

Métodos algebraicos para el análisis y síntesis de circuitos lógicos

2.1 Fundamentos del álgebra booleana

Las herramientas de análisis y síntesis presentadas en este capítulo se basan en los conceptos fundamentales del álgebra booleana y, por tanto, analizaremos ahora este tema. En 1849, George Boole presentó una formulación algebraica de los procesos del pensamiento y el razonamiento lógico [1]. Esta formulación se conoce como álgebra booleana, que resumiremos a continuación.

2.1.1 Postulados básicos

La descripción básica de la formulación del álgebra booleana se basa en conceptos de la teoría de conjuntos, donde se define formalmente un *álgebra booleana* como un conjunto matemático distributivo y complementado [2]. Resumiremos aquí esta definición mediante un conjunto de postulados que sintetiza los elementos y propiedades básicos de un álgebra booleana.

Postulado 1. Definición Un *álgebra booleana* es un sistema algebraico cerrado formado por un conjunto K de dos o más elementos y los dos operadores \cdot y $+$; de manera alternativa, para cada a y b un conjunto K , $a \cdot b$ pertenece a K y $a + b$ pertenece a K ($+$ se llama OR y \cdot se llama AND).

Postulado 2. Existencia de los elementos 1 y 0 En el conjunto K existen los elementos 1 (uno) y 0 (cero), únicos, tales que para toda a en K

(a) $a + 0 = a$,

(b) $a \cdot 1 = a$,

donde 0 es el elemento neutro para la operación $+$ y 1 es el elemento neutro para la operación \cdot .

Postulado 3. Comutatividad de las operaciones $+$ y \cdot Para toda a y b en K

(a) $a + b = b + a$,

(b) $a \cdot b = b \cdot a$.

Postulado 4. Asociatividad de las operaciones + y • Para toda a , b y c en K

- (a) $a + (b + c) = (a + b) + c,$
- (b) $a \cdot (b \cdot c) = (a \cdot b) \cdot c.$

Postulado 5. Distributividad de + sobre • y de • sobre + Para toda a , b y c en K

- (a) $a + (b \cdot c) = (a + b) \cdot (a + c),$
- (b) $a \cdot (b + c) = (a \cdot b) + (a \cdot c).$

Postulado 6. Existencia del complemento Para toda a en K existe un único elemento llamado \bar{a} (*complemento* de a) en K tal que

- (a) $a + \bar{a} = 1,$
- (b) $a \cdot \bar{a} = 0.$

Con este conjunto de premisas, podemos desarrollar otras relaciones útiles que llamaremos teoremas. Para simplificar la notación en el resto del texto, suprimiremos el punto (*) al indicar la operación •.

EJEMPLO 2.1

$$a + b \cdot c = (a + b) \cdot (a + c)$$

$$a + bc = (a + b)(a + c)$$

Antes de desarrollar los teoremas, examinemos los postulados con detalle para entender exactamente lo que significan.

2.1.2 Diagramas de Venn para los postulados [2]

Podemos representar los postulados de manera gráfica en forma de diagramas de Venn. Esta descripción gráfica es posible, ya que el álgebra de conjuntos es un álgebra booleana en la que los conjuntos son los elementos del álgebra, la operación de intersección corresponde a • y la operación de unión corresponde a +. En el diagrama de Venn, los conjuntos se muestran como contornos cerrados, es decir, círculos, cuadrados, elipses, etc. Los diagramas de Venn para los conjuntos a , b , $a \cdot b$ y $a + b$ aparecen en la figura 2.1. Otras notaciones frecuentes para $a + b$ es $a \vee b$ o $a \cup b$ y para $a \cdot b$, ab , $a \wedge b$ o $a \cap b$.

Podemos utilizar los diagramas de Venn para ilustrar los postulados. Elegimos como ejemplo el postulado 5.

EJEMPLO 2.2

Utilizar el diagrama de Venn para ilustrar el postulado 5.

Del análisis de la figura 2.2, es evidente que el conjunto $a + bc$ y el conjunto $(a + b)(a + c)$ son dos representaciones de la misma área sombreada y, por tanto, $a + bc$ es igual a $(a + b)(a + c)$.

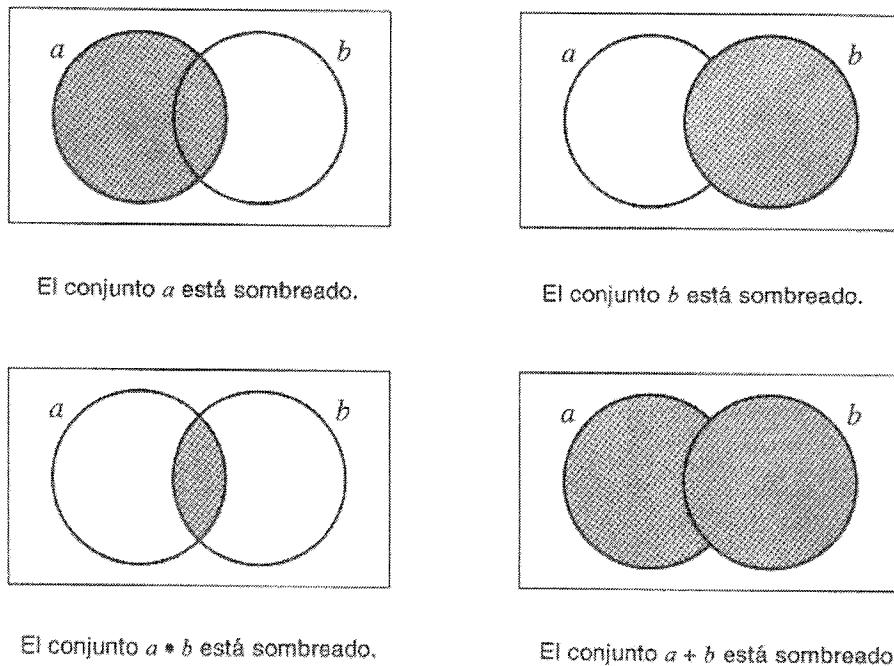


Figura 2.1 Ejemplos de diagramas de Venn.

Es interesante examinar algunas facetas del postulado 6. Este postulado se refiere al complemento de a . Si a es el conjunto sombreado de la figura 2.3, el complemento de a , \bar{a} , es el área fuera de a en el conjunto universal. En otras palabras, a y \bar{a} son mutuamente excluyentes y se encuentran dentro del conjunto universal. Como son mutuamente excluyentes, no tienen un área común y, por tanto, su intersección es el conjunto vacío: $a * \bar{a} = 0$. La unión de a y \bar{a} es por definición el conjunto universal: $a + \bar{a} = 1$.

Además, como el conjunto universal 1 contiene a todos los demás conjuntos, su complemento debe ser el conjunto vacío, 0. Por tanto, $\bar{1} = 0$ y $\bar{0} = 1$.

El diagrama de Venn es una herramienta muy útil no sólo para visualizar los postulados ya presentados, sino también los teoremas importantes del álgebra booleana que describimos a continuación.

2.1.3 Dualidad

El principio de *dualidad* es un concepto muy importante en el álgebra booleana. En pocas palabras, el principio de dualidad establece que, si una expresión es válida en el álgebra booleana, entonces su expresión dual también es válida. Determinaremos la expresión dual remplazando todos los operadores + por *, todos los operadores * por +, todos los unos por ceros y todos los ceros por unos.

EJEMPLO 2.3

Determinar la expresión dual de

$$a + (bc) = (a + b)(a + c)$$

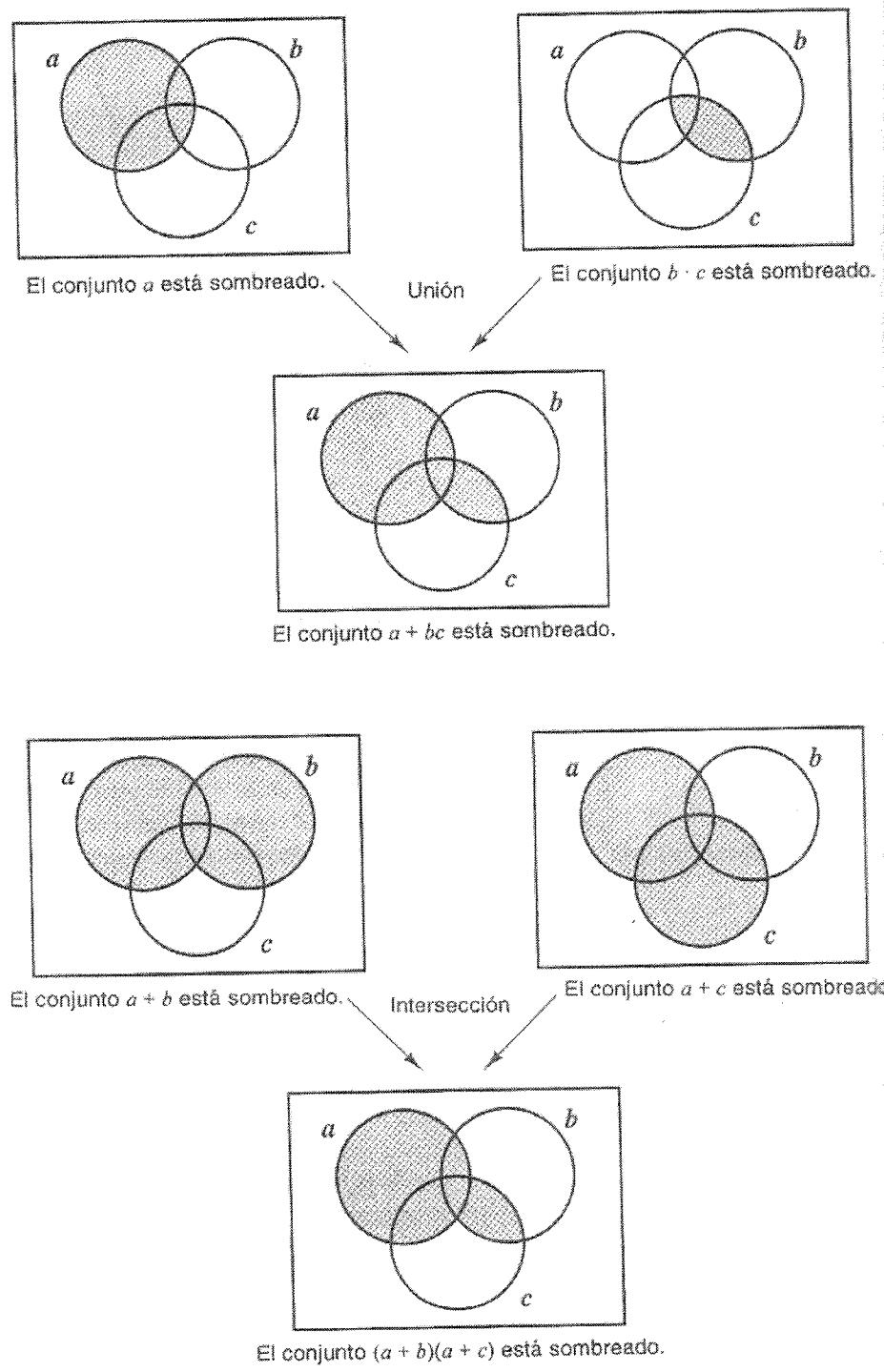


Figura 2.2 Diagramas de Venn para el postulado 5.

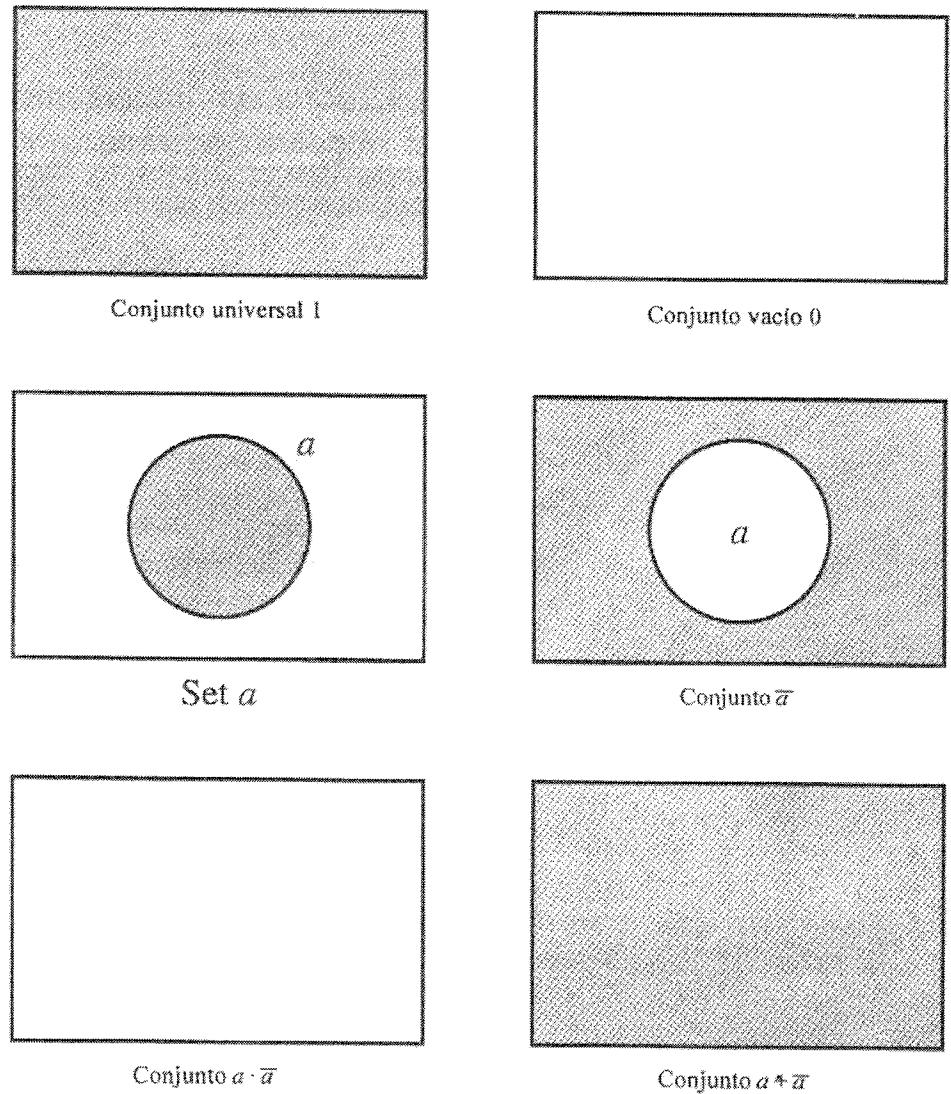


Figura 2.3 Diagramas de Venn que ilustran el postulado 6.

Al cambiar todos los operadores $+$ por \cdot y viceversa, obtenemos la expresión dual

$$a(b + c) = ab + ac$$

Al obtener el dual, no debemos alterar la posición de los paréntesis existentes. Observe que las dos expresiones del último ejemplo son las partes (a) y (b) del postulado 5. De hecho, presentamos los postulados 2 a 6 como expresiones duales.

Utilizaremos ampliamente el principio de dualidad al demostrar los teoremas del álgebra booleana. De hecho, una vez que hayamos utilizado los

postulados y los teoremas demostrados anteriormente para demostrar la validez de una expresión, podremos usar la dualidad para demostrar la validez de la expresión dual.

2.1.4 Teoremas fundamentales del álgebra booleana

Ahora enunciaremos algunos teoremas útiles del álgebra booleana. En estos teoremas, las letras a, b, c, \dots representan los elementos de un álgebra booleana. El primer teorema describe la idempotencia y se expresa como sigue.

Teorema 1. Idempotencia

$$(a) a + a = a.$$

$$(b) a \cdot a = a.$$

Demostración. Podemos demostrar la parte (a) o (b) de este teorema. Supongamos que queremos demostrar la parte (a):

$$\begin{aligned} a + a &= (a + a)1 && [P2(b)] \\ &= (a + a)(a + \bar{a}) && [P6(a)] \\ &\equiv a + a\bar{a} && [P5(a)] \\ &\equiv a + 0 && [P6(b)] \\ &\equiv a && [P2(a)] \end{aligned}$$

Indicamos a la derecha los postulados que justifican un paso en particular. Hay que recordar que podemos utilizar de manera indistinta las expresiones de lados opuestos de una igualdad; por ejemplo, el teorema 1 dice que podemos intercambiar $\{a\}$ por $\{a \cdot a\}$ y viceversa.

El siguiente teorema enfatiza las propiedades de los elementos únicos 1 y 0.

Teorema 2. Elementos neutros para los operadores + y \cdot

$$(a) a + 1 = 1.$$

$$(b) a \cdot 0 = 0.$$

Demostración. De nuevo, demostraremos la parte (a) del teorema.

$$\begin{aligned} a + 1 &= (a + 1)1 && [P2(b)] \\ &= 1 \cdot (a + 1) && [P3(b)] \\ &= (a + \bar{a})(a + 1) && [P6(a)] \\ &\equiv a + \bar{a} \cdot 1 && [P5(a)] \\ &\equiv a + \bar{a} && [P2(b)] \\ &\equiv 1 && [P6(a)] \end{aligned}$$

Como la parte (a) de este teorema es válida, el principio de dualidad implica que la parte (b) también es válida.

Teorema 3. Involución

$$\overline{\overline{a}} = a.$$

Demostración. Por el postulado 5, $a \cdot \bar{a} = 0$ y $a + \bar{a} = 1$. Por tanto, \bar{a} es el complemento de a , y también a es el complemento de \bar{a} . Como el complemento de \bar{a} es único, se sigue que $\bar{\bar{a}} = a$.

Ahora utilizaremos el material anterior para resumir todas las propiedades de los elementos únicos 1 y 0 en la tabla 2.1. Las propiedades \cdot (AND) de 1 y 0 nos recuerdan las propiedades fundamentales de la multiplicación en las matemáticas ordinarias; sin embargo, las propiedades $+$ (OR) hacen ver de inmediato que no estamos tratando con las matemáticas que hemos estudiado con anterioridad, y no podemos suponer ninguna de las propiedades matemáticas usuales para su uso en el álgebra booleana. Sólo debemos utilizar los postulados y teoremas que estamos desarrollando, pues trabajamos en un sistema completamente nuevo y diferente.

En el siguiente teorema establecemos la propiedad de absorción del álgebra booleana. La absorción no tiene contraparte en el álgebra “ordinaria”.

Teorema 4. Absorción

- (a) $a + ab = a$.
- (b) $a(a + b) = a$.

Demostración. Demostraremos la parte (a).

$$\begin{aligned} a + ab &= a \cdot 1 + ab && [\text{P2(b)}] \\ &= a(1 + b) && [\text{P5(b)}] \\ &= a(b + 1) && [\text{P3(b)}] \\ &= a \cdot 1 && [\text{T2(a)}] \\ &= a && [\text{P2(b)}] \end{aligned}$$

Podemos visualizar fácilmente el teorema 4 con un diagrama de Venn. Los siguientes ejemplos ilustran el uso de este teorema.

EJEMPLO 2.4

$$(X + Y) + (X + Y)Z = X + Y \quad [\text{T4(a)}]$$

EJEMPLO 2.5

$$A\bar{B}(A\bar{B} + \bar{B}C) = A\bar{B} \quad [\text{T4(b)}]$$

EJEMPLO 2.6

$$A\bar{B}C + \bar{B} = \bar{B} \quad [\text{T4(a)}]$$

TABLA 2.1 PROPIEDADES DE LOS ELEMENTOS 0 Y 1

OR	AND	COMPLEMENTO
$a + 0 = a$	$a \cdot 0 = 0$	$\bar{0} = 1$
$a + 1 = 1$	$a \cdot 1 = a$	$\bar{1} = 0$

Los siguientes tres teoremas son similares a la absorción en el sentido de que pueden servir para eliminar elementos adicionales de una expresión booleana.

Teorema 5.

- (a) $a + \bar{a}b = a + b$.
- (b) $a(\bar{a} + b) = ab$.

Demostración. La parte (a) del teorema se demuestra como sigue:

$$\begin{aligned} a + \bar{a}b &= (a + \bar{a})(a + b) & [\text{P5(a)}] \\ &= 1 \cdot (a + b) & [\text{P6(a)}] \\ &= (a + b) \cdot 1 & [\text{P3(b)}] \\ &= (a + b) & [\text{P2(b)}] \end{aligned}$$

Los siguientes ejemplos ilustran el uso del teorema 5 para simplificar las expresiones booleanas.

EJEMPLO 2.7

$$B + A\bar{B}\bar{C}D = B + A\bar{C}D \quad [\text{T5(a)}]$$

EJEMPLO 2.8

$$\bar{Y}(X + Y + Z) = \bar{Y}(X + Z) \quad [\text{T5(b)}]$$

EJEMPLO 2.9

$$(X + Y)(\overline{(X + Y)} + Z) = (X + Y)Z \quad [\text{T5(b)}]$$

EJEMPLO 2.10

$$AB + \overline{(AB)}C\bar{D} = AB + C\bar{D} \quad [\text{T5(a)}]$$

Teorema 6.

- (a) $ab + a\bar{b} = a$.
- (b) $(a + b)(a + \bar{b}) = a$.

Demostración. La parte (a) del teorema se demuestra como sigue:

$$\begin{aligned} ab + a\bar{b} &= a(b + \bar{b}) & [\text{P5(b)}] \\ &= a \cdot 1 & [\text{P6(a)}] \\ &= a & [\text{P2(b)}] \end{aligned}$$

Los siguientes ejemplos ilustran el uso del teorema 6 para simplificar expresiones booleanas.

EJEMPLO 2.11

$$ABC + A\bar{B}C = AC \quad [\text{T6(a)}]$$

EJEMPLO 2.12

$$(AD + B + C)(AD + \overline{(B + C)}) = AD \quad [\text{T6(b)}]$$

EJEMPLO 2.13

Simplificar $(\bar{W} + \bar{X} + \bar{Y} + \bar{Z})(\bar{W} + \bar{X} + \bar{Y} + Z)$
 $(\bar{W} + \bar{X} + Y + \bar{Z})(\bar{W} + \bar{X} + Y + Z).$

$$\begin{aligned} &= (\bar{W} + \bar{X} + \bar{Y})(\bar{W} + \bar{X} + Y + \bar{Z})(\bar{W} + \bar{X} + Y + Z) && [\text{T6(b)}] \\ &= (\bar{W} + \bar{X} + \bar{Y})(\bar{W} + \bar{X} + Y) && [\text{T6(b)}] \\ &= (\bar{W} + \bar{X}) && [\text{T6(b)}] \end{aligned}$$

Teorema 7.

- (a) $ab + a\bar{b}c = ab + ac.$
 (b) $(a + b)(a + \bar{b} + c) = (a + b)(a + c).$

Demostración. La parte (a) del teorema se demuestra como sigue:

$$\begin{aligned} ab + a\bar{b}c &= a(b + \bar{b}c) && [\text{P5(b)}] \\ &= a(b + c) && [\text{T5(a)}] \\ &= ab + ac && [\text{P5(b)}] \end{aligned}$$

Los siguientes ejemplos ilustran el uso del teorema 7 para simplificar las expresiones booleanas.

EJEMPLO 2.14

$$xy + x\bar{y}(\bar{w} + \bar{z}) = xy + x(\bar{w} + \bar{z}) \quad [\text{T7(a)}]$$

EJEMPLO 2.15

$$(\bar{x}\bar{y} + z)(w + \bar{x}\bar{y} + \bar{z}) = (\bar{x}\bar{y} + z)(w + \bar{x}\bar{y}) \quad [\text{T7(b)}]$$

EJEMPLO 2.16

$$\begin{aligned} (\bar{A} + \bar{B} + \bar{C})(\bar{B} + C)(A + \bar{B}) &= (\bar{A} + \bar{B})(\bar{B} + C)(A + \bar{B}) && [\text{T7(b)}] \\ &= \bar{B}(\bar{B} + C) && [\text{T6(b)}] \\ &= \bar{B} && [\text{T4(b)}] \end{aligned}$$

EJEMPLO 2.17

$$\begin{aligned} w\bar{y} + w\bar{x}y + wxyz + wx\bar{z} &= w\bar{y} + w\bar{x}y + wxy + wx\bar{z} && [\text{T7(a)}] \\ &= w\bar{y} + wy + ux\bar{z} && [\text{T6(a)}] \\ &= w + ux\bar{z} && [\text{T6(a)}] \\ &= w && [\text{T4(a)}] \end{aligned}$$

En los siguientes capítulos veremos que estos teoremas forman la base de algunos métodos estándar y automatizados por computadora para simplificar expresiones booleanas.

