ENRIQUE MANDADO
CATEDRATICO DE TECNOLOGIA ELECTRONICA
DEPARTAMENTO DE TECNOLOGIA ELECTRONICA
DE LAS ESCUELAS TECNICAS DE INGENIERIA
DE LA UNIVERSIDAD DE VIGO

SISTEMAS ELECTRONICOS DIGITALES

Tomo I
Circuitos combinacionales y secuenciales

marcombo
BOIXAREU EDITORES

BARCELONA

Capítulo

2

Algebra de Boole

2.1 DEFINICION Y POSTULADOS

Un álgebra de Boole es toda clase o conjunto de elementos que pueden tomar dos valores perfectamente diferenciados, que designaremos por 0 y 1 y que están relacionados por dos operaciones binarias denominadas suma (+) y producto (.)* lógicos que cumplen los siguientes postulados:

a) Ambas operaciones son conmutativas, es decir si a y b son elementos del álgebra, se verifica:

$$a + b = b + a; \quad a \cdot b = b \cdot a$$

b) Dentro del álgebra existen dos elementos neutros, el 0 y el 1, que cumplen la propiedad de identidad con respecto a cada una de dichas operaciones:

$$0 + a = a; \quad 1 \cdot a = a$$

c) Cada operación es distributiva con respecto a la otra:

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

d) Para cada elemento, a, del álgebra existe un elemento denominado, $\bar{a}$, tal que:

$$a + \bar{a} = 1; \quad a \cdot \bar{a} = 0$$

Este postulado define realmente una nueva operación fundamental que es la inversión o complementación de una variable. La variable $\bar{a}$ se encuentra siempre en un estado binario contrario al de a.

La primera ecuación expresa la imposibilidad de que a y $\bar{a}$ tomen el valor lógico cero al mismo tiempo y la segunda ecuación indica que nunca pueden tener el valor lógico uno al mismo tiempo. Por lo tanto la tabla de verdad de la inversión o complementación es:

a	$\bar{a}$
0	1
1	0

**Nota:* La operación producto se indica en general simplemente mediante la ausencia de símbolo entre dos variables.

Puede elegirse como postulado un grupo distinto del adoptado con tal de que se cumpla la condición de que ninguno pueda ser deducido de cualquiera de los demás.

De lo explicado anteriormente se deduce que el álgebra de Boole es un ente matemático. En realidad, físicamente son varios los conjuntos que poseen dos operaciones binarias que cumplen los postulados desarrollados. Ejemplo de estos conjuntos son el álgebra de las proposiciones o juicios formales y el álgebra de la conmutación formada también por elementos que pueden tomar dos estados perfectamente diferenciados. Estos elementos son los circuitos lógicos cuyo estudio desarrollaremos en capítulos sucesivos.

Los primeros circuitos de conmutación o lógicos utilizados han sido los contactos y, aunque poco a poco han sido desplazados por los circuitos electrónicos, pueden ser empleados para memorizar más fácilmente las leyes del álgebra de Boole antes expresadas y los teoremas que se desarrollan seguidamente.

La operación suma se asimila a la conexión en paralelo de contactos y la operación producto a la conexión en serie. El inverso de un contacto es otro cuyo estado es siempre el opuesto del primero, es decir está cerrado cuando aquél está abierto y viceversa. El elemento 0 es un contacto que está siempre abierto y el elemento 1 un contacto que está siempre cerrado. Además se considera una función de transmisión entre los dos terminales de un circuito de contactos, que toma el valor 1, cuando existe un camino para la circulación de corriente entre ellos (cortocircuito) y el valor 0 al no existir dicho camino (circuito abierto).

En la figura 2.1, se expresa gráficamente que el álgebra de los contactos cumple las leyes del álgebra de Boole.

2.2 TEOREMAS DEL ALGEBRA DE BOOLE

Basándose en los postulados anteriores se deducen los teoremas que expondremos seguidamente. Su demostración se puede realizar algebraicamente o mediante la llamada tabla de verdad. La tabla de verdad de una expresión algebraica binaria representa los valores que dicha expresión puede tomar para cada combinación de estados de las variables que forman parte de la misma. Dos expresiones algebraicas que tienen la misma tabla de verdad son equivalentes.

Teorema 1: Cada identidad deducida de los anteriores postulados del álgebra de Boole permanece válida si la operación «+» y «·» y los elementos 0 y 1 se intercambian entre sí.

Este principio, llamado de dualidad, se deduce inmediatamente de la simetría de los cuatro postulados con respecto a ambas operaciones y a ambos elementos neutros.

