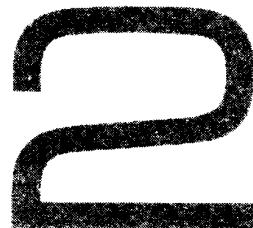

Algebra booleana y compuertas lógicas



2-1 DEFINICIONES BASICAS

El álgebra booleana, como cualquier otro sistema matemático deductivo, puede definirse con un conjunto de elementos, un conjunto de operadores y un número de axiomas no probados o postulados. Un *conjunto* de elementos es cualquier colección de objetos que tienen una propiedad común. Si S es un conjunto y, x y y son ciertos objetos, entonces $x \in S$ denota que x es un miembro del conjunto S y, $y \notin S$ denota que y no es un elemento de S . Un conjunto con un número denumerable de elementos se especifica por llaves: $A = \{ 1, 2, 3, 4 \}$, esto es, los elementos del conjunto A son los números 1, 2, 3 y 4. Un *operador binario* definido en un conjunto S de elementos es una regla que asigna a cada par de elementos de S un elemento único de S . Como ejemplo, considérese la relación $a * b = c$. Se dice que $*$ es un operador binario y especifica una regla para encontrar c mediante el par (a, b) y también si $a, b, c \in S$. Sin embargo, $*$ no es un operador binario si $a, b \in S$, si la regla encuentra que $c \notin S$.

Los postulados de un sistema matemático forman los supuestos básicos mediante los cuales es posible deducir las reglas, teoremas y propiedades del sistema. Los postulados más comunes que se utilizan para formular diversas estructuras algebraicas son:

1. *Cierre*. Un conjunto S está cerrado con respecto a un operador binario si, para cada par de elementos de S , el operador binario especifica una regla para obtener un elemento único de S . Por ejemplo, el conjunto de los números naturales $N = \{ 1, 2, 3, 4, \dots \}$ está cerrado con respecto al operador binario más (+) por las reglas de la adición aritmética, ya que para cualquier $a, b \in N$ se obtiene una única $c \in N$ por la operación $a + b = c$. El conjunto de los números naturales no está cerrado con respecto al operador binario menos (−) por las reglas de la resta aritmética debido a que $2 - 3 = -1$ y $-1 \notin N$, ya que $(-1) \notin N$.
2. *Ley asociativa*. Un operador binario $*$ en un conjunto S se dice que es asociativo siempre que

$$(x*y)*z = x*(y*z) \text{ para todos } x, y, z \in S$$

3. *Ley conmutativa.* Un operador binario $*$ en un conjunto S se dice que es conmutativo siempre que:

$$x*y = y*x \text{ para todos } x, y \in S$$

4. *Elemento identidad.* Un conjunto S se dice que tiene un elemento identidad respecto a una operación binaria $*$ en S si existe un elemento $e \in S$ con la propiedad:

$$e*x = x*e = x \quad \text{para cada } x \in S$$

Ejemplo: El elemento 0 es un elemento identidad con respecto a la operación $+$ en el conjunto de enteros $I = \{ \dots, -3, -2, -1, 0, 1, 2, 3, \dots \}$ ya que:

$$x + 0 = 0 + x = x \text{ para cualquier } x \in I$$

El conjunto de los números naturales N no tiene elemento identidad ya que 0 está excluido del conjunto.

5. *Inversa.* Un conjunto S que tiene el elemento identidad e con respecto a un operador binario $*$ se dice que tiene una inversa siempre que, para cada $x \in S$, existe un elemento $y \in S$ tal que:

$$x*y = e$$

Ejemplo: En el conjunto de enteros I con $e = 0$, la inversa de un elemento a es $(-a)$ ya que $a + (-a) = 0$.

6. *Ley distributiva.* Si $*$ y \cdot son dos operadores binarios en un conjunto S , $*$ se dice que es distributivo sobre \cdot siempre que:

$$x*(y \cdot z) = (x*y) \cdot (x*z)$$

Un ejemplo de una estructura algebraica es un *campo*. Un *campo* es un conjunto de elementos, junto con dos operadores binarios, cada uno teniendo las propiedades 1 a 5 y ambos operadores combinados para dar la propiedad 6. El conjunto de los números reales junto con los operadores binarios $+$ y \cdot forman el campo de los números reales. El campo de los números reales es la base de la aritmética y del álgebra ordinaria. Los operadores y los postulados tienen los siguientes significados:

El operador binario $+$ define la adición.

La identidad aditiva es 0.

La inversa aditiva define la sustracción.

El operador binario \cdot define la multiplicación.

La identidad multiplicativa es 1.

La inversa multiplicativa de $a = 1/a$ define la división, esto es, $a \cdot 1/a = 1$.

La única ley distributiva aplicable es la de \cdot sobre $+$:

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

2-2 DEFINICION AXIOMATICA DEL ALGEBRA BOOLEANA

En 1854 George Boole (1) introdujo un tratamiento sistemático de la lógica y desarrolló para este propósito un sistema algebraico que ahora se conoce como álgebra booleana. En 1938 C.E. Shannon (2) introdujo un álgebra booleana de dos valores denominada *álgebra de interruptores*, en la cual demostró que las propiedades de los circuitos eléctricos y estables con interruptores pueden representarse con esta álgebra. Para la definición formal del álgebra booleana, se emplean los postulados formulados por E.V. Huntington (3) en 1904. Estos postulados o axiomas no son únicos para definir el álgebra booleana. Se han usado otros conjuntos de postulados.* El álgebra booleana es una estructura algebraica definida en un conjunto de elementos B junto con dos operadores binarios $+$ y \cdot siempre que se satisfagan los siguientes postulados (Huntington):

1. (a) Cierre con respecto al operador $+$.
(b) Cierre con respecto al operador \cdot
2. (a) Un elemento identidad con respecto a $+$, designado por 0: $x + 0 = 0 + x = x$.
(b) Un elemento identidad con respecto a \cdot , designado por 1: $x \cdot 1 = 1 \cdot x = x$.
3. (a) Comutativo con respecto a $+$: $x + y = y + x$.
(b) Comutativo con respecto a \cdot : $x \cdot y = y \cdot x$.
4. (a) \cdot es distributivo sobre $+$: $x \cdot (y + z) = (x \cdot y) + (x \cdot z)$.
(b) $+$ es distributivo sobre \cdot : $x + (y \cdot z) = (x + y) \cdot (x + z)$.
5. Para cada elemento $x \in B$, existe un elemento $x' \in B$ (denominado complemento de x) tal que: (a) $x + x' = 1$ y (b) $x \cdot x' = 0$.
6. Existen cuando menos dos elementos $x, y \in B$ tales que $x \neq y$.

Al comparar el álgebra booleana con la aritmética y el álgebra ordinaria (el campo de los números reales), se observan las siguientes diferencias:

1. Los postulados de Huntington no incluyen la ley asociativa. No obstante, esta ley es válida para el álgebra booleana y puede derivarse (para ambos operadores) mediante los otros postulados.

*Véase, por ejemplo, Birk off y Bartee (4), Capítulo 5.

2. La ley distributiva de $+$ sobre \cdot , esto es, $x + (y \cdot z) = (x + y) \cdot (x + z)$, es válida para el álgebra booleana, pero no para el álgebra ordinaria.
3. El álgebra booleana no tiene inversas aditiva o multiplicativa; por lo tanto, no hay operaciones de sustracción o división.
4. El postulado 5 define un operador llamado *complemento* que no se encuentra en el álgebra ordinaria.
5. El álgebra ordinaria trata con números reales, los cuales constituyen un conjunto infinito de elementos. El álgebra booleana trata con el conjunto todavía no definido de elementos B , pero en el álgebra booleana de dos valores que se define más adelante (y de interés en el uso subsecuente de esta álgebra), B se define como un conjunto con sólo dos elementos, 0 y 1.

El álgebra booleana se parece en algunos aspectos al álgebra ordinaria. La elección de los símbolos $+$ y \cdot es intencional para facilitar las manipulaciones algebraicas booleanas por las personas que ya están familiarizadas con el álgebra ordinaria. Aunque puede utilizarse cierto conocimiento del álgebra ordinaria para tratar con el álgebra booleana, el principiante debe tener cuidado de no sustituir las reglas del álgebra ordinaria cuando no son aplicables.

Es importante distinguir entre los elementos del conjunto de una estructura algebraica y las variables de un sistema algebraico. Por ejemplo, los elementos de campos de los números reales son números, en tanto que variables como a , b , c , etc., que se usan en el álgebra ordinaria, son símbolos que representan números reales. En forma semejante, en el álgebra booleana se definen los elementos del conjunto B y variables como x , y , z son simplemente símbolos que representan los elementos. En este punto, es importante tener en cuenta que con objeto de tener un álgebra booleana, deben mostrarse:

1. los elementos del conjunto B ,
2. las reglas de operación para los dos operadores binarios y,
3. que el conjunto de elementos B , junto con los dos operadores, satisfacen los seis postulados de Huntington.

Pueden formularse muchas álgebras booleanas, dependiendo de la elección de los elementos de B y las reglas de operación. * En el trabajo subsecuente, se tratará sólo con el álgebra booleana de dos valores, esto es, una con sólo dos elementos. El álgebra booleana de dos valores tiene aplicaciones en la teoría de conjuntos (el álgebra de clases) y en la lógica proposicional. El interés aquí es la aplicación del álgebra booleana a los circuitos tipo compuerta.

*Véase, por ejemplo, Hohn (6), Whitesitt (7) o Birkhoff y Bartee (4).

Algebra booleana de dos valores

Un álgebra booleana de dos valores se define en un conjunto de dos elementos, $B = \{ 0, 1 \}$, con las reglas para dos operadores binarios $+$ y \cdot como se muestra en las siguientes tablas de operadores (la regla para el operador complemento es para la verificación del postulado 5):

x	y	$x \cdot y$	x	y	$x + y$	x		x'
0	0	0	0	0	0	0		1
0	1	0	0	1	1	1		0
1	0	0	1	0	1			
1	1	1	1	1	1			

Estas reglas son exactamente las mismas que las operaciones AND, OR y NOT, respectivamente, definidas en la Tabla 1-6. Ahora debe mostrarse que los postulados de Huntington son válidos para el conjunto $B = \{ 0, 1 \}$ y los dos operadores binarios que se definieron antes.

1. *Cierre* es obvio por las tablas, ya que el resultado de cada operación es, ya sea 1 o 0 y $1, 0 \in B$.
2. A partir de las tablas puede verse que:

$$(a) 0 + 0 = 0 \quad 0 + 1 = 1 + 0 = 1 \\ (b) 1 \cdot 1 = 1 \quad 1 \cdot 0 = 0 \cdot 1 = 0$$

establece que los dos *elementos identidad* son 0 para $+$ y 1 para \cdot como se define por el postulado 2.

3. Las leyes *comutativas* son obvias por la simetría de las tablas del operador binario.
4. (a) La ley *distributiva* $x \cdot (y + z) = (x \cdot y) + (x \cdot z)$ puede mostrarse que es verdadera por las tablas del operador, al formar una tabla de verdad de todos los valores posibles de x, y y z . Para cada combinación, se deriva $x \cdot (y + z)$ y se muestra que el valor es el mismo que $(x \cdot y) + (x \cdot z)$.

x	y	z	$y + z$	$x \cdot (y + z)$	$x \cdot y$	$x \cdot z$	$(x \cdot y) + (x \cdot z)$
0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	0	0	0	0
1	0	0	0	0	0	0	0
1	0	1	1	1	0	1	1
1	1	0	1	1	1	0	1
1	1	1	1	1	1	1	1

- (b) La ley *distributiva* de $+$ sobre \cdot puede mostrarse que es válida mediante una tabla de verdad semejante a la anterior.
5. Mediante la tabla de complemento es fácil mostrar que:
- $x + x' = 1$, ya que $0 + 0' = 0 + 1 = 1$ y $1 + 1' = 1 + 0 = 1$.
 - $x \cdot x' = 0$, ya que $0 \cdot 0' = 0 \cdot 1 = 0$ y $1 \cdot 1' = 1 \cdot 0 = 0$ lo cual verifica el postulado 5.
6. El postulado 6 se satisface ya que el álgebra booleana de dos valores tiene dos elementos distintos 1 y 0 con $1 \neq 0$.

Acaba de establecerse un álgebra booleana de dos valores que tiene un conjunto de dos elementos, 1 y 0, dos operadores binarios con reglas de operación equivalentes a las operaciones AND y OR y un operador complemento equivalente al operador NOT. En consecuencia, el álgebra booleana se ha definido de una manera matemáticamente formal y se ha mostrado que es equivalente a la lógica binaria que se presentó en forma heurística en la Sección 1-8. La presentación heurística es de ayuda para entender la aplicación del álgebra booleana a los circuitos tipo compuerta. La presentación formal es necesaria para desarrollar los teoremas y las propiedades del sistema algebraico. El álgebra booleana de dos valores que se define en esta sección también se conoce como “álgebra de interruptores” (o de conmutación) entre los ingenieros. Para dar énfasis a las similitudes entre el álgebra booleana de dos valores y otros sistemas binarios, esta álgebra se denominó “lógica binaria” en la Sección 1-8. De aquí en adelante, se eliminará el calificativo “dos valores” del álgebra booleana en las exposiciones subsecuentes.