Al trabajar con álgebra booleana, con frecuencia necesitamos determinar el complemento de una expresión booleana. El siguiente teorema proporciona la base para esta operación.

Los siguientes tres teoremas son similares a la absorción en el sentido de que pueden servir para eliminar elementos adicionales de una expresión booleana.

Teorema 5.

- (a) $a + \bar{a}b = a + b$,
- (b) $a(\bar{a} + b) = ab$.

Demostración. La parte (a) del teorema se demuestra como sigue:

$$\begin{aligned} a + \bar{a}b &= (a + \bar{a})(a + b) && [\text{P5(a)}] \\ &= 1 \cdot (a + b) && [\text{P6(a)}] \\ &= (a + b) \cdot 1 && [\text{P3(b)}] \\ &= (a + b) && [\text{P2(b)}] \end{aligned}$$

Los siguientes ejemplos ilustran el uso del teorema 5 para simplificar las expresiones booleanas.

EJEMPLO 2.7

$$B + A\bar{B}\bar{C}D = B + A\bar{C}D \quad [\text{T5(a)}]$$

EJEMPLO 2.8

$$\bar{Y}(X + Y + Z) = \bar{Y}(X + Z) \quad [\text{T5(b)}]$$

EJEMPLO 2.9

$$(X + Y)(\overline{(X + Y)} + Z) = (X + Y)Z \quad [\text{T5(b)}]$$

EJEMPLO 2.10

$$AB + \overline{(AB)}C\bar{D} = AB + C\bar{D} \quad [\text{T5(a)}]$$

Teorema 6.

- (a) $ab + \bar{a}\bar{b} = a$,
- (b) $(a + b)(a + \bar{b}) = a$.

Demostración. La parte (a) del teorema se demuestra como sigue:

$$\begin{aligned} ab + \bar{a}\bar{b} &= a(b + \bar{b}) && [\text{P5(b)}] \\ &= a \cdot 1 && [\text{P6(a)}] \\ &= a && [\text{P2(b)}] \end{aligned}$$

Los siguientes ejemplos ilustran el uso del teorema 6 para simplificar expresiones booleanas.

EJEMPLO 2.11

$$ABC + A\bar{B}C = AC \quad [\text{T6(a)}]$$

EJEMPLO 2.12

$$(AD + B + C)(AD + \overline{(B + C)}) = AD \quad [\text{T6(b)}]$$

EJEMPLO 2.13

Simplificar $(\bar{W} + \bar{X} + \bar{Y} + \bar{Z})(\bar{W} + \bar{X} + \bar{Y} + Z)$
 $(\bar{W} + \bar{X} + Y + \bar{Z})(\bar{W} + \bar{X} + Y + Z).$

$$\begin{aligned} &= (\bar{W} + \bar{X} + \bar{Y})(\bar{W} + \bar{X} + Y + \bar{Z})(\bar{W} + \bar{X} + Y + Z) && [\text{T6(b)}] \\ &= (\bar{W} + \bar{X} + \bar{Y})(\bar{W} + \bar{X} + Y) && [\text{T6(b)}] \\ &= (\bar{W} + \bar{X}) && [\text{T6(b)}] \end{aligned}$$

Teorema 7.

- (a) $ab + a\bar{b}c = ab + ac$.
 (b) $(a + b)(a + \bar{b} + c) = (a + b)(a + c)$.

Demostración. La parte (a) del teorema se demuestra como sigue:

$$\begin{aligned} ab + a\bar{b}c &= a(b + \bar{b}c) && [\text{P5(b)}] \\ &= a(b + c) && [\text{T5(a)}] \\ &= ab + ac && [\text{P5(b)}] \end{aligned}$$

Los siguientes ejemplos ilustran el uso del teorema 7 para simplificar las expresiones booleanas.

EJEMPLO 2.14

$$xy + x\bar{y}(\bar{w} + \bar{z}) = xy + x(\bar{w} + \bar{z}) \quad [\text{T7(a)}]$$

EJEMPLO 2.15

$$(\bar{x}\bar{y} + z)(w + \bar{x}\bar{y} + \bar{z}) = (\bar{x}\bar{y} + z)(w + \bar{x}\bar{y}) \quad [\text{T7(b)}]$$

EJEMPLO 2.16

$$\begin{aligned} (\bar{A} + \bar{B} + \bar{C})(\bar{B} + C)(A + \bar{B}) &= (\bar{A} + \bar{B})(\bar{B} + C)(A + \bar{B}) && [\text{T7(b)}] \\ &= \bar{B}(\bar{B} + C) && [\text{T6(b)}] \\ &= \bar{B} && [\text{T4(b)}] \end{aligned}$$

EJEMPLO 2.17

$$\begin{aligned} w\bar{y} + w\bar{x}y + wx\bar{y} + wx\bar{z} &= w\bar{y} + w\bar{x}y + wx\bar{y} + wx\bar{z} && [\text{T7(a)}] \\ &= w\bar{y} + wy + wx\bar{z} && [\text{T6(a)}] \\ &= w + wx\bar{z} && [\text{T6(a)}] \\ &= w && [\text{T4(a)}] \end{aligned}$$

En los siguientes capítulos veremos que estos teoremas forman la base de algunos métodos estándar y automatizados por computadora para simplificar expresiones booleanas.

Al trabajar con álgebra booleana, con frecuencia necesitamos determinar el complemento de una expresión booleana. El siguiente teorema proporciona la base para esta operación.

Teorema 8. Teorema de DeMorgan

- (a) $\overline{a + b} = \bar{a} \cdot \bar{b}$.
 (b) $\overline{a \cdot b} = \bar{a} + \bar{b}$.

Demostración. Demostremos la parte (a):

Si $X = a + b$, entonces $\overline{X} = \overline{(a + b)}$. Por el postulado 6, $X \cdot \overline{X} = 0$ y $X + \overline{X} = 1$. Si $X \cdot Y = 0$ y $X + Y = 1$, entonces $Y = \overline{X}$, ya que el complemento de X es único. Por tanto, sea $Y = \overline{ab}$ y calculemos $X \cdot Y$ y $X + Y$:

$$\begin{aligned} X \cdot Y &= (a + b)(\bar{a}\bar{b}) \\ &= (\bar{a}\bar{b})(a + b) && [\text{P3(b)}] \\ &= (\bar{a}\bar{b})a + (\bar{a}\bar{b})b && [\text{P5(b)}] \\ &= a(\bar{a}\bar{b}) + (\bar{a}\bar{b})b && [\text{P3(b)}] \\ &= (a\bar{a})\bar{b} + \bar{a}(b\bar{b}) && [\text{P4(b)}] \\ &= 0 \cdot \bar{b} + \bar{a}(b \cdot \bar{b}) && [\text{P6(b), P3(b)}] \\ &= \bar{b} \cdot 0 + \bar{a} \cdot 0 && [\text{P3(b), P6(b)}] \\ &= 0 + 0 && [\text{T2(b)}] \\ &= 0 && [\text{P2(a)}] \end{aligned}$$

$$\begin{aligned} X + Y &= (a + b) + \bar{a}\bar{b} \\ &= (b + a) + \bar{a}\bar{b} && [\text{P3(a)}] \\ &= b + (a + \bar{a}\bar{b}) && [\text{P4(a)}] \\ &= b + (a + \bar{b}) && [\text{T5(a)}] \\ &= (a + \bar{b}) + b && [\text{P3(a)}] \\ &= a + (\bar{b} + b) && [\text{P4(a)}] \\ &= a + (b + \bar{b}) && [\text{P3(a)}] \\ &= a + 1 && [\text{P6(a)}] \\ &= 1 && [\text{T2(a)}] \end{aligned}$$

Por tanto, por la unicidad de \overline{X} , $Y = \overline{X}$, y en consecuencia

$$\overline{ab} = \overline{a + b}$$

Podemos generalizar el teorema 8 como sigue.

- (a) $\overline{a + b + \dots + z} = \bar{a} \cdot \bar{b} \cdot \dots \cdot \bar{z}$.
 (b) $\overline{ab \dots z} = \bar{a} + \bar{b} + \dots + \bar{z}$.

La regla para complementar una expresión es utilizar la relación (a) o (b), reemplazando cada operador + (OR) por un operador • (AND) y viceversa y reemplazando cada variable por su complemento.

Debemos tener cuidado: al aplicar el teorema de DeMorgan, hay que respetar la precedencia de los operadores: • tiene precedencia sobre +. El siguiente ejemplo ilustra este punto importante.

EJEMPLO 2.18

Complementar la expresión $a + bc$.

$$\begin{aligned}\overline{a + b \cdot c} &= \overline{a + (b \cdot c)} \\ &= \bar{a} \cdot \overline{(b \cdot c)} \\ &= \bar{a} \cdot (\bar{b} + \bar{c}) \\ &= \bar{a}\bar{b} + \bar{a}\bar{c}\end{aligned}$$

Observe que: $\overline{a + b \cdot c} \neq \bar{a} \cdot \bar{b} + \bar{c}$

Los siguientes ejemplos ilustran el uso del teorema de DeMorgan.

EJEMPLO 2.19

$$\begin{aligned}\overline{X + Y} &= \bar{X} \cdot \overline{\bar{Y}} && [\text{T8(a)}] \\ &= \bar{X} \cdot Y && [\text{T3}]\end{aligned}$$

EJEMPLO 2.20

Complementar la expresión $a(b + z(x + \bar{a}))$, y simplificar el resultado de modo que los únicos términos complementados sean variables individuales.

$$\begin{aligned}a(\overline{b + z(x + \bar{a})}) &= \bar{a} + \overline{(b + z(x + \bar{a}))} && [\text{T8(b)}] \\ &= \bar{a} + \bar{b} \overline{(z(x + \bar{a}))} && [\text{T8(a)}] \\ &= \bar{a} + \bar{b}(\bar{z} + \overline{(x + \bar{a})}) && [\text{T8(b)}] \\ &= \bar{a} + \bar{b}(\bar{z} + \bar{x} \cdot \bar{a}) && [\text{T8(a)}] \\ &= \bar{a} + \bar{b}(\bar{z} + \bar{x}a) && [\text{T3}] \\ &= \bar{a} + \bar{b}(\bar{z} + \bar{x}) && [\text{T5(a)}]\end{aligned}$$

EJEMPLO 2.21

Repetir el ejemplo 2.20 para la expresión $a(b + c) + \bar{a}b$.

$$\begin{aligned}\overline{a(b + c) + \bar{a}b} &= \overline{\bar{a}b + ac + \bar{a}b} && [\text{P5(b)}] \\ &= \overline{\bar{a}b} && [\text{T6(a)}] \\ &= \bar{b}(\overline{ac}) && [\text{T8(a)}] \\ &= \bar{b}(\bar{a} + \bar{c}) && [\text{T8(b)}]\end{aligned}$$

Como ilustra este último ejemplo, con frecuencia podemos simplificar el proceso de complementar una expresión reduciéndola antes de aplicar el teorema de DeMorgan.

Así, el teorema de DeMorgan presenta la técnica general para complementar expresiones booleanas. Será de especial utilidad en la manipulación de expresiones booleanas a fin de darles formatos adecuados para su realización con tipos específicos de compuertas lógicas.

El último teorema fundamental del álgebra booleana que consideraremos es el teorema de consenso.

Teorema 9. Consenso

- (a) $ab + \bar{a}c + bc = ab + \bar{a}c$,
- (b) $(a + b)(\bar{a} + c)(b + c) = (a + b)(\bar{a} + c)$.

Demostración. De aquí en adelante, utilizaremos los postulados 3 y 4 sin indicarlo de manera explícita.

$$\begin{aligned}
 ab + \bar{a}c + bc &= ab + \bar{a}c + 1 \cdot bc && [\text{P2(b)}] \\
 &= ab + \bar{a}c + (a + \bar{a})bc && [\text{P6(a)}] \\
 &= ab + \bar{a}c + abc + \bar{a}bc && [\text{P5(b)}] \\
 &= (ab + abc) + (\bar{a}c + \bar{a}cb) \\
 &= ab + \bar{a}c && [\text{T4(a)}]
 \end{aligned}$$

La clave para utilizar este teorema es un elemento y su complemento, encontrar los términos asociados y eliminar el término incluido (el término de “consenso”), el cual está compuesto de los términos asociados.

El teorema de consenso es útil para reducir expresiones booleanas y para desarrollar expresiones en los diversos algoritmos de minimización automatizada que describiremos más adelante.

EJEMPLO 2.22

$$AB + \bar{A}CD + BCD = AB + \bar{A}CD \quad [\text{T9(a)}]$$

EJEMPLO 2.23

$$(a + \bar{b})(\bar{a} + c)(\bar{b} + c) = (a + \bar{b})(\bar{a} + c) \quad [\text{T9(b)}]$$

EJEMPLO 2.24

$$\begin{aligned}
 ABC + \bar{A}D + \bar{B}D + CD &= ABC + (\bar{A} + \bar{B})D + CD && [\text{P5(b)}] \\
 &= ABC + \overline{ABD} + CD && [\text{T8(b)}] \\
 &= ABC + \overline{ABD} && [\text{T9(a)}] \\
 &= ABC + (\bar{A} + \bar{B})D && [\text{T8(b)}] \\
 &= ABC + \bar{A}D + \bar{B}D && [\text{P5(b)}]
 \end{aligned}$$

En cada uno de los ejemplos anteriores, un elemento o expresión y su complemento son la clave para reducir la expresión.

Es importante señalar que es fácil demostrar los teoremas presentados por medio de diagramas de Venn. Por tanto, aconsejamos a los lectores utilizar esta imagen gráfica como ayuda para recordar los teoremas importantes. La tabla 2.2 resume los postulados y teoremas básicos del álgebra booleana. Más adelante presentaremos el teorema 10, incluido en esta tabla.

TABLA 2.2 POSTULADOS Y TEOREMAS DEL ÁLGEBRA BOOLEANA

Expresión	Dual
$P2(a) : a + 0 = a$	$P2(b) : a \cdot 1 = a$
$P3(a) : a + b = b + a$	$P3(b) : ab = ba$
$P4(a) : a + (b + c) = (a + b) + c$	$P4(b) : a(bc) = (ab)c$
$P5(a) : a + bc = (a + b)(a + c)$	$P5(b) : a(b + c) = ab + ac$
$P6(a) : a + \bar{a} = 1$	$P6(b) : a \cdot \bar{a} = 0$
$T1(a) : a + a = a$	$T1(b) : a \cdot a = a$
$T2(a) : a + 1 = 1$	$T2(b) : a \cdot 0 = 0$
$T3 : \bar{\bar{a}} = a$	
$T4(a) : a + ab = a$	$T4(b) : a(a + b) = a$
$T5(a) : a + \bar{a}b = a + b$	$T5(b) : a(\bar{a} + b) = ab$
$T6(a) : ab + a\bar{b} = a$	$T6(b) : (a + b)(a + \bar{b}) = a$
$T7(a) : ab + a\bar{b}c = ab + ac$	$T7(b) : (a + b)(a + \bar{b} + c) = (a + b)(a + c)$
$T8(a) : \overline{a + b} = \bar{a}\bar{b}$	$T8(b) : \overline{ab} = \bar{a} + \bar{b}$
$T9(a) : ab + \bar{a}c + bc = ab + \bar{a}c$	$T9(b) : (a + b)(\bar{a} + c)(b + c) = (a + b)(\bar{a} + c)$
$T10(a) : f(x_1, x_2, \dots, x_n) = x_1 f(1, x_2, \dots, x_n) + \bar{x}_1 f(0, x_2, \dots, x_n)$	
$T10(b) : f(x_1, x_2, \dots, x_n) = [x_1 + f(0, x_2, \dots, x_n)][\bar{x}_1 + f(1, x_2, \dots, x_n)]$	

■ 2.2 Funciones de conmutación

Hemos enunciado los postulados y teoremas del álgebra booleana en forma general, sin especificar los elementos del conjunto K . Por tanto, los resultados son válidos para cualquier álgebra booleana. En el siguiente análisis, nos centraremos en el álgebra booleana en la que $K=\{0, 1\}$. Esta formulación se conoce como *álgebra de conmutación*.

El concepto de función es bien conocido para los que están familiarizados con el álgebra ordinaria. Las funciones de conmutación representan el concepto correspondiente para el álgebra de conmutación y se pueden definir como sigue. Sean X_1, X_2, \dots, X_n símbolos llamados variables, cada uno de los cuales representa el elemento 0 o 1 de un álgebra de conmutación (se dice que 0 o 1 es el *valor* de la variable) y sea $f(X_1, X_2, \dots, X_n)$ una función de conmutación de X_1, X_2, \dots, X_n . La función f tiene el valor 0 o 1 según el conjunto de valores asignado a X_1, X_2, \dots, X_n . Como hay n variables y cada variable tiene dos posibles valores, hay 2^n maneras de asignar estos valores a las n variables. Además, existen dos valores posibles para la función $f(x_1, x_2, \dots, x_n)$. Por tanto, hay 2^n diferentes funciones de conmutación de n variables.

Si $n=0$, las dos funciones de conmutación de cero variables son

$$f_0 = 0 \quad f_1 = 1$$

92 Capítulo 2 Métodos algebraicos para el análisis y síntesis de circuitos lógicos

Si $n = 1$, las cuatro funciones de la variable A son

$$\begin{aligned}f_0 &= 0, & f_2 &= A \\f_1 &= \bar{A}, & f_3 &= 1\end{aligned}$$

Deduciremos ahora las 16 funciones de dos variables A y B . Definamos $f(A, B)$ como sigue:

$$f_i(A, B) = i_3 AB + i_2 A\bar{B} + i_1 \bar{A}B + i_0 \bar{A}\bar{B}$$

donde $(i)_{10} = (i_3 i_2 i_1 i_0)_2$ adopta los valores binarios 0000, 0001, 0010, ..., 1111. Las 16 funciones resultantes son:

$$\begin{aligned}f_0(A, B) &= 0 \\f_1(A, B) &= \bar{A}\bar{B} \\f_2(A, B) &= \bar{A}B \\f_3(A, B) &= AB + \bar{A}\bar{B} = \bar{A} \\f_4(A, B) &= A\bar{B} \\f_5(A, B) &= A\bar{B} + \bar{A}\bar{B} = \bar{B} \\f_6(A, B) &= A\bar{B} + \bar{A}B \\f_7(A, B) &= A\bar{B} + \bar{A}B + \bar{A}\bar{B} = \bar{A} + \bar{B} \\f_8(A, B) &= AB \\f_9(A, B) &= AB + \bar{A}\bar{B} \\f_{10}(A, B) &= AB + \bar{A}B = B \\f_{11}(A, B) &= AB + \bar{A}B + \bar{A}\bar{B} = \bar{A} + B \\f_{12}(A, B) &= AB + A\bar{B} = A \\f_{13}(A, B) &= AB + A\bar{B} + \bar{A}\bar{B} = A + \bar{B} \\f_{14}(A, B) &= AB + A\bar{B} + \bar{A}B = A + B \\f_{15}(A, B) &= AB + A\bar{B} + \bar{A}B + \bar{A}\bar{B} = 1\end{aligned}$$

Al evaluar cada una de estas funciones para cada combinación de A y B , podemos resumir la información anterior en forma de tabla, como en la tabla 2.3.

TABLA 2.3 DIECISEIS FUNCIONES DE DOS VARIABLES

AB	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}
00	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
01	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
10	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
11	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

Podemos describir una función de conmutación mediante una expresión de conmutación como sigue:

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

Si $A = 1$ y $B = C = 0$, el valor de la función f es 1, lo que verificamos como sigue:

$$\begin{aligned} f(1, 0, 0) &= 1 \cdot 0 + \bar{1} \cdot 0 + 1 \cdot \bar{0} \\ &= 1 \cdot 0 + 0 \cdot 0 + 1 \cdot 1 && [\text{T3}] \\ &= 0 + 0 + 1 \cdot 1 && [\text{T2(b)}] \\ &= 1 \cdot 1 && [\text{P2(a)}] \\ &= 1 && [\text{P2(b)}] \end{aligned}$$

Podemos calcular otros valores de manera similar; por ejemplo, si $A = 0$, $B = 1$ y $C = 0$, podemos ver que $f = 0$.

2.2.1 Tablas de verdad

Podemos representar una función de conmutación mediante varias expresiones de conmutación diferentes, pero equivalentes. Si evaluamos una función de conmutación para todas las posibles combinaciones de entradas y presentamos los resultados como una tabla, obtenemos una representación única de la función llamada *tabla de verdad*.

Por ejemplo, podemos utilizar las tablas de verdad, como las tablas 2.4a, b y c, para mostrar las operaciones básicas OR, AND y de complemento utilizadas en el álgebra de conmutación, considerando cada una como una función de conmutación y exhibiendo todas las posibles combinaciones de los elementos.

Si evaluamos la función $f(A, B, C) = AB + \bar{A}C + A\bar{C}$ para todas las combinaciones de entradas posibles y las presentamos en forma de tabla, obtenemos la tabla de verdad que se muestra como tabla 2.5a. Al reemplazar cada 0 de la tabla 2.5a por *F* (falso) y cada 1 por *T* (verdadero) obtenemos una forma alternativa de la tabla de verdad, que se muestra en la tabla 2.5b y que

TABLA 2.4 TABLAS DE VERDAD PARA LAS FUNCIONES OR, AND Y NOT

a	b	$f(a, b) = a + b$	a	b	$f(a, b) = a \cdot b$	a	$f(a) = \bar{a}$
0	0	0	0	0	0	0	1
0	1	1	0	1	0	1	0
1	0	1	1	0	0	1	0
1	1	1	1	0	1	0	1

(a)

(b)

(c)

TABLA 2.5 TABLAS DE VERDAD PARA

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

ABC	$f(A, B, C)$	ABC	$f(A, B, C)$
0 0 0	0 ✓	F F F	F
0 0 1	1 ✓	F F T	T
0 1 0	0 ✓	F T F	F
0 1 1	1 ✓	F T T	T
1 0 0	1 ✓	T F F	T
1 0 1	0 ✓	T F T	F
1 1 0	1 ✓	T T F	T
1 1 1	1 ✓	T T T	T

(a)

(b)

94 Capítulo 2 Métodos algebraicos para el análisis y síntesis de circuitos lógicos

demuestra la correspondencia uno a uno que existe entre el álgebra de conmutación y el cálculo funcional de verdad [4].