Teorema 2: Para cada elemento a de un álgebra de Boole se verifica:

$$a + 1 = 1 \text{ y } a \cdot 0 = 0$$

Demostraremos la primera igualdad y con ello quedará demostrada por dualidad la segunda. En efecto, se verifica:

$$1 = a + \bar{a} = a + \bar{a} \cdot 1 = (a + \bar{a}) \cdot (a + 1) = 1 \cdot (a + 1) = a + 1$$

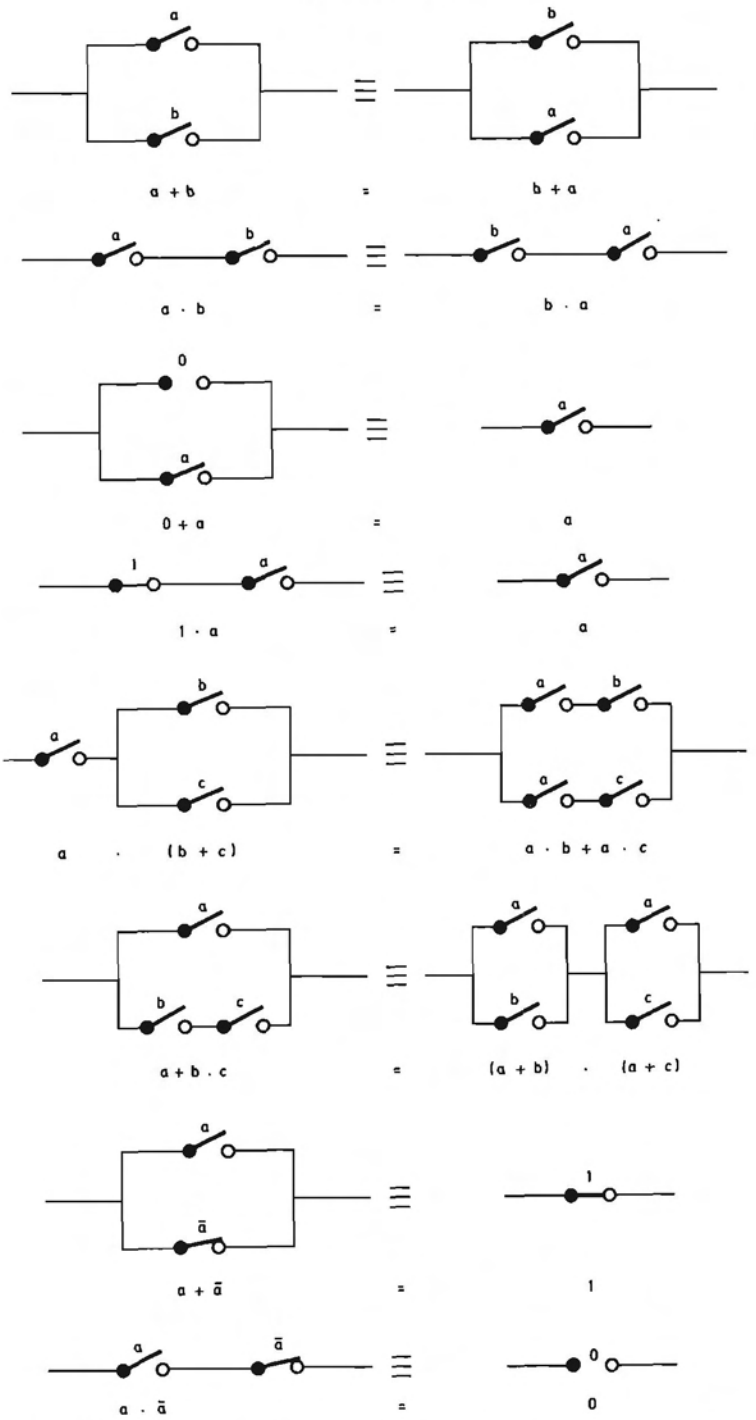


FIGURA 2.1.—**Representación de las leyes del álgebra de Boole mediante asociación de contactos.**

