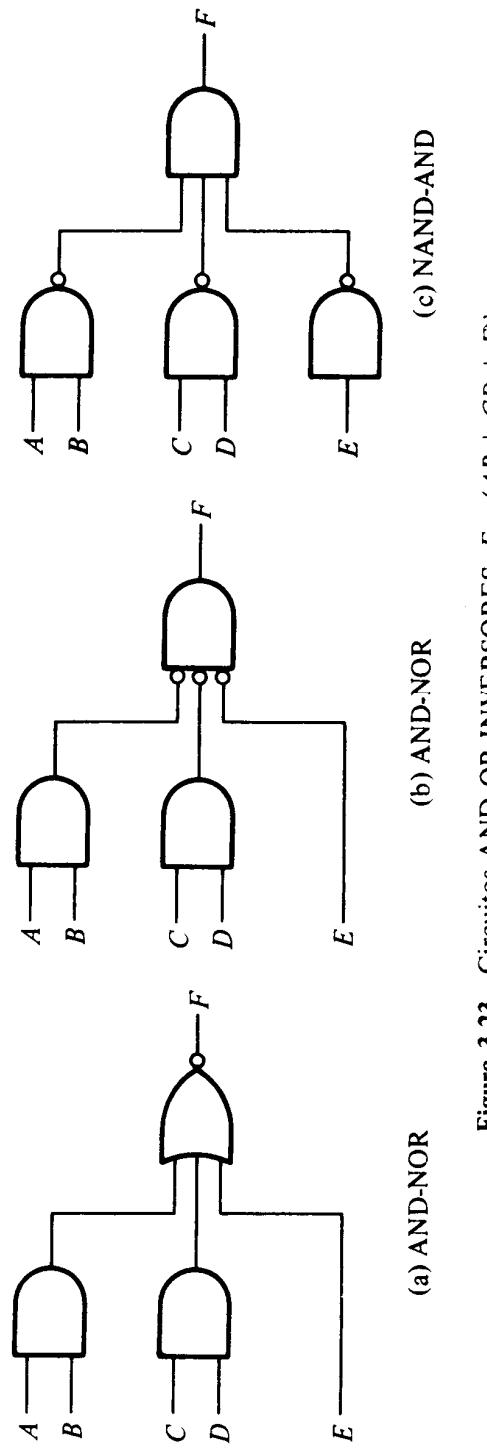


Figura 3-23 Circuitos AND-OR-INVERSORES;  $F = (AB + CD + E)'$ .

Una implementación AND-OR requiere una expresión en suma de productos. La implementación AND-OR-INVERTIDA es similar excepto por la inversión. Por consiguiente, si el complemento de la función se simplifica en suma de productos (por la combinación de los 0 en el mapa), será posible implantar  $F'$  con la parte AND-OR de la función. Cuando  $F'$  pasa a través de la inversión de salida siempre presente (la parte inversora), generará la salida  $F$  de la función. Un ejemplo de la implementación AND-OR-INVERTIDA se mostrará posteriormente.

### Implementación con OR-AND-INVERTIDA

Las formas OR-NAND y NOR-OR realizan la función OR-AND-INVERTIDA. Esto se muestra en la Fig. 3-24. La forma OR-NAND se asemeja a la forma OR-AND, a excepción de la inversión hecha por el círculo en la compuerta NAND. Implanta la función:

$$F = [(A + B)(C + D)E]'$$

Por el uso del símbolo gráfico alterno para la compuerta NAND, se obtiene el diagrama de la Fig. 3-24(b). El circuito en (c) se obtiene moviendo los círculos pequeños de las entradas de la compuerta de segundo nivel a las salidas de las compuertas de primer nivel. El circuito de la Fig. 3-24(c) es una forma NOR-OR y se mostró en la Fig. 3-22 para implementar la función OR-AND-INVERTIDA.

La implementación OR-AND-INVERTIDA requiere una expresión en producto de sumas. Si el complemento de la función se simplifica en producto de sumas, puede implementarse  $F'$  con la parte OR-AND de la función. Cuando  $F'$  pasa a través de la parte INVERSORA, se obtiene el complemento de  $F'$ , o  $F$ , en la salida.

### Resumen tabular y ejemplo

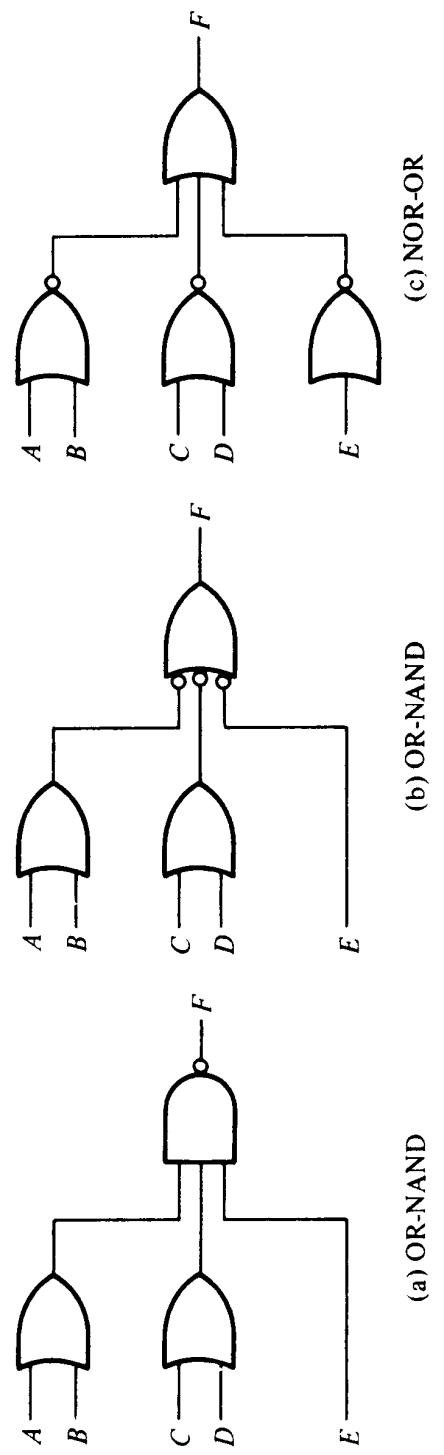
En la Tabla 3-4 se resumen los procedimientos para implementarse una función booleana en cualquiera de las cuatro formas de nivel dos. Debido a la parte INVERSORA en cada caso, es conveniente utilizar la simplificación de  $F'$  (el complemento) de la función. Cuando se implanta  $F'$  en una de esas formas, se obtiene el complemento de la función en la forma AND-OR o OR-AND. Las cuatro formas de nivel dos invierten esta función, dando una salida que es el complemento de  $F'$ . Esta es la salida normal  $F$ .