2-3 TEOREMAS BASICOS Y PROPIEDADES DEL ALGEBRA BOOLEANA

Dualidad

Los postulados de Huntington se listaron en pares y se designaron en la parte (a) y la (b). Una parte puede obtenerse de la otra si los operadores binarios y los elementos identidad se intercambian. Esta propiedad importante del álgebra booleana se denomina *principio de dualidad*. Establece que cada expresión algebraica deducida de los postulados del álgebra booleana permanece válida si los operadores y los elementos identidad se intercambian. En una álgebra booleana de dos valores, los elementos identidad y los elementos del conjunto B son los mismos: 1 y 0. El principio de dualidad tiene muchas aplicaciones. Si se desea el dual de una expresión algebraica, simplemente se intercambian los operadores OR y AND y se reemplazan los 1 por 0 y los 0 por 1.

Teoremas básicos

En la Tabla 2-1 se listan seis teoremas del álgebra booleana y cuatro de sus postulados. La notación se simplifica omitiendo el \cdot siempre que esto no provoque confusiones. Los teoremas y postulados que se listan son las relaciones más básicas en el álgebra

TABLA 2-1 Postulados y teoremas del álgebra booleana

Postulado 2	(a) $x + 0 = x$	(b) $x \cdot 1 = x$
Postulado 5	(a) $x + x' = 1$	(b) $x \cdot x' = 0$
Teorema 1	(a) $x + x = x$	(b) $x \cdot x = x$
Teorema 2	(a) $x + 1 = 1$	(b) $x \cdot 0 = 0$
Teorema 3, involución	$(x')' = x$	
Postulado 3, conmutativo	(a) $x + y = y + x$	(b) $xy = yx$
Teorema 4, asociativo	(a) $x + (y + z) = (x + y) + z$	(b) $x(yz) = (xy)z$
Postulado 4, distributivo	(a) $x(y + z) = xy + xz$	(b) $x + yz = (x + y)(x + z)$
Teorema 5, de De Morgan	(a) $(x + y)' = x'y'$	(b) $(xy)' = x' + y'$
Teorema 6, absorción	(a) $x + xy = x$	(b) $x(x + y) = x$

booleana. Se aconseja al lector que se familiarice con ellos tan pronto como le sea posible. Los teoremas, al igual que los postulados, se listan en pares; cada relación es el dual de su pareja. Los postulados son axiomas básicos de la estructura algebraica y no necesitan prueba. Los teoremas deben probarse mediante los postulados. Las pruebas de los teoremas con una variables se presentan más adelante. A la derecha se lista el número del postulado que justifica cada paso de la prueba.

TEOREMA 1(a): $x + x = x$.

$$\begin{aligned}
 x + x &= (x + x) \cdot 1 && \text{por el postulado: 2(b)} \\
 &= (x + x)(x + x') && 5(a) \\
 &= x + xx' && 4(b) \\
 &= x + 0 && 5(b) \\
 &= x && 2(a)
 \end{aligned}$$

TEOREMA 1(b): $x \cdot x = x$.

$$\begin{aligned}
 x \cdot x &= xx + 0 && \text{por el postulado: 2(a)} \\
 &= xx + xx' && 5(b) \\
 &= x(x + x') && 4(a) \\
 &= x \cdot 1 && 5(a) \\
 &= x && 2(b)
 \end{aligned}$$

Obsérvese que el teorema 1(b) es el dual del teorema 1(a) y que cada paso de la prueba en la parte (b) es el dual de la parte (a). Cualquier teorema dual puede derivarse en forma similar de la prueba de su pareja correspondiente.

TEOREMA 2(a): $x + 1 = 1$.

$$\begin{aligned}
 x + 1 &= 1 \cdot (x + 1) && \text{por el postulado: 2(b)} \\
 &= (x + x')(x + 1) && 5(a) \\
 &= x + x' \cdot 1 && 4(b) \\
 &= x + x' && 2(b) \\
 &= 1 && 5(a)
 \end{aligned}$$

TEOREMA 2(b): $x \cdot 0 = 0$ por la dualidad.

TEOREMA 3: $(x')' = x$. Por el postulado 5, se tiene $x + x' = 1$ y $x \cdot x' = 0$, lo cual define el complemento de x . El complemento de x' es x y también es $(x')'$. Por tanto, ya que el complemento es único, se tiene que $(x')' = x$.

Los teoremas que implican dos o tres variables pueden probarse en forma algebraica por los postulados y teoremas que ya se han probado. Por ejemplo, tómese el teorema de absorción.

TEOREMA 6(a): $x + xy = x$.

$$\begin{aligned} x + xy &= x \cdot 1 + xy && \text{por el postulado 2(b)} \\ &= x(1 + y) && \text{por el postulado 4(a)} \\ &= x(y + 1) && \text{por el postulado 3(a)} \\ &= x \cdot 1 && \text{por el postulado 2(a)} \\ &= x && \text{por el postulado 2(b)} \end{aligned}$$

TEOREMA 6(b): $x(x + y) = x$ por dualidad.

Puede demostrarse que los teoremas del álgebra booleana son válidos mediante las tablas de verdad. En estas tablas, ambos lados de la relación se verifican para que den resultados idénticos en todas las combinaciones posibles de las variables implicadas. La siguiente tabla de verdad verifica el primer teorema de absorción.

		$=$	
x	y	xy	$x + xy$
0	0	0	0
0	1	0	0
1	0	0	1
1	1	1	1

Las pruebas algebraicas de la ley asociativa y del teorema de De Morgan son largas y no se mostrarán aquí. Sin embargo, su validez se ilustra fácilmente con tablas de verdad. Por ejemplo, la tabla de verdad para el primer teorema de De Morgan $(x + y)' = x'y'$ se muestra a continuación.

x	y	$x + y$	$(x + y)'$	x'	y'	$x'y'$
0	0	0	1	1	1	1
0	1	1	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	0

Precedencia de los operadores

La precedencia de los operadores para evaluar las expresiones booleanas es (1) paréntesis, (2) NOT, (3) AND y (4) OR. En otras palabras, la expresión entre paréntesis debe evaluarse antes que las otras operaciones. La siguiente operación que toma precedencia es el complemento, entonces sigue AND y, por último, OR. Como ejemplo, considérese la tabla de verdad para el teorema de De Morgan. El lado izquierdo de la expresión es $(x + y)'$. Por consiguiente, la expresión entre paréntesis se evalúa primero y entonces se toma el complemento del resultado. El lado derecho de la expresión es $x'y'$. Así que, el complemento de x y el complemento de y se evalúan primero y el resultado se opera por AND. Obsérvese que en aritmética ordinaria es válida la misma precedencia (excepto para el complemento) cuando la multiplicación y la suma se reemplazan por AND y OR, respectivamente.

Diagrama de Venn

Una ilustración de ayuda que es posible utilizar para visualizar las relaciones entre las variables de una expresión booleana es el *diagrama de Venn*. Este diagrama consta de un rectángulo, como el que se muestra en la Fig. 2-1, dentro del cual se dibujan círculos traslapados, uno para cada variable. Cada círculo se etiqueta por una variable. Se designan todos los puntos dentro de un círculo como pertenecientes a la variable etiquetada y todos los puntos fuera del círculo como no pertenecientes a la variable. Tómese, por ejemplo, el círculo etiquetado x . Si se considera el interior del círculo, se dice que $x = 1$; si se considera el exterior, se dice que $x = 0$. Ahora, con dos círculos traslapados, hay cuatro áreas distintas dentro del rectángulo: el área que no pertenece ya sea a x o y ($x'y'$). El área dentro del círculo y pero fuera de x ($x'y$), el área en el interior del círculo x pero fuera de y (xy') y el área dentro de ambos círculos (xy).

Los diagramas de Venn pueden usarse para ilustrar los postulados del álgebra booleana o para mostrar la validez de los teoremas. En la Fig. 2-2, por ejemplo, se ilustra que el área que pertenece a xy está en el interior del círculo x y, por lo tanto, $x + xy = x$. En la Fig. 2-3 se muestra la ley distributiva $x(y + z) = xy + xz$. En este diagrama se tienen tres círculos traslapados, uno para cada una de las variables x , y y z . Es posible distinguir ocho áreas distintas en un diagrama de Venn de tres variables. Para este ejemplo particular, la ley distributiva se demuestra observando que el área intersecada por el círculo x , con el área que encierra y o z , es la misma área que pertenece a xy o xz .

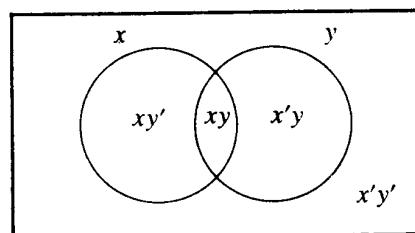


Figura 2-1 Diagrama de Venn para dos variables.

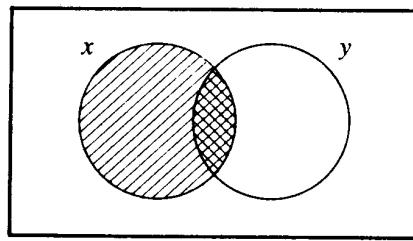


Figura 2-2 Ilustración en el diagrama de Venn de $x = xy + x$.

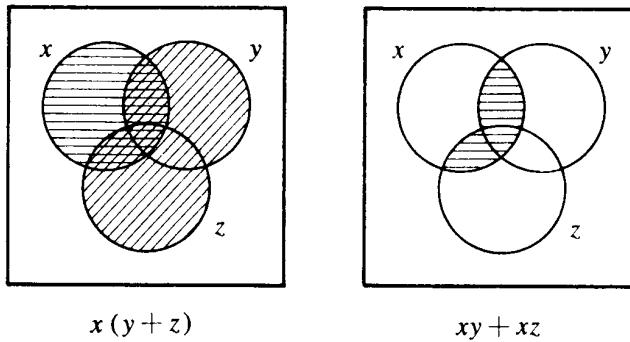


Figura 2-3 Ilustración en el diagrama de Venn de la ley distributiva.

2-4 FUNCIONES BOOLEANAS

Una variable binaria puede tomar el valor de 0 o 1. Una función booleana es una expresión formada por variables binarias, los dos operadores binarios OR y AND, operador unitario NOT, paréntesis y signo de igual. Para un valor dado de variables, la función puede ser 0 o bien 1. Considérese, por ejemplo, la función booleana:

$$F_1 = xyz'$$

La función F_1 es igual a 1 si $x = 1$ y $y = 1$ y $z' = 1$; de otra manera, $F_1 = 0$. Este es un ejemplo de una función booleana representada como una función algebraica. Una función booleana también puede representarse en una tabla de verdad. Para representar una función en una tabla de verdad, se necesita una lista de las 2^n combinaciones de 1 y 0 de las n variables binarias y, una columna que muestre las combinaciones para las cuales la función es igual a 1 o 0. Como se muestra en la Tabla 2-2, hay ocho combinaciones distintas posibles para asignar bits a tres variables. La columna etiquetada F_1 contiene un 0 o bien un 1, para cada una de estas combinaciones. En la tabla se muestra que la función F_1 es igual a 1 sólo cuando $x = 1$, $y = 1$ y $z = 0$. De otra manera, es igual a 0. (Obsérvese que el enunciado $z' = 1$ es equivalente a decir que $z = 0$.) Considérese ahora la función:

$$F_2 = x + y'z$$

$F_2 = 1$ si $x = 1$ o si $y = 0$, mientras $z = 1$. En la Tabla 2-2, $x = 1$ en los últimos cuatro renglones y $yz = 01$ en los renglones 001 y 101. La última combinación se aplica también para $x = 1$. Por tanto, hay cinco combinaciones que hacen $F_2 = 1$. Como un tercer ejemplo, considérese la función:

$$F_3 = x'y'z + x'yz + xy'$$

Esto se muestra en la Tabla 2-2 con cuatro números 1 y cuatro números 0. F_4 es la misma que F_3 y se considera a continuación.

Cualquier función booleana puede representarse en una tabla de verdad. El número de renglones en la tabla es 2^n , donde n es el número de variables binarias en la función. Las combinaciones de 1 y 0 para cada renglón se obtienen fácilmente mediante los números binarios contando desde 0 a $2^n - 1$. Para cada renglón de la tabla, hay un valor para la función igual ya sea a 1 o 0. Surge ahora la pregunta, ¿es única una expresión algebraica de una función booleana dada? En otras palabras, ¿es posible encontrar dos expresiones algebraicas que especifiquen la misma función? La respuesta a esta pregunta es afirmativa. De hecho, la manipulación del álgebra booleana se aplica principalmente al problema de encontrar expresiones más simples para la misma función. Considérese, por ejemplo, la función:

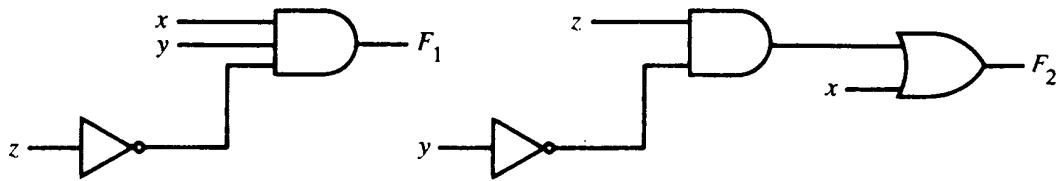
$$F_4 = xy' + x'z$$

Mediante la Tabla 2-2, se encuentra que F_4 es la misma que F_3 , ya que ambas tienen 1 idénticos y 0 idénticos para cada combinación de valores de las tres variables binarias. En general, se dice que dos funciones de n variables binarias son iguales si tienen el mismo valor para todas las 2^n combinaciones posibles de las n variables.