También podemos utilizar la tabla de verdad como un medio conveniente para evaluar las funciones de conmutación. Por ejemplo, consideremos nuestra función anterior

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

Podemos obtener cada término de la tabla de verdad, uno a la vez, como sigue:

A, B, C	AB	\bar{A}	$\bar{A}C$	$AB + \bar{A}C$	\bar{C}	$A\bar{C}$	$(AB + \bar{A}C) + A\bar{C}$	$f(A, B, C)$
0 0 0	$0 \cdot 0 = 0$	$\bar{0} = 1$	$1 \cdot 0 = 0$	$0 + 0 = 0$	$\bar{0} = 1$	$0 \cdot 1 = 0$	$0 + 0 = 0$	0
0 0 1	$0 \cdot 0 = 0$	$\bar{0} = 1$	$1 \cdot 1 = 1$	$0 + 1 = 1$	$\bar{1} = 0$	$0 \cdot 0 = 0$	$1 + 0 = 1$	1
0 1 0	$0 \cdot 1 = 0$	$\bar{0} = 1$	$1 \cdot 0 = 0$	$0 + 0 = 0$	$\bar{0} = 1$	$0 \cdot 1 = 0$	$0 + 0 = 0$	0
0 1 1	$0 \cdot 1 = 0$	$\bar{0} = 1$	$1 \cdot 1 = 1$	$0 + 1 = 1$	$\bar{1} = 0$	$0 \cdot 0 = 0$	$1 + 0 = 1$	1
1 0 0	$1 \cdot 0 = 0$	$\bar{1} = 0$	$0 \cdot 0 = 0$	$0 + 0 = 0$	$\bar{0} = 1$	$1 \cdot 1 = 1$	$0 + 1 = 1$	1
1 0 1	$1 \cdot 0 = 0$	$\bar{1} = 0$	$0 \cdot 1 = 0$	$0 + 0 = 0$	$\bar{1} = 0$	$1 \cdot 0 = 0$	$0 + 0 = 0$	0
1 1 0	$1 \cdot 1 = 1$	$\bar{1} = 0$	$0 \cdot 0 = 0$	$1 + 0 = 1$	$\bar{0} = 1$	$1 \cdot 1 = 1$	$1 + 1 = 1$	1
1 1 1	$1 \cdot 1 = 1$	$\bar{1} = 0$	$0 \cdot 1 = 0$	$1 + 0 = 1$	$\bar{1} = 0$	$1 \cdot 0 = 0$	$1 + 0 = 1$	1

2.2.2 Formas algebraicas de las funciones de conmutación

Hasta ahora hemos visto varias formas diferentes de las funciones de conmutación, incluidas las expresiones algebraicas, las tablas de verdad y los diagramas de Venn. Ahora definiremos otras formas específicas de las funciones que demostrarán ser muy útiles.

Formas SOP y POS

Las funciones de conmutación en la forma de *suma de productos* (SOP) se construyen al sumar (OR) términos producto (AND), donde cada término producto se forma mediante el AND de varias variables complementadas o no complementadas, cada una de las cuales es una *literal*. Un ejemplo de forma SOP de una función de cuatro variables es

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

Las funciones de conmutación en la forma de *producto de sumas* (POS) se construyen al considerar el producto (AND) de términos suma (OR), donde cada término suma se obtiene mediante el OR de varias literales. Un ejemplo de la forma POS de una función de cuatro variables es

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

Formas canónicas

Las formas canónicas de las funciones de conmutación son ciertas formas SOP y POS con características especiales. Como hemos visto, podemos representar una función de conmutación mediante expresiones de conmutación diferentes pero equivalentes. En cambio, las formas canónicas SOP y POS son únicas para cada función.

Mintérminos. Para una función de n variables, si un término producto contiene cada una de las n variables exactamente una vez, ya sea en forma complementada o no complementada, el término producto es un *mintérmino*. Si una funci

representa como una suma sólo de mintérminos, decimos que la función tiene la forma de *suma canónica de productos (SOP canónica)*. Por ejemplo,

$$f_\alpha(A, B, C) = \bar{A}B\bar{C} + AB\bar{C} + \bar{A}BC + ABC \quad (2.1)$$

es la forma SOP canónica de la función $f_\alpha(A, B, C)$, con cuatro mintérminos.

Para simplificar la escritura de la forma canónica SOP, con frecuencia se usa una notación especial, en la que cada mintérmino se representa mediante un código binario de n bits. Cada bit representa una de las variables del mintérmino como sigue:

Variable no complementada: 1
Variable complementada: 0

Las variables se enumeran en el mismo orden en cada mintérmino. Lo importante de esta notación es que, para que un mintérmino valga 1, cada variable no complementada del mintérmino debe valer 1 y cada variable complementada debe valer 0. Con este código, podemos escribir los mintérminos $f_\alpha(A, B, C)$ en una de las siguientes formas equivalentes:

Mintérmino	Código del mintérmino	Número de mintérmino
$\bar{A}B\bar{C}$	010	m_2
$AB\bar{C}$	110	m_6
$\bar{A}BC$	011	m_3
ABC	111	m_7

Escribimos cada mintérmino en forma abreviada como m_i , donde i es el entero decimal igual al código binario correspondiente para el mintérmino. Así, podemos escribir $f_\alpha(A, B, C)$ en forma compacta como

$$f_\alpha(A, B, C) = m_2 + m_3 + m_6 + m_7 \quad (2.2)$$

Podemos simplificar aún más si escribimos la función en *forma de lista de mintérminos* como sigue:

$$f_\alpha(A, B, C) = \sum m(2, 3, 6, 7) \quad (2.3)$$

Las tres ecuaciones (2.1), (2.2) y (2.3) ilustran tres formas diferentes, pero equivalentes, de representar la forma canónica SOP para $f_\alpha(A, B, C)$.

El orden de las variables en la notación funcional de las ecuaciones (2.2) y (2.3) es muy importante, ya que determina el orden de los bits en los números del mintérmino. Podemos demostrar este hecho fácilmente, cambiando la relación de orden de las variables en la función $f_\alpha(A, B, C)$ a $f_\alpha(B, C, A)$ como sigue:

$$\begin{aligned} f_\beta(B, C, A) &= \sum m(2, 3, 6, 7) \\ &= \underbrace{m_2}_{010} + \underbrace{m_3}_{011} + \underbrace{m_6}_{110} + \underbrace{m_7}_{111} \\ &= \bar{B}C\bar{A} + \bar{B}CA + BC\bar{A} + BCA \\ &= \bar{A}\bar{B}C + A\bar{B}C + \bar{A}BC + ABC \end{aligned} \quad (2.4)$$

Observe que la ecuación (2.4) no es idéntica a la (2.1), aunque las listas de mintérminos sean iguales. Continuamos trabajando con la ecuación (2.4) para obtener

$$\begin{aligned}
 f_\beta(A, B, C) &= f_\beta(B, C, A) \\
 &= \underbrace{\bar{A}\bar{B}C}_{001} + \underbrace{\bar{A}BC}_{011} + \underbrace{A\bar{B}C}_{101} + \underbrace{ABC}_{111} \\
 &= m_1 + m_3 + m_5 + m_7 \\
 &= \sum m(1, 3, 5, 7)
 \end{aligned} \tag{2.5}$$

Las ecuaciones (2.4) y (2.5) son iguales; la diferencia en las listas de mintérminos refleja el orden de las variables en la notación funcional.

Podemos deducir fácilmente la tabla de verdad para $f_\beta(A, B, C)$ a partir de su forma SOP canónica:

Fila númer. (<i>i</i>)	Entradas <i>A</i> <i>B</i> <i>C</i>	m_1 $\bar{A}\bar{B}C$	m_3 $\bar{A}BC$	m_5 $A\bar{B}C$	m_7 ABC	Salidas $f_\beta(A, B, C)$
0	0 0 0	0	0	0	0	0
1	0 0 1	1	0	0	0	1
2	0 1 0	0	0	0	0	0
3	0 1 1	0	1	0	0	1
4	1 0 0	0	0	0	0	0
5	1 0 1	0	0	1	0	1
6	1 1 0	0	0	0	0	0
7	1 1 1	0	0	0	1	1

Un análisis cuidadoso de la tabla muestra que cada fila se numera según su código decimal, y que los únicos unos que aparecen en la tabla son los de las filas *i*, que corresponden a mintérminos m_i . Por tanto, en general, podemos eliminar todos los pasos intermedios y sólo escribir la tabla de verdad directamente a partir de la lista de mintérminos, como se muestra a continuación para la función $f_\alpha(A, B, C)$:

Fila númer. (<i>i</i>)	Entradas <i>A</i> <i>B</i> <i>C</i>	Salidas: $f_\alpha(A, B, C) = \sum m(2, 3, 6, 7)$	Complemento $\bar{f}_\alpha(A, B, C) = \sum m(0, 1, 4, 5)$
0	0 0 0	0	1 $\leftarrow m_0$
1	0 0 1	0	1 $\leftarrow m_1$
2	0 1 0	1 $\leftarrow m_2$	0
3	0 1 1	1 $\leftarrow m_3$	0
4	1 0 0	0	1 $\leftarrow m_4$
5	1 0 1	0	1 $\leftarrow m_5$
6	1 1 0	1 $\leftarrow m_6$	0
7	1 1 1	1 $\leftarrow m_7$	0

Además, podemos ver que la tabla de verdad para $\bar{f}_\alpha(A, B, C)$ tiene unos en las filas 0, 1, 4 y 5. Por tanto,

$$f_\alpha(A, B, C) = \sum m(2, 3, 6, 7)$$

y

$$\bar{f}_\alpha(A, B, C) = \sum m(0, 1, 4, 5)$$

Observe que todos los mintérminos compuestos por tres variables (que suman $2^3 = 8$) están en la lista de mintérminos para $f_A(A, B, C)$ o la lista para $\bar{f}_A(A, B, C)$. En general, cada uno de los 2^n mintérminos de n variables siempre aparecerá en la forma SOP canónica para $f(x_1, x_2, \dots, x_n)$ o la de $\bar{f}(x_1, x_2, \dots, x_n)$.

EJEMPLO 2.25

Dada la función

$$f(A, B, Q, Z) = \bar{A}\bar{B}\bar{Q}\bar{Z} + \bar{A}\bar{B}\bar{Q}Z + \bar{A}B\bar{Q}\bar{Z} + \bar{A}BQZ,$$

expresar las funciones $f(A, B, Q, Z)$ y

$\bar{f}(A, B, Q, Z)$ en forma de lista de mintérminos.

$$\begin{aligned} f(A, B, Q, Z) &= \bar{A}\bar{B}\bar{Q}\bar{Z} + \bar{A}\bar{B}\bar{Q}Z + \bar{A}B\bar{Q}\bar{Z} + \bar{A}BQZ \\ &= m_0 + m_1 + m_6 + m_7 \\ &= \sum m(0, 1, 6, 7) \end{aligned}$$

$\bar{f}(A, B, Q, Z)$ contendrá los 12 ($2^4 - 4$) mintérminos restantes. La lista de mintérminos para esta función es

$$\begin{aligned} \bar{f}(A, B, Q, Z) &= m_2 + m_3 + m_4 + m_5 + m_8 + m_9 \\ &\quad + m_{10} + m_{11} + m_{12} + m_{13} + m_{14} + m_{15} \\ &= \sum m(2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 14, 15) \end{aligned}$$

Debemos recordar del álgebra de conmutación que

$$f(x_1, x_2, \dots, x_n) + \bar{f}(x_1, x_2, \dots, x_n) = 1$$

Sin embargo, como

$$f(x_1, x_2, \dots, x_n) + \bar{f}(x_1, x_2, \dots, x_n) = \sum_{i=0}^{2^n-1} m_i$$

entonces

$$\sum_{i=0}^{2^n-1} m_i = 1 \tag{2.6}$$

En otras palabras, la suma (OR) de todos los mintérminos de n variables (m_0, \dots, m_{2^n-1}) es igual a 1. Por último, es importante observar que, si bien

$$AB + \bar{A}\bar{B} = 1 \quad [P6(a)]$$

y

$$AB + \bar{A} + \bar{B} = 1 \quad [T7(b)]$$

son expresiones válidas,

$$AB + \bar{A}\bar{B} \neq 1.$$

Un error común de los estudiantes del álgebra de conmutación es igualar la expresión anterior a 1.

Maxtérmino. Si un término suma de una función de n variables contiene cada una de las n variables exactamente una vez en forma complementada o no complementada, el término suma es un *maxtérmino*. Si representamos una

función como producto de términos suma, cada uno de los cuales es un maxtrérmino, decimos que la función tiene la *forma canónica de producto de sumas (POS canónica)*. Por ejemplo,

$$f_y(A, B, C) = (A + \bar{B} + C)(A + B + \bar{C})(\bar{A} + B + C)(\bar{A} + B + \bar{C}) \quad (2.7)$$

es la forma POS canónica de una función $f_y(A, B, C)$, que tiene cuatro maxtrérminos.

Adoptamos una notación especial para los maxtrérminos, como hicimos con los mintérminos, con la diferencia de que el código se intercambia como sigue:

Variable no complementada: 0

Variable complementada: 1

Lo importante de esta notación es que, para que un maxtrérmino valga 0, toda variable no complementada del maxtrérmino debe valer 0 y toda variable complementada debe valer 1. Así, podemos representar los maxtrérminos de $f_y(A, B, C)$ como sigue:

Maxtrérmino	Código del maxtrérmino	Lista de mintérmino
$A + B + C$	000	M_0
$A + B + \bar{C}$	001	M_1
$\bar{A} + B + C$	100	M_4
$\bar{A} + B + \bar{C}$	101	M_5

Escribimos cada maxtrérmino en forma abreviada como M_i , donde i es el entero decimal del código binario correspondiente para el maxtrérmino. Así,

$$f_y(A, B, C) = M_0 M_1 M_4 M_5 \quad (2.8)$$

$$= \prod M(0, 1, 4, 5) \quad (2.9)$$

La última forma es la *forma de lista de maxtrérminos*. Las ecuaciones (2.7), (2.8) y (2.9) son formas POS canónicas equivalentes para $f_y(A, B, C)$. Como en el caso de las ecuaciones (2.2) y (2.3), el orden de las variables en las ecuaciones (2.8) y (2.9) es muy importante. La tabla de verdad para $f_y(A, B, C)$ es

Núm. de fila (i)	Entradas ABC	M_0 $A + B + C$	M_1 $A + B + \bar{C}$	M_4 $\bar{A} + B + C$	M_5 $\bar{A} + B + \bar{C}$	Salidas $f_y(A, B, C)$
0	0 0 0	0	1	1	1	0
1	0 0 1	1	0	1	1	0
2	0 1 0	1	1	1	1	1
3	0 1 1	1	1	1	1	1
4	1 0 0	1	1	0	1	0
5	1 0 1	1	1	1	0	0
6	1 1 0	1	1	1	1	1
7	1 1 1	1	1	1	1	1

Cada fila de la tabla se numera según el código decimal, como lo hicimos con los mintérminos. Observe que los únicos ceros de la tabla aparecen en los renglones i correspondientes a los maxtrérminos M_i . Por tanto, como hicimos

con los mintérminos, podemos generar la tabla de verdad mediante una inspección directa de la lista de maxterminos. Al comparar las tablas de verdad para $f_\alpha(A, B, C)$ y $f_\gamma(A, B, C)$ vemos que

$$\begin{aligned} f_\alpha(A, B, C) &= \sum m(2, 3, 6, 7) \\ &= f_\gamma(A, B, C) \\ &= \prod M(0, 1, 4, 5) \end{aligned} \quad (2.10)$$

Por tanto, las funciones $f_\alpha(A, B, C)$ y $f_\gamma(A, B, C)$ son iguales y la ecuación (2.10) muestra las formas SOP y POS canónicas para $f_\alpha(A, B, C)$.

Ejemplo 2.26

Dada la función $f(A, B, C) =$

$(A + B + \bar{C})(A + \bar{B} + \bar{C})(\bar{A} + B + \bar{C})(\bar{A} + \bar{B} + C)$,

construir la tabla de verdad y expresar la función con maxterminos y con mintérminos.

$$\begin{aligned} f(A, B, C) &= \underbrace{(A + B + \bar{C})}_{001} \underbrace{(A + \bar{B} + \bar{C})}_{011} \underbrace{(\bar{A} + B + \bar{C})}_{101} \underbrace{(\bar{A} + \bar{B} + C)}_{111} \\ &= M_1 M_3 M_5 M_7 \\ &= \prod M(1, 3, 5, 7) \end{aligned}$$

Los maxterminos colocan ceros en las filas 1, 3, 5 y 7 de la tabla de verdad.

Núm. de fila (i)	Entradas ABC	Salidas $f(A, B, C)$	$= \prod M(1, 3, 5, 7)$
0	0 0 0	1	
1	0 0 1	0	$\leftarrow M_1$
2	0 1 0	1	
3	0 1 1	0	$\leftarrow M_3$
4	1 0 0	1	
5	1 0 1	0	$\leftarrow M_5$
6	1 1 0	1	
7	1 1 1	0	$\leftarrow M_7$

De la tabla de verdad para $f(A, B, C)$, observamos que

$$f(A, B, C) = \sum m(0, 2, 4, 6)$$

Por tanto,

$$\begin{aligned} \bar{f}(A, B, C) &= \sum m(1, 3, 5, 7) \\ &= \underbrace{m_1}_{001} + \underbrace{m_3}_{011} + \underbrace{m_5}_{101} + \underbrace{m_7}_{111} \\ &= \bar{A}\bar{B}C + \bar{A}BC + A\bar{B}C + ABC \end{aligned}$$

En consecuencia,

$$\begin{aligned} f(A, B, C) &= \overline{\bar{A}\bar{B}C + \bar{A}BC + A\bar{B}C + ABC} \\ &= \overline{\bar{A}BC} \cdot \overline{\bar{A}BC} \cdot \overline{A\bar{B}C} \cdot \overline{ABC} \end{aligned}$$

$$\begin{aligned}
 &= \underbrace{(A + B + \bar{C})}_{001} \underbrace{(A + \bar{B} + \bar{C})}_{011} \underbrace{(\bar{A} + B + \bar{C})}_{101} \underbrace{(\bar{A} + \bar{B} + \bar{C})}_{111} \\
 &= M_1 M_3 M_5 M_7 \\
 &= \prod M(1, 3, 5, 7)
 \end{aligned}$$

Por tanto, hemos demostrado de manera algebraica que

$$f(A, B, C) = \prod M(1, 3, 5, 7) = \sum m(0, 2, 4, 6)$$

lo que es obvio si examinamos la tabla de verdad.

Al manejar funciones, queda clara una relación entre los mintérminos m_i y los maxtérminos M_i . Por ejemplo, para una función $f(A, B, C)$ tenemos que

$$\tilde{m}_1 = \overbrace{\bar{A}\bar{B}C}^{001} = \underbrace{A + B + \bar{C}}_{001} = M_1$$

(código del mintérmino) (código del maxtérmino)

y viceversa. Esto ocurre en el caso general; es decir,

$$\tilde{m}_i = M_i \quad (2.11)$$

$$M_i = \tilde{m}_i = m_i \quad (2.12)$$

Por tanto, los mintérminos y maxtérminos son complementarios entre sí.

EJEMPLO 2.27

Dada la función $f(A, B, C)$ del ejemplo 2.26, determinar la relación entre los maxtérminos para la función y su complemento.

La tabla de verdad es:

Núm. de fila (i)	Entradas $A\bar{B}C$	Salidas $f(A, B, C)$	Salidas $\bar{f}(A, B, C)$	$= \prod M(0, 2, 4, 6)$
0	0 0 0	1	0	$\leftarrow M_0$
1	0 0 1	0	1	
2	0 1 0	1	0	$\leftarrow M_2$
3	0 1 1	0	1	
4	1 0 0	1	0	$\leftarrow M_4$
5	1 0 1	0	1	
6	1 1 0	1	0	$\leftarrow M_6$
7	1 1 1	0	1	

Como los renglones 0, 2, 4 y 6 tienen ceros, la forma canónica de $\bar{f}(A, B, C)$ es

$$\bar{f}(A, B, C) = \prod M(0, 2, 4, 6)$$

y, por tanto,

$$f(A, B, C) = \prod M(1, 3, 5, 7)$$

La función $f(A, B, C)$ tiene tres variables y, por tanto, ocho maxterminos, todos los cuales aparecen en la lista de $f(A, B, C)$ o bien de $\bar{f}(A, B, C)$. Por el álgebra de conmutación, sabemos que

$$f(A, B, C) \cdot \bar{f}(A, B, C) = 0$$

y por tanto

$$(M_0 M_2 M_4 M_6)(M_1 M_3 M_5 M_7) = 0$$

o bien

$$\prod_{i=0}^{2^3-1} M_i = 0$$

Este ejemplo ilustra una relación que se cumple en general:

$$\prod_{i=0}^{2^n-1} M_i = 0 \quad (2.13)$$

Por último, observemos de la tabla de verdad que

$$f(A, B, C) = \sum m(0, 2, 4, 6) = \prod M(1, 3, 5, 7)$$

y

$$\bar{f}(A, B, C) = \sum m(1, 3, 5, 7) = \prod M(0, 2, 4, 6)$$

2.2.3 Deducción de formas canónicas

En los ejemplos anteriores mostramos el modo de traducir directamente las formas POS y SOP canónicas de una función en tablas de verdad y viceversa. Si expresamos una función en forma no canónica, con frecuencia es más conveniente utilizar el álgebra de conmutación para convertirla en su forma POS o SOP canónica, sin tener que deducir primero la tabla de verdad.

El siguiente teorema se utiliza con frecuencia en el desarrollo de expresiones de conmutación a una forma canónica.

Teorema 10. Teorema de desarrollo de Shannon

- (a) $f(x_1, x_2, \dots, x_n) = x_1 \cdot f(1, x_2, \dots, x_n) + \bar{x}_1 \cdot f(0, x_2, \dots, x_n)$
- (b) $f(x_1, x_2, \dots, x_n) = [x_1 + f(0, x_2, \dots, x_n)][\bar{x}_1 + f(1, x_2, \dots, x_n)]$

La base de este teorema la constituyen por el postulado 2 y el teorema 1. Como $x_1 = x_1 \cdot x_1 = x_1 \cdot 1$, podemos reemplazar cualquier x_1 dentro de la función de la parte (a) del teorema por 1, y de igual manera $x_1 = x_1 + x_1 = x_1 + 0$, lo que nos permite reemplazar cualquier x_1 dentro de la función de la parte (b) del teorema por 0. El teorema 10 es muy útil para desarrollar funciones o añadir literales a los términos producto.

EJEMPLO 2.28

Convertir la siguiente función de conmutación a la forma SOP canónica.

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

Aplicamos de manera sistemática el teorema 10a a esta función para las tres variables A, B y C . Para la variable A , el teorema 10a da

$$\begin{aligned} f(A, B, C) &= AB + A\bar{C} + \bar{A}\bar{C} \\ &= A \cdot f(1, B, C) + \bar{A} \cdot f(0, B, C) \end{aligned}$$

$$\begin{aligned}
 &= A(1 \cdot B + 1 \cdot \bar{C} + \bar{1} \cdot C) + \bar{A}(0 \cdot B + 0 \cdot \bar{C} + \bar{0} \cdot C) \\
 &= A(B + \bar{C}) + \bar{A}C
 \end{aligned}$$

Para la variable B tenemos

$$\begin{aligned}
 f(A, B, C) &= A(B + \bar{C}) + \bar{A}C \\
 &= B[A(1 + \bar{C}) + \bar{A}C] + \bar{B}[A(0 + \bar{C}) + \bar{A}C] \\
 &= B[A + \bar{A}C] + \bar{B}[A\bar{C} + \bar{A}C] \\
 &= AB + \bar{A}BC + A\bar{B}\bar{C} + \bar{A}\bar{B}C
 \end{aligned}$$

Por último, para la variable C ,

$$\begin{aligned}
 f(A, B, C) &= AB + \bar{A}BC + A\bar{B}\bar{C} + \bar{A}\bar{B}C \\
 &= C[AB + \bar{A}B \cdot 1 + A\bar{B} \cdot \bar{1} + \bar{A}\bar{B} \cdot 0] \\
 &\quad + \bar{C}[AB + \bar{A}B \cdot 0 + A\bar{B} \cdot \bar{0} + \bar{A}\bar{B} \cdot 0] \\
 &= ABC + \bar{A}BC + \bar{A}\bar{B}C + AB\bar{C} + A\bar{B}\bar{C}
 \end{aligned}$$

Otro método para convertir expresiones a una forma SOP o POS canónica consiste en aplicar el teorema 6 para agregar literales a los términos producto o suma hasta obtener mintérminos o maxtérminos. Los siguientes ejemplos ilustran este procedimiento.