De este teorema y del postulado *b*) se deducen las siguientes igualdades:

$$0 + 0 = 0 \qquad 0 \cdot 0 = 0$$
$$0 + 1 = 1 \qquad 0 \cdot 1 = 0$$
$$1 + 1 = 1 \qquad 1 \cdot 1 = 1$$

y, por tanto, las tablas de verdad de las operaciones lógicas suma (función O) y producto (función Y) son las siguientes:

a	b	s
0	0	0
0	1	1
1	0	1
1	1	1

$s = a + b$

a	b	p
0	0	0
0	1	0
1	0	0
1	1	1

$p = a \cdot b$

Teorema 3: Para cada elemento a de un álgebra de Boole se verifica:

$$a + a = a \qquad \text{y} \qquad a \cdot a = a$$

Demostraremos la primera igualdad:

$$a = a + 0 = a + a\bar{a} = (a + a) \cdot (a + \bar{a}) = a + a$$

Teorema 4: Para cada par de elementos de un álgebra de Boole a y b, se verifica:

$$a + ab = a \qquad \text{y} \qquad a\,(a + b) = a$$

Esta ley se llama de absorción.

a	b	$a + ab$
0	0	0
0	1	0
1	0	1
1	1	1

TABLA 2.1

Lo demostraremos algebraicamente y mediante la tabla de verdad. En efecto, algebraicamente:

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

En la tabla 2.1 se comprueba que la columna correspondiente a $a + ab$ es igual a la columna de la variable a y por tanto se deduce la igualdad:

$$a = a + ab$$

Teorema 5: En un álgebra de Boole, las operaciones suma y producto son asociativas

$$a + (b + c) = (a + b) + c = a + b + c$$
$$a\,(bc) = (ab)\,c = abc$$

Este teorema se demuestra fácilmente mediante la tabla de verdad.

Teorema 6: Para todo elemento $\bar{a}$ de un álgebra de Boole se verifica:

$$\bar{\bar{a}} = a$$

Su demostración es inmediata mediante la tabla de verdad, lo cual se verifica seguidamente

a	$\bar{a}$	$\bar{\bar{a}}$
0	1	0
1	0	1

De este teorema y el postulado *d)* se deduce que en toda álgebra de Boole:

$$\bar{0} = 1 \qquad \text{y} \qquad \bar{1} = 0$$

Teorema 7: En toda álgebra de Boole se verifica:

(1) $\overline{a + b + c + d + \ldots} = \bar{a}\,\bar{b}\,\bar{c}\,\bar{d}\ldots$

(2) $\overline{a\,b\,c\,d\ldots} = \bar{a} + \bar{b} + \bar{c} + \bar{d}\ldots$

Demostraremos la primera de estas igualdades, denominadas leyes de De Morgan, con lo cual la segunda quedará demostrada por dualidad.

Realizaremos primero la demostración para dos variables.

De acuerdo con el postulado *d)*, para que $\overline{a + b} = \bar{a}\bar{b}$ se ha de verificar:

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

En efecto, aplicando los postulados y teoremas ya estudiados resulta

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

La generalización para un número cualquiera de variables resulta ahora muy sencilla:

Denominaremos $b + c + d + \ldots = p$ y aplicando la ley de De Morgan que hemos demostrado para dos variables, resulta:

$$\overline{a + b + c + d} + \ldots = \overline{a + p} = \bar{a}\bar{p} = \bar{a}\overline{b + c + d + \ldots}$$

Llamando ahora $q = c + d + \ldots$ resulta:

$$\overline{a + b + c + d + \ldots} = \bar{a}\overline{b + q} = \bar{a}\bar{b}\bar{q} = \bar{a}\bar{b}\overline{c + d + \ldots}$$

Repitiendo este proceso se obtiene:

$$\overline{a + b + c + d + \ldots} = \bar{a}\bar{b}\bar{c}\bar{d}\ldots$$

c.q.d.

Este teorema define realmente dos nuevas funciones lógicas de gran importancia que, como veremos en el capítulo dedicado a estudiar la tecnología, serán utilizadas como elementos básicos para la realización de los sistemas digitales. Estas dos funciones, que realizan las expresiones (1) y (2), se denominan respectivamente NO-O (NOR) y NO-Y (NAND).

Las tres funciones elementales, suma, producto, e inversión lógica, pueden ser realizadas mediante las funciones NO-Y y NO-O.

En efecto, aplicando el teorema de De Morgan tenemos:

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

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

y la inversión se realiza con una función NO-O o NO-Y de una sola entrada.

Para representar las funciones lógicas fue necesario crear símbolos adecuados. El desarrollo de la electrónica digital ha sido tan rápido que hizo que se crearan símbolos sin un estudio minucioso. Inicialmente se adoptaron símbolos diferentes para cada función, de los que son un ejemplo los de las funciones Y y O representados en la figura 2.2.