Una función booleana puede transformarse de una expresión algebraica en un diagrama lógico compuesto de compuertas AND, OR y NOT. El implante de las cuatro funciones que se introdujo en la exposición anterior se muestra en la Fig. 2-4.

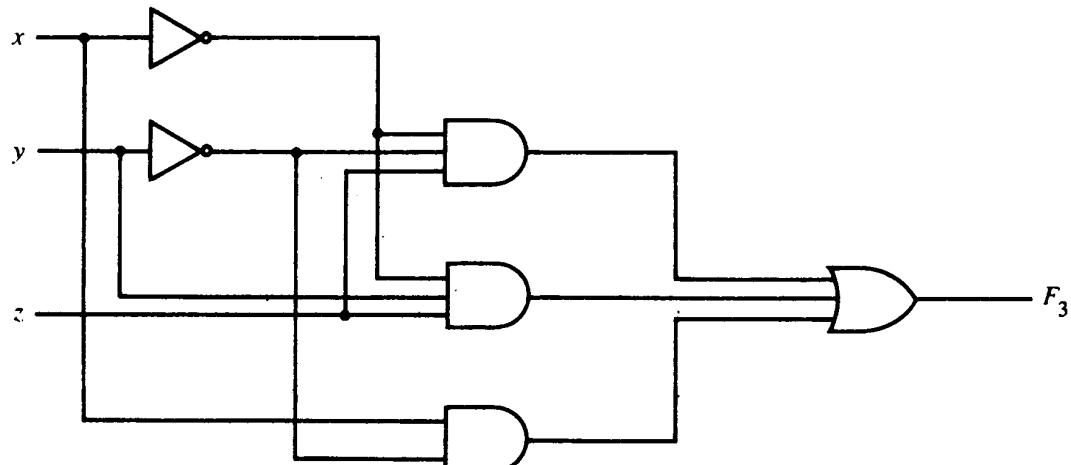
TABLA 2-2 Tablas de verdad para $F_1 = xyz'$, $F_2 = x + y'z$, $F_3 = x'y'z + x'yz + xy'$, y $F_4 = xy' + x'z$

x	y	z	F_1	F_2	F_3	F_4
0	0	0	0	0	0	0
0	0	1	0	1	1	1
0	1	0	0	0	0	0
0	1	1	0	0	1	1
1	0	0	0	1	1	1
1	0	1	0	1	1	1
1	1	0	1	1	0	0
1	1	1	0	1	0	0

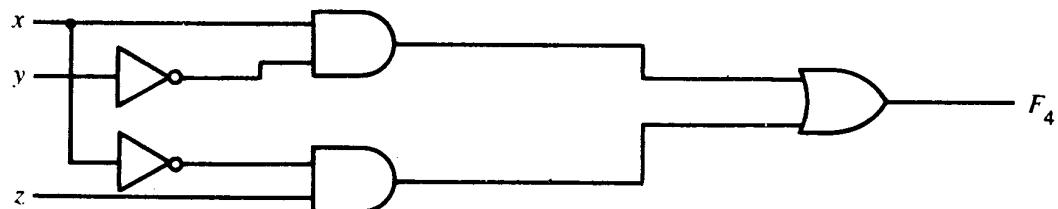


$$(a) \quad F_1 = xyz'$$

$$(b) \quad F_2 = x + y'z$$



$$(c) \quad F_3 = x'y'z + x'yz + xy'$$



$$(d) \quad F_4 = xy' + x'z$$

Figura 2-4 Implementación de funciones booleanas con compuertas.

El diagrama lógico incluye un circuito inversor para cada variable presente en su forma de complemento. (El inversor es innecesario si está disponible el complemento de la variable.) Hay una compuerta Y para cada término en la expresión y, se usa una compuerta O para combinar dos o más términos. Para los diagramas, es obvio que el implante de F_4 requiere menos compuertas y menos entradas que F_3 . Ya que F_4 y F_3 son funciones booleanas iguales, es más económico implantar la forma F_4 que la F_3 . Para encontrar circuitos más simples, debe conocerse cómo manipular las funciones booleanas para obtener expresiones iguales y más simples. Lo que constituye la mejor forma de una función booleana depende de la aplicación particular. En esta sección, se toma en consideración el criterio de minimización de equipo.

Manipulación algebraica

Una *literal* es una variable prima o no prima. Cuando una función booleana se implanta con compuertas lógicas, cada literal en la función denota una entrada a una compuerta, y cada término se implanta con una compuerta. La minimización del número de literales y el número de términos resulta en un circuito con menos equipo. No siempre es posible minimizar ambos en forma simultánea; por lo común, debe disponerse de más criterios. Por el momento, se reduce el criterio de minimización a la minimización de literales. Se expondrán otros criterios en el Capítulo 5. El número de literales en una función booleana puede minimizarse por manipulaciones algebraicas. Desafortunadamente, no hay reglas específicas que seguir que garanticen la respuesta final. El único método disponible es un procedimiento de corte y ensayo empleando los postulados, teoremas básicos y cualquier otro método de manipulación que llegue a ser familiar con el uso. Los siguientes ejemplos ilustran este procedimiento.

EJEMPLO 2-1: Simplifique la siguiente función booleana a un número mínimo de literales.

1. $x + x'y = (x + x')(x + y) = 1 \cdot (x + y) = x + y$
2. $x(x' + y) = xx' + xy = 0 + xy = xy$
3. $x'y'z + x'yz + xy' = x'z(y' + y) + xy' = x'z + xy'$
4. $xy + x'z + yz = xy + x'z + yz(x + x')$
 $= xy + x'z + xyz + x'yz$
 $= xy(1 + z) + x'z(1 + y)$
 $= xy + x'z$
5. $(x + y)(x' + z)(y + z) = (x + y)(x' + z)$ por la dualidad de la función 4.

Las funciones 1 y 2 son duales una de otra y utilizan expresiones duales en los pasos correspondientes. La función 3 muestra la igualdad de las funciones F_3 y F_4 , expuestas con anterioridad. La cuarta ilustra el hecho de que un incremento en el número de literales algunas veces conduce a una expresión final más simple. La función 5 no se minimiza en forma directa, pero puede derivarse del dual de los pasos usados para derivar la función 4.

Complemento de una función

El complemento de una función F es F' y se obtiene por el intercambio de números 0 a números 1 y de números 1 a números 0 en el valor de F . El complemento de una función puede derivarse en forma algebraica mediante el teorema de De Morgan. Este par de teoremas se lista en la Tabla 2-1 para dos variables. Los teoremas de De Morgan pueden ampliarse a tres o más variables. La forma de tres variables del primer

teorema de De Morgan se deriva a continuación. Los postulados y los teoremas son los que se listan en la Tabla 2-1.

$$\begin{aligned}
 (A + B + C)' &= (A + X)' && \text{sea } B + C = X \\
 &= A'X' && \text{por el teorema 5(a) (De Morgan)} \\
 &= A' \cdot (B + C)' && \text{se sustituye } B + C = X \\
 &= A' \cdot (B'C') && \text{por el teorema 5(a) (De Morgan)} \\
 &= A'B'C' && \text{por el teorema 4(b) (asociativo)}
 \end{aligned}$$

Los teoremas de De Morgan para cualquier número de variables son semejantes en forma al caso de dos variables y pueden derivarse por sustituciones sucesivas en forma similar al método usado en la derivación anterior. Estos teoremas pueden generalizarse como sigue:

$$\begin{aligned}
 (A + B + C + D + \dots + F)' &= A'B'C'D' \dots F' \\
 (ABCD \dots F)' &= A' + B' + C' + D' + \dots + F'
 \end{aligned}$$

La forma generalizada del teorema de De Morgan enuncia que el complemento de una función se obtienen por el intercambio de los operadores AND y OR y complementando cada literal.

EJEMPLO 2-2: Encuentre el complemento de las funciones $F_1 = x'yz' + x'y'z$ y $F_2 = x(y'z' + yz)$. Se aplica el teorema de De Morgan cuantas veces sea necesario y se obtienen los complementos como sigue:

$$\begin{aligned}
 F'_1 &= (x'yz' + x'y'z)' = (x'yz')'(x'y'z)' = (x + y' + z)(x + y + z') \\
 F'_2 &= [x(y'z' + yz)]' = x' + (y'z' + yz)' = x' + (y'z')' \cdot (yz)' \\
 &= x' + (y + z)(y' + z')
 \end{aligned}$$

Un procedimiento más simple para derivar el complemento de una función es tomar la dual de la función y complementar cada literal. Este método se sigue del teorema generalizado de De Morgan. Recuérdese que la dual de una función se obtiene por el intercambio de los operadores AND y OR y los 1 y los 0.

EJEMPLO 2-3: Obtenga el complemento de las funciones F_1 y F_2 del Ejemplo 2-2, tomando sus duales y complementando cada literal.

1. $F_1 = x'yz' + x'y'z$.
La dual de F_1 es $(x' + y + z')(x' + y' + z)$.
Complemento de cada literal: $(x + y' + z)(x + y + z') = F'_1$.
2. $F_2 = x(y'z' + yz)$.
La dual de F_2 es $x + (y' + z')(y + z)$.
Complemento de cada literal: $x' + (y + z)(y' + z') = F'_2$.

2-5 FORMAS CANONICA Y ESTANDAR

Mintérminos y maxtérminos

Una variable binaria puede aparecer ya sea en forma normal (x) o en su forma complementaria (x'). Ahora considérense dos variables binarias x y y combinadas con un operador AND. Ya que cada variable puede aparecer en cualquier forma, hay cuatro combinaciones posibles: $x'y'$, $x'y$, xy' y xy . Cada uno de esos cuatro términos AND representa una de las áreas diferentes en el diagrama de Venn en la Fig. 2-1 y se denomina un *mintérmino* o un *producto estándar*. En forma semejante, pueden combinararse n variables para formar 2^n mintérminos. Los 2^n mintérminos diferentes pueden determinarse por un método similar al que se muestra en la Tabla 2-3 para tres variables. Los números binarios desde 0 a $2^n - 1$ se listan bajo las n variables. Cada mintérmino se obtiene de un término AND de las n variables, con cada variable vuelta prima si el bit correspondiente del número binario es un 0 y no prima si es un 1. En la tabla también se muestra un símbolo para cada mintérmino y está en la forma m_j , donde j indica el equivalente decimal del número binario del mintérmino denotado.

De manera semejante, n variables forman un término OR, con cada variable vuelta prima o no prima, proporcionando 2^n combinaciones posibles, denominadas *maxtérminos* o *sumas estándar*. Los ocho maxtérminos para tres variables, junto con su denominación simbólica, se listan en la Tabla 2-3. Cualesquiera 2^n maxtérminos para n variables pueden determinarse en forma similar. Cada maxtérmino se obtiene de un término OR de las n variables, con cada variable no prima si el bit correspondiente es 0 y prima si es un 1.* Obsérvese que cada maxtérmino es el complemento de su mintérmino correspondiente y viceversa.

*En algunos libros se define maxtérmino como un término OR de n variables, con cada variable sin prima si el bit es un 1 y con prima si es un 0. La definición que se ha adoptado en este libro es preferible, ya que lleva a conversiones más simples entre funciones del tipo maxtérmino y mintérmino.

TABLA 2-3 Mintérminos y maxtérminos para tres variables binarias.

Mintérminos				Maxtérminos	
x	y	z	Término	Término	Designación
0	0	0	$x'y'z'$	$x + y + z$	M_0
0	0	1	$x'y'z$	$x + y + z'$	M_1
0	1	0	$x'yz'$	$x + y' + z$	M_2
0	1	1	$x'yz$	$x + y' + z'$	M_3
1	0	0	$xy'z'$	$x' + y + z$	M_4
1	0	1	$xy'z$	$x' + y + z'$	M_5
1	1	0	xyz'	$x' + y' + z$	M_6
1	1	1	xyz	$x' + y' + z'$	M_7

Una función booleana puede expresarse en forma algebraica mediante una tabla de verdad dada, formando un mintérmino para cada combinación de variables que produce un 1 en la función y, tomando entonces los OR de todos esos términos. Por ejemplo, la función F_1 en la Tabla 2-4 se determina al expresar las combinaciones 001, 100 y 111 como $x'y'z$, $xy'z'$ y xyz , respectivamente. Ya que cada uno de estos min térmimos resulta en $F_1 = 1$, se debe tener:

$$f_1 = x'y'z + xy'z' + xyz = m_1 + m_4 + m_7$$

De manera semejante, puede verificarse con facilidad que:

$$f_2 = x'yz + xy'z + xyz' + xyz = m_3 + m_5 + m_6 + m_7$$

Estos ejemplos demuestran una propiedad importante del álgebra booleana: Cualquier función booleana puede expresarse como una suma de mintérminos (por “suma” se entiende la aplicación del operador OR en los términos).