**EJEMPLO 3-11:** Implante la función de la Fig. 3-19(a) con las cuatro formas de nivel dos que se listan en la Tabla 3-4. El complemento de la función se simplifica en suma de productos por la combinación de los 0 en el mapa:

$$F' = x'y + xy' + z$$

La salida normal para esta función puede expresarse como

$$F = (x'y + xy' + z)'$$



**Figura 3-24** Circuitos OR-AND-INVERSORES;  $F = [(A + B)(C + D)E]'$ .

TABLA 3-4 Implementación con otras formas de dos niveles

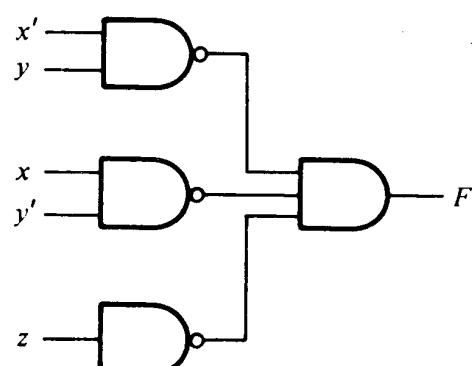
| Forma equivalente no degenerada | Implementación de función        | Simplificar $F'$ en                                                                                     | Para obtener una salida de |
|---------------------------------|----------------------------------|---------------------------------------------------------------------------------------------------------|----------------------------|
| (a)                             | (b)*                             |                                                                                                         |                            |
| <b>AND-NOR</b>                  | <b>NAND-AND OR-AND-INVERSORA</b> | Suma de productos por la combinación de números 0 en el mapa                                            | $F$                        |
| <b>OR-NAND</b>                  | <b>NOR-OR</b>                    | AND-OR-INVERSORA<br>Producto de sumas por la combinación de números 1 en el mapa, complementar entonces | $F$                        |

\* La forma (b) requiere una compuerta NAND de una entrada o NOR (inversora) para un solo término literal.

la cual está en la forma AND-OR-INVERTIDA. Las implementaciones AND-NOR y NAND-AND se muestran en la Fig. 3-25(a). Obsérvese que se necesita una compuerta de una entrada NAND o inversora en la imple-

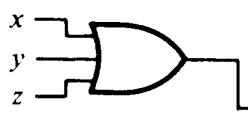


AND-NOR

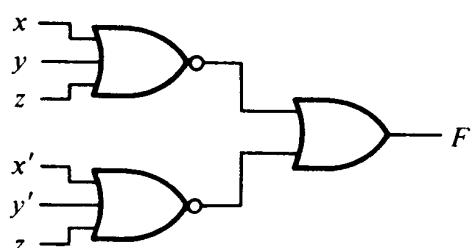


NAND-AND

$$(a) F = (x'y + xy' + z)'$$



OR-NAND



NOR-OR

$$(b) F = [(x + y + z)(x' + y' + z)]'$$

Figura 3-25 Otras implementaciones de dos niveles.

mentación de la NAND-AND, pero no en el caso AND-NOR. La inversora puede eliminarse si se aplica la variable de entrada  $z'$  en lugar de  $z$ . Las formas OR-AND-INVERTIDA requieren una expresión simplificada del complemento de la función en producto de sumas. Para obtener esta expresión, primero deben combinarse los 1 en el mapa:

$$F = x'y'z' + xyz'$$

Entonces se toma el complemento de la función:

$$F' = (x + y + z)(x' + y' + z)$$

Ahora la salida normal  $F$  puede expresarse en la forma:

$$F = [(x + y + z)(x' + y' + z)]'$$

la cual está en la forma OR-AND-INVERTIDA. Mediante esta expresión puede implantarse la función en las formas OR-NAND y NOR-OR como se muestra en la Fig. 3-25(b).

### 3-8 CONDICIONES NO IMPORTA

Los 1 y los 0 en el mapa significan la combinación de variables que hacen la función igual a 1 o 0, respectivamente. Las combinaciones por lo común se obtienen de una tabla de verdad donde se listan las condiciones bajo las cuales la función es un 1. La función se supone que es igual a 0 bajo todas las otras condiciones. Esta suposición no siempre es verdadera ya que hay aplicaciones donde ciertas combinaciones de variables de entrada nunca ocurren. Un código decimal de cuatro bits, por ejemplo, tiene seis combinaciones que no se usan. Cualquier circuito digital que utiliza este código opera bajo la suposición de que estas combinaciones no usadas nunca ocurrirán mientras el sistema trabaje apropiadamente. Como resultado, no importa cuál es la salida de la función para esas combinaciones de las variables, ya que está garantizado que nunca ocurrirán. Estas condiciones no importa pueden usarse en un mapa para proporcionar simplificación adicional de la función.

Debe tomarse en cuenta que una combinación no importa no puede marcarse con un 1 en el mapa, debido a que requeriría que la función siempre fuera un 1 para tal combinación de entrada. En forma similar, colocar un 0 en el cuadro requiere que la función sea 0. Para distinguir las condiciones no importa de números 1 y 0, se usará una  $X$ .

Cuando se escogen cuadros adyacentes para simplificar la función en el mapa, puede suponerse que las  $X$  son ya sea 0 o 1, lo que dé la expresión más simple. Además, en absoluto es necesario usar  $X$  si no contribuye a cubrir un área más grande. En cada caso, la elección depende sólo de la simplificación que pueda lograrse.

**EJEMPLO 3-12:** Simplifique la función booleana:

$$F(w, x, y, z) = \Sigma(1, 3, 7, 11, 15)$$

y las condiciones de no importa:

$$d(w, x, y, z) = \Sigma(0, 2, 5)$$

Los mintérminos de  $F$  son las combinaciones de variables que hacen que la función sea igual a 1. Los mintérminos de  $d$  son las combinaciones no importa que se sabe nunca ocurren. La minimización se muestra en la Fig. 3-26. Los mintérminos de  $F$  se marcan con números 1, los de  $d$  se marcan con  $X$  y los cuadros restantes se llenan con números 0. En (a), los 1 y  $X$ , se combinan en cualquier forma conveniente de modo que incluyan el número máximo de cuadros adyacentes. No es necesario incluir todas o cualesquiera de las  $X$ , sino sólo las que son útiles para simplificar un término. Una combinación que da una función mínima encierra una  $X$  y deja dos fuera. Esto resulta en una función simplificada de suma de productos:

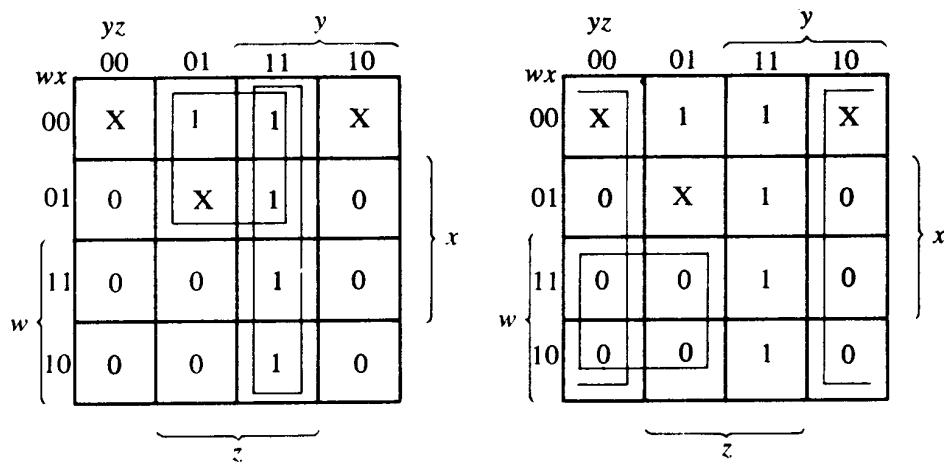
$$F = w'z + yz$$

En (b), los 0 se combinan con cualesquiera  $X$  convenientes para simplificar el complemento de la función. Los mejores resultados se obtienen y se encierran las dos  $X$  como se muestra. La función complemento se simplifica a:

$$F' = z' + wy'$$

Complementando de nuevo, se obtiene una función simplificada de producto de sumas:

$$F = z(w' + y)$$



(a) Combinación de 1 y  $X$   $F = w'z + yz$       (b) Combinación de 0 y  $X$   $F = z(w' + y)$

Figura 3-26 Ejemplo con condiciones no importa.