FIGURA 2.2.—**Símbolos lógicos no normalizados de las funciones Y y O.**

La búsqueda de una simplificación en la representación de los elementos lógicos llevó a la realización de estudios por parte de asociaciones de ingeniería (IEEE) y organizaciones de normalización (ISO) que han cristalizado en la adopción de un sistema normalizado de representación por parte de la Comisión Electrotécnica Internacional. Al estudio de este sistema se dedica el apéndice 1 de este libro. El nuevo sistema internacional llegó después de más de diez años durante los cuales los fabricantes de semiconductores crearon sus propios símbolos. Por ello se ha considerado lo más conveniente en esta edición combinar adecuadamente la simbología antigua y la nueva para que el lector pueda adaptarse de forma paulatina al cambio que supone pasar de una a otra.

FIGURA 2.3.—**Símbolos lógicos normalizados de las funciones Y y O.**

En la figura 2.3 se representan los símbolos de las puertas Y y O en el nuevo sistema.

En los símbolos antiguos la inversión unida a otras funciones se puede representar mediante un círculo; por tanto, los símbolos de la función NO-O (NOR) y NO-Y (NAND) se deducen respectivamente de los de las funciones O e Y añadiéndoles un círculo (fig. 2.4)

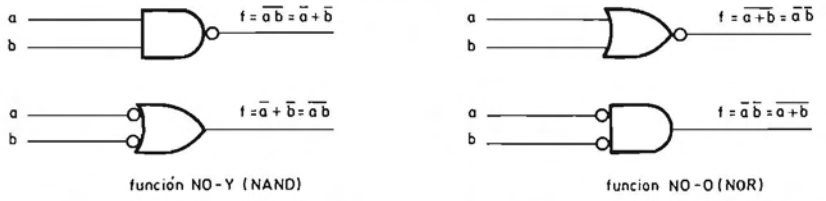


FIGURA 2.4.—**Símbolos lógicos no normalizados de las funciones NO-Y (NAND) Y NO-O (NOR).**

En el nuevo sistema la inversión unida a otras funciones se puede representar mediante un círculo pero en el caso de utilizar el criterio de lógica positiva (descrito en el apartado 5.1) se sustituye por un triángulo rectángulo (ver apéndice 1). En este libro se utiliza el círculo y por ello se representan las puertas NO-Y (NAND) y NO-O (NOR) mediante los símbolos de la figura 2.5.

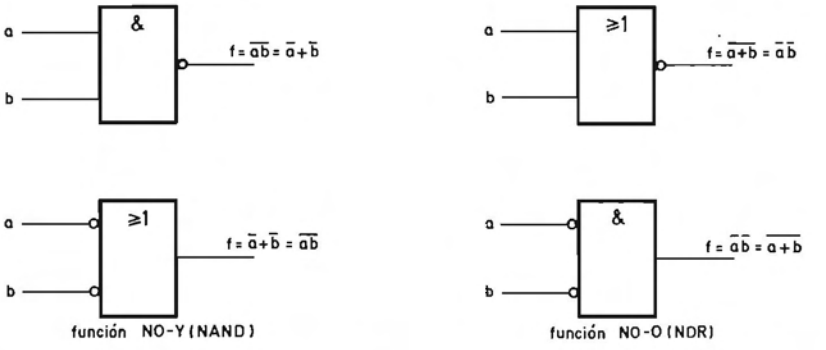


FIGURA 2.5.—**Símbolos lógicos normalizados de las funciones NO-Y (NAND) y NO-O (NOR).**

El teorema de De Morgan indica que existen dos formas de expresar la función NO-O (NOR) y la función NO-Y (NAND):

$$\overline{a + b} = \bar{a}\,\bar{b}$$

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

La segunda expresión de la función NO-O se puede representar mediante el símbolo de la función Y precedido de dos inversiones. Igualmente la función NO-Y se puede representar mediante el símbolo de la función O precedido de dos inversiones. En las figuras 2.4 y 2.5 se indican ambos símbolos.