Ahora considérese el complemento de una función booleana. A partir de la tabla de verdad puede leerse al formar un mintérmino para cada combinación que produce un 0 en la función y aplicando el operador OR a esos términos. El complemento de f_1 se lee como:

$$f'_1 = x'y'z' + x'yz' + x'yz + xy'z + xyz'$$

Si se toma el complemento de f'_1 , se obtiene la función f_1 :

$$\begin{aligned} f_1 &= (x + y + z)(x + y' + z)(x + y' + z')(x' + y + z')(x' + y' + z) \\ &= M_0 \cdot M_2 \cdot M_3 \cdot M_5 \cdot M_6 \end{aligned}$$

En forma similar, es posible leer la expresión para f_2 de la tabla:

$$\begin{aligned} f_2 &= (x + y + z)(x + y + z')(x + y' + z)(x' + y + z) \\ &= M_0 M_1 M_2 M_4 \end{aligned}$$

TABLA 2-4 Función de tres variables

x	y	z	Función f_1	Función f_2
0	0	0	0	0
0	0	1	1	0
0	1	0	0	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Estos ejemplos demuestran una segunda propiedad importante del álgebra booleana: Cualquier función booleana puede expresarse como un producto de maxterminos (por “producto” se entiende que se aplica el operador AND a los términos). El procedimiento para obtener el producto de los maxterminos en forma directa de la tabla de verdad es como sigue. Fórmese un maxtermino para cada combinación de las variables que produce un 0 en la función, y entonces fórmese AND de todos los maxterminos. Las funciones booleanas expresadas como una suma de mintérminos o producto de maxterminos se dice que están en *forma canónica*.

Suma de mintérminos

Con anterioridad se enunció que para n variables binarias, pueden obtenerse 2^n mintérminos diferentes y, que cualquier función booleana puede expresarse como una suma de mintérminos. Los mintérminos cuya suma define la función booleana son los que dan los 1 de la función en una tabla de verdad. Ya que la función puede ser 1 o bien 0 para cada mintérmino, y puesto que hay 2^n mintérminos, pueden calcularse las funciones posibles que es factible formarse con n variables para hacer 2^{2^n} . Algunas veces es conveniente expresar la función booleana en la forma de su suma de mintérminos. Si no puede hacerse en esta forma, entonces puede realizarse primero por la expansión de la expresión en una suma de términos AND. Después cada término se inspecciona para ver si contiene todas las variables. Si se han perdido una o más variables, se aplica el operador AND con una expresión como $x + x'$, en donde x es una de las variables perdidas. El siguiente ejemplo aclara este procedimiento.

EJEMPLO 2-4: Exprese la función booleana $F = A + B'C$ en una suma de mintérminos. La función tiene tres variables A , B y C . El primer término A pierde dos variables; por tanto:

$$A = A(B + B') = AB + AB'$$

Todavía se pierde una variable:

$$\begin{aligned} A &= AB(C + C') + AB'(C + C') \\ &= ABC + ABC' + AB'C + AB'C' \end{aligned}$$

El segundo término $B'C$ pierde una variable:

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

Combinando todos los términos, se tiene:

$$\begin{aligned} F &= A + B'C \\ &= ABC + ABC' + AB'C + AB'C' + AB'C + A'B'C \end{aligned}$$

Pero $AB'C$ aparece dos veces y, de acuerdo con el teorema 1 ($x + x = x$), es posible quitar uno de ellos. Reordenando los mintérminos de manera ascendente, por último se obtiene:

$$\begin{aligned} F &= A'B'C + AB'C' + AB'C + ABC' + ABC \\ &= m_1 + m_4 + m_5 + m_6 + m_7 \end{aligned}$$

Algunas veces es conveniente expresar la función booleana, cuando está en su suma de mintérminos, en la siguiente notación abreviada:

$$F(A, B, C) = \Sigma (1, 4, 5, 6, 7)$$

El símbolo de suma Σ representa el operador OR que opera en los términos; los números siguientes son los mintérminos de la función. Las letras entre paréntesis que siguen a F forman una lista de las variables en el orden tomado cuando el mintérmino se convierte en un término AND.

Producto de los maxtérminos

Cada una de las funciones 2^n de n variables binarias también puede expresarse como un producto de maxtérminos. Para expresar la función booleana como un producto de maxtérminos, primero debe llevarse a una forma de términos OR. Es posible hacer esto por el uso de la ley distributiva $x + yz = (x + y)(x + z)$. Entonces, cualquier variable perdida x en cada término 0 se opera a OR con xx' . Este procedimiento se aclara en el siguiente ejemplo.

EJEMPLO 2-5: Exprese la función booleana $F = xy + x'z$ en un producto de forma maxtérmino. Primero convierta la función en términos OR usando la ley distributiva:

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

La función tiene tres variables: x , y y z . Cada término OR pierde una variable; por tanto:

$$\begin{aligned} x' + y &= x' + y + zz' = (x' + y + z)(x' + y + z') \\ x + z &= x + z + yy' = (x + y + z)(x + y' + z) \\ y + z &= y + z + xx' = (x + y + z)(x' + y + z) \end{aligned}$$

Se combinan todos los términos y se eliminan los que aparecen más de una vez y, por último, se obtiene:

$$\begin{aligned} F &= (x + y + z)(x + y' + z)(x' + y + z)(x' + y + z') \\ &= M_0M_2M_4M_5 \end{aligned}$$

Una forma conveniente de expresar esta función es como sigue:

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

El símbolo de producto, Π , denota la operación AND de maxtérminos; los números son los maxtérminos de la función.

Conversión entre formas canónicas

El complemento de una función expresada como suma de mintérminos es igual a la suma de los mintérminos perdidos de la función original. Esto se debe a que la función original está expresada por los mintérminos que hacen la función igual a 1, mientras que su complemento es un 1 para los términos en los que la función es un 0. Como ejemplo, considérese la función:

$$F(A, B, C) = \Sigma(1, 4, 5, 6, 7)$$

Esta tiene un complemento que puede expresarse como:

$$F'(A, B, C) = \Sigma(0, 2, 3) = m_0 + m_2 + m_3$$

Ahora bien, si se toma el complemento de F' por el teorema de De Morgan, se obtiene F en una forma diferente:

$$F = (m_0 + m_2 + m_3)' = m'_0 \cdot m'_2 \cdot m'_3 = M_0 M_2 M_3 = \Pi(0, 2, 3)$$

La última conversión se sigue de la definición de mintérminos y maxtérminos como se muestra en la Tabla 2-3. Por la tabla, es claro que la siguiente relación es válida:

$$m'_j = M_j$$

Esto es, el maxtérmino con subíndice j es un complemento del mintérmino con el mismo subíndice j y viceversa.

El último ejemplo demuestra la conversión entre una función expresada en sumas de mintérminos y su equivalente en producto de maxtérminos. Un argumento similar mostrará que la conversión entre el producto de maxtérminos y la suma de mintérminos es similar. Ahora se enuncia un procedimiento general de conversión. Para convertir de una forma canónica a otra, se intercambian los símbolos Π y Σ y se listan los números perdidos de la forma original. Como otro ejemplo, la función:

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

se expresa en la forma de producto de maxtérminos. Su conversión en suma de mintérminos es:

$$F(x, y, z) = \Sigma(1, 3, 6, 7)$$

Obsérvese que, con objeto de encontrar los términos perdidos, debe tomarse en cuenta que el número total de mintérminos o maxtérminos es 2^n , donde n es el número de variables binarias en la función.

Formas estándar

Las dos formas canónicas del álgebra booleana son formas básicas que se obtienen al leer una función de la tabla de verdad. Estas formas muy rara vez son las que tienen el menor número de literales, debido a que cada mintérmino y maxtérmino debe contener, por definición, todas las variables ya sea complementadas o sin complementar.

Otra forma de expresar las funciones booleanas es la forma estándar. En esta configuración, los términos que forman la función pueden contener uno, dos o cualquier número de literales. Hay dos tipos de formas estándar: la suma de productos y el producto de sumas.

La *suma de productos* es una expresión booleana que contiene términos AND, llamados *términos producto*, de una o más literales cada uno. La *suma* denota la operación OR de esos términos. Un ejemplo de una función expresada en suma de productos es:

$$F_1 = y' + xy + x'yz'$$

La expresión tiene tres términos producto de una, dos y tres literales cada uno, respectivamente. Su suma es, en efecto, una operación OR.

Un *producto de sumas* es una expresión booleana que contiene términos OR, llamados *términos suma*. Cada término puede tener cualquier número de literales. El *producto* denota la operación AND de esos términos. Un ejemplo de una función expresada en producto de sumas es:

$$F_2 = x(y' + z)(x' + y + z' + w)$$

Esta expresión tiene tres términos suma de una, dos y cuatro literales cada uno. El producto es una operación AND. El uso de las palabras *producto* y *suma* surge de la similitud de la operación AND con el producto aritmético (multiplicación) y la semejanza de la operación OR con la suma aritmética (adición).

Una función booleana puede expresarse en una forma no estándar. Por ejemplo, la función:

$$F_3 = (AB + CD)(A'B' + C'D')$$

no es una suma de productos ni un producto de suma. Puede cambiarse a una forma estándar usando la ley distributiva para eliminar los paréntesis:

$$F_3 = A'B'CD + ABC'D'$$

2-6 OTRAS OPERACIONES LOGICAS

Cuando los operadores binarios AND y OR se colocan entre dos variables x y y , forman dos funciones booleanas $x \cdot y$ y $x + y$, respectivamente. Se enunció previamente que hay 2^n funciones para n variables binarias. Para dos variables, $n = 2$ y el

número de funciones booleanas posibles es 16. Por tanto, las funciones AND y OR son sólo dos de un total de 16 funciones posibles formadas con dos variables binarias. Sería instructivo encontrar las otras 14 funciones e investigar sus propiedades.

Las tablas de verdad para las 16 funciones formadas con dos variables binarias x y y se listan en la Tabla 2-5. En esta tabla, cada una de las 16 columnas, de F_0 a F_{15} , representa una tabla de verdad de una función posible para las dos variables dadas x y y . Obsérvese que la función está determinada por las 16 combinaciones binarias que pueden asignarse a F . Algunas de las funciones se muestran con un símbolo de operador. Por ejemplo, F_1 representa la tabla de verdad para AND y F_7 representa la tabla de verdad para OR. Los símbolos de los operadores para esas funciones son (\cdot) y $(+)$, respectivamente.

Las 16 funciones que se listan en forma de tabla de verdad pueden expresarse de manera algebraica mediante expresiones booleanas. Esto se muestra en la primera columna de la Tabla 2-6. Las expresiones booleanas que se listan se simplifican a su número mínimo de literales.

Aun cuando cada función puede expresarse en términos de las operaciones booleanas AND, OR y NOT, no hay razón para que no puedan asignarse símbolos especiales de operador para expresar las otras funciones. Tales símbolos de operador se listan en la segunda columna de la Tabla 2-6. Sin embargo, todos los nuevos símbolos que se muestran, excepto para el símbolo del operador OR-excluyente \oplus , no son de uso común por los diseñadores digitales.

Cada una de las funciones de la Tabla 2-6 se lista con un nombre que la acompaña y un comentario que explica la función en cierta forma. Las 16 funciones listadas pueden subdividirse en tres categorías:

1. Dos funciones que producen un constante 0 o 1.
2. Cuatro funciones con operaciones unitarias de complemento y transferencia.
3. Diez funciones con operadores binarios que definen ocho operaciones diferentes AND, OR, NAND, NOR, O-excluyente, equivalencia, inhibición e implicación.

Cualquier función puede ser igual a una constante, pero una función binaria puede ser igual sólo a 1 o 0. La función complemento produce el complemento de cada

TABLA 2-5 Tablas de verdad para las 16 funciones de dos variables binarias

x	y	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}
0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	
1	1	0	1	0	1	0	1	0	1	0	1	1	0	1	0	1	
Símbolo Operador		.	/	/		\oplus	+	\downarrow	\odot	'	\subset	'	\supset	\uparrow			

TABLA 2-6 Expresiones booleanas para las 16 funciones de dos variables

Funciones booleanas	Símbolo del operador	Nombre	Comentarios
$F_0 = 0$		Nulo	Constante binaria 0
$F_1 = xy$	$x \cdot y$	AND	x y y
$F_2 = xy'$	x/y	Inhibición	x pero no y
$F_3 = x$		Transferencia	x
$F_4 = x'y$	y/x	Inhibición	y pero no x
$F_5 = y$		Transferencia	y
$F_6 = xy' + x'y$	$x \oplus y$	Excluyente-OR	x o y pero no ambas
$F_7 = x + y$	$x + y$	OR	x o y
$F_8 = (x + y)'$	$x \downarrow y$	NOR	NOT-OR
$F_9 = xy + x'y'$	$x \odot y$	Equivalencia*	x igual a y
$F_{10} = y'$	y'	Complemento	No y
$F_{11} = x + y'$	$x \subset y$	Implicación	Si y , entonces x
$F_{12} = x'$	x'	Complemento	No x
$F_{13} = x' + y$	$x \supset y$	Implicación	Si x , entonces y
$F_{14} = (xy)'$	$x \uparrow y$	NAND	NOT-AND
$F_{15} = 1$		Identidad	Constante binaria 1

*La equivalencia también se conoce como *igualdad*, *coincidencia* y *excluyente NOR*.

una de las variables binarias. Una función que es igual a una variable de entrada recibe el nombre de *transferencia*, ya que la variable x o y se transfiere a través de la compuerta que forma la función sin cambiar su valor. De los diez operadores binarios, cuatro (que corresponden a las funciones de inhibición e implicación) los utilizan especialistas en lógica, pero rara vez se usan en la lógica de computadora. Se han mencionado operadores AND y OR junto con el álgebra booleana. Las otras cuatro funciones se emplean en forma extensa en el diseño de sistemas digitales.