Las dos expresiones que se obtuvieron en el Ejemplo 3-12 dan dos funciones que puede mostrarse que son algebraicamente iguales. Este no es siempre el caso cuando están implicadas condiciones no importa. De hecho, si se usa una  $X$  como un 1 cuando se combinan los 1 y otra vez como un 0 cuando se combinan los 0, las dos funciones resultantes no darán respuestas algebraicamente iguales. La selección de las condiciones no importa como un 1 en el primer caso y como un 0 en el segundo resultan en expresiones de mintérminos diferentes y, por lo tanto, funciones diferentes. Esto puede verse en el Ejemplo 3-12. En la solución de este ejemplo, la  $X$  que se elige para ser 1 no se escogió para ser un 0. Ahora, si en la Fig. 3-26(a) se elige el término  $w'x'$  en lugar de  $w'z'$ , se sigue obteniendo una función minimizada:

$$F = w'x' + yz$$

Pero no es igual algebraicamente a la obtenida en producto de sumas debido a que las mismas  $X$  se utilizaron como números 1 en la primera minimización y como números 0 en la segunda.

Este ejemplo también demuestra que una expresión con el número mínimo de literales no es necesariamente única. Algunas veces el diseñador se enfrenta con una elección entre dos términos con un número igual de literales, y cualquier elección resulta en una expresión minimizada.

### 3-9 METODO DE TABULACION

El método de mapa de simplificación es conveniente en tanto que el número de variables no exceda cinco o seis. Conforme aumenta el número de variables, el número excesivo de cuadros evita una selección razonable de cuadros adyacentes. La desventaja obvia del mapa es que en esencia es un procedimiento de ensayo y error, que depende de la habilidad del usuario humano para reconocer ciertos patrones. Para funciones de seis o más variables, es difícil tener la seguridad de que se ha hecho la mejor selección.

El método de tabulación supera esta dificultad. Es un procedimiento específico de paso a paso que está garantizado para producir una expresión simplificada en forma estándar para una función. Puede aplicarse a problemas con muchas variables y tiene la ventaja de ser adecuado para computación por máquina. Sin embargo, es bastante tedioso para el uso humano y propenso a errores debido a su proceso rutinario y monótono. El método de tabulación lo formuló por vez primera Quine (3) y lo mejoró posteriormente McCluskey (4). También se conoce como método de Quine-McCluskey.

El método tabular de simplificación consta de dos partes. La primera es encontrar por una búsqueda exhaustiva todos los términos que son candidatos para su inclusión en la función simplificada. Estos términos se denominan implicantes primos. La segunda operación es escoger entre los implicantes primos los que dan una expresión con el menor número de literales.

### 3-10 DETERMINACION DE LOS IMPLICANTES PRIMOS\*

El punto de inicio del método de tabulación es la lista de los mintérminos que especifican la función. La primera operación tabular es encontrar los implicantes primos usando un proceso de comparación. Este proceso compara cada mintérmino con cada uno de los otros mintérminos. Si dos mintérminos difieren sólo en una variable, esta variable se elimina y se encuentra un término con una literal menos. Este proceso se repite para cada mintérmino hasta que se completa la búsqueda exhaustiva. El ciclo del proceso de comparación se repite para los nuevos términos que acaban de encontrarse. Los ciclos tercero y posteriores se continúan hasta que un paso único a través de un ciclo no rinde más eliminación de literales. Los términos restantes y todos los términos que no comparan durante el proceso comprenden los implicantes primos. Este método de tabulación se ilustra en el siguiente ejemplo.

**EJEMPLO 3-13:** Simplifique la siguiente función booleana utilizando el método de tabulación:

$$F = \Sigma(0, 1, 2, 8, 10, 11, 14, 15)$$

Paso 1: Se hace la representación binaria de grupo de los mintérminos de acuerdo con el número de 1's contenidos, como se muestra en la Tabla 3-5, columna (a). Esto se hace agrupando los mintérminos en cinco secciones separadas por líneas horizontales. La primera sección contiene el número sin números 1 en ella. La segunda sección contiene los números que tienen sólo un 1. La tercera, cuarta y quinta secciones contienen los números binarios con dos, tres y cuatro números 1, respectivamente. Los equivalentes decimales de los mintérminos también se llevan para identificación.

Paso 2: Cualesquiera dos mintérminos que difieran uno del otro sólo por una variable pueden combinarse, y la variable que no compara se elimina. Dos números mintérminos caen en esta categoría si ambos tienen el mismo valor de bit en todas las posiciones excepto una. Los mintérminos en una sección se comparan con los de la siguiente hacia abajo solamente, debido a que dos términos que difieren por más de un bit no pueden comparar. El mintérmino en la primera sección se compara con cada uno de los tres mintérminos en la segunda sección. Si dos números cualesquiera son los mismos en cada posición excepto una, se coloca una marca a la derecha de ambos mintérminos para mostrar que se han utilizado. El término resultante, junto con los equivalentes decimales, se lista en la columna (b) de la tabla. La variable eliminada durante la comparación se indica con un guión en su posición original.

\* Esta sección y la siguiente pueden omitirse sin pérdida de continuidad.

**TABLA 3-5** Determinación de implicantes primos para el Ejemplo 3-13

| (a)                          | (b)                                  | (c)                                              |
|------------------------------|--------------------------------------|--------------------------------------------------|
| w x y z                      | w x y z                              | w x y z                                          |
| 0 0 0 0 0 ✓                  | 0, 1 0 0 0 -<br>0, 2 0 0 - 0 ✓       | 0, 2, 8, 10 - 0 - 0<br>0, 8, 2, 10 - 0 - 0       |
| 1 0 0 0 1 ✓<br>2 0 0 1 0 ✓   | 0, 8 - 0 0 0 ✓                       | 10, 11, 14, 15 1 - 1 -<br>10, 14, 11, 15 1 - 1 - |
| 8 1 0 0 0 ✓                  | 2, 10 - 0 1 0 ✓<br>8, 10 1 0 - 0 ✓   |                                                  |
| 10 1 0 1 0 ✓                 | 10, 11 1 0 1 - ✓                     |                                                  |
| 11 1 0 1 1 ✓<br>14 1 1 1 0 ✓ | 10, 14 1 - 1 0 ✓                     |                                                  |
| 15 1 1 1 1 ✓                 | 11, 15 1 - 1 1 ✓<br>14, 15 1 1 1 - ✓ |                                                  |

En este caso  $m_0$  (0000) combina con  $m_1$  (0001) para formar (000-). Esta combinación es equivalente a la operación algebraica:

$$m_0 + m_1 = w'x'y'z' + w'x'y'z = w'x'y'$$

El mintérmino  $m_0$  también combina con  $m_2$  para formar (00-0) y con  $m_8$  para formar (-000). El resultado de esta comparación se coloca en la primera sección de la columna (b). Los mintérminos de las secciones dos y tres de la columna (a) se comparan a continuación para producir los términos que se listan en la segunda sección de la columna (b). Todas las otras secciones de (a) se comparan en forma similar y se forman las secciones subsecuentes en (b). Este proceso de comparación exhaustivo resulta en las cuatro secciones de (b).