EJEMPLO 2.29

Convertir la siguiente función a forma SOP canónica:

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

Aplicamos el teorema 6a a cada uno de los tres términos producto de esta expresión.

$$\begin{aligned}
 AB &= AB\bar{C} + ABC = m_6 + m_7 \\
 A\bar{C} &= A\bar{C}\bar{B} + A\bar{C}B = A\bar{B}\bar{C} + AB\bar{C} = m_4 + m_5 \\
 \bar{A}C &= \bar{A}C\bar{B} + \bar{A}CB = \bar{A}\bar{B}C + \bar{A}BC = m_1 + m_3
 \end{aligned}$$

Por tanto,

$$\begin{aligned}
 f(A, B, C) &= AB + A\bar{C} + \bar{A}C \\
 &= (m_6 + m_7) + (m_4 + m_5) + (m_1 + m_3) \\
 &= \sum m(1, 3, 4, 6, 7)
 \end{aligned}$$

EJEMPLO 2.30

Desarrollar la siguiente función en forma POS canónica:

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

Podemos aplicar el teorema 6a como sigue para obtener los maxtérminos:

$$\begin{aligned}
 A &= (A + \bar{B})(A + B) \\
 &= (A + \bar{B} + \bar{C})(A + \bar{B} + C)(A + B + \bar{C})(A + B + C) \\
 &= M_3 M_2 M_1 M_0 \\
 (A + \bar{C}) &= (A + \bar{C} + \bar{B})(A + \bar{C} + B) \\
 &= (A + \bar{B} + \bar{C})(A + B + \bar{C})
 \end{aligned}$$

$$= M_3 M_1$$

Por tanto,

$$\begin{aligned} A(A + C) &= (M_3 M_2 M_1 M_0)(M_3 M_1) \\ &= \prod M(0, 1, 2, 3) \end{aligned}$$

2.2.4 Funciones con especificación incompleta

En el diseño de circuitos digitales, con frecuencia ocurre que la función de conmutación no tiene una especificación completa. En otras palabras, puede ocurrir que una función deba contener ciertos mintérminos y omitir otros, y que los mintérminos restantes sean opcionales. En este caso, podemos incluir los mintérminosopcionales en el diseño lógico si éstos ayudan a simplificar el circuito lógico, o bien omitirlos. Un mintérmino opcional es un *mintérmino prescindible*. Si expresamos una función mediante sus maxterminos, por lo general escribimos los mintérminos prescindibles en la forma correspondiente de maxterminos, llámándose en este caso *maxterminos prescindibles*.

Los términos prescindibles surgen de dos formas. En la primera, podría ocurrir que ciertas combinaciones de entrada nunca se aplican a una red de conmutación en particular; por tanto, como nunca aparecen, podemos utilizar sus mintérminos a nuestro antojo. Estas condiciones prescindibles ocurren de manera natural en muchas aplicaciones prácticas. Por ejemplo, suponga que una red de conmutación tiene entradas $a_3 a_2 a_1 a_0$, las cuales representan dígitos del código binario decimal (BCD) definido en la tabla 2.6. (Recuerde que analizamos el código BCD en el capítulo 1.) Sólo pueden aparecer 10 mintérminos, $m_0 \dots m_9$, los correspondientes a los 10 dígitos decimales. Los otros seis mintérminos, $m_{10} \dots m_{15}$, no pueden aparecer y, por tanto, son prescindibles en todas las situaciones. En consecuencia, podemos incluir u omitir estos términos en las expresiones de conmutación para cualquier función $f(a_3 a_2 a_1 a_0)$ de estas entradas.

TABLA 2.6 CÓDIGO DECIMAL CODIFICADO EN BINARIO (BCD)

Dígito decimal	Código BCD $a_3 a_2 a_1 a_0$	Dígito decimal	Código BCD $a_3 a_2 a_1 a_0$
0	0000	5	0101
1	0001	6	0110
2	0010	7	0111
3	0011	8	1000
4	0100	9	1001

Las condiciones prescindibles también surgen cuando ocurren todas las combinaciones de entradas de una red dada, pero sólo se requiere que la salida sea 1 o 0 para ciertas combinaciones.

Al escribir las expresiones de conmutación, indicaremos los mintérminos prescindibles como d_i en vez de m_i , y los maxterminos prescindibles como D_i en vez de M_i , como en el siguiente ejemplo.

EJEMPLO 2.31

Sea $f(A, B, C)$ una función con mintérminos m_0, m_3 y m_7 , y condiciones prescindibles d_4 y d_5 . Expressar la función y su complemento con mintérminos y con maxterminos; reducir después la función a su forma más sencilla.

La forma de lista de mintérminos para esta función es

$$f(A, B, C) = \sum m(0, 3, 7) + d(4, 5)$$

y la lista de maxterminos es

$$f(A, B, C) = \prod M(1, 2, 6) \cdot D(4, 5)$$

Observe que los maxterminos prescindibles D , son sencillamente los mintérminos prescindibles, ya que los términos pueden ser 1 o 0. De aquí que

$$\begin{aligned} \bar{f}(A, B, C) &= \sum m(1, 2, 6) + d(4, 5) \\ &= \prod M(0, 3, 7) \cdot D(4, 5) \end{aligned}$$

Para simplificar la expresión $f(A, B, C)$, enumeramos los términos como

$$f(A, B, C) = \bar{A}\bar{B}\bar{C} + \bar{A}BC + ABC + d(A\bar{B}\bar{C} + A\bar{B}C)$$

Observe que el segundo y tercer término difieren sólo en una literal y, por tanto podemos agruparlos para obtener

$$f(A, B, C) = \bar{A}\bar{B}\bar{C} + BC + d(A\bar{B}\bar{C} + A\bar{B}C)$$

Si el uso de los prescindibles, no podríamos simplificar aún más la función. Sin embargo, recordemos que los prescindibles, por definición, pueden ser cero o uno. Por tanto, podemos utilizarlos u omitirlos, según ayuden o no en la simplificación. En la función anterior, si optamos por utilizar d_4 y omitir d_5 , la función se convierte en

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

que es la forma más sencilla de la función. Podemos realizar un análisis similar con la función en forma de maxterminos.

2.3 Circuitos de conmutación

Los circuitos lógicos digitales, o *circuitos de conmutación*, como se les llama con frecuencia, constan de combinaciones seriales y paralelas de elementos de conmutación llamados *compuertas*, o se implantan mediante arreglos lógicos programables o dispositivos similares. Desde un punto de vista matemático, las compuertas son sólo rutas de señales abiertas o cerradas. Desde un punto de vista tecnológico, las compuertas son dispositivos electrónicos de conmutación de gran velocidad que pueden activarse o desactivarse en pocos nanosegundos. En este capítulo analizaremos el uso de compuertas para la construcción de circuitos lógicos que realicen funciones de conmutación. Analizaremos el diseño con arreglos lógicos programables en el capítulo 6.

2.3.1 Compuertas lógicas electrónicas

En los circuitos lógicos digitales, podemos asociar las variables de conmutación a las condiciones de entrada de las compuertas; es decir, a niveles de voltaje altos o bajos aplicados a las entradas de una compuerta. Las funciones de conmutación pueden corresponder a la salida de una compuerta o sistema de compuertas, representadas por un nivel de voltaje alto o bajo en la salida.

Señales eléctricas y valores lógicos

En los libros de datos, las tablas de verdad que definen la operación de los circuitos de compuertas lógicas se presentan en términos de un voltaje alto (H) y otro bajo (L). El diseñador puede utilizar estos niveles de voltaje para representar los valores lógicos 0 y 1 de diversas formas. La convención de *lógica positiva* utiliza un voltaje alto (H) para representar el 1 lógico y un voltaje bajo (L) para representar el 0 lógico. La convención de *lógica negativa* utiliza L para representar el 1 lógico y H para el 0 lógico. En un sistema de *lógica mixta*, se utiliza la lógica positiva para algunas señales y la negativa para otras.

Una señal de 1 lógico es *afirmada, activa o verdadera*. Una señal *alta activa* se afirma cuando es alta (en lógica positiva), mientras que una señal *baja activa* se afirma cuando es baja (en lógica negativa). Si una señal está afirmada, es decir, si indica 0 lógico, es una señal *no afirmada, negada o falsa*. Utilizamos el término *polaridad* para referirnos a la naturaleza alta activa o baja activa de una señal lógica.

Al representar las señales mediante variables lógicas, escribimos los nombres de la señal baja activa en forma complementada (por ejemplo, \bar{a} , a' , a^*), y los de la señal alta activa en forma no complementada (a). Debemos elegir cada nombre de señal de modo que el nombre sugiera el propósito de ésta. Por ejemplo, el nombre de señal *RUN* sugiere una señal afirmada (alta) para que comience a trabajar un equipo. Si la señal es baja activa, deberemos utilizar el nombre de señal *RUN* para indicar que el equipo trabajará cuando la señal esté afirmada baja. En este texto utilizaremos principalmente señales altas activas, pero como muchos módulos de circuitos comerciales tienen entradas o salidas bajas activas, también presentaremos varios ejemplos en los que se utilicen señales bajas activas.

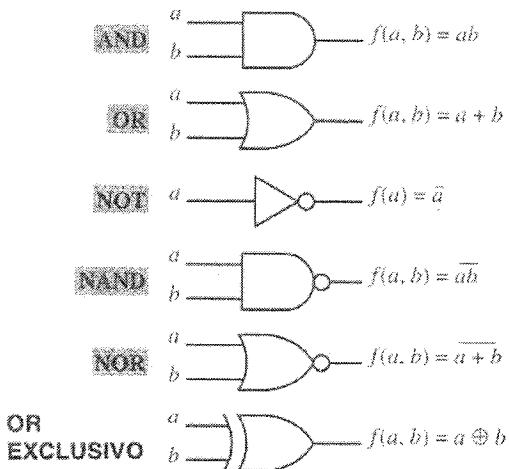
Símbolos de compuerta

Representamos cada compuerta en un diagrama de circuitos lógicos mediante un símbolo que incluye las entradas y las salidas. El número de entradas de una compuerta se conoce como su *fan-in* (ábanico de entrada). Hay módulos de circuitos estándar que contienen compuertas AND, OR, NAND y NOR con un número limitado de opciones de *fan-in*; por lo general, las compuertas tienen dos, tres, cuatro u ocho entradas. Los dispositivos lógicos programables y los circuitos adaptados proporcionan por lo general una amplia gama de opciones *fan-in*, lo que hace posible una correspondencia más directa entre el circuito y la expresión lógica por realizar.

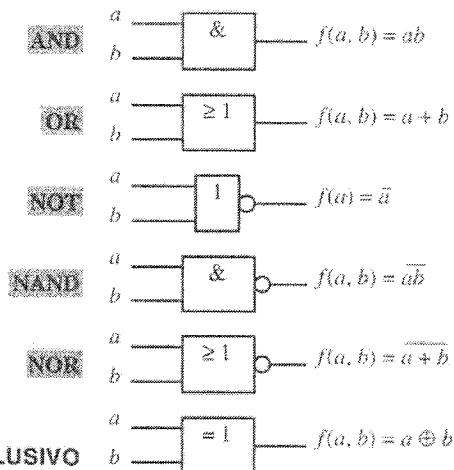
La forma del cuerpo del símbolo representa la función lógica básica, u operación booleana, realizada por la compuerta (OR, AND, NOT u otras). Las *burbujas* dibujadas en las entradas o salidas de un símbolo lógico indican señales bajas activas. Una burbuja en una entrada indica que la entrada es baja activa, es decir, que debe estar afirmada baja para obtener un 1 lógico como entrada de la función. La ausencia de burbuja indica una entrada alta activa; la entrada se

afirma con el valor 1 lógico. De igual manera, una burbuja en una salida indica una salida baja activa, lo que implica que si la evaluación de la función produce 1, se obtiene un 0 lógico en la salida.

Para entender los circuitos lógicos, debemos dominar los operadores de la figura 2.4. Los conjuntos de símbolos ilustrados se definen en la norma IEEE/ANSI Estándar 91-1984 [7]. Analizaremos cada uno de estos operadores, o *compuertas*, en la siguiente sección. En este texto, cuando dibujemos circuitos compuestos por compuertas discretas, utilizaremos principalmente el conjunto de símbolos 1, en el que se identifican las funciones lógicas por medio de formas distintivas de los símbolos. Utilizaremos el conjunto de símbolos 2 en capítulos posteriores para describir módulos funcionales mayores.



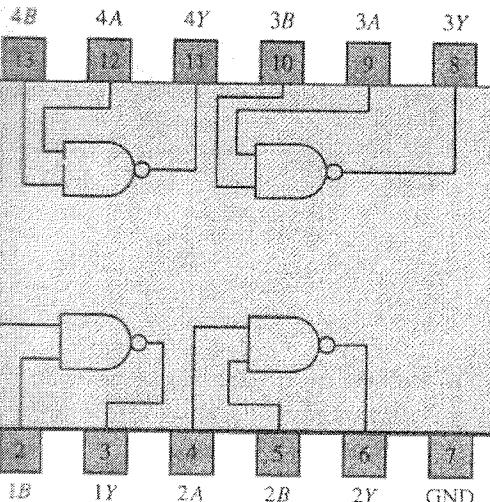
Conjunto de símbolos 1



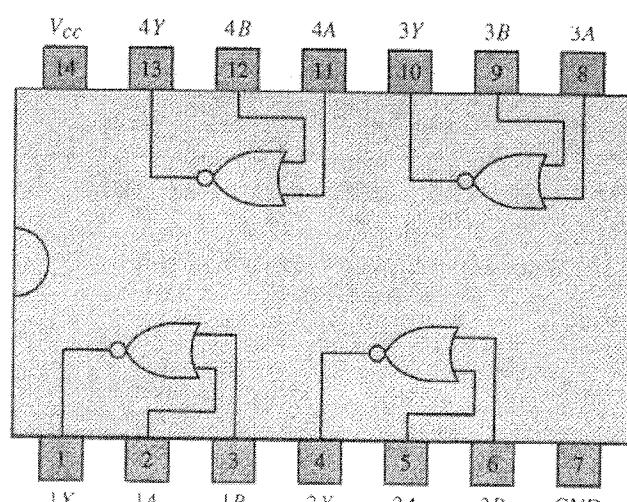
Conjunto de símbolos 2 (Estándar ANSI/IEEE 91-1984)

Figura 2.4 Símbolos para dispositivos de conmutación.

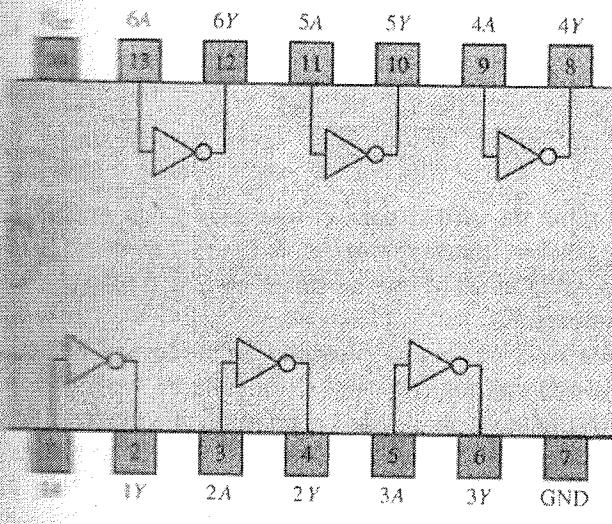
Los módulos de circuitos disponibles en el mercado que contienen configuraciones específicas de compuertas lógicas discretas aparecen en la figura 2.5. Estos módulos se utilizan para construir circuitos útiles para aplicaciones prácticas. Los grupos de compuertas individuales implantados en un módulo lógico se caracterizan como módulos con integración a pequeña escala (SSI) y contienen entre 10 y 100 transistores que conforman el módulo total. Los módulos descritos están disponibles en paquetes duales en línea (DIP) con las asignaciones

7400: $Y = \overline{AB}$

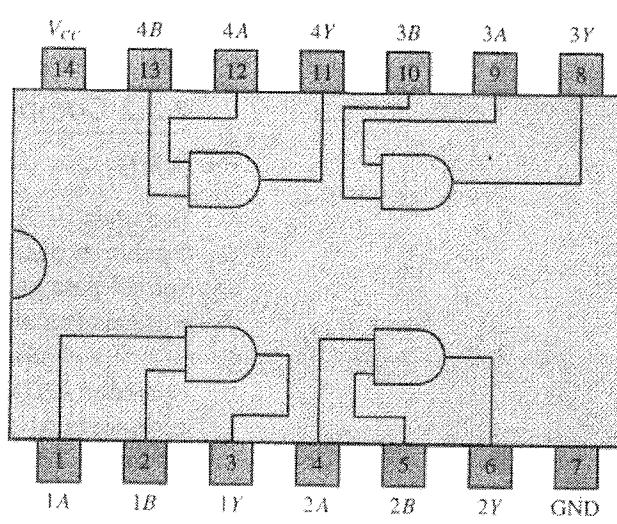
Compuertas NAND cuádruples de dos entradas

7402: $Y = \overline{A + B}$

Compuertas NOR cuádruples de dos entradas

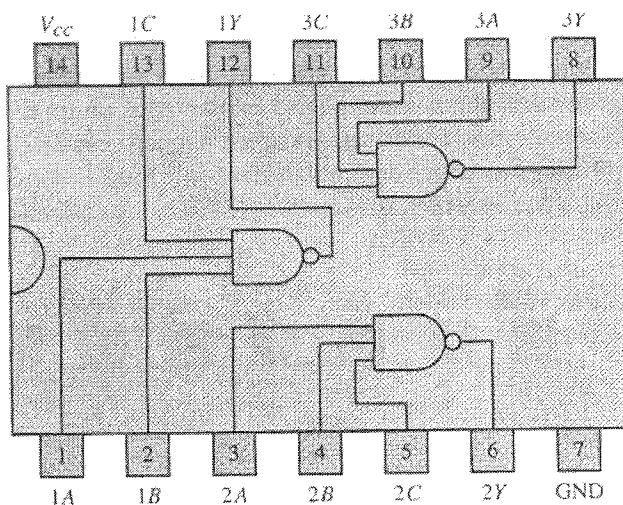
7404: $Y = \overline{A}$

Inversores sextuplo

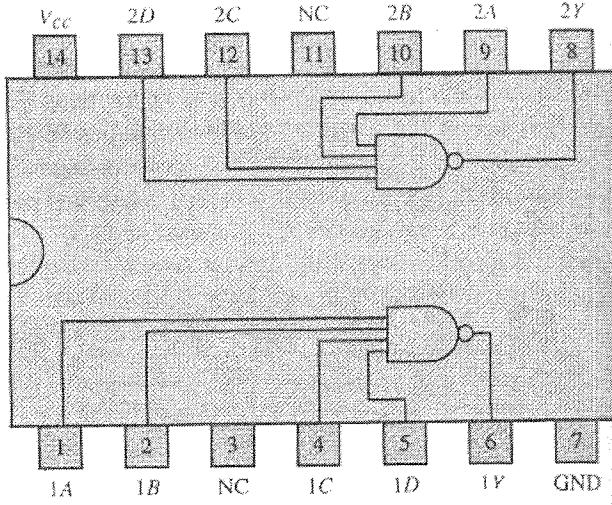
7408: $Y = AB$

Compuertas AND cuádruples de dos entradas

Figura 2.5 Dispositivos TTL estándar de circuitos integrados a pequeña escala (vista superior).



7410: $Y = \overline{ABC}$
Compuertas NAND triples de tres entradas



7420: $Y = \overline{ABCD}$
Compuertas NAND duales de cuatro entradas

Figura 2.5 Dispositivos TTL estándar de circuitos integrados a pequeña escala (continuación).

de pines indicadas en la figura 2.5. Con frecuencia, las funciones completas se realizan mediante un dispositivo de circuito integrado a muy gran escala (VLSI). En estos casos, el diseño se efectúa con compuertas individuales o módulos funcionales que contienen patrones predefinidos de compuertas. Trataremos el diseño modular en el capítulo 4.

2.3.2 Componentes funcionales básicos

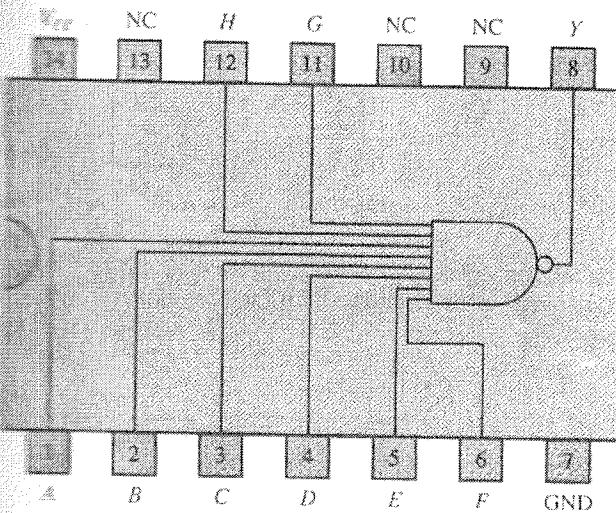
AND

Podemos determinar la tabla de verdad para el operador AND mediante la álgebra de conmutación, cuyo resultado aparece en la figura 2.6a. Esta tabla de verdad para el operador AND muestra que su salida es 1 si y sólo si ambas entradas son simultáneamente 1.

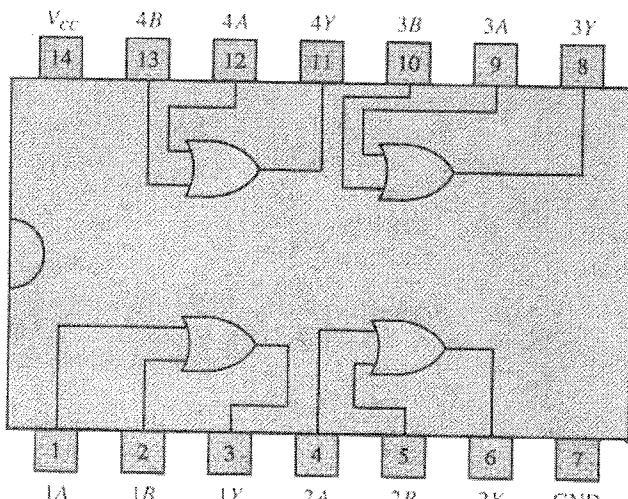
La compuerta AND electrónica está diseñada de modo que realice el operador AND en un sistema con lógica positiva. La tabla de verdad de una compuerta AND se da en la figura 2.6b, donde L representa un voltaje bajo y H un voltaje alto. Observe que el operador AND de la figura 2.6a se realiza al sustituir 0 por L y 1 por H en la tabla de verdad de la compuerta AND. Los símbolos estándar para la compuerta AND aparecen en las figuras 2.6c y d. En la figura 2.6d, observe que el símbolo de bloque estándar IEEE utiliza el símbolo $\&$ para indicar que la operación AND se ejecuta dentro del bloque.