La función NOR es el complemento de la función OR y su nombre es la abreviatura de *no-OR*. En forma similar, NAND es el complemento de AND y es una abreviatura de *no-AND*. Excluyente OR se abrevia XOR o EOR es similar a OR, pero excluye la combinación *tanto de x como de y* cuando son iguales a 1. La equivalencia es una función que es 1 cuando dos variables binarias son iguales, esto es, cuando ambas son 0 o ambas son 1. La excluyente OR y las funciones de equivalencia son los complementos una de otra. Esto puede verificarse con facilidad por la inspección de la Tabla 2-5. La tabla de verdad para la excluyente OR es F_6 y para la equivalencia es F_9 , y, estas dos funciones son los complementos una de la otra. Por esta razón, la función de equivalencia con frecuencia se denomina excluyente NOR, es decir, excluyente-OR-NOT.

El álgebra booleana, como se define en la Sección 2-2, tiene dos operadores binarios, que se han denominado AND y OR y un operador unario, NOT (complemento). Por las definiciones, se deduce cierto número de propiedades de estos operadores y ahora se han definido otros operadores binarios en términos de ellos. No hay

nada excepcional en este procedimiento. Se puede empezar también con el operador NOR (\downarrow) , por ejemplo, y definir después AND, OR y NOT en términos de él. Sin embargo, hay buenas razones para introducir el álgebra booleana en el modo que se ha hecho. Los conceptos de “y”, “o” y “no” son familiares y las personas los utilizan para expresar ideas lógicas en la vida cotidiana. No obstante, los postulados de Huntington reflejan la naturaleza dual del álgebra, con énfasis en la simetría de + y · de uno con respecto a otro.

2-7 COMPUERTAS LOGICAS DIGITALES

Ya que las funciones booleanas se expresan en términos de operaciones AND, OR y NOT, es fácil implantar una función booleana con estos tipos de compuertas. La posibilidad de construir compuertas para otras operaciones lógicas es de interés práctico. Los factores que hay que pesar cuando se considera la construcción de otros tipos de compuertas lógicas son (1) la factibilidad y economía de producir la compuerta con componentes físicos, (2) la posibilidad de extender la compuerta a más de dos entradas, (3) las propiedades básicas del operador binario como conmutatividad y asociatividad y (4), la habilidad de la compuerta para implantar compuertas booleanas solas o junto con otras compuertas.

De las 16 funciones que se definen en la Tabla 2-6, dos son iguales a una constante y otras cuatro se repiten dos veces. Solo que dan diez funciones que considerar como candidatos para compuertas lógicas. Dos, inhibición y complicación, no son conmutativas o asociativas y, por tanto, no es práctico usarlas como compuertas lógicas estándar. Las otras ocho: complemento, transferencia, AND, OR, NAND, NOR, excluyente-OR, y equivalencia, se utilizan como compuertas estándar en el diseño digital.

Los símbolos gráficos y las tablas de verdad de las ocho compuertas se muestran en la Fig. 2-5. Cada compuerta tiene una o dos variables binarias de entrada designadas por x y y y una variable binaria de salida designada por F . Los circuitos AND, OR e inversor se definen en la Fig. 1-6. El circuito inversor invierte el sentido lógico de una variable binaria. Produce la función NOR o complemento. El pequeño círculo en la salida del símbolo gráfico de un inversor designa el complemento lógico. El símbolo de triángulo por sí mismo denota un circuito buffer. Un buffer produce la función de *transferencia* pero no produce alguna operación lógica particular, ya que el valor binario de la salida es igual al valor binario de la entrada. El circuito se usa simplemente para amplificación de potencia de la señal y es equivalente a dos inversores conectados en cascada.

La función NAND es el complemento de la función AND, como se indica por un símbolo gráfico, que consta de un símbolo gráfico AND seguido de un círculo pequeño. La función NOR es el complemento de la función OR y usa un símbolo gráfico OR seguido de un círculo pequeño. Las compuertas NAND y NOR se utilizan en forma extensa como compuertas lógicas estándar y de hecho se emplean más que las compuertas AND y OR. Esto se debe a que las compuertas NAND y NOR se construyen fácilmente con circuitos de transistores y a que las funciones booleanas pueden implementarse con sencillez con dichas compuertas.

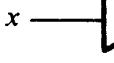
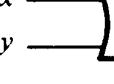
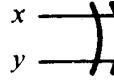
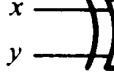
Nombre	Símbolo gráfico	Función algebraica	Tabla de verdad															
AND		$F = xy$	<table border="1" data-bbox="1175 359 1321 538"> <tr> <th>x</th><th>y</th><th>F</th></tr> <tr> <td>0</td><td>0</td><td>0</td></tr> <tr> <td>0</td><td>1</td><td>0</td></tr> <tr> <td>1</td><td>0</td><td>0</td></tr> <tr> <td>1</td><td>1</td><td>1</td></tr> </table>	x	y	F	0	0	0	0	1	0	1	0	0	1	1	1
x	y	F																
0	0	0																
0	1	0																
1	0	0																
1	1	1																
OR		$F = x + y$	<table border="1" data-bbox="1175 570 1321 749"> <tr> <th>x</th><th>y</th><th>F</th></tr> <tr> <td>0</td><td>0</td><td>0</td></tr> <tr> <td>0</td><td>1</td><td>1</td></tr> <tr> <td>1</td><td>0</td><td>1</td></tr> <tr> <td>1</td><td>1</td><td>1</td></tr> </table>	x	y	F	0	0	0	0	1	1	1	0	1	1	1	1
x	y	F																
0	0	0																
0	1	1																
1	0	1																
1	1	1																
Inversor		$F = x'$	<table border="1" data-bbox="1175 770 1321 897"> <tr> <th>x</th><th>F</th></tr> <tr> <td>0</td><td>1</td></tr> <tr> <td>1</td><td>0</td></tr> </table>	x	F	0	1	1	0									
x	F																	
0	1																	
1	0																	
Buffer		$F = x$	<table border="1" data-bbox="1175 918 1321 1045"> <tr> <th>x</th><th>F</th></tr> <tr> <td>0</td><td>0</td></tr> <tr> <td>1</td><td>1</td></tr> </table>	x	F	0	0	1	1									
x	F																	
0	0																	
1	1																	
NAND		$F = (xy)'$	<table border="1" data-bbox="1175 1056 1321 1235"> <tr> <th>x</th><th>y</th><th>F</th></tr> <tr> <td>0</td><td>0</td><td>1</td></tr> <tr> <td>0</td><td>1</td><td>1</td></tr> <tr> <td>1</td><td>0</td><td>1</td></tr> <tr> <td>1</td><td>1</td><td>0</td></tr> </table>	x	y	F	0	0	1	0	1	1	1	0	1	1	1	0
x	y	F																
0	0	1																
0	1	1																
1	0	1																
1	1	0																
NOR		$F = (x + y)'$	<table border="1" data-bbox="1175 1267 1321 1446"> <tr> <th>x</th><th>y</th><th>F</th></tr> <tr> <td>0</td><td>0</td><td>1</td></tr> <tr> <td>0</td><td>1</td><td>0</td></tr> <tr> <td>1</td><td>0</td><td>0</td></tr> <tr> <td>1</td><td>1</td><td>0</td></tr> </table>	x	y	F	0	0	1	0	1	0	1	0	0	1	1	0
x	y	F																
0	0	1																
0	1	0																
1	0	0																
1	1	0																
Excluyente-OR (XOR)		$F = xy' + x'y \\ = x \oplus y$	<table border="1" data-bbox="1175 1457 1321 1636"> <tr> <th>x</th><th>y</th><th>F</th></tr> <tr> <td>0</td><td>0</td><td>0</td></tr> <tr> <td>0</td><td>1</td><td>1</td></tr> <tr> <td>1</td><td>0</td><td>1</td></tr> <tr> <td>1</td><td>1</td><td>0</td></tr> </table>	x	y	F	0	0	0	0	1	1	1	0	1	1	1	0
x	y	F																
0	0	0																
0	1	1																
1	0	1																
1	1	0																
Excluyente-NOR o equivalente		$F = xy + x'y' \\ = x \odot y$	<table border="1" data-bbox="1175 1668 1321 1848"> <tr> <th>x</th><th>y</th><th>F</th></tr> <tr> <td>0</td><td>0</td><td>1</td></tr> <tr> <td>0</td><td>1</td><td>0</td></tr> <tr> <td>1</td><td>0</td><td>0</td></tr> <tr> <td>1</td><td>1</td><td>1</td></tr> </table>	x	y	F	0	0	1	0	1	0	1	0	0	1	1	1
x	y	F																
0	0	1																
0	1	0																
1	0	0																
1	1	1																

Figura 2-5 Compuertas lógicas digitales.

La compuerta excluyente-OR tiene un símbolo gráfico similar al de la compuerta OR excepto por la línea adicional curva en el lado de entrada. La equivalencia, o compuerta excluyente NOR es el complemento de la excluyente OR, como se indica por el círculo pequeño en el lado de salida del símbolo gráfico.

Extensión a entradas múltiples

Las compuertas que se muestran en la Fig. 2-5, excepto por el inversor y el buffer, pueden extenderse para tener más de dos entradas. Una compuerta puede extenderse para tener entradas múltiples si la operación binaria que representa es conmutativa y asociativa. Las operaciones AND y OR, definidas en el álgebra booleana, poseen esas dos propiedades. Para la función OR se tiene:

$$x + y = y + x \quad \text{conmutativa}$$

y

$$(x + y) + z = x + (y + z) = x + y + z \quad \text{asociativa}$$

lo cual indica que las entradas de compuerta pueden intercambiarse y que la función O puede extenderse a tres o más variables.

Las funciones NAND y NOR son conmutativas y sus compuertas pueden extenderse para tener más de dos entradas, siempre que se modifique ligeramente la definición de la operación. La dificultad es que los operadores NAND y NOR no son asociativos, esto es, $(x \downarrow y) \downarrow z \neq x \downarrow (y \downarrow z)$, como se muestra en la Fig. 2-6 y a continuación:

$$\begin{aligned} (x \downarrow y) \downarrow z &= [(x + y)' + z]' = (x + y)z' = xz' + yz' \\ x \downarrow (y \downarrow z) &= [x + (y + z)']' = x'(y + z) = x'y + x'z \end{aligned}$$

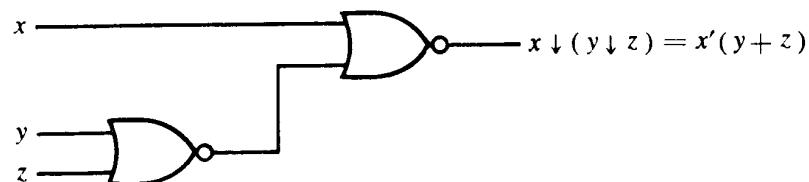
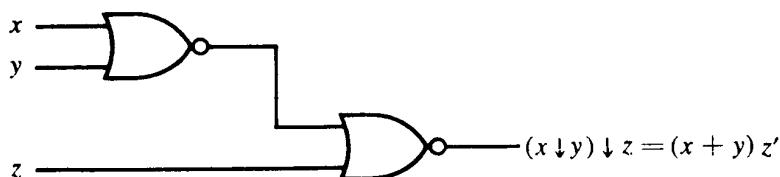


Figura 2-6 Demostración de la no asociabilidad del operador NOR; $(x \downarrow y) \downarrow z \neq x \downarrow (y \downarrow z)$.

Para superar esta dificultad, se define la compuerta múltiple NOR (o NAND) como una compuerta complementada OR (o AND). Así, por definición, se tiene:

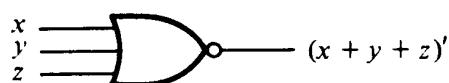
$$\begin{aligned}x \downarrow y \downarrow z &= (x + y + z)' \\x \uparrow y \uparrow z &= (xyz)'\end{aligned}$$

Los símbolos gráficos para las compuertas de tres entradas se muestran en la Fig. 2-7. Al indicar por escrito operaciones NOR y NAND en cascada, deben utilizarse los paréntesis correctos para indicar la secuencia apropiada de las compuertas. Para demostrar esto, considérese el circuito de la Fig. 2-7(c). La función booleana para el circuito debe escribirse como:

$$F = [(ABC)'(DE)']' = ABC + DE$$

La segunda expresión se obtiene a partir del teorema de De Morgan. Muestra también que una expresión en suma de productos puede implantarse con compuertas NAND. En las Secciones 3-6, 4-7 y 4-8 puede encontrarse la exposición detallada de las compuertas NAND y NOR.