Paso 3: Los términos de la columna (b) tienen sólo tres variables. Un 1 bajo la variable indica que no tiene prima, un 0 significa que tiene prima, y un guion significa que la variable no se incluye en el término. El proceso de búsqueda y comparación se repite para los términos en la columna (b) para formar los términos de dos variables de la columna (c). De nuevo, los términos en cada sección necesitan compararse sólo si tienen guiones en la misma posición. Observe que el término (000-) no compara con cualquier otro término. Por tanto, no tiene marca de verificación a la derecha. Los equivalentes decimales se escriben en el lado izquierdo de cada entrada para propósitos de identificación. El proceso de comparación debe llevarse a cabo otra vez en la columna (c).

y en las columnas subsecuentes, en tanto se encuentre una comparación apropiada. En este ejemplo, la operación se detiene en la tercera columna.

Paso 4: Los términos que no están marcados en la tabla forman los implicantes primos. En este ejemplo, se tiene el término  $w'x'y'$  (000-) en la columna (b), y los términos  $x'z'$  (-0-0) y  $wy$  (1-1-) en la columna (c). Observe que cada término en la columna (c) aparece dos veces en la tabla, y en tanto el término forma un implicante primo, es innecesario usar el mismo término dos veces. La suma de los implicantes primos da una expresión simplificada de la función. Esto se debe a que cada término marcado en la tabla lo ha tomado en cuenta una entrada en un término más simple en una columna subsecuente. Por tanto, las entradas no marcadas (implicantes primos) son los términos que se dejan para formular la función. Para este ejemplo, la suma de los implicantes primos de la función minimizada en suma de productos:

$$F = w'x'y' + x'z' + wy$$

Vale la pena comparar estas respuestas con la que se obtuvo por el método de mapa. En la Fig. 3-27 se muestra el mapa de simplificación de esta función. Las combinaciones de cuadros adyacentes dan los tres implicantes primos de la función. La suma de estos tres términos es la expresión simplificada en suma de productos.

Es importante puntualizar que el Ejemplo 3-13 se escogió con el propósito de dar la función simplificada a partir de la suma de los implicantes primos. En la mayoría de otros casos, la suma de implicantes primos no necesariamente forma la expresión con el número mínimo de términos. Esto se demuestra en el Ejemplo 3-14.

La manipulación tediosa que debe seguirse cuando se utiliza el método de tabulación se reduce si se compara con la que se hace con números decimales en lugar de binarios. Se mostrará un método en el que se usa la sustracción de números decimales en lugar de comparar e igualar los números binarios. Obsérvese que cada 1 en un número binario representa el coeficiente multiplicado por una potencia de 2.

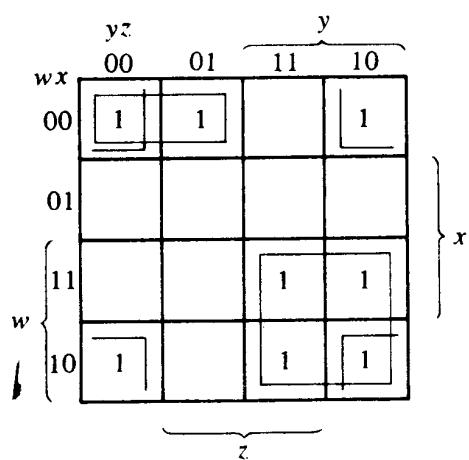


Figura 3-27 Mapa para la función del Ejemplo 3-13;  $F = w'x'y' + x'z' + wy$ .

Cuando dos mintérminos son los mismos en cada posición excepto uno, el mintérmino con el 1 adicional debe ser mayor que el número de otros mintérminos por una potencia de 2. Por tanto, dos mintérminos pueden combinarse si el número del primer mintérmino difiere en una potencia de 2 respecto a un segundo número más grande en la siguiente sección abajo en la tabla. Se ilustrará este procedimiento repitiendo el Ejemplo 3-13.

Como se muestra en la Tabla 3-6, en la columna (a), los mintérminos se ordenan en secciones como antes, excepto que ahora sólo se listan los equivalentes decimales de los mintérminos. El proceso de comparar mintérminos es como sigue: Inspecciónense cada dos números decimales en secciones adyacentes de la tabla. Si el número en la sección abajo es *mayor* que el número que en la sección arriba por una potencia de 2 (esto es, 1, 2, 4, 8, 16, etc.), verifíquense ambos números para mostrar que se han usado y se escribe abajo en la columna (b). El par de números transferidos a la columna (b) incluye un tercer número entre paréntesis que designa la potencia de 2 por la cual difieren los números. El número entre paréntesis indica la posición del guión en la notación binaria. El resultado de todas las comparaciones de la columna (a) se muestra en la columna (b).

La comparación entre secciones adyacentes en la columna (b) se lleva a cabo de manera semejante, excepto que sólo se comparan los términos con el mismo número entre paréntesis. El par de números en una sección debe diferir por una potencia de 2 del par de números en la sección siguiente. Y los números en la sección siguiente abajo deben ser *mayores* para que tenga lugar la combinación. En la columna (c), se escriben todos los cuatro números decimales con los dos números entre paréntesis designando la posición de los guiones. Una comparación de las Tablas 3-5 y 3-6 puede ser de ayuda para entender las derivaciones en la Tabla 3-6.

Los implicantes primos son los términos no señalados en la tabla. Estos son los mismos de antes, excepto que están dados en notación decimal. Para convertir de la notación decimal a la binaria, conviértanse todos los números decimales en el término en binarios y entonces insértese un guión en las posiciones designadas por los números entre paréntesis. Por eso, 0, 1 (1) se convierte en binario como 0000, 0001; un guión en la primera posición de cualquier número da como resultado (000-). En forma similar, 0, 2, 8, 10 (2, 8), se convierten a la notación binaria de 0000, 0010, 1000 y 1010, y se inserta un guión en las posiciones 2 y 8, para producir (-0-0).

**EJEMPLO 3-14:** Determine los implicantes primos de la función:

$$F(w, x, y, z) = \Sigma(1, 4, 6, 7, 8, 9, 10, 11, 15)$$

Los números mintérminos se agrupan en secciones como se muestra en la Tabla 3-7, columna (a). El equivalente binario del mintérmino se incluye con el propósito de contar el número de 1. Los números binarios en la primera sección tienen sólo un 1; en la segunda sección, dos 1; etc. Los números mintérminos se comparan por el método decimal y se encuentra un par si el número en la sección de abajo es mayor que el que está en la sección superior. Si el número en la sección inferior es menor

**TABLA 3-6** Determinación de implicantes primos del Ejemplo 3-13 en notación decimal

| (a)         | (b)                 | (c)                   |
|-------------|---------------------|-----------------------|
| <u>0</u> ✓  | 0, 1 (1)            | 0, 2, 8, 10 (2, 8)    |
|             | 0, 2 (2) ✓          | 0, 2, 8, 10 (2, 8)    |
| <u>1</u> ✓  | <u>0, 8 (8)</u> ✓   |                       |
| <u>2</u> ✓  |                     | 10, 11, 14, 15 (1, 4) |
| <u>8</u> ✓  | 2, 10 (8) ✓         | 10, 11, 14, 15 (1, 4) |
|             | <u>8, 10 (2)</u> ✓  |                       |
| <u>10</u> ✓ |                     |                       |
|             | 10, 11 (1) ✓        |                       |
| <u>11</u> ✓ | <u>10, 14 (4)</u> ✓ |                       |
| <u>14</u> ✓ |                     |                       |
|             | 11, 15 (4) ✓        |                       |
| <u>15</u> ✓ | <u>14, 15 (1)</u> ✓ |                       |

que el de arriba, no se registra un par aun si los dos números difieren por una potencia de 2. La búsqueda exhaustiva en la columna (a) resulta en los términos de la columna (b), con todos los mintérminos en la columna (a) marcados. Hay sólo dos pares de términos en la columna (b). Cada uno da el mismo término de dos literales registrado en la columna (c). Los implicantes primos constan de todos los términos no marcados en la tabla. La conversión de la notación decimal en la binaria se muestra en la parte inferior de la tabla. Los implicantes primos se encuentran que son  $x'y'z$ ,  $w'xz'$ ,  $w'xy$ ,  $xyz$ ,  $wyz$  y  $wx'$ .