OR

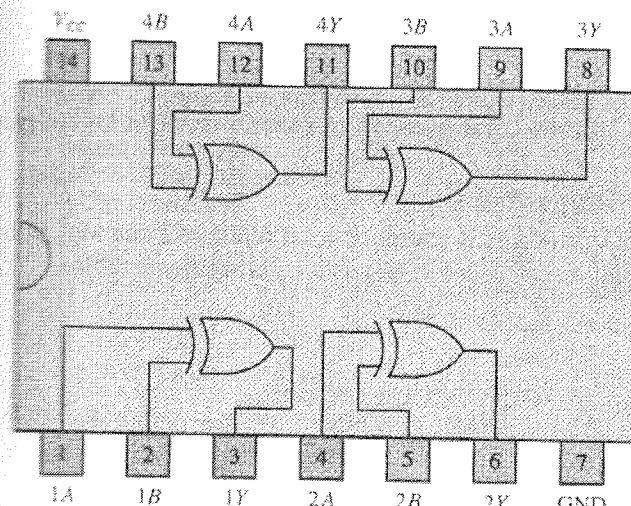
La función OR es idéntica al operador OR del álgebra de conmutación; su tabla de verdad aparece en la figura 2.7a. Observe que la salida es 0 si y sólo si ambas entradas son 0, y 1 si una o más entradas son 1. La tabla de verdad correspondiente

7430: $Y = \overline{ABCDEF}H$

Compuerta NAND de ocho entradas

7432: $Y = A + B$

Compuertas OR cuádruples de dos entradas

7486: $Y = A \oplus B$

Compuertas OR exclusivo cuádruples con dos entradas

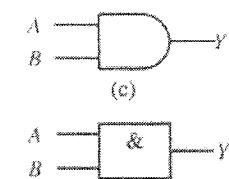
Figura 2.5 Dispositivos TTL estándar de circuitos integrados a pequeña escala (continuación).

de una compuerta OR electrónica se da en la figura 2.7b. Cabe señalar que la compuerta OR realiza el operador OR en un sistema con lógica positiva. Los símbolos estándar de la compuerta OR aparecen en las figuras 2.7c y d. En la figura 2.7d, observe que el símbolo de bloque de IEEE contiene la designación ≥ 1 . Esto significa que la suma matemática de los valores de las variables de

a	b	$f_{\text{AND}}(a, b) = ab$
0	0	0
0	1	0
1	0	0
1	1	1



<i>A</i>	<i>B</i>	<i>Y</i>
L	L	L
L	H	L
H	L	L
H	H	R



133

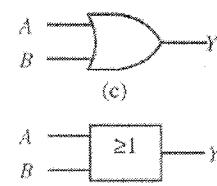
Figura 2.6 La función lógica AND y la compuerta AND. (a) La función lógica AND. (b) Compuerta AND electrónica. (c) Símbolo estándar. (d) Símbolo de bloque IEEE.

a	b	$\text{for}(a, b) = a + b$
0	0	0
0	1	1
1	0	1
1	1	1

三

A	B	Y
L	L	L
L	H	H
H	L	H
H	H	R

(



(d)

Figura 2.7 La función lógica OR y la compuerta OR. (a) La función lógica OR. (b) Compuerta OR electrónica. (c) Símbolo estándar. (d) Símbolo de bloque IEEE.

entrada a y b determina la salida de la compuerta. La salida es 1 cuando la suma de a y b es mayor o igual que 1, como se muestra en la siguiente tabla:

a	b	$\text{sum}(a, b)$	$\text{sum}(a, b) \geq 1?$	$f_{OR}(a, b) = a + b$
0	0	0	Falso	0
0	1	1	Verdadero	1
1	0	1	Verdadero	1
1	1	2	Verdadero	2

NOT

Una compuerta NOT (Fig. 2.8), o *inversor*, siempre tiene exactamente una entrada y se utiliza para implantar el concepto de complemento del álgebra de conmutación. Cualquier variable tiene sus formas verdadera (no complementada) y falsa (complementada), a y \bar{a} , respectivamente. Utilizamos una compuerta NOT para obtener una a partir de la otra.

Los símbolos estándar para la compuerta NOT, que se muestran en las figuras 2.8c y d, incluyen una burbuja en la salida de la compuerta. Como ya hemos señalado, una burbuja en la salida de cualquier elemento de circuito lógico indica que un 1 lógico interno produce un 0 lógico externo y, de manera similar, un 0 lógico interno produce un 1 lógico externo. La compuerta NOT no realiza ninguna otra función lógica; por tanto, el valor lógico de la salida de una compuerta NOT es sólo el complemento del valor lógico de su entrada.

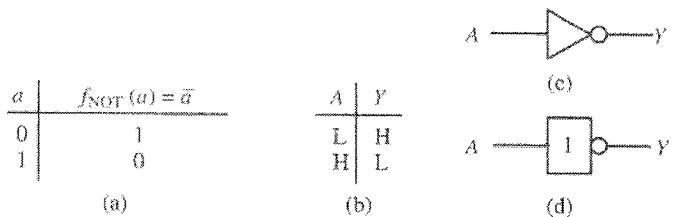


Figura 2.8 La función lógica NOT y la compuerta NOT. (a) La función lógica NOT. (b) La compuerta NOT electrónica. (c) Símbolo estándar. (d) Símbolo de bloque IEEE.

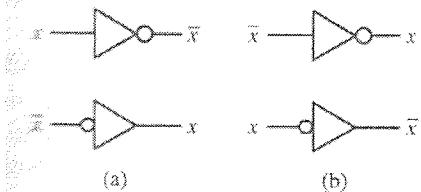


Figura 2.9 Concordancia de polaridad de la señal con las entradas y salidas de la compuerta NOT. (a) Uso preferido. (b) Uso inadecuado.

Podemos visualizar una compuerta NOT como un cambio de polaridad de una señal alta activa a baja activa, o viceversa. En consecuencia, podemos dibujar el símbolo de la compuerta NOT con la burbuja en la entrada o en la salida. Por convención, dibujamos la burbuja en la entrada de la compuerta cuando la señal de entrada es baja activa, y en la salida de la compuerta si la señal emitida es baja activa. La figura 2.9a muestra el uso recomendado del símbolo de compuerta NOT, haciendo concordar las burbujas con la señal baja activa, \bar{x} . Los diagramas de la figura 2.9b, aunque no son incorrectos, no se recomiendan.

Lógica positiva contra lógica negativa

Si utilizamos la convención de la lógica positiva para todas las entradas y salidas de las compuertas, es decir, si las señales conectadas a las entradas y salidas de la compuerta son todas altas activas, las funciones lógicas AND y OR se realizan mediante las compuertas AND y OR, respectivamente. Cuando las señales conectadas a las entradas y salidas de la compuerta son bajas activas, se invierten los papeles de estas compuertas.

Recuerde que en la convención de lógica negativa, el 1 se representa con un voltaje bajo y el 0 con un voltaje alto. Por tanto, podemos deducir la función realizada por una compuerta AND en el sistema de lógica negativa al sustituir 0 por *H* y 1 por *L* en la tabla de verdad de la compuerta AND de la figura 2.6b. La tabla resultante, que aparece en la figura 2.10a, es idéntica a la tabla de verdad del operador OR de la figura 2.7a. Así, podemos considerar que una compuerta AND con entradas y salidas bajas activas realiza la función lógica OR.

Podemos verificar esto con el álgebra de conmutación si aplicamos la involución (teorema 3) y el teorema de DeMorgan (teorema 8) a la expresión de la función lógica AND, como sigue:

$$\begin{aligned}
 y &= \overline{a \cdot b} \\
 &= \overline{\overline{a} \cdot \overline{b}} \\
 &= \overline{\overline{a} + \overline{b}} \\
 &= \bar{f}_{\text{OR}}(\bar{a}, \bar{b})
 \end{aligned} \tag{2.14}$$

La ecuación 2.14 indica que podemos dibujar un símbolo de compuerta AND como una función OR con entradas y salida bajas activas, como se muestra en la figura 2.10b. Aunque al principio parezca extraño, este símbolo ilustra la función

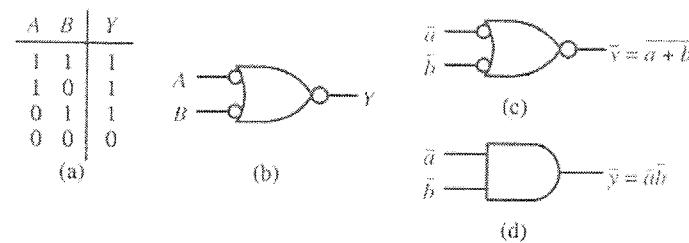


Figura 2.10 Uso de la compuerta AND en un sistema con lógica negativa. (a) Tabla de verdad de la compuerta AND ($L = 1$, $H = 0$). (b) Símbolo alternativo de la compuerta AND (lógica negativa). (c) Uso preferido. (d) Uso inadecuado.

realizada por una compuerta AND en un sistema con lógica negativa mejor que el símbolo estándar de la figura 2.6c.

Por ejemplo, consideremos la compuerta de la figura 2.10c. Las entradas de la compuerta están conectadas a señales bajas activas \bar{a} y \bar{b} y la salida a la señal baja activa \bar{y} . Formamos la expresión lógica para la salida \bar{y} tomando el complemento de cada una de las entradas, aplicando un OR y complementando el resultado, como sigue:

$$\begin{aligned}\bar{y} &= \overline{\overline{(\bar{a})} + \overline{(\bar{b})}} \\ &= \overline{\overline{a} + \overline{b}} \\ &= \bar{f}_{OR}(a, b)\end{aligned}\quad (2.15)$$

Por tanto, \bar{y} está afirmada (baja) cuando una o ambas entradas están afirmadas. Observe que la forma alternativa, que aparece en la figura 2.10d, no es incorrecta pero es más difícil de analizar y, por tanto, debemos evitarla al utilizar lógica negativa.

De manera similar, una compuerta OR realiza el operador lógico AND cuando sus entradas y salida son señales bajas activas. Podemos obtener la función realizada por una compuerta OR en un sistema con lógica negativa al sustituir 0 por H y 1 por L en la tabla de verdad de la compuerta OR de la figura 2.7b. La tabla resultante, que aparece en la figura 2.11a, es idéntica a la tabla de verdad del operador AND de la figura 2.6a. Por tanto, podemos pensar que una compuerta OR con entradas y salida bajas activas realiza la función lógica AND. Podemos verificar esto mediante el álgebra de conmutación, como hicimos antes para la compuerta AND.

$$\begin{aligned}y &= a + b \\ &= \overline{\overline{a} + \overline{b}} \\ &= \overline{\overline{a} \cdot \overline{b}} \\ &= \bar{f}_{AND}(\bar{a}, \bar{b})\end{aligned}\quad (2.16)$$

La ecuación 2.16 indica que podemos dibujar un símbolo de compuerta OR como una función AND con entradas y salida bajas activas, como se muestra en la figura 2.11b. Este símbolo alternativo ilustra la función realizada por una compuerta OR en un sistema con lógica negativa mejor que el símbolo estándar de la figura 2.7c.

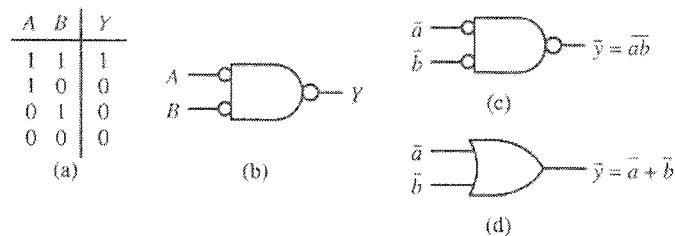


Figura 2.11 Uso de la compuerta OR en un sistema con lógica negativa.
 (a) Tabla de verdad de la compuerta OR ($L = 1$, $H = 0$). (b) Símbolo alternativo de la compuerta OR (lógica negativa). (c) Uso preferido. (d) Uso inadecuado.

Por ejemplo, consideremos la compuerta de la figura 2.11c. Si escribimos la expresión lógica para la salida baja activa \bar{y} en términos de las entradas bajas activas \bar{a} y \bar{b} obtenemos

$$\begin{aligned}\bar{y} &= \overline{(\bar{a})} \cdot \overline{(\bar{b})} \\ &= \overline{a \cdot b} \\ &= \bar{f}_{AND}(a, b)\end{aligned}\quad (2.17)$$

Por tanto, afirmamos \bar{y} sólo cuando \bar{a} y \bar{b} se afirman en forma simultánea. Esta operación es más difícil de ver si la compuerta se dibuja como en la figura 2.11d. En consecuencia, siempre debemos utilizar la forma de la figura 2.11c con la lógica negativa.

El siguiente ejemplo demuestra el uso de las compuertas AND y OR con los dispositivos que tienen entradas y salidas bajas activas.

EJEMPLO 2.32

Diseñar un circuito lógico que implante un sistema de alarma contra incendios en un edificio.

El edificio debe estar protegido por un sistema de alarma contra incendios que tenga dos detectores de humo, un aspersor y un marcador telefónico automático que llame a los bomberos. El aspersor se debe activar si cualquiera de los detectores detecta humo, y hay que llamar a los bomberos si ambos detectores detectan humo. Los detectores tienen salidas bajas activas, $\overline{D1}$ y $\overline{D2}$, que se afirman siempre que detecten partículas de humo. El aspersor tiene una entrada baja activa SPK que debe afirmarse para que el aspersor se active. De manera similar, el marcador telefónico inicia una llamada cuando su señal de entrada baja activa $DIAL$ se afirma.

Obtenemos las ecuaciones lógicas para el aspersor y el marcador telefónico determinando las condiciones que deben activar cada dispositivo. El aspersor debe activarse cuando cualquiera de las salidas de los detectores de humo quede afirmada. La operación deseada es $SPK = D1 + D2$. Puesto que estas señales sólo están disponibles en forma baja activa, escribimos

$$SPK = D1 + D2 \quad (2.18)$$

De manera similar, el marcador debe activarse cuando las salidas de los dos detectores de humo quedan afirmadas; así, $DIAL = D1 \cdot D2$. Puesto que estas señales sólo están disponibles en forma baja activa, escribimos

Las ecuaciones 2.18 y 2.19 se realizan mediante el diagrama lógico de la figura 2.12. Observe que la compuerta $G1$ es una compuerta AND utilizada para realizar la función OR de la ecuación 2.18, mientras que $G2$ es una compuerta OR utilizada para realizar la función AND de la ecuación 2.19.

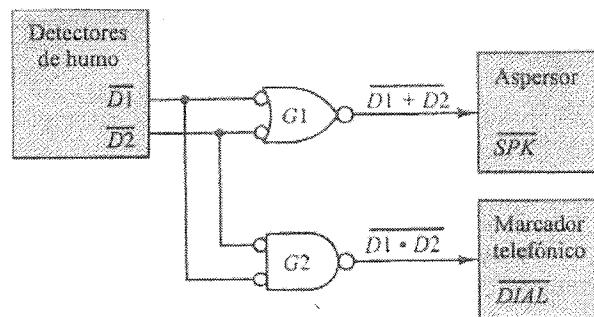


Figura 2.12 Sistema de alarma contra incendios, que ilustra la lógica negativa.

Las compuertas AND y OR se utilizan siempre que las entradas y salidas tienen la misma polaridad. Las dos compuertas que presentaremos enseguida, NAND y NOR, se utilizan en los sistemas con lógica mixta, es decir, cuando las entradas son altas activas y las salidas bajas activas, o viceversa.

NAND

La compuerta NAND es una combinación de una compuerta AND seguida de una compuerta NOT. Definimos la función NAND como

$$f_{NAND}(a, b) = \overline{ab} \quad (2.20)$$

De esta ecuación, queda claro que la compuerta NAND realiza la función lógica AND cuando sus señales de entrada son altas activas y su salida baja activa. Obtenemos las tablas de verdad para la función NAND y la compuerta NAND complementando las columnas de salida de las tablas de verdad para la función y la compuerta AND, respectivamente. Las tablas resultantes aparecen en las figuras 2.13a y b. La clave para entender la función NAND es observar que la salida es 0 si y sólo si sus entradas son simultáneamente 1.

Las figuras 2.13c, d y e muestran los símbolos estándar para la compuerta NAND. La burbuja en la terminal de salida de la figura 2.13c indica la operación NOT, estableciendo la diferencia respecto a la compuerta AND. Obtenemos la forma de la figura 2.13d aplicando el teorema de DeMorgan a la expresión de commutación de la función NAND de la ecuación 2.20:

$$f_{NAND}(a, b) = \overline{ab} = \bar{a} + \bar{b} \quad (2.21)$$

a	b	$f_{\text{NAND}}(a, b) = \bar{ab}$
0	0	1
0	1	1
1	0	1
1	1	0

(a)

A	B	Y
L	L	H
L	H	H
H	L	H
H	H	L

(b)

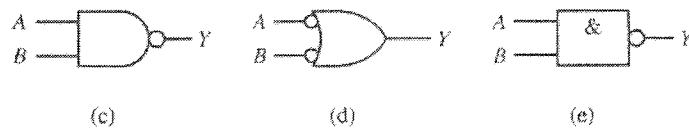


Figura 2.13 La función lógica NAND y la compuerta NAND. (a) La función lógica NAND. (b) Compuerta NAND electrónica. (c) Símbolo estándar. (d) Símbolo alternativo. (e) Símbolo de bloque IEEE.

Así, utilizamos una compuerta NAND para realizar la función OR cuando las señales de entrada son bajas activas y la salida es alta activa. Como explicamos en el caso de la compuerta NOT, las burbujas en el símbolo de la compuerta NAND siempre deben coincidir con las señales bajas activas. Así, utilizamos el símbolo de la figura 2.13c cuando la señal de salida es baja activa, y el de la figura 2.13d cuando las señales de entrada son bajas activas. Las figuras 2.14a y b muestran el uso correcto e incorrecto, respectivamente, de los dos símbolos de compuerta NAND.

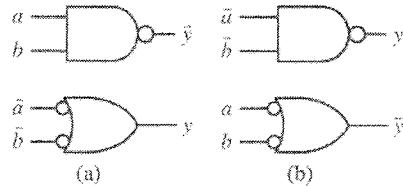


Figura 2.14 Concordancia de la polaridad de la señal con las entradas y salidas de la compuerta NAND. (a) Uso preferido. (b) Uso inadecuado.

A continuación mostramos otras propiedades interesantes de la compuerta NAND:

$$f_{\text{NAND}}(a, a) = \overline{a \cdot a} = \overline{a} = f_{\text{NOT}}(a)$$

$$\tilde{f}_{\text{NAND}}(a, b) = \overline{\overline{a} \cdot b} = a \cdot b = f_{\text{AND}}(a, b)$$

$$f_{\text{NAND}}(\tilde{a}, \tilde{b}) = \overline{\tilde{a} \cdot \tilde{b}} = a + b = f_{\text{OR}}(a, b)$$

Por tanto, una compuerta NAND con ambas entradas controladas por la misma señal equivale a una compuerta NOT; una compuerta NAND cuya salida se complementa equivale a una compuerta AND, y una compuerta NAND con entradas complementadas actúa como una compuerta OR.

Así, podemos utilizar las compuertas NAND para implantar los tres operadores elementales (AND, OR y NOT), como se muestra en la figura 2.15. En consecuencia, podemos construir cualquier función de conmutación, utilizando sólo compuertas NAND. Las compuertas con esta propiedad se llaman *primitivas* o *funcionalmente completas*.

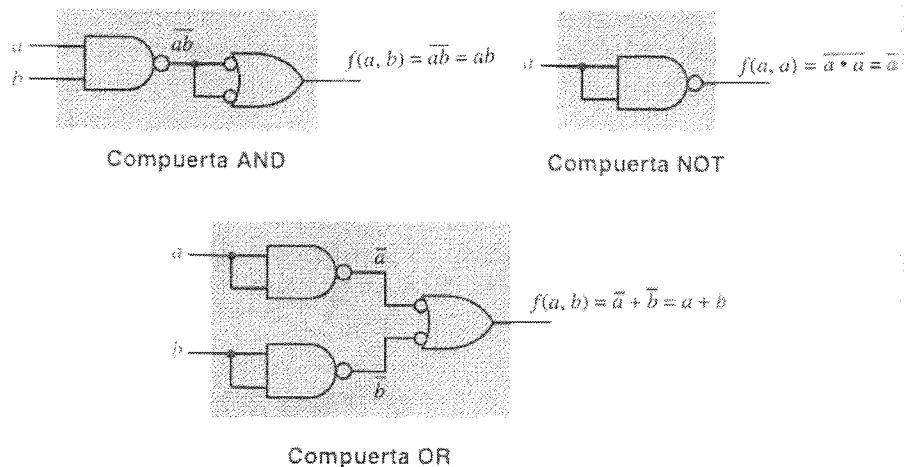


Figura 2.15 Compuertas AND, OR y NOT construidas exclusivamente mediante compuertas NAND.

NOR

La compuerta NOR es una combinación de compuerta OR seguida de una compuerta NOT, lo que representa la función

$$f_{\text{NOR}}(a, b) = \overline{a + b} \quad (2.22)$$

La compuerta NOR realiza la función lógica OR con entradas altas activas y una salida baja activa. Por tanto, la tabla de verdad para la función NOR y la compuerta NOR se obtienen complementando las columnas de salida de las tablas de verdad de la función OR y la compuerta OR, respectivamente. Las tablas resultantes aparecen en las figuras 2.16a y b. La clave para recordar la función de una compuerta NOR es la primera fila de la tabla de verdad; la salida de una compuerta NOR es 1 si y sólo si ambas entradas son simultáneamente 0.

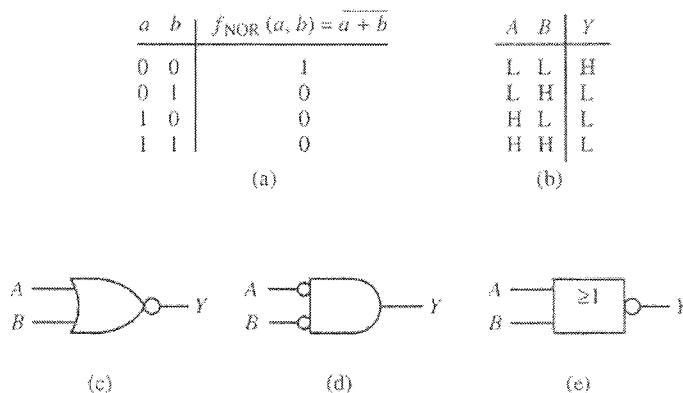


Figura 2.16 La función lógica NOR y la compuerta NOR. (a) La función lógica NOR. (b) Compuerta NOR electrónica. (c) Símbolo estándar. (d) Símbolo alternativo. (e) Símbolo de bloque IEEE.

Los símbolos estándar para la compuerta NOR se presentan en las figuras 2.16c, d y e. La burbuja en la terminal de salida indica la operación NOT, lo que establece su diferencia con la compuerta OR. Obtenemos la forma de la figura 2.16d aplicando el teorema de DeMorgan a la definición de la función NOR definida mediante la ecuación 2.22:

$$f_{\text{NOR}}(a, b) = \overline{a + b} = \bar{a} \cdot \bar{b} \quad (2.23)$$

Así, podemos utilizar una compuerta NOR para realizar la función AND con entradas bajas activas y una salida alta activa. Como en el caso de la compuerta NAND, utilizamos el símbolo de la figura 2.16c cuando la señal de salida es baja activa, y el de la figura 2.16d, cuando las señales de entrada son bajas activas. Las figuras 2.17a y b ilustran el uso correcto e incorrecto de los símbolos para la compuerta NOR, respectivamente.

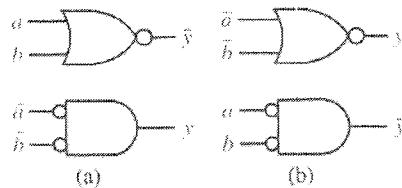


Figura 2.17 Concordancia de la polaridad de la señal con las entradas y salidas de la compuerta NOR. (a) Uso preferido. (b) Uso inadecuado.

Como en el caso de las compuertas NAND, las compuertas NOR también son elementos primitivos, en el sentido de que pueden servir para generar las operaciones AND, OR y NOT, como mostramos a continuación.

$$\begin{aligned} f_{\text{NOR}}(a, a) &= \overline{a + a} = \bar{a} &= f_{\text{NOT}}(a) \\ f_{\text{NOR}}(a, b) &= \overline{a + b} = a + b = f_{\text{OR}}(a, b) \\ f_{\text{NOR}}(\bar{a}, \bar{b}) &= \overline{\bar{a} + \bar{b}} = a \cdot b = f_{\text{AND}}(a, b) \end{aligned}$$

La figura 2.18 presenta estas tres operaciones en forma simbólica.

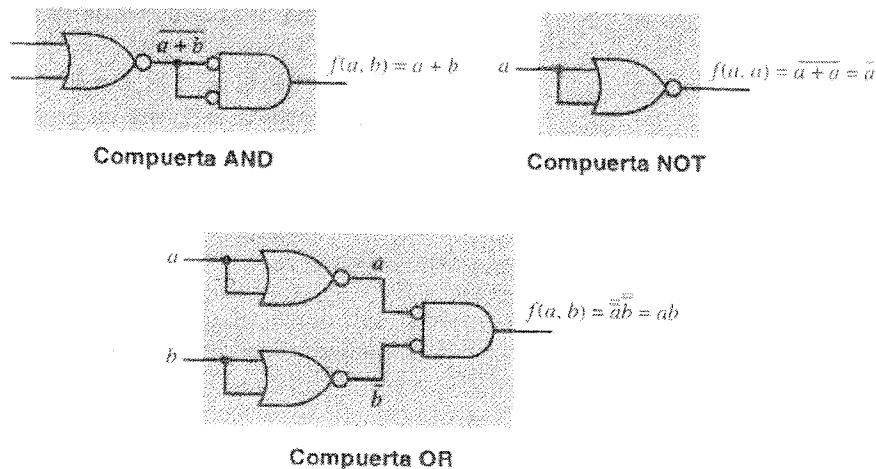


Figura 2.18 Compuertas AND, OR y NOT construidas exclusivamente mediante compuertas NOR.