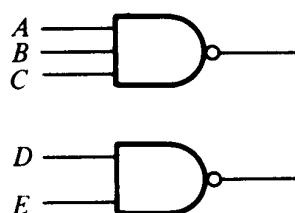
Las compuertas excluyente OR y de equivalencia son conmutativas y asociativas y pueden extenderse a más de dos entradas. Sin embargo, las compuertas excluyente OR de entradas múltiples no son usuales desde el punto de vista del hardware. De hecho, incluso una función de dos entradas por lo común se construye con otros tipos de compuertas. Además, la definición de esas funciones debe modificarse cuando se extienden a más de dos variables. La excluyente-OR es una función *ímpar*, esto es, es igual a 1 si las variables de entrada tienen un número ímpar de 1. La función de equivalencia es una función *par*, es decir, es igual a 1 si las variables de entrada tienen un número par de 0. La construcción de una función excluyente OR de tres entradas se muestra en la Fig. 2-8. En forma normal se implementa con compuertas de dos entradas en cascada, como se muestra en (a). De manera gráfica, puede representarse con una sola compuerta de tres entradas como se muestra en (b). La tabla de verdad en



(a) Compuerta NOR de tres entradas

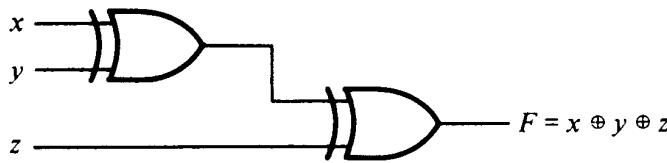


(b) Compuerta NAND de tres entradas

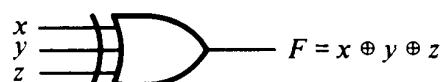


(c) Compuertas NAND en cascada

Figura 2-7 Compuertas de entradas múltiples NOR y NAND puestas en cascada.



(a) Uso de compuertas con dos entradas



(b) Una compuerta de tres entradas

x	y	z	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

(c) Tabla de verdad

Figura 2-8 Compuerta excluyente OR de tres entradas.

(c) indica con claridad que la salida F es igual a 1 si sólo una entrada es igual a 1 o si todas las tres entradas son iguales a 1, esto es, cuando el número total de 1 en las variables de entrada es *impar*. En la Sección 4-9 puede encontrarse un análisis adicional de la excluyente OR y la equivalencia.

2-8 FAMILIAS LOGICAS DIGITALES IC

El IC se introdujo en la Sección 1-9, donde se estableció que los circuitos digitales se construyen en forma invariable con IC. En las secciones previas se han expuesto diversas compuertas lógicas digitales, ahora ya están dadas las condiciones para presentar las compuertas IC y exponer sus propiedades generales.

Las compuertas digitales IC se clasifican no sólo por su operación lógica, sino también por la familia de circuitos lógicos a las cuales pertenecen. Cada familia lógica tiene su propio circuito electrónico básico con el cual se desarrollan circuitos y funciones digitales más complejos. El circuito básico de cada familia es una compuerta NAND o bien una compuerta NOR. Los componentes electrónicos que se emplean en la construcción del circuito básico por lo general se utilizan para nombrar la familia lógica. En el comercio se han introducido muchas familias lógicas diferentes de IC digitales. Las que han alcanzado un amplio uso popular se listan a continuación.

TTL	Lógica de transistor-transistor
ECL	Lógica de emisor acoplado
MOS	Semiconductor de óxido metálico
CMOS	Semiconductor complementario de óxido metálico
I ² L	Lógica de inyección integrada

La lógica TTL tiene una lista extensa de funciones digitales y hoy día es la familia lógica más popular. La lógica ECL se utiliza en sistemas que requieren

operaciones de alta velocidad. Las MOS e I²L se usan en circuitos que requieren alta densidad de componentes y la CMOS se emplea en sistemas que necesitan bajo consumo de potencia.

El análisis del circuito electrónico en cada familia básica se presenta en el Capítulo 10. El lector familiarizado con la electrónica básica puede consultar el Capítulo 10 para así conocer estos circuitos electrónicos. Aquí la exposición se limita a las propiedades generales de las diversas compuertas IC disponibles en forma comercial.

Debido a la alta densidad con la cual pueden fabricarse los transistores en MOS e I²L, estas dos familias son las que más se utilizan para las funciones LSI. Las otras tres familias, TTL, ECL, y CMOS, tienen dispositivos LSI y también un gran número de dispositivos MSI y SSI. Los dispositivos SSI son los que incluyen un pequeño número de compuertas o flip-flops (presentados en la Sección 6-2) en un paquete IC. El límite del número de circuitos en los dispositivos SSI es el número de clavijas en el paquete. Por ejemplo, un paquete de 14 clavijas puede acomodar sólo cuatro compuertas de dos entradas, debido a que cada compuerta requiere tres clavijas externas, dos para cada una de las entradas y una para la salida, con un total de 12 clavijas. Las dos clavijas restantes se necesitan para suministrar potencia a los circuitos.

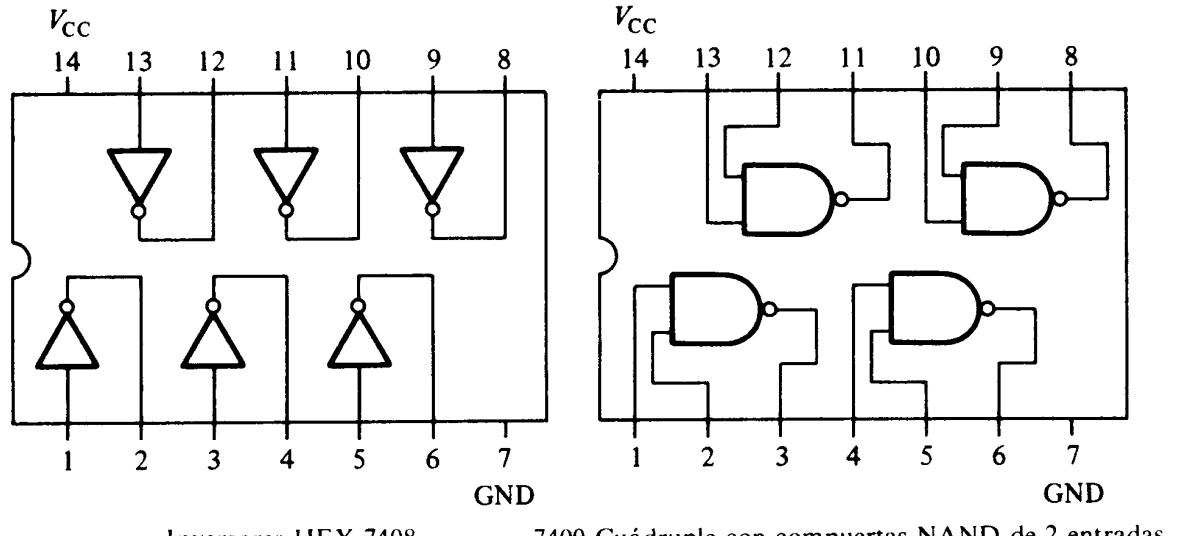
Algunos circuitos típicos SSI se muestran en la Fig. 2-9. Cada IC se encapsula un paquete de 14 o 16 clavijas. Las clavijas se numeran a lo largo de los dos lados del paquete y especifican las conexiones que pueden hacerse. Las compuertas dibujadas dentro de los IC son sólo para información y no pueden verse debido a que el paquete IC real aparece como se muestra en la Fig. 1-8.

Los IC de la familia TTL por lo común se distinguen por designaciones numéricas como las series 5400 y 7400. La primera tiene amplios márgenes de temperatura de operación, adecuados para uso militar y, la segunda tiene márgenes más reducidos de temperatura, adecuados para uso industrial. La designación numérica de la serie 7400 significa que los paquetes IC están numerados como 7400, 7401, 7402, etc. Algunos proveedores ponen a la disposición IC de la familia TTL con denominaciones numéricas diferentes, como las series 9000 u 8000.

En la Fig. 2-9(a) se muestran dos circuitos TTL SSI. La serie 7404 proporciona seis (hex) inversores en un paquete. La serie 7400 proporciona cuatro (cuádruple) puertas NAND de dos entradas. Las terminales marcadas V_{CC} y GND son las clavijas de suministro de potencia que requieren un voltaje de 5 volts para la operación apropiada.

El tipo más común de ECL se designa como la serie 10 000. En la Fig. 2-9(b) se muestran dos circuitos ECL. La serie 10102 proporciona compuertas NOR de dos entradas. Obsérvese que una compuerta ECL puede tener dos salidas, una para la función NOR y otra para la función 0 (clavija 9 del 10102 IC). El 10107 IC proporciona tres compuertas excluyentes OR. Aquí hay de nuevo dos salidas para cada compuerta; la otra salida de la función excluyente NOR o de equivalencia. Las compuertas ECL tienen tres terminales para suministro de potencia. V_{CC1} y V_{CC2} por lo común se conectan a tierra y V_{EE} a un suministro de -5.2 volt.

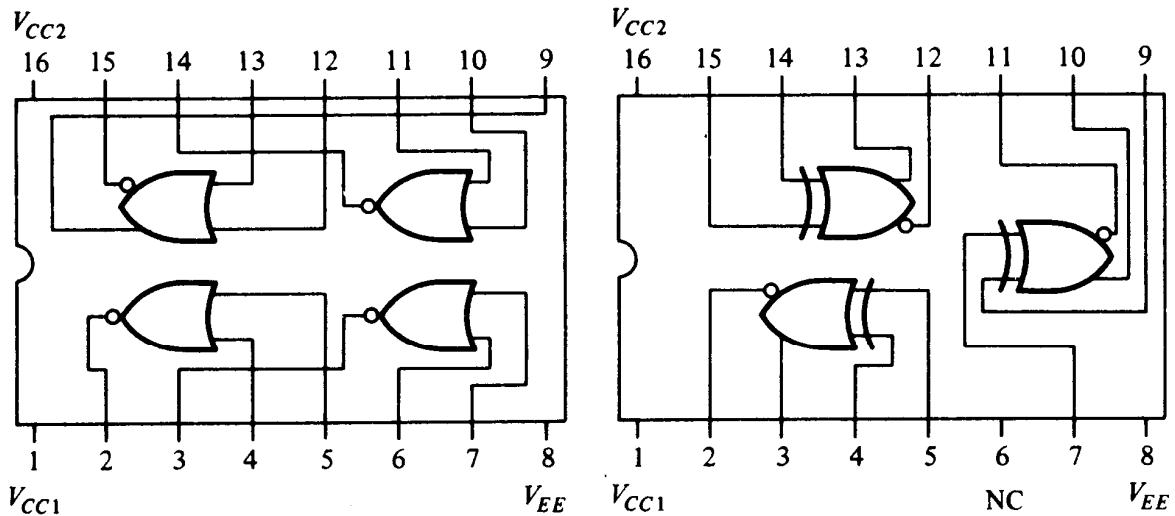
Los circuitos CMOS de la serie 4000 se muestran en la Fig. 2-9(c). Sólo pueden acomodarse en el 4002 dos compuertas NOR de cuatro entradas, debido a la limitación de clavijas. El tipo 4059 proporciona seis compuertas buffer. Ambos ICs tienen



Inversores HEX-7408

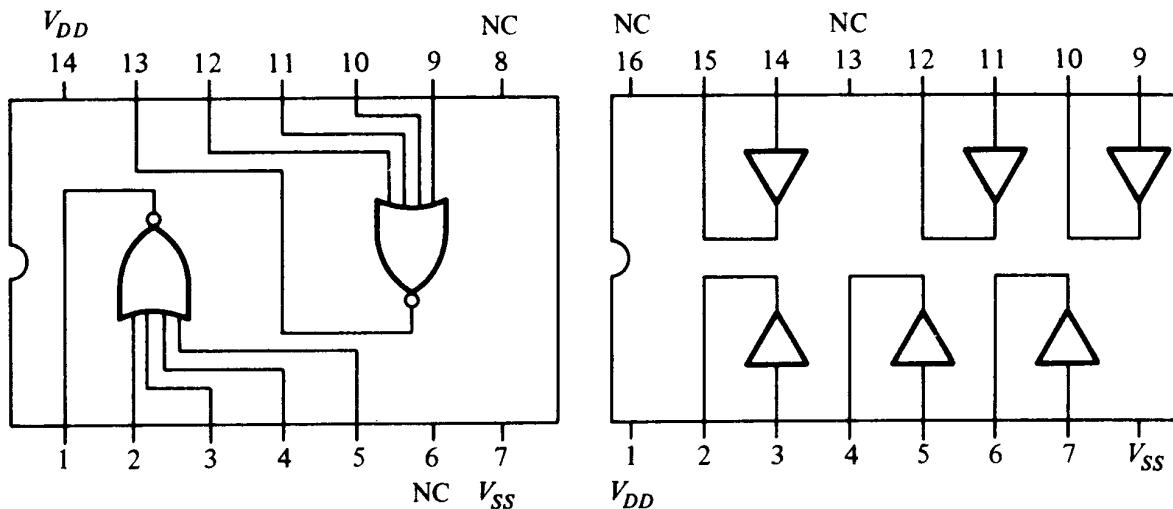
7400-Cuádruple con compuertas NAND de 2 entradas

(a) Compuertas TTL.