La suma de los implicantes primos da una expresión algebraica válida para la función. Sin embargo, esta expresión no necesariamente es la que tiene el número mínimo de términos. Esto puede demostrarse inspeccionando el mapa para la función del Ejemplo 3-14. Como se muestra en la Fig. 3-28, la función minimizada se reconoce que es:

$$F = x'y'z + w'xz' + xyz + wx'$$

la cual consta de la suma de cuatro de los seis implicantes primos que se derivaron en el Ejemplo 3-14. El procedimiento tabular para seleccionar los implicantes primos que da la función minimizada es tema de la siguiente sección.

### 3-11 SELECCION DE IMPLICANTES PRIMOS

La selección de implicantes primos que forman la función minimizada se realiza mediante una tabla de implicantes primos. En esta tabla, cada implicante primo se

TABLA 3-7 Determinación de implicantes primos para el Ejemplo 3-14

| (a)   |       |       | (b)    |       |  | (c)          |        |  |
|-------|-------|-------|--------|-------|--|--------------|--------|--|
| 0001  | 1     | ✓     | 1, 9   | (8)   |  | 8, 9, 10, 11 | (1, 2) |  |
| 0100  | 4     | ✓     | 4, 6   | (2)   |  | 8, 9, 10, 11 | (1, 2) |  |
| <hr/> | <hr/> | <hr/> | 8, 9   | (1) ✓ |  |              |        |  |
| 1000  | 8     | ✓     | 8, 10  | (2) ✓ |  |              |        |  |
| <hr/> | <hr/> | <hr/> | <hr/>  | <hr/> |  |              |        |  |
| 0110  | 6     | ✓     | 6, 7   | (1)   |  |              |        |  |
| 1001  | 9     | ✓     | 9, 11  | (2) ✓ |  |              |        |  |
| 1010  | 10    | ✓     | <hr/>  | <hr/> |  |              |        |  |
| <hr/> | <hr/> | <hr/> | 10, 11 | (1) ✓ |  |              |        |  |
| 0111  | 7     | ✓     | <hr/>  | <hr/> |  |              |        |  |
| 1011  | 11    | ✓     | 7, 15  | (8)   |  |              |        |  |
| <hr/> | <hr/> | <hr/> | <hr/>  | <hr/> |  |              |        |  |
| 1111  | 15    | ✓     | 11, 15 | (4)   |  |              |        |  |
| <hr/> | <hr/> | <hr/> | <hr/>  | <hr/> |  |              |        |  |

Implicantes primos

| Decimal             | Binario |   |   |   | Término |
|---------------------|---------|---|---|---|---------|
|                     | w       | x | y | z |         |
| 1, 9 (8)            | —       | 0 | 0 | 1 | $x'y'z$ |
| 4, 6 (2)            | 0       | 1 | — | 0 | $w'xz'$ |
| 6, 7 (1)            | 0       | 1 | 1 | — | $w'xy$  |
| 7, 15 (8)           | —       | 1 | 1 | 1 | $xyz$   |
| 11, 15 (4)          | 1       | — | 1 | 1 | $wyz$   |
| 8, 9, 10, 11 (1, 2) | 1       | 0 | — | — | $wx'$   |

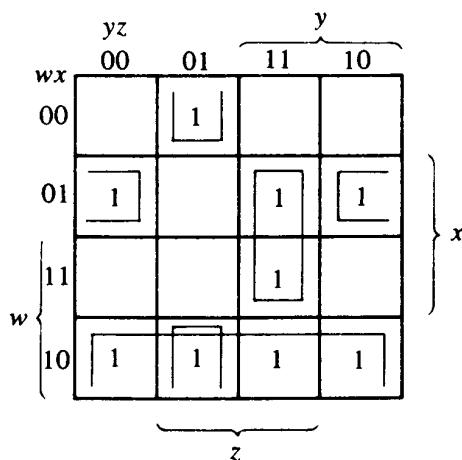


Figura 3-28 Mapa para la función del Ejemplo 3-14;  $F = x'y'z + w'xz' + xyz + wx'$ .

representa en un renglón y cada mintérmino en una columna. Se marcan cruces en cada renglón para mostrar la composición de mintérminos que hacen los implicantes primos. Un conjunto mínimo de implicantes primos se elige entonces para que cubra todos los mintérminos en la función. Este procedimiento se ilustra en el Ejemplo 3-15.

**EJEMPLO 3-15:** Minimice la función del Ejemplo 3-14. La tabla de implicantes primos para este ejemplo se muestra en la Tabla 3-8. Hay seis renglones, uno para cada implicante primo (derivado en el Ejemplo 3-14), y nueve columnas, cada una representando un mintérmino de la función. Se marcan cruces en cada renglón para indicar los mintérminos contenidos en el implicante primo de ese renglón. Por ejemplo, las dos cruces en el primer renglón indican que los mintérminos 1 y 9 están contenidos en el implicante primo  $x'y'z$ . Es aconsejable incluir la equivalente decimal del implicante primo en cada renglón, ya que es más conveniente dar los mintérminos contenidos en él. Después de marcar todas las cruces, proceda a seleccionar un número mínimo de implicantes primos.

La tabla completada de implicantes primos se inspecciona para buscar las columnas que contienen sólo una cruz. En este ejemplo, hay cuatro mintérminos cuyas columnas dan una sola cruz: 1, 4, 8 y 10. El mintérmino 1 está cubierto por el implicante primo  $x'y'z$ , esto es, la selección del implicante primo  $x'y'z$  garantiza que el mintérmino 1 esté incluido en la función. De manera semejante, el mintérmino 4 está cubierto por el implicante primo  $w'xz'$  y los mintérminos 8 y 10 por el  $wx'$ . Los implicantes primos que cubren mintérminos con una sola cruz en su columna se denominan *implicantes primos esenciales*. Para permitir que la expresión simplificada final contenga todos los mintérminos, no se tiene otra alternativa que incluir los implicantes primos esenciales. En la tabla se coloca una marca de verificación junto a los implicantes primos esenciales para indicar que se han seleccionado.