Como ambas compuertas, NAND y NOR, son funcionalmente completas, resultan valiosas porque se puede implantar un diseño completo usando un único tipo de elemento. Es más fácil construir un circuito integrado si todas sus compuertas son NAND (o todas NOR) en vez de combinar compuertas AND, OR y NOT. Además, los circuitos con compuertas electrónicas NAND y NOR son por lo general más rápidos y fáciles de fabricar que las compuertas AND y OR equivalentes y, por tanto, tienen un efecto importante sobre el costo.

OR exclusivo (XOR)

La operación de OR exclusivo (o XOR) se define de manera funcional como

$$f_{XOR}(a, b) = a \oplus b = ab + \bar{a}\bar{b} \quad (2.24)$$

La tabla de verdad que se obtiene de la ecuación 2.24 aparece en la figura 2.19a. La tabla de verdad correspondiente para la compuerta XOR aparece en la figura 2.19b, y los símbolos lógicos estándar, en las figuras 2.19c y d.

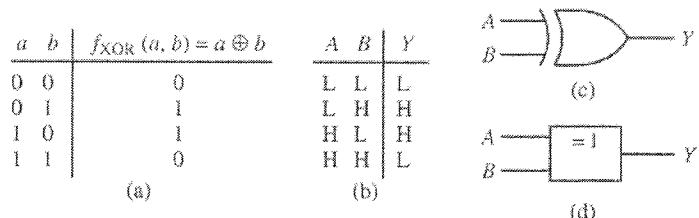


Figura 2.19 La función lógica OR exclusivo (XOR) y la compuerta XOR. (a) La función lógica XOR. (b) Compuerta XOR electrónica. (c) Símbolo estándar. (d) Símbolo de bloque IEEE.

La salida de la compuerta OR exclusivo es 1 si y sólo si sus entradas no son iguales en forma simultánea. En otras palabras, cuando las entradas son diferentes, la salida es 1. El OR exclusivo se llama así debido a su relación con la compuerta OR. Las dos difieren en la combinación de entradas $a = 1, b = 1$. El OR exclusivo excluye esta combinación, dando como salida 0, mientras que la compuerta OR la incluye, por lo que se le llama también OR-inclusivo.

Obtenemos la forma de producto de sumas para el OR-exclusivo de la forma de suma de productos como sigue:

$$\begin{aligned} a \oplus b &= \bar{a}b + a\bar{b} \\ &= \bar{a}a + \bar{a}b + a\bar{b} + b\bar{b} \quad [\text{P2(a), P6(b)}] \\ &= \bar{a}(a + b) + \bar{b}(a + b) \quad [\text{P5(b)}] \\ &= (\bar{a} + \bar{b})(a + b) \quad [\text{P5(b)}] \end{aligned}$$

Las siguientes son otras relaciones útiles relativas al OR-exclusivo:

$$a \oplus a = 0 \quad (2.25)$$

$$a \oplus \bar{a} = 1 \quad (2.26)$$

$$a \oplus 0 = a \quad (2.27)$$

$$a \oplus 1 = \bar{a} \quad (2.28)$$

$$\hat{a} \oplus \hat{b} = a \oplus b \quad (2.29)$$

$$a \oplus b = b \oplus a \quad (2.30)$$

$$a \oplus (b \oplus c) = (a \oplus b) \oplus c \quad (2.31)$$

El lector puede verificar la validez de estas relaciones construyendo sus tablas de verdad.

El símbolo de bloque estándar IEEE para la compuerta de OR-exclusivo indica que la salida se afirma cuando la suma matemática de las entradas es igual a 1:

$a \ b$	$\text{sum}(a, b)$	$\text{sum}(a, b) = 1?$	$f(a, b) = a \oplus b$
0 0	0	Falso	0
0 1	1	Verdadero	1
1 0	1	Verdadero	1
1 1	2	Falso	0

Podemos ver de esta tabla que la salida de una compuerta OR-exclusivo es la suma módulo 2 de sus entradas. Por esta razón, las compuertas OR-exclusivo se utilizan con frecuencia en el diseño de circuitos aritméticos que realizan sumas y restas binarias. En el capítulo 4 analizaremos esto con mayor detalle.

NOR-exclusivo (XNOR)

Una función común relacionada con el OR-exclusivo es la operación de coincidencia, o NOR-exclusivo (XNOR), que es sencillamente el complemento del OR-exclusivo. Definimos esta función como:

$$f_{\text{XNOR}}(a, b) = \overline{a \oplus b} = a \odot b \quad (2.32)$$

Las tablas de verdad y los símbolos lógicos de la compuerta XNOR aparecen en

a	b	$f_{\text{XNOR}}(a, b) = a \oplus b$	A	B	Y
0	0	1	L	L	H
0	1	0	L	H	L
1	0	0	H	L	L
1	1	1	H	H	H

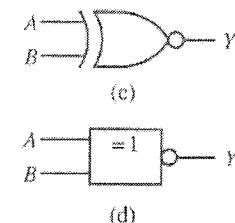


Figura 2.20 La función lógica NOR exclusivo (XNOR) y la compuerta XNOR. (a) La función lógica XNOR. (b) Compuerta XNOR electrónica. (c) Simbolo estándar. (d) Simbolo de bloque IEEE.

la figura 2.20. Las formas de suma de productos y producto de sumas de la operación de coincidencia son las siguientes:

$$\begin{aligned} a \odot b &= \overline{a \oplus b} \\ &= \overline{ab} + \overline{ab} \\ &= \overline{ab} \cdot \overline{ab} \end{aligned} \quad \begin{matrix} [\text{P2}] \\ [\text{T8(a)}] \end{matrix}$$

Capítulo 2. M étodos algebraicos para el an lisis y sntesis de circuitos l gicos

$$\begin{aligned} &= (a + \bar{b})(\bar{a} + b) && [\text{T8(b)}] \\ &= a\bar{a} + ab + \bar{a}\bar{b} + \bar{b}b && [\text{P5(b)}] \\ &= ab + \bar{a}\bar{b} && [\text{P6(b), P2(a)}] \end{aligned}$$

Tambi n se puede verificar f cilmente que

$$a \oplus \bar{b} = a \odot b \quad (2.33)$$

2.4 An lisis de circuitos combinatorios

Los circuitos digitales se dise an transformando una descripci n verbal de una funci n en un conjunto de ecuaciones de conmutaci n y realizando despu s las ecuaciones mediante compuertas, dispositivos l gicos programables (PLD) u otros elementos l gicos. El an lisis de los circuitos digitales es el problema inverso. Partimos de una realizaci n en hardware de un circuito digital y obtenemos una descripci n del circuito en forma de expresiones de conmutaci n, tablas de verdad, diagramas de tiempos, u otras descripciones de su comportamiento. Utilizamos el an lisis para determinar el comportamiento de un circuito l gico, para verificar que el comportamiento de un circuito corresponda con sus especificaciones, o como apoyo para convertir el circuito a una forma diferente, ya sea con el fin de reducir el n mero de compuertas o para realizarlo con diferentes elementos.

En este cap tulo presentamos el an lisis y sntesis de los circuitos digitales, incluidos el dise o y uso de m dulos de bloques de construcci n que sirven para implantar dise os de mayor tama o. En el cap tulo 4 analizaremos m dulos m s complejos, con integraci n a mediana escala (MSI). Estos m dulos son dispositivos de mayor nivel que contienen de 100 a 1000 transistores. En el cap tulo 5 examinaremos el uso de los dispositivos l gicos programables para dise ar circuitos digitales.

2.4.1 M etodo algebraico

Podemos construir redes l gicas interconectando las compuertas presentadas en la secci n anterior. Utilizamos estos circuitos para realizar funciones especficas dentro de un sistema digital de c mputo. Cualquier red de conmutaci n se puede representar por completo mediante una expresi n o funci n de conmutaci n, y con ello podemos aplicar toda la capacidad del l gbra de conmutaci n para llevar la funci n de conmutaci n a una forma deseada.

Hay que recordar que todas las expresiones de conmutaci n se pueden escribir en t rmicos de las operaciones AND, OR y NOT. Por tanto, podemos construir cualquier red de conmutaci n utilizando s olo elementos primitivos, como las compuertas NAND (o las NOR), como se muestra en la figura 2.15 (o la figura 2.18).

EMPLO 2.33

Determinar una expresi n de conmutaci n simplificada y la red l gica para el circuito l gico de la figura 2.21a.

Escribimos una expresi n de conmutaci n para la salida de cada compuerta.

$$\begin{aligned} P_1 &= \overline{ab} \\ P_2 &= \overline{\overline{a} + c} \end{aligned}$$

$$\begin{aligned}P_3 &= b \oplus \bar{c} \\P_4 &= P_1 \cdot P_2 = \overline{ab} (\bar{a} + c)\end{aligned}$$

La salida es

$$\begin{aligned}f(a, b, c) &= \overline{P_3 + P_4} \\&= \overline{(b \oplus \bar{c}) + \overline{ab} (\bar{a} + c)}\end{aligned}$$

Para analizar esta función, podemos llevarla a una forma más sencilla empleando el álgebra de conmutación:

$$\begin{aligned}\bar{f}(a, b, c) &= (b \oplus \bar{c}) + \overline{ab} \overline{\bar{a} + c} \\&= bc + \bar{b}\bar{c} + \overline{ab} \overline{\bar{a} + c} \quad [\text{Ec. 2.24}] \\&= bc + \bar{b}\bar{c} + (\bar{a} + \bar{b})a\bar{c} \quad [\text{T8}] \\&= bc + \bar{b}\bar{c} + ab\bar{c} \quad [\text{T5(b)}] \\&= bc + \bar{b}\bar{c} \quad [\text{T4(a)}] \\f(a, b, c) &= b \odot c \quad [\text{Ec. 2.32}]\end{aligned}$$

Por tanto, de la ecuación 2.32,

$$f(a, b, c) = \overline{b \odot c} = b \oplus c$$

Hemos reducido esta función a una compuerta OR exclusivo, que se muestra en la figura 2.21b. Las dos redes de conmutación de la figura 2.21 tienen la misma

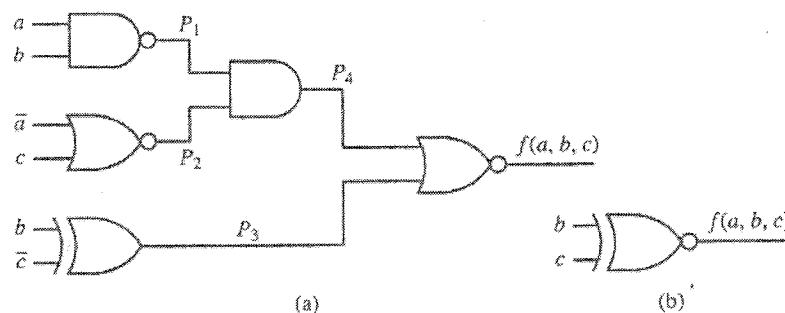


Figura 2.21 Redes de conmutación equivalentes.

tabla de verdad y, por ende, son equivalentes. Queda claro que la figura 2.21b es más recomendable, pues es menos compleja.

EJEMPLO 2.34

Determinar una expresión de conmutación y un circuito simplificado para la red de la figura 2.22.

La expresión lógica para cada salida de la compuerta aparece en el diagrama lógico de la figura 2.22. A partir de ellas, podemos obtener la expresión de salida como

$$\begin{aligned}f(a, b, c) &= \overline{(a \oplus b)(b \oplus c) \cdot (\overline{\bar{a} + \bar{b}} + a + c)} \\&= \overline{(a \oplus b)(b \oplus c)} + \overline{\bar{a} + \bar{b}} + a + c \quad [\text{T8(b)}]\end{aligned}$$

$$\begin{aligned}
 &= (a \oplus b)(b \oplus c) + (\bar{a} + \bar{b})(a + c) && [\text{T8(a)}] \\
 &= (a\bar{b} + \bar{a}b)(b\bar{c} + \bar{b}c) + (\bar{a} + \bar{b})(a + c) && [\text{Ec. 2.24}] \\
 &= a\bar{b}\bar{c} + a\bar{b}c + \bar{a}b\bar{c} + \bar{a}bc + \bar{a}a + \bar{a}c + a\bar{b} + \bar{b}c && [\text{P5(b)}] \\
 &= \bar{a}\bar{b}\bar{c} + \bar{a}\bar{b}c + \bar{a}c + a\bar{b} + \bar{b}c && [\text{P6(b), T4(a)}] \\
 &= \bar{a}\bar{b}\bar{c} + \bar{a}c + a\bar{b} + \bar{b}c && [\text{T4(a)}] \\
 &= \bar{a}\bar{b}\bar{c} + \bar{a}c + a\bar{b} && [\text{T9(a)}] \\
 &= \bar{a}\bar{b} + \bar{a}c + a\bar{b} && [\text{T7(a)}] \\
 &= \bar{a}c + a \oplus b && [\text{Ec. 2.24}]
 \end{aligned}$$

Observe que ésta es la forma de la red de conmutación de la figura 2.22b.

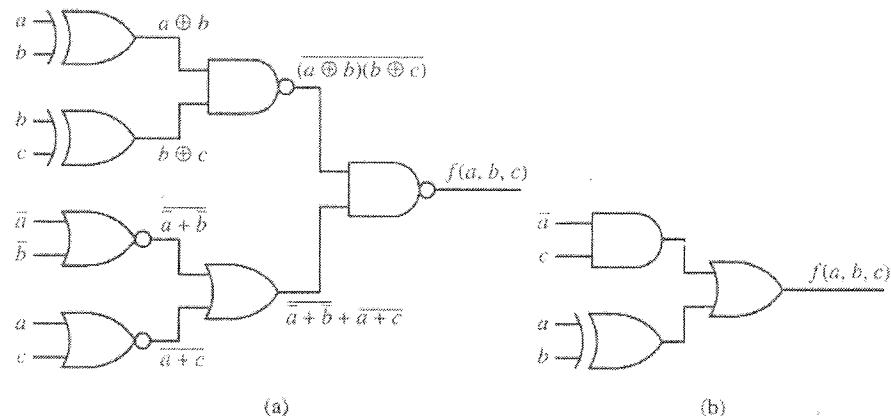


Figura 2.22 Redes de conmutación equivalentes.

Método de la tabla de verdad

Ya hemos obtenido la tabla de verdad para una función a partir de una expresión de conmutación evaluando la expresión, parte por parte. Es posible seguir el mismo método con los diagramas lógicos, obteniendo la tabla de verdad compuesta por compuerta.

Podemos obtener la tabla de verdad para la función del último ejemplo utilizando la técnica del álgebra de conmutación ya descrita:

abc	$\bar{a}c$	$a \oplus b$	$f(a, b, c)$
0 0 0	0	0	0
0 0 1	1	0	1
0 1 0	0	1	1
0 1 1	1	1	1
1 0 0	0	1	1
1 0 1	0	1	1
1 1 0	0	0	0
1 1 1	0	0	0

La columna $\bar{a}c$ es 1 siempre que $\bar{a}=1$ y $c=1$, o siempre que $a=0$ y $c=1$. La columna $a \oplus b$ es 1 siempre que $a \neq b$. Obtenemos el OR de estas dos columnas para crear $f(a, b, c)$; por tanto, $f(a, b, c)$ vale 0 si $\bar{a}c$ y $a \oplus b$ son ambos 0.

En esta sección consideramos una red de conmutación, la analizamos escribiendo la función de conmutación, la simplificamos mediante el álgebra de conmutación, y obtuvimos una red equivalente, pero menos compleja.

2.4.2 Análisis de diagramas de tiempos

Hasta ahora, hemos analizado los diagramas de circuitos lógicos obteniendo expresiones o tablas de verdad para las funciones lógicas realizadas por los circuitos. Otro método de análisis consiste en aplicar una serie de valores a las entradas de un circuito en cierto lapso, ya sea en forma experimental o con un programa de simulación lógica, y tomar nota de la relación entre las entradas y la serie correspondiente de salidas en forma de un diagrama de tiempos. De este diagrama, podemos deducir la función lógica realizada por el circuito y estudiar los efectos de los retardos de propagación de las compuertas sobre el funcionamiento del circuito.

Diagramas de tiempos

Un *diagrama de tiempos* es una representación gráfica de las relaciones entre las señales de entrada y salida de una red de conmutación, como podrían verse en la pantalla de un osciloscopio o analizador lógico o en un programa de simulación lógica. Con frecuencia, también se muestran las señales intermedias. Además, los diagramas de tiempos pueden mostrar los retardos por propagación introducidos por los dispositivos de conmutación cuando las señales se propagan a través de la red. Un diagrama de tiempos adecuado puede mostrar toda la información contenida en la tabla de verdad, como muestra el ejemplo siguiente.

Ejemplo 2.35

El circuito de la figura 2.23a se estimula con una serie de entradas, produciendo el diagrama de tiempos de la figura 2.23b. En este ejemplo, 1 se representa con una señal alta y 0 con una baja.

Determinar la tabla de verdad y las listas de mintérminos para las dos funciones $f_a(A, B, C)$ y $f_b(A, B, C)$ realizadas por este circuito.

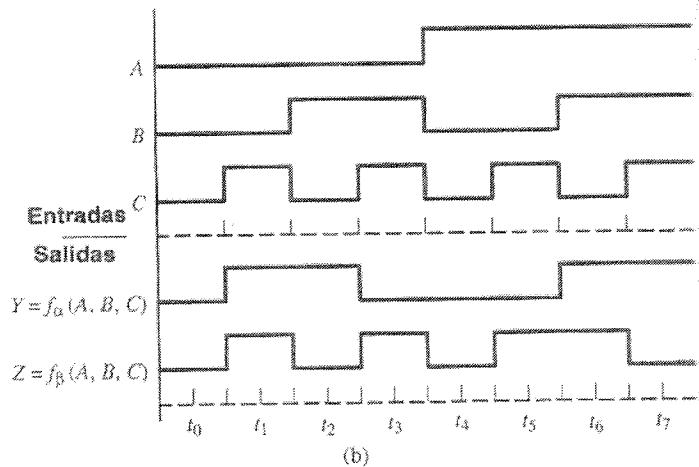
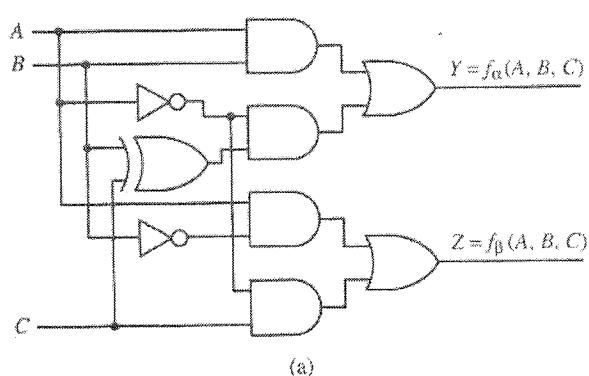
Las señales de entrada y salida se indican sobre el diagrama del circuito. Hemos seleccionado los patrones de entrada de modo que en una unidad de tiempo aparezcan todas las combinaciones posibles de las entradas A , B y C .

Al analizar el diagrama de tiempos en los instantes t_0, t_1, \dots, t_7 , determinamos los valores de las entradas y las salidas en cada instante y los escribimos en forma de tabla de verdad, como en la figura 2.23c. A partir de esta tabla de verdad, podemos escribir la lista de mintérminos y después obtener una expresión lógica simplificada para cada función, de la manera siguiente:

$$\begin{aligned}f_a(A, B, C) &= \sum m(1, 2, 6, 7) \\&= \bar{A}\bar{B}C + \bar{A}B\bar{C} + AB\bar{C} + ABC \\&= \bar{A}\bar{B}C + B\bar{C} + AB\end{aligned}$$

$$\begin{aligned}
 f_B(A, B, C) &= \sum m(1, 3, 5, 6) \\
 &= \bar{A}\bar{B}C + \bar{A}BC + A\bar{B}C + ABC \\
 &= \bar{A}C + A\bar{B}C + AB\bar{C}
 \end{aligned}$$

En el ejemplo anterior, mostramos todas las salidas de las compuertas como si cambiaron en forma instantánea después de un cambio en la entrada. En realidad, siempre hay un retardo entre el momento de un cambio de entrada y el cambio correspondiente en la salida de un circuito. En la siguiente sección consideraremos los efectos de los retardos por propagación de las compuertas sobre el funcionamiento del circuito.



Instante	Entradas ABC	Salidas	
		$f_A(A, B, C)$	$f_B(A, B, C)$
t_0	0 0 0	0	0
t_1	0 0 1	1	1
t_2	0 1 0	1	0
t_3	0 1 1	0	1
t_4	1 0 0	0	0
t_5	1 0 1	0	1
t_6	1 1 0	1	1
t_7	1 1 1	1	0

(c)

Figura 2.23 Obtención de una tabla de verdad a partir de un diagrama de tiempos. (a) Circuito lógico que realiza dos funciones. (b) Diagrama de tiempos. (c) Tabla de verdad.

Retardo por propagación

Además de la función lógica, el diseñador debe preocuparse por varias características físicas de los circuitos lógicos digitales, como las siguientes:

- Retardos por propagación
- Restricciones de *fan-in* y *fan-out* de las compuertas
- Consumo de energía
- Tamaño y peso

Éstas son propiedades físicas de los circuitos de bajo nivel utilizados para crear las compuertas lógicas, y dependen del número y configuración de las compuertas en un circuito dado. El diseño de bajo nivel no es un objetivo de este libro; el lector puede consultar Wakerly [9]. Sin embargo, los retardos por propagación y las restricciones de *fan-in* y *fan-out* tienen un efecto importante sobre el diseño lógico, por lo que deben tomarse en cuenta durante cualquier análisis o diseño de cualquier circuito lógico digital.

Una compuerta lógica física requiere una cantidad no nula de tiempo para reaccionar a los cambios de entrada y producir modificaciones en su estado de salida. Este retardo entre el instante de cambio de la entrada y el cambio correspondiente en la salida es el *retardo por propagación*. Así, si un circuito lógico realiza una función $z = f(x_1, \dots, x_n)$, el retardo por propagación es el tiempo que los cambios tardan en “propagarse” desde alguna entrada x_i a través del circuito hasta la salida z . Los retardos por propagación dependen de la complejidad del circuito, de la tecnología electrónica utilizada, y de factores como el *fan-out* (el número de entradas de otras compuertas controladas por una única salida de la compuerta), la temperatura y el voltaje del circuito.

Después de varios cambios de entrada, las salidas de los circuitos de compuertas lógicas electrónicas pueden tardar un tiempo distinto en cambiar de bajo a alto que de alto a bajo. Por tanto, generalmente especificamos dos parámetros de retardo por propagación para una compuerta lógica dada:

t_{PLH} = tiempo de retardo por propagación, con salida de bajo a alto nivel

t_{PHL} = tiempo de retardo por propagación, con salida de alto a bajo nivel

donde t_{PLH} y t_{PHL} se miden a partir del momento de cambio en la entrada hasta el momento del cambio correspondiente en la salida.

Si no se necesita información precisa de tiempos, utilizamos un único parámetro de retardo por propagación, denotado por t_{PD} , como aproximación de t_{PLH} y t_{PHL} . Por lo general, t_{PD} se calcula como la media de t_{PLH} y t_{PHL} :

$$t_{PD} = \frac{t_{PLH} + t_{PHL}}{2}$$

Para la compuerta AND de la figura 2.24a, las figuras 2.24b-d ilustran la respuesta de la salida de la compuerta a una serie de cambios en sus valores de entrada. En la figura 2.24b se muestra el caso ideal, en el que las salidas tienen un cambio instantáneo; es decir, el retardo por propagación es nulo. En la figura 2.24c se muestran todos los cambios de salida retrasados por un retardo de propagación medio t_{PD} . La figura 2.24d presenta una imagen más precisa de los tiempos, con parámetros distintos para t_{PLH} y t_{PHL} .