10102-Cuádruple con compuertas NOR de 2 entradas 10107-Triple con compuertas excluyente OR/NOR

(b) Compuertas ECL



4002-Dual con compuertas NOR de 4 entradas

4050-Buffer Hex.

(c) Compuertas CMOS.

Figura 2-9 Algunas compuertas típicas en circuitos integrados.

dos terminales sin uso marcadas NC (no conexión). La terminal marcada V_{DD} requiere un voltaje en el suministro de potencia de 3 a 15 volts, en tanto V_{SS} por lo común se conecta a tierra.

Lógicas positiva y negativa

La señal binaria en la entrada o salida de cualquier compuerta puede tener uno de dos valores, excepto durante la transición. Un valor de señal representa la lógica 1 y el otro, la lógica 0. Ya que se asignan dos valores de señal a dos valores lógicos, existen dos diferentes asignaciones de señales a lógica. Debido al principio de dualidad de álgebra booleana, un intercambio en la asignación del valor de señal resulta en el implante de una función dual.

Considérense los dos valores de una señal binaria tal como se muestra en la Fig. 2-10. un valor debe ser más alto que el otro, ya que los dos valores deben ser diferentes con objeto de distinguir entre ellos. Se designa el nivel alto por H y el nivel bajo por L . Hay dos elecciones para la asignación del valor de lógica. La elección del nivel alto H para que represente la lógica 1, como se muestra en la Fig. 2-10(a), define un sistema de *lógica positiva*. La elección del nivel bajo L para representar la lógica 1, como se muestra en la Fig. 2-10(b), define un sistema de *lógica negativa*. Los términos *positiva* y *negativa* algunas veces pueden ser engañosos, ya que ambas señales de valor pueden ser positivas o negativas. No es la polaridad de la señal la que determina el tipo de lógica, sino más bien la asignación de valores lógicos de acuerdo con las amplitudes relativas de las señales.

Las hojas de datos de los circuitos integrados definen las funciones digitales no en términos de la lógica 1 o lógica 0, sino más bien en términos de los niveles H y L . Se deja al usuario decidir la asignación de una lógica positiva o negativa. Los voltajes de alto nivel y bajo nivel para las tres familias lógicas digitales IC se listan en la Tabla 2-7. En cada familia, hay unos márgenes de valores de voltaje que el circuito reconocerá como nivel alto o bajo. El valor típico es el que más se encuentra por lo común. En la tabla también se listan los requisitos del suministro de voltaje para cada familia como una referencia.

La familia TTL tiene valores típicos de $H = 3.5$ volts y $L = 0.2$ volts. La familia ECL tiene dos valores negativos, con $H = -0.8$ volts y $L = -1.8$ volts. Obsérvese que aunque ambos niveles son negativos, el más elevado es -0.8 . Las compuertas CMOS pueden usar un voltaje de suministro V_{DD} en cualquier parte entre 3 y 15 volts; en forma típica, utilizan ya sea 5 o 10 volts. Los valores de señal en las CMOS son una función

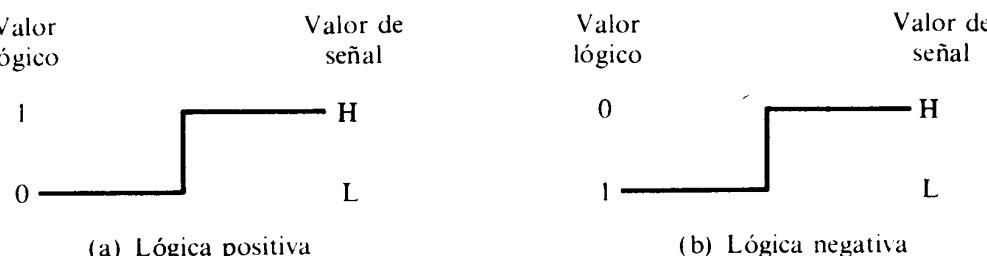


Figura 2-10 Asignación de amplitud de señal y tipo de lógica.

TABLA 2-7 Niveles H y L en las familias lógicas IC

Tipo de familia IC	Voltaje de suministro (V)	Alto nivel de voltaje (V)		Bajo nivel de voltaje (V)	
		Márgenes	Típico	Márgenes	Típico
TTL	$V_{CC} = 5$	2.4–5	3.5	0–0.4	0.2
ECL	$V_{EE} = -5.2$	-0.95–-0.7	-0.8	-1.9–-1.6	-1.8
CMOS	$V_{DD} = 3-10$	V_{DD}	V_{DD}	0–0.5	0
Lógica positiva:			lógica 1		lógica 0
Lógica negativa:			lógica 0		lógica 1

del voltaje de suministro con $H = V_{DD}$ y $L = 0$ volts. Las asignaciones de polaridad para lógica positiva y negativa también se indican en la tabla.

A la luz de esta exposición, es necesario justificar los símbolos lógicos usados para los IC que se listan en la Fig. 2-9. Tómese, por ejemplo, una de las compuertas del IC 7400. Esta compuerta se muestra en forma de diagrama de bloques en la Fig. 2-11(b). La tabla de verdad del fabricante para esta compuerta dada en una hoja de datos se muestra en la Fig. 2-11(a). En esta tabla se especifica el comportamiento físico de la compuerta, con H de 3.5 volts en forma típica y L de 0.2 volts. Esta compuerta física puede funcionar ya sea como compuerta NAND o NOR, dependiendo de la asignación de polaridad.

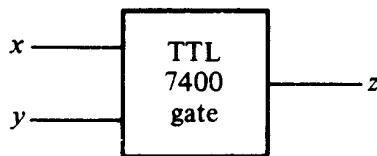
En la tabla de verdad de la Fig. 2-11(c) se supone la asignación de lógica positiva con $H = 1$ y $L = 0$. Al verificar esta tabla de verdad en la Fig. 2-5, se reconoce como una compuerta NAND. El símbolo gráfico para una compuerta NAND de lógica positiva se muestra en la Fig. 2-11(b) y es similar a la que se adoptó con anterioridad.

Ahora considérese la asignación de lógica negativa a esta compuerta física con $L = 1$ y $H = 0$. El resultado es la tabla de verdad que se muestra en la Fig. 2-11(e). Puede reconocerse que esta tabla representa la función NOR aun cuando sus entradas están listadas hacia atrás. El símbolo gráfico para una compuerta NOR de lógica negativa se muestra en la Fig. 2-11(f). El pequeño triángulo en los alambres de entrada y salida designa un *indicador de polaridad*. La presencia de este indicador de polaridad a lo largo de una terminal indica que se asigna una lógica negativa a la terminal. Por tanto, la misma compuerta física puede funcionar ya sea como una NAND de lógica positiva o como una NOR de lógica negativa. La que está dibujada en el diagrama depende por completo de la asignación de polaridad que deseé emplear el diseñador.

De manera semejante, es posible mostrar que una NOR de lógica positiva es la misma compuerta física que una NAND de lógica negativa. La misma relación es válida entre las compuertas AND y OR o entre las compuertas excluyente-OR y equivalencia. En cualquier caso, si se supone lógica negativa en cualquier terminal de entrada o salida, es necesario incluir el símbolo del triángulo indicador de polaridad junto a la terminal. Algunos diseñadores digitales utilizan esta convención para facilitar el diseño de circuitos digitales cuando se usan exclusivamente compuertas NAND o NOR. En este libro no se emplea esta simbología, pero se recurre a otros

x	y	z
L	L	H
L	H	H
H	L	H
H	H	L

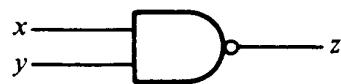
(a) Tabla de verdad de términos de H y L.



(b) Diagrama de bloque de compuerta.

x	y	z
0	0	1
0	1	1
1	0	1
1	1	0

(c) Tabla de verdad para lógica positiva:
 $H = 1, L = 0$.



(d) Símbolo gráfico para la compuerta NAND de lógica positiva.

x	y	z
1	1	0
1	0	0
0	1	0
0	0	1

(e) Tabla de verdad para lógica negativa:
 $L = 1, H = 0$.



(f) Símbolo gráfico para la compuerta NOR de lógica negativa.

Figura 2-11 Demostración de las lógicas positiva y negativa.

métodos para diseñar con las compuertas NAND y NOR. Obsérvese que los IC que se presentan en la Fig. 2-9 se muestran con sus símbolos gráficos de lógica positiva. Podrían haberse ilustrado con sus símbolos de lógica negativa si así se hubiera deseado.

La conversión de lógica positiva en lógica negativa y viceversa es en esencia una operación que cambia los 1 en 0 y los 0 en 1, tanto en las entradas como en las salidas de una computadora. Ya que esta operación produce el dual de una función, el cambio de todas las terminales de una polaridad a la otra resulta en tomar la dual de la función. El resultado de esta conversión es que todas las operaciones AND se convierten en operaciones OR (o símbolos gráficos) y viceversa. Además, no debe olvidarse incluir el indicador de polaridad en los símbolos gráficos cuando se supone lógica negativa.

El pequeño triángulo que representa un indicador de polaridad y el pequeño círculo que representa una complementación tienen efectos similares pero diferente significados. Por tanto, puede reemplazarse uno por otro, pero la interpretación es diferente. Un círculo seguido por un triángulo, como en la Fig. 2-11(f), representa una complementación seguida por un indicador de polaridad de lógica negativa. Los dos se cancelan uno a otro y ambos pueden eliminarse. Pero si se eliminan ambos, entonces las entradas y salidas de la compuerta representarán polaridades diferentes.

Características especiales

Las características de las familias IC de lógica digital por lo común se comparan por el análisis de circuito de la compuerta, básica en cada familia. Los parámetros más importantes que se evalúan y comparan son la salida en abanico (multiplicidad de conexiones en la salida), disipación de potencia, retardo de propagación y margen de ruido. Se explicarán primero las propiedades de este parámetro y después se utilizarán para comparar las familias IC lógicas.

El *abanico de salida* especifica el número de cargas estándar que pueden impulsar la salida de una compuerta sin menoscabar su operación normal. Una carga estándar por lo común se define como la cantidad de corriente necesaria por una entrada de otra compuerta en la misma familia IC. Algunas veces el término *cargado* se usa en lugar de abanico de salida. Este término se deriva del hecho de que la salida de una compuerta puede suministrar una cantidad limitada de corriente, arriba de la cual cesa su operación apropiada y se dice que está sobrecargada. La salida de una compuerta por lo general se conecta a las entradas de otras compuertas similares. Cada entrada consume una cierta cantidad de potencia de la entrada de la compuerta, de modo que cada conexión adicional se agrega a la carga de la compuerta. Las "reglas de carga" por lo común se listan para una familia de circuitos digitales estándar. Estas reglas especifican la máxima cantidad de carga permitida para cada salida de cada circuito. El exceder la carga máxima especificada puede causar un mal funcionamiento debido a que el circuito no puede suministrar la potencia demandada de él. El abanico de salida es el número máximo de entradas (a otros circuitos) que pueden conectarse a la salida de una compuerta y se expresa por un número.

Las capacidades del abanico de salida de una compuerta pueden considerarse cuando se simplifican las funciones booleanas. Debe tenerse cuidado de no desarrollar expresiones que resulten en una compuerta sobrecargada. Los amplificadores no inversores o buffer algunas veces se emplean para proporcionar capacidades adicionales de impulsión para cargas pesadas.

La *disipación de potencia* es la potencia suministrada requerida para operar la compuerta. Este parámetro se expresa en miliwatts (mW) y representa la potencia real disipada en la compuerta. El número que representa este parámetro no incluye la potencia suministrada por otra compuerta; más bien, representa la potencia suministrada a la compuerta por el suministro de potencia. Un IC con cuatro compuertas requerirá, de su suministro de potencia, cuatro veces la potencia disipada por cada compuerta. En un sistema dado, puede haber muchos IC y, la potencia requerida por cada IC debe considerarse. La disipación total de potencia en un sistema es la suma total de la potencia disipada en todos los IC.

El *retardo de propagación* es el retardo de tiempo de transición promedio para que una señal se propague desde la entrada a la salida cuando la señal binaria cambia en valor. Las señales a través de una compuerta toman cierta cantidad de tiempo para propagarse desde las entradas a la salida. Este intervalo de tiempo se define como el retardo de propagación de la compuerta. El retardo de propagación se expresa en nanosegundos (ns) y, un ns es igual a 10^{-9} de un segundo.