A continuación se verifica cada columna cuyo mintérmino está cubierto por los implicantes primos esenciales que se seleccionaron. Por ejemplo, el implicante primo seleccionado  $x'y'z$  cubre los mintérmi-

**TABLA 3-8** Tabla de implicantes primos para el Ejemplo 3-15

|              | 1            | 4 | 6 | 7 | 8 | 9 | 10 | 11 | 15 |
|--------------|--------------|---|---|---|---|---|----|----|----|
| $\vee x'y'z$ | 1, 9         | X |   |   |   | X |    |    |    |
| $\vee w'xz'$ | 4, 6         |   | X | X |   |   |    |    |    |
| $w'xy$       | 6, 7         |   |   | X | X |   |    |    |    |
| $xyz$        | 7, 15        |   |   |   | X |   |    |    | X  |
| $wyz$        | 11, 15       |   |   |   |   |   |    | X  | X  |
| $\vee wx'$   | 8, 9, 10, 11 |   |   |   | X | X | X  | X  |    |
|              | ✓            | ✓ | ✓ |   | ✓ | ✓ | ✓  | ✓  | ✓  |

nos 1 y 9. Se inserta una marca de verificación en la parte inferior de las columnas. En forma similar el implicante primo  $w'xz'$  cubre los mintérminos 4 y 6, y  $wx'$  cubre los mintérminos 8, 9, 10 y 11. La inspección de la tabla de primo-implicandos muestra que la selección de los implicantes primos esenciales cubre todos los mintérminos de la función, excepto 7 y 15. Estos dos mintérminos deben incluirse por la selección de uno o más primo-implicandos. En este ejemplo, es claro que el implicante primo  $xyz$  cubre ambos mintérminos y, por tanto, es el que se selecciona. Así se ha encontrado el conjunto mínimo de implicante primo cuya suma da la función minimizada requerida:

$$F = x'y'z + w'xz' + wx' + xyz$$

Todas las expresiones simplificadas que se derivan de los ejemplos anteriores están en la forma de suma de productos. El método de tabulación puede adaptarse para dar una expresión simplificada en producto de sumas. Como en el método de mapa, se principia con el complemento de la función tomando los 0 como la lista inicial de mintérminos. Esta lista contiene los mintérminos que no se incluyen en la función original, los cuales son numéricamente iguales a los maxterminos de la función. El proceso de tabulación se lleva a cabo con los 0 de la función y termina con una expresión simplificada en suma de productos del complemento de la función. Tomando de nuevo el complemento, se obtiene la expresión simplificada en producto de sumas.

Una función con condiciones no importa puede simplificarse por el método de tabulación después de una ligera modificación. Los términos no importa se incluyen en la lista de mintérminos cuando se determinan los implicantes primos. Esto permite la derivación de implicantes primos con el número más bajo de literales. Los términos no importa no se incluyen en la lista de mintérminos cuando la tabla de implicantes primos se establece, porque los términos no importa no tienen que cubrirse por los primo-implicandos seleccionados.

### 3-12 COMENTARIOS CONCLUYENTES

En este capítulo se introdujeron dos métodos de simplificación de función booleana. El criterio que se tomó para la simplificación fue la minimización del número de literales en las expresiones en suma de productos o producto de sumas. Tanto los métodos de mapa como de tabulación tienen capacidades restringidas, ya que son útiles para simplificar solamente funciones booleanas que se expresen en las formas estándar. Aunque es una desventaja de los métodos, no es muy crítica. La mayoría de las aplicaciones requieren las formas estándar sobre cualquier otra forma. Mediante la Fig. 3-15 se vio que el implante en compuerta de las expresiones en formas estándar consta de no más de dos niveles de compuertas. Las expresiones que no están en la forma estándar se implementan con más de dos niveles. Humphrey (5) muestra una extensión del método de mapa que produce expresiones simplificadas de nivel múltiple.

Debe reconocerse que la secuencia de código reflejado elegida para los mapas no es única. Es posible dibujar un mapa y asignar una secuencia de código reflejado binario a los renglones y columnas diferentes de la secuencia que aquí se emplea. En tanto que la secuencia binaria elegida produzca un cambio sólo de un bit entre cuadros adyacentes, producirá un mapa válido y útil.

En la Fig. 3-29 se muestran dos versiones alternas de mapas de tres variables que con frecuencia se encuentran en la literatura referente a la lógica digital. Los números mintérminos se escriben en cada cuadro para referencia. En (a), la asignación de variables a los renglones y columnas es diferente a la que se utiliza en este libro. En (b), el mapa se ha girado a una posición vertical. La asignación del número mintérmino en todos los mapas permanece en el orden  $xyz$ . Por ejemplo, el cuadro para el mintérmino 6 se encuentra por la asignación de las variables ordenadas del número binario  $xyz = 110$ . El cuadro de este mintérmino se encuentra en (a) la columna marcada  $xy = 11$  y el renglón con  $z = 0$ . El cuadro correspondiente en (b) pertenece a la columna marcada con  $x = 1$  y el renglón con  $yz = 10$ . El procedimiento de simplificación con estos mapas es exactamente el mismo que se describe en este capítulo, excepto, por supuesto, la variación en la asignación en mintérmino y variable.

En la Fig. 3-30 se muestran otras dos versiones del mapa de cuatro variables. El mapa en (a) tiene gran preferencia y con mucha frecuencia se utiliza en la literatura. De nuevo, aquí la diferencia es ligera y se manifiesta por un simple intercambio de asignación de variables de renglones a columnas y viceversa. El mapa en (b) es el diagrama original de Veitch (1) el cual modificó Karnaugh (2) a la forma que se muestra en (a). De nuevo, los procedimientos de simplificación no cambian cuando estos mapas se usan en lugar del que se emplea en este libro. También hay variaciones de los mapas de cinco y seis variables. En cualquier caso, cualquier mapa de diferente apariencia respecto al que se utiliza en este libro, o que reciba un nombre distinto, se reconoce en forma simple como una variación de la asignación de mintérminos a los cuadros en el mapa.

Como es evidente a partir de los Ejemplos 3-13 y 3-14, el método de tabulación tiene la desventaja de la ocurrencia inevitable de errores que se dan al tratar de comparar números en listas largas. El método de mapa parece ser preferible pero, para

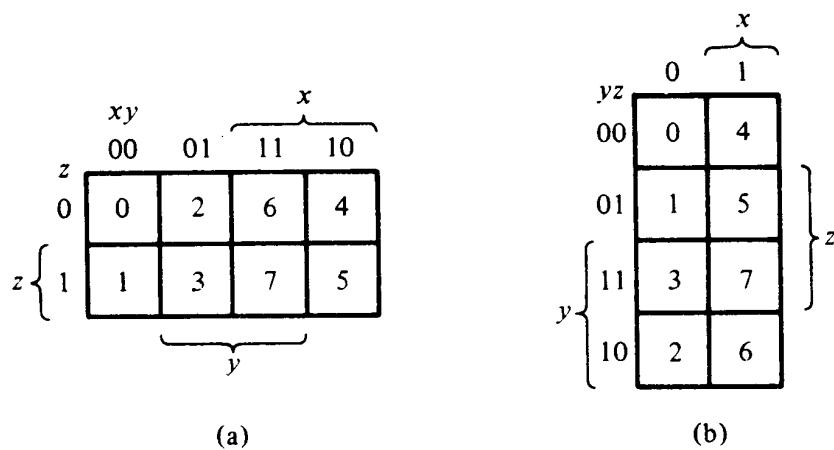
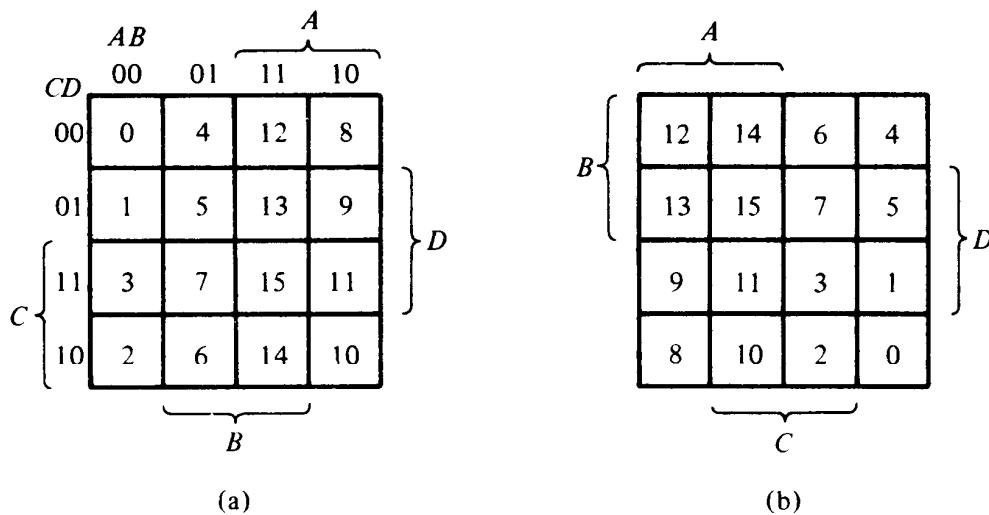


Figura 3-29 Variaciones del mapa de tres variables.