La tabla 2.7 enumera los valores de t_{PD} para compuertas NAND con dos entradas de varias familias lógicas diferentes, junto con la disipación de energía por compuerta para cada una.

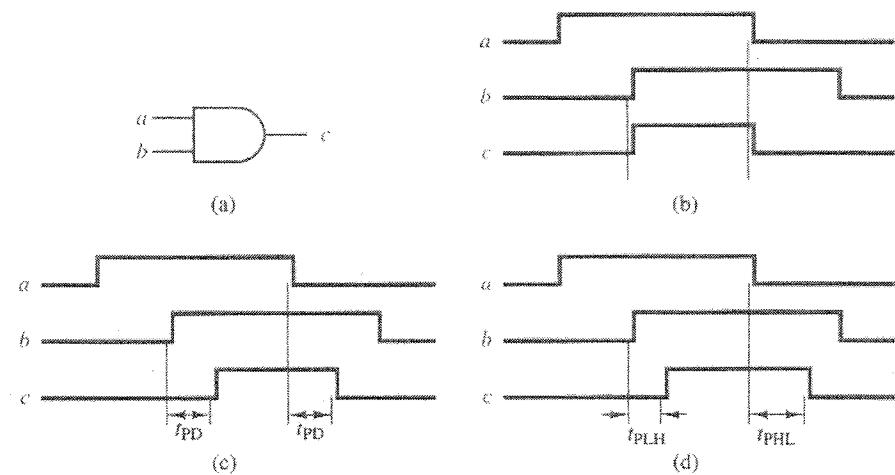


Figura 2.24 Retardo por propagación a través de una compuerta lógica. (a) Compuerta AND con dos entradas. (b) Retardo ideal (nulo). (c) $t_{PD} = t_{PLH} = t_{PHL}$. (d) $t_{PLH} < t_{PHL}$.

Se dispone de compuertas lógicas con varios tipos de tecnología, ocho de las cuales se muestran en la tabla. Como puede verse, existe cierto juego entre la velocidad y la disipación de energía en las diversas tecnologías. Por ejemplo, los dispositivos 74Sxx son más rápidos que los dispositivos equivalentes 74xx o 74LSxx, mientras que los dispositivos 74LSxx son más lentos, pero consumen menos energía. Con frecuencia, un diseñador sacrifica la velocidad en aras de un menor consumo de energía en aplicaciones en las que la corriente de la fuente de potencia es limitada, como es el caso de los sistemas energizados por baterías.

TABLA 2.7 DISIPACIÓN DE ENERGÍA Y RETARDOS POR PROPAGACIÓN PARA VARIAS FAMILIAS LÓGICAS [8]

Familia lógica	Retardo por propagación	Disipación de energía por compuerta (mW)	Tecnología
7400	10	10	TTL estándar
74H00	6	22	TTL de alta velocidad
74L00	33	1	TTL de baja potencia
74LS00	9.5	2	TTL Schottky de baja potencia
74S00	3	19	TTL Schottky
74ALS00	3.5	1.3	TTL Schottky avanzado de baja potencia
74AS00	3	8	TTL Schottky avanzado
74HC00	8	0.17	CMOS de alta velocidad

Los retardos por propagación también difieren entre las diferentes compuertas implantadas con la misma tecnología, ya que sus circuitos a nivel de transistores son diferentes. La tabla 2.8 enumera los parámetros t_{PHL} y t_{PLH} para cinco compuertas primitivas (AND, OR, NAND, NOR y NOT) de la familia

74LS. Observe que damos un valor típico y un valor máximo para cada parámetro. Los retardos por propagación varían de un dispositivo a otro, y son afectados por la cantidad de carga controlada por la compuerta. Por ello, la mayor parte de las hojas de datos de los dispositivos especifica un tiempo máximo de retardo, correspondiente a las peores condiciones de carga, además del tiempo típico de retardo para cada dispositivo.

TABLA 2.8 RETARDO POR PROPAGACIÓN DE COMPUERTAS PRIMITIVAS DE LA SERIE 74LS [8]

Círculo	Función	t_{PLH}		t_{PHL}	
		Típico	Máximo	Típico	Máximo
74LS04	NOT	9	15	10	15
74LS00	NAND	9	15	10	15
74LS02	NOR	10	15	10	15
74LS08	AND	8	15	10	20
74LS32	OR	14	22	14	22

2.36

Se aplica una serie de entradas al circuito de la figura 2.25a, con lo cual se obtiene el diagrama de tiempos de la figura 2.25b. Cada compuerta tiene un retardo por propagación t_{po} de una unidad de tiempo. Queremos determinar la tabla de verdad y la expresión de conmutación mínima para este circuito.

Partimos del diagrama de tiempos y obtenemos la tabla de verdad de la figura 2.25c examinando las salidas de cada compuerta después de cada uno de los cambios en la entrada. Como las señales tardan diferentes cantidades de tiempo en propagarse hasta la salida de la compuerta, debemos esperar hasta que la propagación de la señal termine antes de determinar la salida correspondiente a la entrada actual. Observe que la señal no se propaga a través de más de tres compuertas, por lo que no transcurren más de tres unidades de tiempo entre un cambio en la entrada y una salida estable.

Por ejemplo, en el instante t_1 , la entrada C cambia de 0 a 1, lo que hace que la salida D del inversor y la salida G de la compuerta AND cambien en el instante $t_1 + 1$. El cambio en G hace que la salida Y de la compuerta OR cambie de 0 a 1 en el instante $t_1 + 2$. Así, el cambio en la entrada necesitó dos unidades de tiempo para propagarse de la entrada del circuito C a la salida del circuito Y . Por tanto, debemos esperar un tiempo de $t_1 + 2$ para determinar el valor final de Y . Observe que el cambio de entrada en el instante t_2 también requirió dos unidades de tiempo para propagarse hasta la salida, mientras que los cambios en los instantes t_4 y t_7 requieren tres unidades de tiempo.

Utilizamos la tabla de verdad para escribir la lista de mintérminos y obtener una expresión mínima de conmutación de la manera siguiente:

$$\begin{aligned} f(A, B, C) &= \sum m(1, 4, 5, 6) \\ &= \bar{A}\bar{B}C + A\bar{B}\bar{C} + A\bar{B}C + AB\bar{C} \\ &= A\bar{C} + \bar{B}C \end{aligned}$$

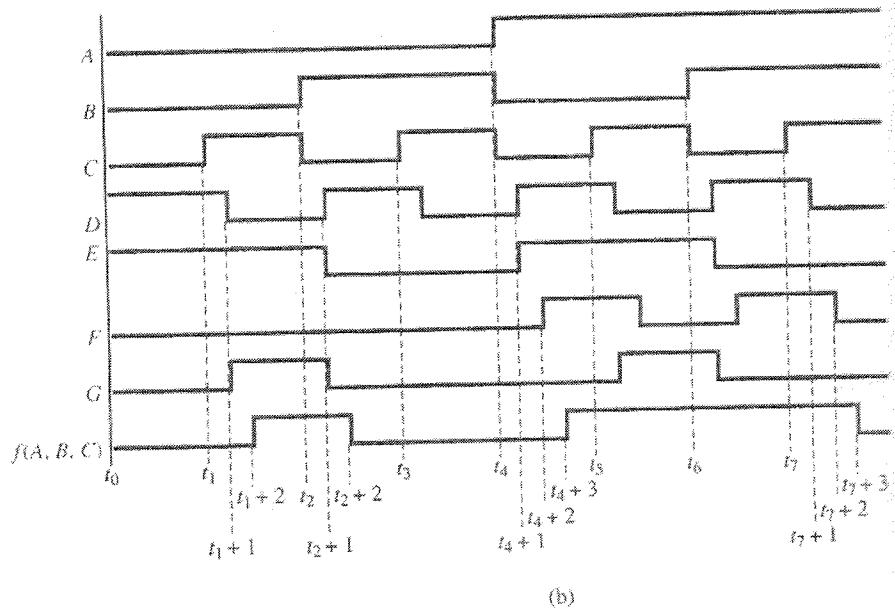
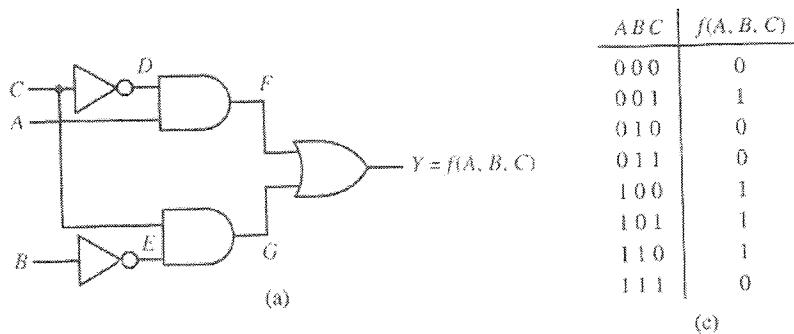


Figura 2.25 Obtención de una tabla de verdad a partir de un diagrama de tiempos.
 (a) Diagrama de circuitos. (b) Diagrama de tiempos. (c) Tabla de verdad.

2.5 Síntesis de circuitos lógicos combinatorios

Hasta ahora, hemos presentado varias herramientas que pueden servir para analizar y sintetizar redes de conmutación. Estas herramientas incluyen al álgebra de conmutación, los dispositivos de conmutación, las tablas de verdad y los diagramas de tiempos. En esta sección utilizaremos algunas de estas herramientas para el diseño e implantación de redes de conmutación.

2.5.1 Redes AND-OR y NAND

Una red AND-OR utiliza compuertas AND para formar términos producto y una compuerta OR para formar la suma de estos productos. Por tanto, una red

de conmutación que deba implantarse en lógica AND-OR debe expresarse en forma de suma de productos (SOP). Por ejemplo, podemos implantar la función

$$f_8(p, q, r, s) = p\bar{r} + qrs + \bar{p}s$$

de manera directa en la lógica AND-OR como se muestra en la figura 2.26a.

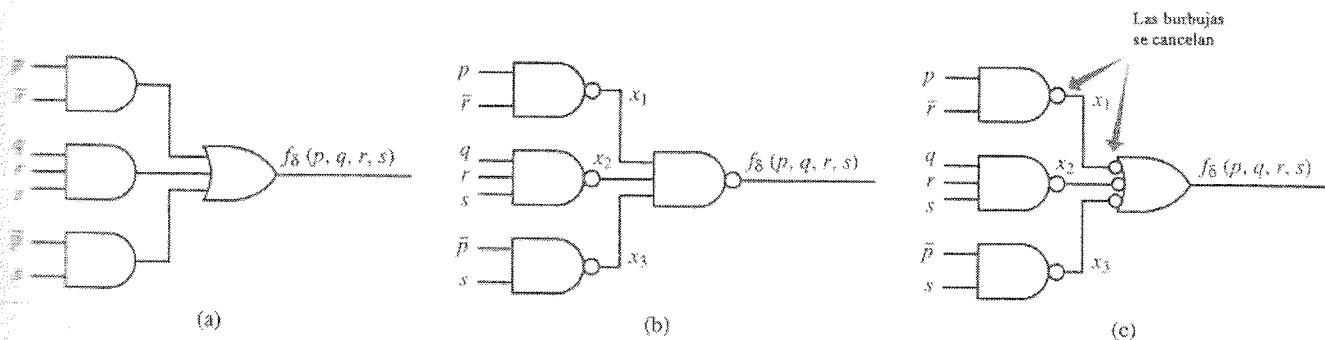


Figura 2.26 Implementaciones de $f_8(p, q, r, s) = p\bar{r} + qrs + \bar{p}s$. (a) Red AND-OR. (b) Red NAND. (c) Red NAND (forma preferida).

Podemos usar una traducción sencilla con álgebra de conmutación para transformar una expresión de suma de productos en una forma adecuada para la implantación NAND directa. Colocamos dos barras sobre toda la función SOP, y utilizamos el teorema de DeMorgan (teorema 8) para determinar la forma NAND de la función.

$$\begin{aligned} f_8(p, q, r, s) &= \overline{\overline{p\bar{r}} + qrs + \bar{p}s} && [\text{T3}] \\ &= \overline{\overline{p\bar{r}} \cdot \overline{qrs} \cdot \overline{\bar{p}s}} && [\text{T8(a)}] \\ &= \overline{x_1 \cdot x_2 \cdot x_3} \end{aligned}$$

donde $x_1 = p\bar{r}$, $x_2 = \overline{qrs}$, y $x_3 = \overline{\bar{p}s}$. En la figura 2.26b mostramos la realización NAND de la función.

La figura 2.26c presenta el mismo circuito, pero con la compuerta NAND de la salida en su forma equivalente de DeMorgan. Si escribimos la expresión lógica para la salida,

$$\begin{aligned} f_8(p, q, r, s) &= \bar{x}_1 + \bar{x}_2 + \bar{x}_3 \\ &= \overline{\overline{p\bar{r}}} + \overline{\overline{qrs}} + \overline{\overline{\bar{p}s}} \\ &= p\bar{r} + qrs + \bar{p}s \end{aligned}$$

Observe que las burbujas de inversión en ambos extremos de las líneas x_1 , x_2 , y x_3 se cancelan, lo que hace que el diagrama recuerde a la figura 2.26a y así ilustre de manera clara el hecho de que se está implantando una forma de suma de productos. Por ello, el formato de la figura 2.26c es preferible al de la figura 2.26b cuando se usa un circuito NAND.

Las técnicas aquí utilizadas pueden aplicarse a cualquier función de suma de productos para obtener una red AND-OR o NAND.

2.5.2 Redes OR-AND y NOR

Una red OR-AND utiliza compuertas OR para formar términos de suma y una compuerta AND para formar el producto de estas sumas. En consecuencia, una función de conmutación dada que vaya a implantarse en lógica OR-AND deberá expresarse en forma de producto de sumas (POS). Por ejemplo, la realización de la función

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

en lógica OR-AND es directa y aparece en la figura 2.27a.

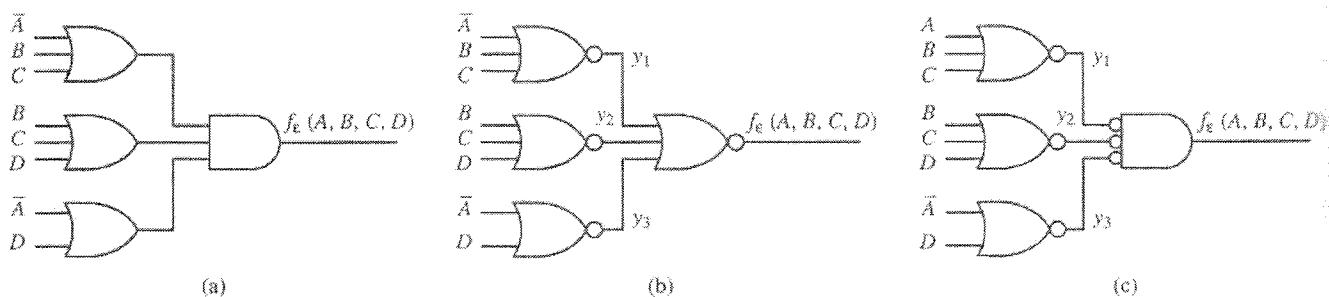


Figura 2.27 Implementaciones de $f_e(A, B, C, D) = (\bar{A} + B + C)(B + C + D)(\bar{A} + D)$.
(a) Red OR-AND. (b) Red NOR. (c) Red NOR (forma preferible).

Podemos aprovechar la misma transformación del álgebra de conmutación que utilizamos con anterioridad y expresar $f_e(A, B, C, D)$ en una forma adecuada para la implantación NOR directa, mediante el teorema de DeMorgan:

$$\begin{aligned} f_e(A, B, C, D) &= \overline{(\bar{A} + B + C)(B + C + D)(\bar{A} + D)} & [T3] \\ &= \overline{\bar{A} + B + C} \cdot \overline{B + C + D} \cdot \overline{\bar{A} + D} & [T8(b)] \\ &= \overline{y_1 + y_2 + y_3} \end{aligned}$$

donde $y_1 = \overline{\bar{A} + B + C}$, $y_2 = \overline{B + C + D}$, y $y_3 = \overline{\bar{A} + D}$. La realización NOR de esta función aparece en la figura 2.27b.

La figura 2.27c muestra el mismo circuito, pero con la compuerta NOR de salida en su forma equivalente de DeMorgan. Si escribimos la expresión lógica de salida, tenemos

$$\begin{aligned} f_e(A, B, C, D) &= \overline{y_1} \cdot \overline{y_2} \cdot \overline{y_3} \\ &= \overline{(A + B + C)} \cdot \overline{(B + C + D)} \cdot \overline{(\bar{A} + D)} \\ &= (\bar{A} + B + C)(B + C + D)(\bar{A} + D) \end{aligned}$$

Como en el caso de los circuitos NAND de dos niveles, las burbujas de inversión se cancelan. Este formato aclara que se está implantando una función en forma de producto de sumas y por ello es la forma preferible para trazar un circuito NOR de dos niveles.

Podemos generalizar el método anterior para implantar cualquier función de producto de sumas en lógica NOR.

2.5.3 Circuitos de dos niveles

Las redes con una estructura similar a la de las figuras 2.26 y 2.27 se conocen como *redes de dos niveles*. Las señales de entrada deben pasar a través de dos niveles de compuertas antes de llegar a la salida. Como muestra la figura 2.28a, definimos el primer nivel como el nivel que contiene la compuerta que produce la salida. Las compuertas que reciben las entradas del circuito están en el segundo nivel. Si se requieren compuertas NOT en las líneas de entrada, se tiene una red de tres niveles, como en la figura 2.28b. Una red tiene n niveles si al menos una señal de entrada debe pasar a través de n compuertas antes de llegar a la salida.

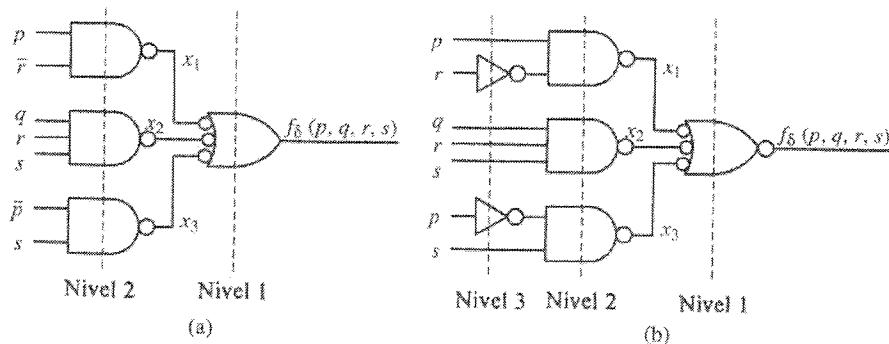


Figura 2.28 Estructuras de red de dos y tres niveles. (a) Red de dos niveles. (b) Red de tres niveles.

Las funciones de conmutación en la forma SOP o POS tienen una implantación directa en las redes de dos niveles si se dispone de las entradas en forma complementada y no complementada. Se necesita una red de tres niveles cuando sólo se dispone de una forma de las entradas. En este caso, sólo se necesitan compuertas NOT en el tercer nivel.

Con frecuencia hay que crear circuitos con más de dos niveles cuando existen límites para el *fan-in* de las compuertas. Por ejemplo, la función $f(a, b, c, d, e) = abcde$ se puede realizar con una única compuerta AND de cinco entradas, como se muestra en la figura 2.29a, pero si el diseñador tiene la restricción de trabajar sólo con compuertas AND de dos entradas, necesitará un circuito de tres o cuatro niveles, como los de las figuras 2.29b y c, respectivamente. El lector debe verificar que ambos circuitos son equivalentes.

Ahora, el lector dispone de todas las herramientas necesarias para tomar una función de conmutación expresada en forma de lista de mintérminos o maxtérminos e implantarla en lógica NAND y NOR, respectivamente. A continuación bosquejamos el proceso de implantación para la lógica NAND; los términos entre paréntesis se usarán si la implantación debe hacerse en lógica NOR.

Paso 1. Expresamos la función en forma de lista de mintérminos (maxtérminos).

Paso 2. Escribimos los mintérminos (maxtérminos) en forma algebraica.

Paso 3. Simplificamos la función en la forma de suma de productos (producto de sumas) mediante el álgebra de conmutación.

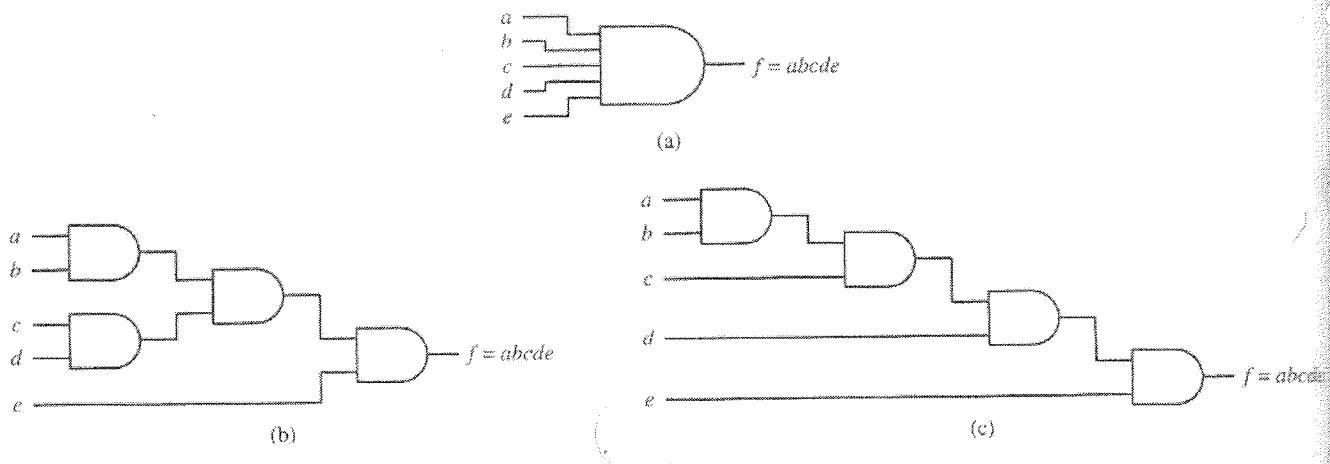


Figura 2.29 Circuito multinivel necesario debido a las restricciones de *fan-in*.
 (a) Una sola compuerta AND con cinco entradas. (b) Red de tres niveles con compuertas de dos entradas. (c) Red de cuatro niveles de compuertas de dos entradas.

Paso 4. Utilizamos el teorema 8a (8b) y el teorema 3 para transformar la expresión a la formulación NAND (NOR).

Paso 5. Trazamos el diagrama lógico NAND (NOR).

Ahora ilustraremos este proceso para la función $f_\phi(X, Y, Z) = \sum m(0, 3, 4, 5, 7)$

EJEMPLO 2.37

Implantar $f_\phi(X, Y, Z) = \sum m(0, 3, 4, 5, 7)$ en lógica NAND.

$$1. f_\phi(X, Y, Z) = \sum m(0, 3, 4, 5, 7)$$

$$2. f_\phi(X, Y, Z) = m_0 + m_3 + m_4 + m_5 + m_7 \\ = \bar{X}\bar{Y}\bar{Z} + \bar{X}YZ + X\bar{Y}\bar{Z} + X\bar{Y}Z + XYZ$$

$$3. f_\phi(X, Y, Z) = \bar{Y}\bar{Z} + YZ + XZ \quad [\text{T6(a)}]$$

$$4a. f_\phi(X, Y, Z) = \overline{\bar{Y}\bar{Z}} + \overline{YZ} + \overline{XZ} \quad [\text{T3}]$$

o bien

$$4b. f_\phi(X, Y, Z) = \overline{\overline{\bar{Y}\bar{Z}} + YZ + XZ} \quad [\text{T3}] \\ = \overline{\bar{Y}\bar{Z}} \cdot \overline{YZ} \cdot \overline{XZ} \quad [\text{T8(a)}]$$

Obtenemos el diagrama lógico de la figura 2.30a a partir de la expresión del paso 4a, y decimos que es una *realización SOP mínima de dos niveles* de la función de conmutación. Este ejemplo ilustra de manera completa el proceso de diseño.