Las funciones NO-O (NOR) y NO-Y (NAND) de una sola variable constituyen la función de inversión, por lo que esta función se puede representar mediante el símbolo de cualquiera de ellas con una sola variable de una entrada o mediante un símbolo especial. En la simbología antigua la función inversión se representa mediante un triángulo seguido de un círculo, o una puerta NO-Y o NO-O de una entrada tal como se indica en la figura 2.6. En el nuevo sistema normalizado se representa mediante el símbolo de un seguidor (un 1 como indicativo) con el triángulo o círculo de inversión a la salida (figura 2.7)

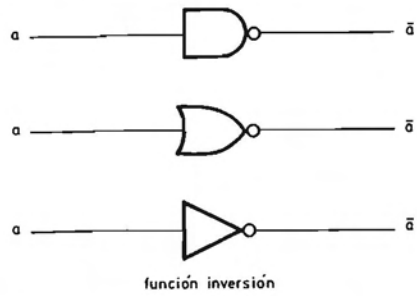


FIGURA 2.6.—**Símbolos lógicos no normalizados de un inversor.**

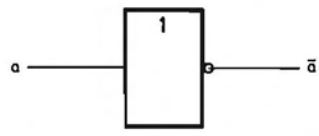


FIGURA 2.7.—**Símbolo lógico normalizado de un inversor.**

La realización de las funciones suma, producto e inversión con las funciones NO-Y y NO-O se representan gráficamente, mediante los símbolos antiguos en la figura 2.8 y los nuevos normalizados en la figura 2.9.

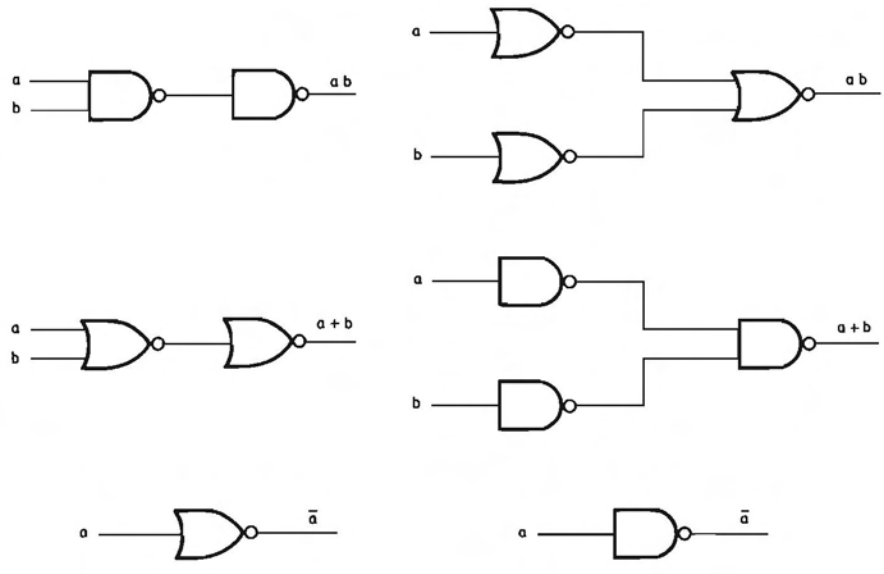


FIGURA 2.8.—**Esquemas de la realización de las funciones producto, suma e inversión con funciones NO-Y (NAND) y NO-O (NOR), representadas con símbolos no normalizados.**