**Figura 3-30** Variaciones del mapa de cuatro variables.

más de cinco variables, no puede tenerse la certeza de que se ha encontrado la mejor expresión simplificada. La ventaja real del método de tabulación se basa en el hecho de que consta de procedimientos específicos paso a paso que garantizan una respuesta. Además, este procedimiento formal es adecuado para mecanización por computadora.

En la Sección 3-9 se estableció que el método de tabulación siempre principia con la lista de mintérminos de la función. Si la función no está en esta forma, debe convertirse. En la mayoría de las aplicaciones, la función que va a simplificarse proviene de una tabla de verdad y ya se dispone de la lista de mintérminos. De otra manera, la conversión en mintérminos agrega considerable trabajo de manipulación al problema. Sin embargo, existe una extensión del método de tabulación para encontrar los implicantes primos de expresiones arbitrarias de suma de productos. Véase, por ejemplo, McCluskey (7).

En este capítulo, se han considerado las simplificaciones de funciones con muchas variables de entrada y una sola variable de salida. Sin embargo, algunos circuitos digitales tienen más de una salida. Tales circuitos se describen por un conjunto de funciones booleanas, una para cada variable de salida. Un circuito con salidas múltiples algunas veces puede tener términos comunes entre las diversas funciones que pueden utilizarse para formar compuertas comunes durante la implementación. Esto resulta en una simplificación adicional que no se toma en consideración cuando cada variable se simplifica por separado. Existe una extensión del método de tabulación para circuitos de salidas múltiples (6, 7). Sin embargo, este método es demasiado especializado y muy tedioso para manipulación humana. Es de importancia práctica sólo si el usuario dispone de un programa de computadora basado en este método.

## BIBLIOGRAFIA

1. Veitch, E. W., "A Chart Method for Simplifying Truth Functions." *Proc. of the ACM* (Mayo 1952), 127-33.

2. Karnaugh, M., "A Map Method for Synthesis of Combinational Logic Circuits." *Trans. AIEE, Comm. and Electronics*, Vol. 72, Parte I (Noviembre 1953), 593-99.
3. Quine, W.V., "The Problem of Simplifying Truth Functions." *Am. Math. Monthly*, Vol. 59, No. 8 (Octubre 1952), 521-31.
4. McCluskey, E.J., Jr., "Minimization of Boolean Functions." *Bell System Tech. J.*, Vol. 35, No. 6 (Noviembre 1956), 1417-44.
5. Humphrey, W. S., Jr., *Switching Circuits with Computer Applications*. New York: McGraw-Hill Book Co., 1958, Cap. 4.
6. Hill, F. J., and G.R. Peterson, *Introduction to Switching Theory and Logical Design*, 3a. ed. New York: John Wiley & Sons, Inc., 1981.
7. McCluskey, E. J., Jr., *Introduction to the Theory of Switching Circuits*. New York: McGraw-Hill Book Co., 1965, Cap. 4.
8. Kohavi, Z., *Switching and Finite Automata Theory*. New York: McGraw-Hill Book Co., 1970.
9. Nagle, H.T. Jr., B.D. Carroll, and J.D. Irwin, *An Introduction to Computer Logic*. Englewood Cliffs, N.J.: Prentice-Hall, Inc., 1975.

### PROBLEMAS

- ✓ 3-1. Obtenga las expresiones simplificadas en suma de productos para las siguientes funciones booleanas:
- (a)  $F(x, y, z) = \Sigma(2, 3, 6, 7)$
  - (b)  $F(A, B, C, D) = \Sigma(7, 13, 14, 15)$
  - (c)  $F(A, B, C, D) = \Sigma(4, 6, 7, 15)$
  - (d)  $F(w, x, y, z) = \Sigma(2, 3, 12, 13, 14, 15)$
- 3-2. Obtenga las expresiones simplificadas en suma de productos para las siguientes funciones booleanas:
- (a)  $xy + x'y'z' + x'yz'$
  - (b)  $A'B + BC' + B'C'$
  - (c)  $a'b' + bc + a'bc'$
  - (d)  $xy'z + xyz' + x'yz + xyz$
- 3-3. Obtenga las expresiones simplificadas en suma de productos para las siguientes expresiones booleanas:
- (a)  $D(A' + B) + B'(C + AD)$
  - (b)  $ABD + A'C'D' + A'B + A'CD' + AB'D'$
  - (c)  $k'l'm' + k'm'n + klm'n' + lmn'$
  - (d)  $A'B'C'D' + AC'D' + B'CD' + A'BCD + BC'D$
  - (e)  $x'z + w'xy' + w(x'y + xy')$
- 3-4. Obtenga las expresiones simplificadas en suma de productos para las siguientes funciones booleanas:
- (a)  $F(A, B, C, D, E) = \Sigma(0, 1, 4, 5, 16, 17, 21, 25, 29)$
  - (b)  $BDE + B'C'D + CDE + A'B'CE + A'B'C + B'C'D'E'$
  - (c)  $A'B'CE' + A'B'C'D' + B'D'E' + B'CD' + CDE' + BDE'$

- \* 3-5. Dada la siguiente tabla de verdad:

| $x$ | $y$ | $z$ | $F_1$ | $F_2$ |
|-----|-----|-----|-------|-------|
| 0   | 0   | 0   | 0     | 0     |
| 0   | 0   | 1   | 1     | 0     |
| 0   | 1   | 0   | 1     | 0     |
| 0   | 1   | 1   | 0     | 1     |
| 1   | 0   | 0   | 1     | 0     |
| 1   | 0   | 1   | 0     | 1     |
| 1   | 1   | 0   | 0     | 1     |
| 1   | 1   | 1   | 1     | 1     |

- (a) Exprese  $F_1$  y  $F_2$  en producto de maxterminos.  
 (b) Obtenga las funciones simplificadas en suma de productos.  
 (c) Obtenga las funciones simplificadas en producto de sumas.

- \* 3-6. Obtenga las expresiones simplificadas en producto de sumas:

- (a)  $F(x, y, z) = \Pi(0, 1, 4, 5)$   
 (b)  $F(A, B, C, D) = \Pi(0, 1, 2, 3, 4, 10, 11)$   
 (c)  $F(w, x, y, z) = \Pi(1, 3, 5, 7, 13, 15)$

- 3-7. Obtenga las expresiones simplificadas en (1) suma de productos y (2) producto de sumas:

- (a)  $x'z' + y'z' + yz' + xyz$   
 (b)  $(A + B' + D)(A' + B + D)(C + D)(C' + D')$   
 (c)  $(A' + B' + D')(A + B' + C')(A' + B + D')(B + C' + D')$   
 (d)  $(A' + B' + D)(A' + D')(A + B + D')(A + B' + C + D)$   
 (e)  $w'yz' + vw'z' + vw'x + v'wz + v'w'y'z'$

- 3-8. Dibuje la implementación de compuertas para las funciones booleanas que se obtuvieron en el problema 3-7 utilizando compuertas AND y OR.