Las señales que viajan de las entradas de un circuito digital a sus salidas pasan a través de una serie de compuertas. La suma de los retardos de propagación a través de las compuertas es el retardo total de propagación del circuito. Cuando la velocidad de operación es importante, cada compuerta debe tener un pequeño retardo de propagación y el circuito digital debe tener un número mínimo de compuertas en serie entre las entradas y las salidas.

En la mayoría de los circuitos digitales las señales de entrada se aplican en forma simultánea a más de una compuerta. Todas las compuertas que reciben sus entradas exclusivamente desde las entradas externas, constituyen el primer nivel lógico del circuito. Las compuertas que reciben cuando menos una entrada de una salida de una compuerta del primer nivel lógico se considera que están en el segundo nivel lógico, y en forma semejante, para el tercer nivel y los más altos. El retardo total de propagación del circuito es igual al retardo de propagación de una compuerta multiplicado por el número de niveles lógicos en el circuito. Luego, una reducción en el número de niveles lógicos produce una reducción del retardo de señal y en circuitos más rápidos. La reducción del retardo de propagación en los circuitos puede ser más importante que la reducción en el número total de compuertas si la velocidad de operación es un factor principal.

El *margen de ruido* es el máximo voltaje de ruido añadido a la señal de entrada de un circuito digital que no causa un cambio indeseable en la salida del circuito. Hay dos tipos de ruido que considerar: el ruido CC es causado por una deriva en los niveles de voltaje de una señal. El ruido CA es un pulso aleatorio que puede crearse por otras señales de interrupción. Por eso, el ruido es un término que se utiliza para denominar una señal indeseable que está superpuesta sobre la señal normal de operación. La capacidad de los circuitos para operar en forma confiable en un ambiente de ruido es importante en muchas aplicaciones. El margen de ruido se expresa en volts (V) y representa la señal de ruido máximo que puede tolerarse por la compuerta.

Características de las familias lógicas IC

El circuito básico de la familia lógica TTL es la compuerta NAND. Hay muchas versiones de la TTL y tres de ellas se listan en la Tabla 2-8. En esta tabla se dan las

TABLA 2-8 Características típicas de las familias lógicas IC

Familia lógica IC	Abanico de salida	Dissipación de potencia (mW)	Retardo de propagación (ns)	Margen de ruido (V)
Estándar TTL	10	10	10	0.4
Schottky TTL	10	22	3	0.4
Baja potencia Schottky TTL	20	2	10	0.4
ECL	25	25	2	0.2
CMOS	50	0.1	25	3

características generales de las familias lógicas IC. Los valores que se listan son representativos en una base de comparación. Para cualquier familia o versión, los valores pueden tener cierta variación.

La compuerta estándar TTL fue la primera versión de la familia TTL. Conforme progresó la tecnología, se agregaron mejoras adicionales. La TTL Schottky es una última mejora que reduce el retardo de propagación, pero resulta en un aumento de la disipación de potencia. La versión TTL Schottky de baja potencia sacrifica cierta velocidad para reducir la disipación de potencia. Tiene el mismo retardo de propagación que la TTL estándar, pero la disipación de potencia se reduce en forma considerable. El abanico de salida de la TTL estándar es 10, pero la versión Schottky de baja potencia tiene un abanico de salida de 20. Bajo ciertas condiciones las otras versiones también pueden tener un abanico de salida de 20. El margen de ruido es mejor que 0.4 V, con un valor típico de 1 V.

El circuito básico de la familia ECL es la compuerta NOR. La ventaja especial de las compuertas ECL es su bajo retardo de propagación. Algunas versiones ECL pueden tener un retardo de propagación tan bajo como 0.5 ns. La disipación de potencia en las compuertas ECL es comparativamente alta y el margen de ruido bajo. Estos dos parámetros imponen una desventaja cuando se elige la ECL sobre las otras familias lógicas. Sin embargo, debido a su bajo retardo de propagación, la ECL ofrece la velocidad más alta entre todas las familias y es la elección final para sistemas muy rápidos.

El circuito más bajo de la CMOS es el inversor por el cual ambas compuertas NAND y NOR pueden construirse. La ventaja especial del CMOS es su disipación de potencia en extremo baja. Bajo condiciones estáticas, la disipación de potencia de la compuerta CMOS es despreciable, con promedios de cerca de 10 nW. Cuando la señal de la compuerta cambia de estado, hay una disipación dinámica de potencia que es proporcional a la frecuencia a la cual se ejerce el circuito. El número que se lista en la tabla es un valor típico de la disipación dinámica de potencia en las compuertas CMOS.

Una desventaja principal de la compuerta CMOS es su alto retardo de propagación. Esto significa que no es práctica para utilizarse en sistemas que requieren operaciones a alta velocidad. Los parámetros característicos de la compuerta CMOS dependen del voltaje de suministro de potencia V_{DD} que se use. La disipación de potencia aumenta conforme aumenta el voltaje de suministro. El retardo de propagación disminuye con el incremento en el voltaje de suministro, y el margen de ruido se estima que es alrededor del 40% del valor del voltaje de suministro.

BIBLIOGRAFIA

1. Boole, G., *An Investigation of the Laws of Thought*. New York: Dover Pub., 1954.
2. Shannon, C. E., "A Symbolic Analysis of Relay and Switching Circuits." *Trans. of the AIEE*, Vol. 57 (1938), 713-23.
3. Huntington, E. V., "Sets of Independent Postulates for the Algebra of Logic." *Trans. Am. Math. Soc.*, Vol. 5 (1904), 288-309.

4. Birkhoff, G., and T. C. Bartee, *Modern Applied Algebra*. New York: McGraw-Hill Book Co., 1970.
5. Birkhoff, G., and S. MacLane, *A Survey of Modern Algebra*, 3a. ed. New York: The Macmillan Co., 1965.
6. Hohn, F. E., *Applied Boolean Algebra*, 2a. ed. New York: The Macmillan Co., 1966.
7. Whitesitt, J. E., *Boolean Algebra and its Applications*. Reading, Mass.: Addison-Wesley Pub. Co., 1961.
8. *The TTL Data Book for Design Engineers*. Dallas, Texas: Texas Instruments Inc., 1976.
9. *MECL Integrated Circuits Data Book*. Phoenix, Ariz.: Motorola Semiconductor Products, Inc., 1972.
10. *RCA Solid State Data Book Series: COS/MOS Digital Integrated Circuits*. Somerville, N. J.: RCA Solid State Div., 1974.

PROBLEMAS

- * 2-1. ¿Cuál de las seis leyes básicas (cierre, asociativa, conmutativa, identidad, inversa y distributiva) se satisface por el par de operadores binarios que se listan a continuación?

+	0	1	2		·	0	1	2
0	0	0	0	·	0	0	1	2
1	0	1	1	·	1	1	1	2
2	0	1	2	·	2	2	2	2

- * 2-2. Muestre que el conjunto de tres elementos $\{0, 1, 2\}$ y los dos operadores binarios $+$ y \cdot como se define en la tabla anterior no es un álgebra booleana. Establezca cuál de los postulados de Huntington no se satisface.
- 2-3. Demuestre mediante tablas de verdad la validez de los siguientes teoremas del álgebra booleana.
- Las leyes asociativas.
 - Los teoremas de De Morgan para tres variables.
 - La ley distributiva $+$ sobre \cdot .
- 2-4. Repita el problema 2.3 utilizando diagramas de Venn.
- * 2-5. Simplifique las siguientes funciones booleanas a un número mínimo de literales.
- $xy + xy'$
 - $(x + y)(x + y')$
 - $xyz + x'y + xyz'$
 - $zx + zx'y$
 - $(A + B)'(A' + B)'$
 - $y(wz' + wz) + xy$
- 2-6. Reduzca las siguientes expresiones booleanas al número requerido de literales.
- | | |
|---|--------------------|
| (a) $ABC + A'B'C + A'BC + ABC' + A'B'C'$ | a cinco literales |
| (b) $BC + AC' + AB + BCD$ | a cuatro literales |
| (c) $[(CD)' + A]' + A + CD + AB$ | a tres literales |
| (d) $(A + C + D)(A + C + D')(A + C' + D)(A + B')$ | a cuatro literales |

- 2-7. Encuentre el complemento de las siguientes funciones booleanas y redúzcalas a un número mínimo de literales.

- $(BC' + A'D)(AB' + CD')$
- $B'D + A'BC' + ACD + A'BC$
- $[(AB)'A][(AB)'B]$
- $AB' + C'D'$

- 2-8. Dadas dos funciones booleanas F_1 y F_2 :

- Muestre que la función booleana $E = F_1 + F_2$, obtenida al aplicar el operador OR a las dos funciones, contiene la suma de todos los mintérminos en F_1 y F_2 .
- Muestre que la función booleana $G = F_1F_2$, obtenida al aplicar el operador AND a las dos funciones, contiene los mintérminos comunes tanto a F_1 como F_2 .

- 2-9. Obtenga la tabla de verdad de la función:

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

- 2-10. Implemente las funciones booleanas simplificadas del problema 2-6 con compuertas lógicas.

- 2-11. Dada la función booleana:

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

- Impleméntela con las compuertas AND, OR y NOT
- Impleméntela sólo con las compuertas OR y NOT
- Impleméntela sólo con las compuertas AND y NOT

- 2-12. Simplifique las funciones T_1 y T_2 a un número mínimo de literales.

A	B	C	T ₁	T ₂
0	0	0	1	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	0	1
1	0	1	0	1
1	1	0	0	1
1	1	1	0	1

- 2-13. Exprese las siguientes funciones de una suma de mintérminos y un producto de maxtérminos.

- $F(A, B, C, D) = D(A' + B) + B'D$
- $F(w, x, y, z) = y'z + wxy' + wxz' + w'x'z$
- $F(A, B, C, D) = (A + B' + C)(A + B')(A + C' + D')(A' + B + C + D')(B + C' + D')$
- $F(A, B, C) = (A' + B)(B' + C)$
- $F(x, y, z) = 1$
- $F(x, y, z) = (xy + z)(y + xz)$

- 2-14. Convierta las siguientes funciones en la otra forma canónica.
- (a) $F(x, y, z) = \Sigma(1, 3, 7)$
 - (b) $F(A, B, C, D) = \Sigma(0, 2, 6, 11, 13, 14)$
 - (c) $F(x, y, z) = \Pi(0, 3, 6, 7)$
 - (d) $F(A, B, C, D) = \Pi(0, 1, 2, 3, 4, 6, 12)$
- 2-15. ¿Cuál es la diferencia entre la forma canónica y la estándar? ¿Cuál forma es preferible cuando se implementa con compuertas una función booleana? ¿Cuál forma se obtiene cuando se lee una función de una tabla de verdad?
- 2-16. La suma de todos los mintérminos de una función booleana de n variables es 1.
- (a) Pruebe la enunciación anterior para $n = 3$.
 - (b) Sugiera un procedimiento para una prueba general.
- 2-17. El producto de todos los maxterminos de una función booleana de n variables es 0.
- (a) Pruebe la enunciación anterior para $n = 3$.
 - (b) Sugiera un procedimiento para una prueba general. ¿Puede usarse el principio de dualidad después de probar (b) del problema 2-16?
- 2-18. Muestre que la dual de la excluyente OR es igual a su complemento.
- 2-19. Por la sustitución de la función booleana equivalente a las operaciones binarias como se definen en la Tabla 2-6, muestre que:
- (a) Los operadores de inhibición e implicación no son ni conmutativos ni asociativos.
 - (b) Los operadores excluyente OR y de equivalencia son conmutativos y asociativos.
 - (c) El operador NAND no es asociativo.
 - (d) Los operadores NOR y NAND no son distributivos.
- 2-20. Una compuerta de *mayoría* es un circuito digital cuya salida es igual a 1 si la mayoría de las entradas son 1. En otra forma, la salida es 0. Mediante una tabla de verdad, encuentre la función booleana implementada por una compuerta de mayoría de 3 entradas. Simplifique la función.
- 2-21. Verifique la tabla de verdad para la compuerta excluyente OR de 3 entradas que se lista en la Fig. 2-8(c). Liste todas las ocho combinaciones de x, y y z ; evalúe $A = x \oplus y$; después evalúe $F = A \oplus z = x \oplus y \oplus z$.
- 2-22. La familia lógica TTL SSI existe principalmente en paquetes de 14 clavijas. Dos clavijas se reservan para suministro de potencia y las otras clavijas se utilizan para terminales de entrada y salida. Cuántas compuertas están encapsuladas en un paquete de esta clase si contiene los siguientes tipos de compuertas:
- (a) Compuertas excluyente-OR de 2 entradas.
 - (b) Compuertas AND de 3 entradas.
 - (c) Compuertas NAND de 4 entradas.
 - (d) Compuertas NOR de 5 entradas.
 - (e) Compuertas NAND de 8 entradas.
- 2-23. Muestre que una compuerta AND de lógica positiva es una compuerta OR de lógica negativa y viceversa.
- 2-24. Una familia lógica IC tiene compuertas NAND con abanico de salida de 5 y compuertas buffer con abanico de salida de 10. Muestre cómo la señal de salida de una sola compuerta NAND puede aplicarse a otras 50 entradas de compuerta.