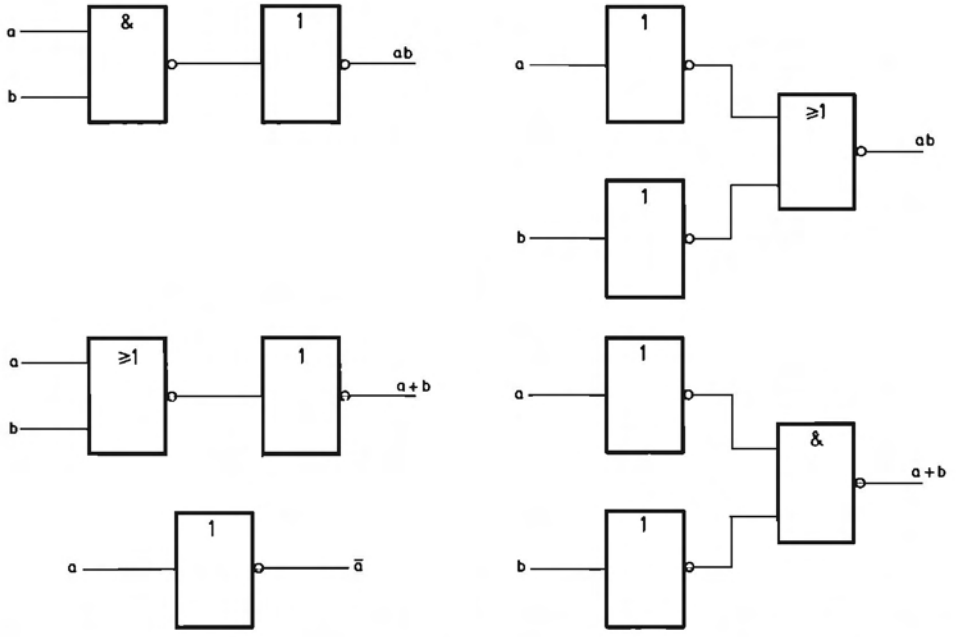


FIGURA 2.9.—**Esquemas de la realización de las funciones producto, suma e inversión con funciones NO-Y (NAND) y NO-O(NOR) representadas mediante símbolos normalizados.**

2.3 FUNCION DE UN ALGEBRA DE BOOLE

Una función de un álgebra de Boole es una variable binaria cuyo valor es igual al de una expresión algebraica en la que se relacionan entre sí las variables binarias por medio de las operaciones básicas, Producto lógico, Suma lógica e Inversión.

Se representa una función lógica por la expresión $f = f(a, b, c, ...)$

El valor lógico de f, depende del de las variables a, b, c, ...

Se llama término canónico de una función lógica a todo producto o suma en la cual aparecen todas las variables en su forma directa o inversa. Al primero de ellos se le llama producto canónico (minterm) y al segundo suma canónica (maxterm).

Por ejemplo sea una función de tres variables $f(a, b, c)$. El término $a\bar{b}c$ es un producto canónico y el término $\bar{a} + b + \bar{c}$ es una suma canónica.

El número máximo de productos canónicos o sumas canónicas viene dado por las variaciones con repetición de dos elementos tomados de n en n. El número de productos o sumas canónicas de n variables es por lo tanto 2^n.

Para mayor facilidad de representación, cada término canónico, se expresa mediante un número decimal equivalente al binario obtenido al sustituir las variables ordenadas con un criterio determinado por un 1 o un 0 según aparezcan en su forma directa o complementada respectivamente.

Por ejemplo, los términos canónicos siguientes se representarán:

$$\bar{d}cb\bar{a} \equiv 0110_2 = 6_{10}$$
$$d + \bar{c} + b + \bar{a} \equiv 1010_2 \equiv 10_{10}$$

Por lo tanto la función lógica $f(a, b, c) = \bar{a}b\bar{c} + ab\bar{c} + a\bar{b}c$ se podrá representar por la expresión: $f(a, b, c) = \Sigma_3(2, 3, 5)$ en la cual el símbolo Σ representa la suma lógica.

De igual forma la función $f(a, b, c) = (a + \bar{b} + \bar{c})(\bar{a} + b + \bar{c})(a + b + c)$ se puede representar por $f(a, b, c) = \mathit{II}_3(1, 2, 7)$ en la cual II indica el producto lógico.

Cuando una función se expresa como una suma de productos canónicos o un producto de sumas canónicas se dice que se encuentra en su forma canónica.

Demostraremos ahora un teorema relativo a las funciones de un álgebra de Boole de gran importancia en la simplificación algebraica de las funciones lógicas.

Teorema. Toda función de un álgebra de Boole se puede expresar de la siguiente forma:

$$f(a, b, c...) = af(1, b, c, ...) + \bar{a}f(0, b, c...)$$
$$f(a, b, c...) = [a + f(0, b, c...)][\bar{a} + f(1, b, c...)]$$

Demostraremos la primera ecuación y la segunda quedará también demostrada por dualidad.

Para demostrarla es suficiente comprobar que la igualdad se cumple tanto para $a = 0$ como para $a = 1$.

En efecto si $a = 0$ y $\bar{a} = 1$; se verifica:

$$f(a, b, c...) = f(0, b, c...) = 0f(1, b, c...) + 1f(0, b, c...) = f(0, b, c...)$$

y si $a = 1$ y $\bar{a} = 0$; se verificará así mismo:

$$f(a, b, c...) = f(1, b, c...) = 1f(1, b, c...) + 0f(0, b, c...) = f(1, b, c...)$$

Quedan por lo tanto demostradas ambas igualdades. Multiplicando la primera de ellas por *a y por* $\bar{a}$ se obtienen respectivamente las relaciones:

$$af(a, b, c...) = af(1, b, c...)$$
$$\bar{a}f(a, b, c...) = \bar{a}f(0, b, c...)$$

Igualmente sumando a y $\bar{a}$ a la segunda igualdad se obtiene:

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

Estas últimas cuatro expresiones se pueden utilizar para simplificar algebraicamente las funciones lógicas. Por ejemplo dada la función:

$$f = abc + \bar{a}(b + a\bar{c} + \bar{a}\bar{b}\bar{c})$$

resulta aplicando la segunda igualdad al segundo sumando:

$$f = abc + \bar{a}(b + \bar{b}\bar{c})$$

El teorema que acabamos de demostrar permite llegar a la conclusión de que

toda función lógica puede transformarse en una función canónica bajo cualquiera de las dos formas anteriormente indicadas.