EJEMPLO 2.38

Implantar $f_\phi(X, Y, Z) = \sum m(0, 3, 4, 5, 7)$ en lógica NOR.

$$1. f_\phi(X, Y, Z) = \prod M(1, 2, 6)$$

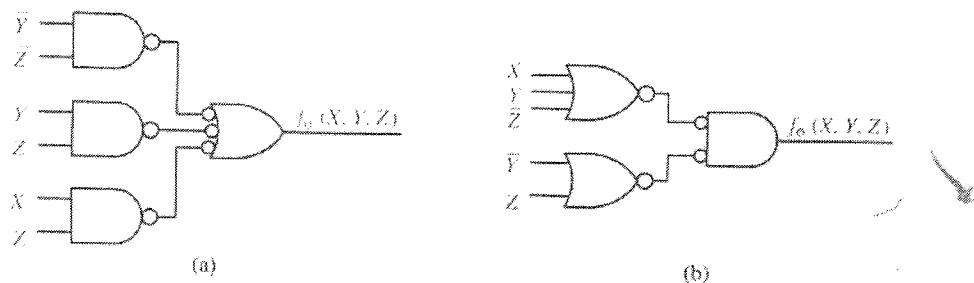


Figura 2.30 Forma canónica de $f_\phi(X, Y, Z) = \sum m(0, 3, 4, 5, 7)$ en redes NAND y NOR. (a) Realización NAND. (b) Realización NOR.

$$2. f_\phi(X, Y, Z) = M_1 \cdot M_2 \cdot M_3 \\ = (X + Y + \bar{Z})(X + \bar{Y} + Z)(\bar{X} + \bar{Y} + Z)$$

$$3. f_\phi(X, Y, Z) = (X + Y + \bar{Z})(\bar{Y} + Z) \quad [\text{T6(b)}]$$

$$4a. f_\phi(X, Y, Z) = \overline{(X + Y + Z)} \cdot \overline{(\bar{Y} + Z)} \quad [\text{T3}]$$

o

$$4b. f_\phi(X, Y, Z) = \overline{(X + Y + \bar{Z})(\bar{Y} + Z)} \quad [\text{T3}]$$

$$= \overline{(X + Y + \bar{Z})} + \overline{(\bar{Y} + Z)} \quad [\text{T8(b)}]$$

La red NOR obtenida del paso 4a. aparece en la figura 2.30b y es una *realización POS mínima de dos niveles* de la función de conmutación. Ambas redes de la figura 2.30 son implantaciones de la función $f_\phi(X, Y, Z)$.

2.5.4 Circuitos AND-OR-inversor

Un circuito AND-OR-inversor (AOI) está formado por un conjunto de compuertas AND, cuyas salidas se alimentan a una compuerta NOR y, por tanto, puede servir para realizar con facilidad los circuitos de suma de productos de dos niveles. En la figura 2.31 aparece una configuración típica, similar a la utilizada en la lógica estándar (7454) de la serie 7400. En general, definimos los circuitos según el número de entradas a las compuertas AND. Por ejemplo, un circuito con tres compuertas AND en el que una tiene dos entradas, otra tres entradas y la última cuatro, se designa como circuito AOI 2-3-4.

El circuito 7454 que aparece en la figura 2.31 es un circuito AOI 2-2-2-2 que realiza la función

F = AB + CD + EF + GH

Aunque podemos utilizar el circuito AOI de varias formas, sólo ilustraremos aquí una aplicación. Si B, D, F y H funcionan como líneas de activación (control) y A, C, E y G son líneas de información, podemos usar el circuito anterior para reunir la información de las cuatro fuentes A, C, E y G en un único canal (este circuito es un *multiplexor 4 a 1*, y se describirá con detalle en el capítulo 4). Por ejemplo, si $A = Y_1, C = Y_2, E = Y_3$ y $G = Y_4$, y $B = F = H = 0$, la salida será $F = \bar{Y}_1$. Si entonces cambiamos las líneas de activación de modo que $D = 1$ y $B = F = H = 0$, la salida será $F = \bar{Y}_2$. Así, al establecer las señales de las líneas de activación en 1 lógico de la manera ilustrada, podemos

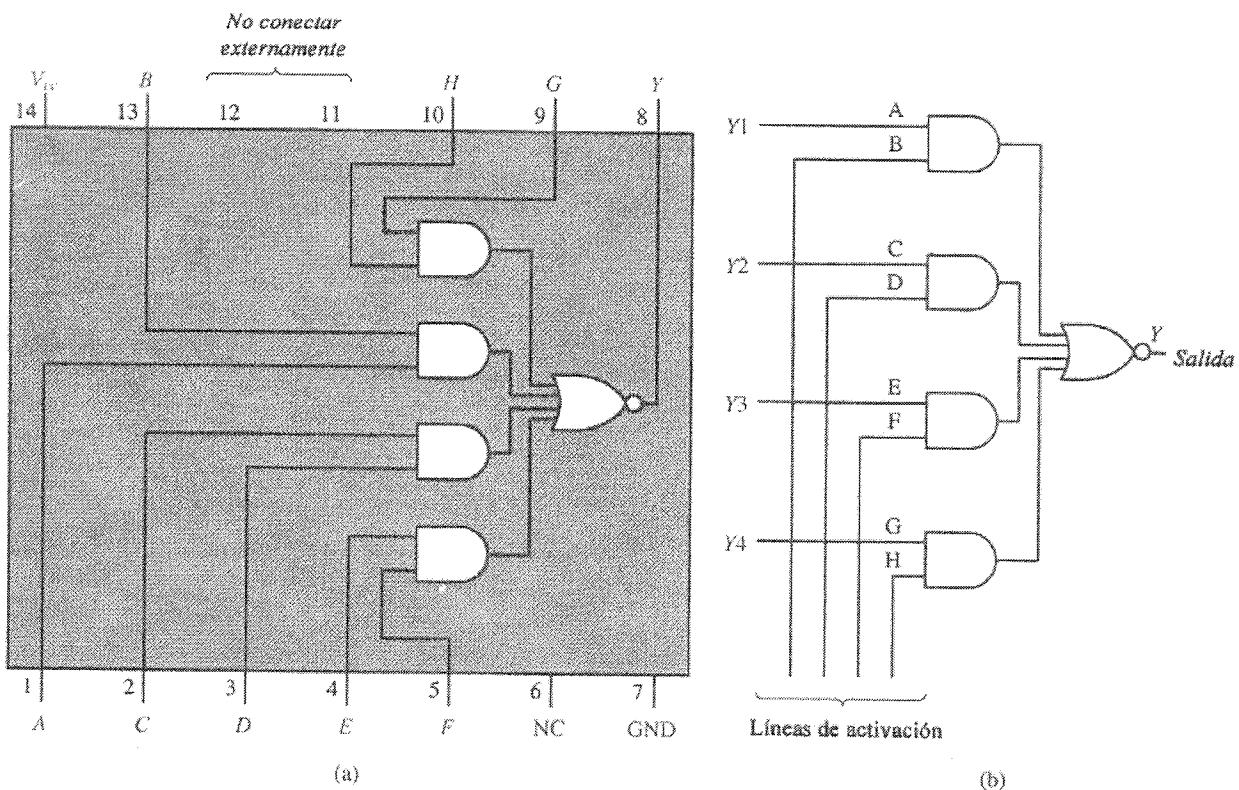


Figura 2.31 Circuito AND-OR-inversor 7454 2-2-2-2. (a) Paquete de circuito 7454 (vista superior). (b) 7454 utilizado como un multiplexor 4 a 1.

reunir un conjunto de flujos de datos representados por A , C , E y G en un único flujo representado por F .

2.5.5 Factorización

La factorización es una técnica que nos permite obtener formas de nivel superior para las funciones de commutación (las cuales requieren circuitos en los que las señales se pueden propagar a través de más de dos niveles de compuertas lógicas). La importancia de estas formas superiores surge del hecho de que muchas veces, necesitan menos hardware y, por tanto, su implantación es más económica. También requerimos formas de nivel superior en situaciones en las que debemos utilizar *fan-in* limitado. En estos casos nos valemos de la factorización para reducir la cantidad de literales en términos grandes de productos o sumas hasta obtener valores menores o iguales a la cantidad disponible de entradas de compuerta. Sin embargo, las formas de nivel superior son más difíciles de diseñar que las formas SOP o POS sencillas y, por lo general, son más lentas por que tienen más de dos niveles de compuertas lógicas.

La factorización, que por lo general implica el uso de la ley distributiva (postulado 5) del álgebra de commutación, es esencialmente un arte en el que la

experiencia tiene un papel importante. La técnica se complica aún más porque debemos agregar cierta redundancia en un paso intermedio para obtener una realización sencilla mediante factorización. Demostraremos este método a través de los ejemplos siguientes.

EJEMPLO 2.39

Suponga la siguiente función de cuatro variables:

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

La realización de dos niveles de esta función usando compuertas NAND aparece en la figura 2.32a.

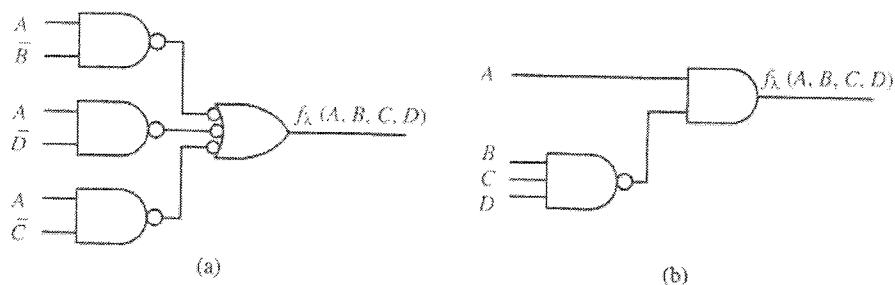


Figura 2.32 Realizaciones de $f_k(A, B, C, D)$. (a) Forma original. (b) Después de la factorización.

Observe que esta realización de segundo orden de la función requiere cuatro compuertas y nueve entradas de compuerta. Sin embargo, si aplicamos factorización a la función, podemos obtener una realización de orden superior, de la manera siguiente:

$$\begin{aligned} f_k(A, B, C, D) &= A\bar{B} + A\bar{D} + A\bar{C} \\ &= A(\bar{B} + \bar{D} + \bar{C}) \\ &= A(\overline{BCD}) \end{aligned}$$

Esta realización de $f_k(A, B, C, D)$, que se muestra en la figura 2.32b, sólo requiere dos compuertas y cinco entradas de compuerta.

EJEMPLO 2.40

Realizar la función $f(a, b, c, d) = \sum m(8, 13)$ utilizando sólo compuertas AND y OR de dos entradas.

Comenzamos escribiendo la forma SOP canónica:

$$\begin{aligned} f(a, b, c, d) &= \sum m(8, 13) \\ &= ab\bar{c}\bar{d} + ab\bar{c}d \end{aligned} \tag{2.34}$$

No podemos reducir los dos términos producto de la ecuación 2.34 mediante el álgebra de conmutación. En consecuencia, se necesitarían dos compuertas AND de cuatro entradas y una compuerta OR de dos entradas para realizar un circuito AND-OR de dos niveles.

Puesto que sólo disponemos de compuertas de dos entradas, podemos aplicar factorización para reducir el tamaño de los términos producto, como sigue:

$$\begin{aligned} f(a, b, c, d) &= ab\bar{c}\bar{d} + ab\bar{c}d \\ &= (a\bar{c})(bd + \bar{b}\bar{d}) \end{aligned} \quad (2.35)$$

En la ecuación 2.35, ningún término de producto o suma contiene más de dos literales. Así, podemos realizar esta expresión de conmutación con compuertas de dos entradas, como se muestra en la figura 2.33. Observe que el circuito contiene cuatro niveles de compuertas lógicas, incluidos los inversores de las entradas.

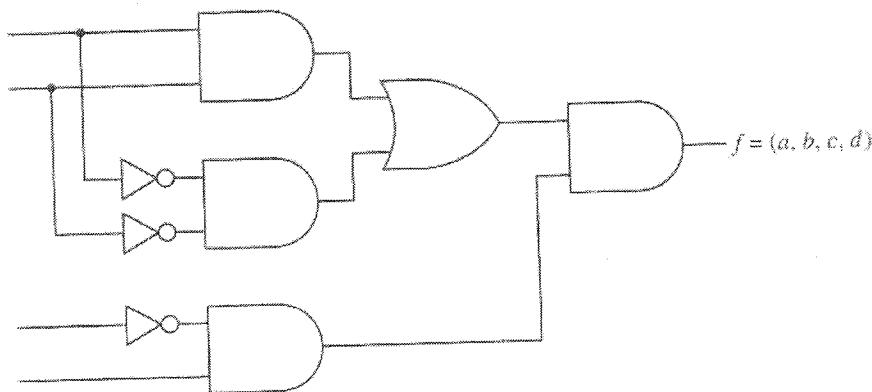


Figura 2.33 Uso de la factorización utilizada para realizar $f(a, b, c, d) = \sum m(8, 13)$ con compuertas de dos entradas.

La estrategia algebraica que aplicamos en este ejemplo se puede realizar también mediante varios métodos gráficos y tabulares, como el mapa de Karnaugh (mapa K), que presentaremos en el capítulo 3. Si desea más detalles sobre el tema, el lector debe consultar [3].

■ 2.6 Aplicaciones

Hasta ahora hemos presentado varias herramientas, como el álgebra de conmutación, las tablas de verdad y los diagramas de Venn, fundamentales para el análisis y la síntesis de las redes lógicas. Además, hemos presentado los conceptos básicos del análisis y diseño de un circuito lógico digital. En los siguientes ejemplos ilustramos el uso de estas técnicas para resolver problemas.

Algunas áreas en que las herramientas básicas tienen amplia aplicación son las de la lógica simbólica y las funciones de verdad. No consideraremos estos temas con detalle, sino que ilustraremos el uso de la lógica en varios ejemplos muy sencillos.

Ejemplo 2.41

Se diseña una alarma contra robos para un banco, de modo que percibe cuatro líneas de señal como entradas. La línea *A* es del interruptor secreto de control, la línea *B* es de un sensor de presión debajo de una caja fuerte que se encuentra en un gabinete cerrado, la línea *C* es de un reloj alimentado por baterías, y la línea *D* se conecta a un interruptor en la puerta del gabinete cerrado. Las siguientes condiciones producen un voltaje de 1 lógico en cada línea:

- A*: El interruptor de control está cerrado.
- B*: La caja fuerte está en su posición normal dentro del gabinete.
- C*: El reloj está entre las 1000 y las 1400 horas.
- D*: La puerta del gabinete está cerrada.

Escribir las ecuaciones de la lógica de control para la alarma contra robos a modo de producir un 1 lógico (suena un timbre) cuando la caja se mueve y el interruptor de control está cerrado, o cuando el gabinete se abre fuera de horas hábiles, o cuando el gabinete está abierto con el interruptor de control abierto.

El enunciado “cuando la caja se mueve y el interruptor de control está cerrado” se representa mediante $A\bar{B}$. “Cuando el gabinete se abre fuera de horas hábiles” se representa mediante $\bar{C}\bar{D}$. “Cuando el gabinete está abierto con el interruptor de control abierto” se representa mediante $\bar{A}\bar{D}$. Por tanto, la ecuación lógica para la alarma contra robos es

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

Ejemplo 2.42

Juan y María Pérez tienen dos hijos, José y Susana. Cuando salen a comer, van a un restaurante que sólo sirve hamburguesas o a uno que sólo sirve pollo. Antes de salir, la familia vota para elegir el restaurante. Gana la mayoría, excepto cuando los papás están de acuerdo, en cuyo caso ellos ganan. Cualquier otro empate implica ir al restaurante de pollo. Diseñar un circuito lógico que seleccione en forma automática el restaurante elegido cuando toda la familia vota.

Si 1 representa un voto por las hamburguesas y 0 un voto por el pollo, la tabla de verdad para el circuito de votación es la de la tabla 2.9. La función lógica es

$$\begin{aligned} f(A, B, C, D) = & \bar{A}\bar{B}\bar{C}\bar{D} + A\bar{B}\bar{C}\bar{D} + A\bar{B}\bar{C}\bar{D} \\ & + A\bar{B}\bar{C}D + A\bar{B}CD + ABCD \end{aligned}$$

$$\begin{aligned}
 &= \bar{A}BCD + A\bar{B}CD + AB \\
 &= AB + ACD + \bar{A}\bar{B}CD \\
 &= AB + ACD + BCD
 \end{aligned}$$

La figura 2.34 muestra el circuito lógico para esta función.

TABLA 2.9 TABLA DE VERDAD PARA LA VOTACIÓN DE LA FAMILIA PÉREZ

Juan A	Maria B	José C	Susana D	Voto por las hamburguesas f
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

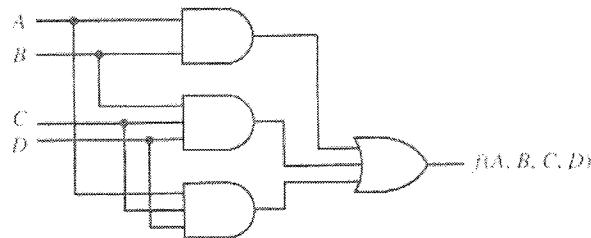


Figura 2.34 Circuito lógico para el ejemplo de la votación por el restaurante.

EJEMPLO 2.43

Deducir las ecuaciones lógicas para un circuito que sume los dos números binarios de 2 bits $(A_1 A_0)_2$ y $(B_1 B_0)_2$ y produzca los bits de suma $(S_1 S_0)_2$ y el bit de acarreo de salida C_1 ; es decir,

$$\begin{array}{r}
 A_1 A_0 \\
 + B_1 B_0 \\
 \hline
 C_1 S_1 S_0
 \end{array}$$

Tenemos cuatro entradas, A_1, A_0, B_1 y B_0 , y tres salidas, C_1, S_1 y S_0 , la tabla de verdad es entonces como se muestra en la tabla 2.10. Deducimos cada fila de la

TABLA 2.10 TABLA DE VERDAD PARA EL SUMADOR BINARIO DE 2 BITS

A_1	A_0	B_1	B_0	C_1	S_1	S_0
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	0	0	1	0
0	0	1	1	0	1	1
0	1	0	0	0	0	1
0	1	0	1	0	1	0
0	1	1	0	0	1	1
→	0	1	1	1	0	0
1	0	0	0	0	1	0
1	0	0	1	0	1	1
1	0	1	0	1	0	0
1	0	1	1	1	0	1
1	1	0	0	0	1	1
1	1	0	1	1	0	0
1	1	1	0	1	0	1
1	1	1	1	1	1	0

tabla de verdad como se muestra en el siguiente cálculo de la octava fila.

$$\begin{array}{r}
 & 1 & 1 \\
 & & 0 & 1 \\
 + & 1 & 1 \\
 \hline
 1 & 0 & 0
 \end{array}$$

Obtenemos la siguiente ecuación lógica para los términos C_1, S_1 y S_0 a partir de la tabla de verdad:

$$\begin{aligned}
 S_0 = & \bar{A}_1 \bar{A}_0 \bar{B}_1 B_0 + \bar{A}_1 \bar{A}_0 B_1 B_0 + \bar{A}_1 A_0 \bar{B}_1 \bar{B}_0 + \bar{A}_1 A_0 B_1 \bar{B}_0 \\
 & + A_1 \bar{A}_0 \bar{B}_1 B_0 + A_1 \bar{A}_0 B_1 B_0 + A_1 A_0 \bar{B}_1 \bar{B}_0 + A_1 A_0 B_1 \bar{B}_0
 \end{aligned}$$

$$\begin{aligned}
 S_1 = & \bar{A}_1 \bar{A}_0 B_1 \bar{B}_0 + \bar{A}_1 \bar{A}_0 B_1 B_0 + \bar{A}_1 A_0 \bar{B}_1 B_0 + \bar{A}_1 A_0 B_1 \bar{B}_0 \\
 & + A_1 \bar{A}_0 \bar{B}_1 \bar{B}_0 + A_1 \bar{A}_0 \bar{B}_1 B_0 + A_1 A_0 \bar{B}_1 \bar{B}_0 + A_1 A_0 B_1 B_0
 \end{aligned}$$

$$\begin{aligned}
 C_1 = & \bar{A}_1 A_0 B_1 B_0 + A_1 \bar{A}_0 B_1 \bar{B}_0 + A_1 \bar{A}_0 B_1 B_0 + A_1 A_0 \bar{B}_1 \bar{B}_0 \\
 & + A_1 A_0 B_1 \bar{B}_0 + A_1 A_0 B_1 B_0
 \end{aligned}$$

Podemos reducir estas expresiones a:

$$S_0 = A_0 \bar{B}_0 + \bar{A}_0 B_0$$

$$S_1 = \bar{A}_1 \bar{A}_0 B_1 + \bar{A}_1 B_1 \bar{B}_0 + \bar{A}_1 A_0 \bar{B}_1 B_0 + A_1 A_0 B_1 B_0 + A_1 \bar{B}_1 \bar{B}_0 + A_1 \bar{A}_0 \bar{B}_1$$

$$C_1 = A_0 B_1 B_0 + A_1 A_0 B_0 + A_1 B_1$$

2.7 Diseño de circuitos lógicos asistido por computadora

2.7.1 El ciclo de diseño

Una gran parte de los circuitos lógicos digitales de la actualidad contienen el equivalente de miles o cientos de miles de compuertas lógicas. La mayor parte de estos circuitos se fabrican en chips de circuitos integrados (IC). El diseño y fabricación de un circuito VLSI (integración a muy grande escala) es un proceso caro y complejo. Por tanto, es necesario verificar que el diseño del circuito lógico sea correcto antes de iniciar el diseño del circuito real, para tener una alta probabilidad de operación correcta desde que se inicie la fabricación del circuito. Esto también es cierto al diseñar sistemas digitales con varios IC y tarjetas de circuitos. Es un hecho que no se pueden desarrollar ni verificar los circuitos y sistemas de tal complejidad sin el apoyo de herramientas de diseño asistido por computadora (CAD).

El ciclo de diseño para un circuito lógico digital abarca varios pasos entre el concepto y la implantación física, incluida la síntesis de diseño, la simulación, la realización y la prueba. La figura 2.35 describe este proceso. A partir de enunciado del problema, el diseñador comienza por desarrollar una solución abstracta que se va transformando sistemáticamente en un circuito lógico digital. Esta transformación es apoyada por el modelado y la evaluación del circuito en cada nivel de abstracción del diseño. Evaluamos un diseño mediante un modelo que simula su operación, lo que permite verificar la respuesta del circuito a diversos estímulos de entrada. El modelo se revisa y simula cuantas veces sea necesario hasta obtener las respuestas correctas. Además de verificar la correcta operación, podemos evaluar los efectos de las diferentes opciones de diseño sobre el desempeño del circuito como apoyo para la toma de decisiones de diseño con un costo realista. Una vez aceptado el comportamiento modelado del diseño, se desarrolla e implanta el diseño físico. Por último, se prueba el circuito terminado, y los resultados de la prueba se comparan con el comportamiento modelado para detectar dispositivos con fallas.

En este capítulo describiremos los procesos y herramientas de CAD empleados en las fases de síntesis y análisis del diseño de circuitos digitales. Examinaremos cada bloque de las fases de síntesis y análisis de la figura 2.35. En primer lugar, analizaremos el modelado del diseño. En la siguiente sección describiremos la captura de esquemas y otras herramientas de CAD que capturan los modelos de circuitos lógicos y los traducen al formato de la base de datos de diseño. Después analizaremos la simulación lógica digital, que se aplica para verificar el comportamiento lógico y los tiempos de un diseño. Por último, examinaremos las herramientas de CAD que deducen expresiones de conmutación mínimas para las funciones lógicas.

2.7.2 Modelado de un circuito digital

El modelado de un circuito lógico o sistema digital tiene varios propósitos. En primer lugar, el proceso de desarrollo de un modelo ayuda al diseñador formalizar una solución. En segundo lugar, una computadora digital puede procesar un modelo de circuito para verificar que no tenga errores de diseño sea correcto, y para predecir las características de tiempos. Además, se disponen