- \* 3-9. Simplifique cada una de las siguientes funciones e implántelas con compuertas NAND. Dé dos alternativas.

- (a)  $F_1 = AC' + ACE + ACE' + A'CD' + A'D'E'$   
 (b)  $F_2 = (B' + D')(A' + C' + D)(A + B' + C' + D)(A' + B + C' + D')$

- \* 3-10. Repita el problema 3-9 para implementaciones NOR.

- 3-11. Implemente las siguientes funciones con compuertas NAND. Suponga que están disponibles entradas tanto normal como complementaria.

- (a)  $BD + BCD + AB'C'D' + A'B'CD'$  con no más de seis compuertas y cada una con tres entradas.  
 (b)  $(AB + A'B')(CD' + C'D)$  con compuertas de dos entradas.

- 3-12. Implemente las siguientes funciones con compuertas NOR. Suponga que están disponibles entradas tanto normal como complementaria.

- (a)  $AB' + C'D' + A'CD' + DC'(AB + A'B') + DB(AC' + A'C)$   
 (b)  $AB'CD' + A'BCD' + AB'C'D + A'BC'D$

- 3-13. Liste las ocho formas degeneradas de dos niveles y muestre que se reducen a una sola operación. Explique cómo las formas degeneradas de dos niveles pueden usarse para extender el abanico de entrada de compuertas.
- 3-14. Implemente las funciones del problema 3-9 con las siguientes formas de dos niveles: NOR-OR, NAND-AND, OR-NAND y AND-NOR.
- 3-15. Simplifique la función booleana  $F$  en suma de productos usando las condiciones no importa  $d$ :
- $F = y' + x'z'$   
 $d = yz + xy$
  - $F = B'C'D' + BCD' + ABC'D$   
 $d = B'CD' + A'BC'D'$
- 3-16. Simplifique la función booleana  $F$  utilizando las condiciones no importa  $d$ , en (1) suma de productos y (2) producto de sumas:
- $F = A'B'D' + A'CD + A'BC$   
 $d = A'BC'D + ACD + AB'D'$
  - $F = w'(x'y + x'y' + xyz) + x'z'(y + w)$   
 $d = w'x(y'z + yz') + wyz$
  - $F = ACE + A'CD'E' + A'C'DE$   
 $d = DE' + A'D'E + AD'E'$
  - $F = B'DE' + A'BE + B'C'E' + A'BC'D'$   
 $d = BDE' + CD'E'$
- 3-17. Implemente las siguientes funciones usando las condiciones no importa. Suponga que están disponibles las entradas tanto normal como complementaria.
- $F = A'B'C' + AB'D + A'B'CD'$  con no más de dos compuertas NOR.  
 $d = ABC + AB'D'$
  - $F = (A + D)(A' + B)(A' + C')$  con no más de tres compuertas NAND.
  - $F = B'D + B'C + ABCD$  con compuertas NAND.  
 $d = A'BD + AB'C'D'$
- 3-18. Implemente la siguiente función ya sea con compuertas NAND o NOR. Use sólo cuatro compuertas. Sólo están disponibles las entradas normales.

$$\begin{aligned} F &= w'xz + w'yz + x'yz' + wxy'z \\ d &= wyz \end{aligned}$$

- 3-19. La siguiente expresión booleana:

$$BE + B'DE'$$

es una versión simplificada de la expresión:

$$A'BE + BCDE + BC'D'E + A'B'DE' + B'C'DE'$$

¿Aquí hay condiciones no importa? Si es así, ¿cuáles son?

- 3-20. Dé tres formas posibles de expresar la función:

$$F = A'B'D' + AB'CD' + A'BD + ABC'D$$

con ocho o menos literales.

- $\times$  3-21. Con el uso de mapas, encuentre la forma más simple en suma de productos de la función  $F = fg$ , en donde  $f$  y  $g$  están dadas por:

$$\begin{aligned}f &= wxy' + y'z + w'yz' + x'yz' \\g &= (w + x + y' + z')(x' + y' + z)(w' + y + z')\end{aligned}$$

*Sugerencia:* Vea el problema 2-8(b).

- 3-22. Simplifique la función booleana del problema 3-2(a) utilizando el mapa que se define en la Fig. 3-29(a). Repita esto con el mapa en la Fig. 3-30(b).
- 3-23. Simplifique la función booleana en el problema 3-3(a) usando el mapa que se define en la Fig. 3-30(a). Repita esto con el mapa en la Fig. 3-30(b).
- $\checkmark$  3-24. Simplifique las funciones booleanas siguientes mediante el método de tabulación.  
 (a)  $F(A, B, C, D, E, F, G) = \Sigma(20, 28, 52, 60)$   
 (b)  $F(A, B, C, D, E, F, G) = \Sigma(20, 28, 38, 39, 52, 60, 102, 103, 127)$   
 (c)  $F(A, B, C, D, E, F) = \Sigma(6, 9, 13, 18, 19, 25, 27, 29, 41, 45, 57, 61)$
- 3-25. Repita el problema 3-6 utilizando el método de tabulación.
- 3-26. Repita el problema 3-16(c) y (d) usando el método de tabulación.

---

---

# Lógica combinacional

# 4

---

---

## 4-1 INTRODUCCION

Los circuitos lógicos para sistemas digitales pueden ser combinacionales o secuenciales. Un circuito combinacional consta de compuertas lógicas cuyas salidas en cualquier momento están determinadas en forma directa por la combinación presente de las entradas sin tomar en cuenta las entradas previas. Un circuito combinacional realiza una operación específica de procesamiento de información, especificada por completo en forma lógica por un conjunto de funciones booleanas. Los circuitos secuenciales emplean elementos de memoria (celdas binarias) además de las compuertas lógicas. Sus salidas son una función de las entradas y el estado de los elementos de memoria. El estado de los elementos de memoria, a su vez, es una función de las entradas previas. Como consecuencia, las salidas de un circuito secuencial dependen no sólo de las entradas presentes, sino también de las entradas del pasado y, el comportamiento del circuito debe especificarse en una secuencia de tiempo de entradas y de estados internos. En el Capítulo 6 se exponen los circuitos secuenciales.

En el Capítulo 1 se aprendió a reconocer los números binarios y los códigos binarios que representan cantidades discretas de información. Estas variables binarias se representan por voltajes eléctricos o alguna otra señal. Las señales pueden manipularse en las compuertas lógicas digitales para realizar las funciones requeridas. En el Capítulo 2 se introdujo el álgebra booleana como una forma para expresar de manera algebraica las funciones lógicas. En el Capítulo 3 se aprendió cómo simplificar las funciones booleanas para lograr la implementación económica de compuertas. El objetivo de este capítulo es usar el conocimiento adquirido en los capítulos previos y formular varios procedimientos sistemáticos de diseño y análisis de los circuitos combinacionales. La solución de algunos ejemplos típicos proporcionará un catálogo útil de funciones elementales importantes para el entendimiento de las computadoras y sistemas digitales.

Un circuito combinacional consta de variables de entrada, compuertas lógicas y variables de salida. Las compuertas lógicas aceptan las señales de las entradas y generan señales a las salidas. Este proceso transforma la información binaria de los datos dados de entrada en los datos requeridos de salida. En forma obvia, tanto los datos de entrada y