En efecto según hemos demostrado:

$f(a, b, c...) = a\,f(1, b, c...) + \bar{a}\,f(0, b, c...)$ y dado que:
$f(1, b, c...) = bf(1, 1, c...) + \bar{b}\,f(1, 0, c...)$ y
$f(0, b, c...) = bf(0, 1, c...) + \bar{b}\,f(0, 0, c...)$ resulta
$f(a, b, c...) = ab\,f(1, 1, c...) + a\bar{b}\,f(1, 0, c, ...) + \bar{a}\,b\,f(0, 1, c...) + \bar{a}\,\bar{b}\,f(0, 0, c...)$ y repitiendo el proceso se obtiene finalmente

$$f(a, b, c...) = (abc...)\,f(1, 1, 1...) + ... + (\bar{a}\,\bar{b}\,\bar{c}...)\,f(0, 0, 0...) \qquad [1]$$

Esta expresión indica que una función es igual a la suma de todos los productos canónicos afectados de un coeficiente igual al valor que toma la función al sustituir cada variable por 1 o 0 según en el producto canónico figure en forma directa o inversa respectivamente.

De igual forma se deduce que la expresión en forma de producto de sumas canónicas es:

$$f(a,b,c...) = (a + b + c + ... + f(0,0,0...)) ... (\bar{a} + \bar{b} + \bar{c} + ... + f(1,1,1...)) \qquad [2]$$

De la expresión [1] se deduce que toda función se puede representar mediante la suma de todos los productos canónicos multiplicados por un coeficiente igual a 1 si el término forma parte de la función e igual a 0 si no forma parte de ella.

Igualmente de la expresión [2] se deduce que la expresión canónica de sumas canónicas de una función es igual al producto de todas las sumas canónicas posibles, sumando a cada una de ellas un coeficiente igual a cero, si el término forma parte de la función e igual a 1 si no forma parte de ella.

Siendo 2^n el número de términos canónicos, el número de funciones canónicas de n variables es igual al de variaciones con repetición de dos elementos, 0 y 1, tomados de 2^n en 2^n, es decir 2^{2^n}.

Utilizando la notación numérica anteriormente indicada para expresar los términos canónicos, ambas ecuaciones [1] y [2] se pueden representar de la forma siguiente:

$$f(a,\ b,\ c,...) = \sum_{i=0}^{2^n-1} f(i)i = \prod_{i=0}^{2^n-1} [f(2^n - 1 - i) + i]$$

Estas dos expresiones numéricas permiten pasar con gran facilidad de una forma canónica a la otra, lo cual aplicaremos al tratar la simplificación de circuitos combinacionales.

En efecto, si un producto canónico i existe en una función debido a que su coeficiente es igual a 1, no existirá en la expresión en forma de productos de sumas canónicas de dicha función el término $2^n - 1 - i$ por ser su coeficiente también igual a la unidad. Por lo tanto si se tiene la expresión canónica en forma suma de productos, la expresión canónica en forma de producto de sumas se obtiene mediante el complemento a $2^n - 1$ de los productos canónicos que no forman parte de la función.

Por ejemplo, si $f = \sum_3 (0, 2, 5)$ tendremos:

$$f = \prod_3 (0,1,3,4,6)$$

Esto se demuestra de otra forma muy fácilmente aplicando los teoremas de De Morgan. Denominando al producto canónico por P con un subíndice igual al número decimal que le corresponde y a la suma canónica con una S y el subíndice respectivo, se verifica:

$$\bar{P}_i = S_{2^n-1-i}$$

Por ejemplo:

$$\overline{\bar{a}b\bar{c}} = \bar{P}_2 = a + \bar{b} + c = S_5$$

De igual forma $S_i = \bar{P}_{2^n-1-i}$

La función inversa de f está formada por los términos canónicos que no pertenecen a f. Invirtiendo la función f se obtiene de nuevo f, pero expresada en forma dual.

Por ejemplo si

$$f = \sum_3 (0,3,7)$$

Se verifica

$$\bar{f} = \sum_3 (1,2,4,5,6)$$

$$\bar{\bar{f}} = f = \overline{\sum_3 (1,2,4,5,6)} = \prod_3 (1,2,3,5,6)$$

Cuando una función lógica se presenta de una forma no canónica su transformación en canónica resulta muy sencilla por procedimientos algebraicos.

Si se desea obtener la expresión canónica en forma de suma de productos canónicos, se operará algebraicamente aplicando las propiedades distributivas del producto con respecto a la suma, hasta obtener una expresión de suma de productos no canónicos. Para convertir cada uno de estos productos en canónicos se le multiplica por la suma de las variables que faltan en él y sus inversas.

Un ejemplo aclarará el procedimiento.

Sea la función $f = a(\bar{b} + \bar{c}) + c$

Aplicando la propiedad distributiva del producto con respecto a la suma, resulta:

$$f = a\bar{b} + a\bar{c} + c$$

De acuerdo con lo explicado anteriormente

$$f = a\bar{b}(c + \bar{c}) + a\bar{c}(b + \bar{b}) + c(a + \bar{a}) \cdot (b + \bar{b})$$

Y aplicando de nuevo la propiedad distributiva del producto con respecto a la suma, resulta:

$$f = a\bar{b}c + a\bar{b}\bar{c} + ab\bar{c} + a\bar{b}\bar{c} + abc + \bar{a}bc + a\bar{b}c + \bar{a}\bar{b}c$$

Suprimiendo los términos repetidos de acuerdo con el teorema 4 resulta:

$$f = a\bar{b}c + a\bar{b}\bar{c} + ab\bar{c} + abc + \bar{a}bc + \bar{a}\bar{b}c$$

La función f se puede expresar en la forma numérica abreviada antes indicada:

$$f = \sum_{3}(1,3,4,5,6,7)$$

De igual forma, si se desea obtener la expresión canónica en forma de producto de sumas canónicas, se operará algebraicamente aplicando la propiedad distributiva de la suma con respecto al producto hasta obtener una expresión de producto de sumas no canónicas. Para convertir cada una de estas sumas en canónica se le suma el producto de cada variable que falta en ella por su inversa. Un ejemplo aclarará el procedimiento.

Utilizaremos la misma función que en el caso anterior.

$$f = a(\bar{b} + \bar{c}) + c$$

Aplicamos la propiedad distributiva de la suma con respecto al producto:

$$f = (a + c)(\bar{b} + \bar{c} + c) = a + c$$

De acuerdo con la regla antes expresada resulta:

$$f = a + c + b\bar{b}$$

Y aplicando de nuevo la propiedad distributiva de la suma con respecto al producto, obtenemos:

$$f = (a + b + c)(a + \bar{b} + c)$$

La función f puede también expresarse en forma numérica

$$f = \prod_{3}(5,7)$$

2.4 TABLA DE VERDAD DE UNA FUNCION LOGICA

La tabla de verdad de una función lógica es una forma de representación de la misma, en la que se indica el valor 1 o 0 que toma la función para cada una de las combinaciones posibles de las variables de las cuales depende. En la tabla 2.2 se representa la tabla de verdad de una función de tres variables.

	c	b	a	f
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	1
4	1	0	0	1
5	1	0	1	0
6	1	1	0	1
7	1	1	1	1

TABLA 2.2

La deducción de la forma canónica de la función por medio de la tabla de verdad resulta sencilla.

Si, para una determinada combinación de las entradas, la función toma el valor lógico 1, el producto canónico de todos los posibles 2^n, que vale 1 para dicha combinación, ha de formar parte de la función. La deducción del producto canónico correspondiente es inmediata asignando al estado 0 la variable inversa y al estado 1 la variable directa.

Por ejemplo, el producto canónico correspondiente a la combinación 100 es el $c\bar{b}\bar{a}$. Sólo $c\bar{b}\bar{a}$ vale 1 cuando $c = 1$ y $b = a = 0$.

De lo dicho se deduce que la forma canónica de la función cuya tabla de verdad es la indicada en la tabla 2.2 es:

$$f = a\bar{b}\bar{c} + ab\bar{c} + \bar{a}\bar{b}c + \bar{a}bc + abc$$

Asignando a cada combinación binaria de entrada el número decimal equivalente se obtiene la expresión abreviada de la función:

$$f = \sum_3 (1,3,4,6,7)$$

Por el método indicado anteriormente se deduce la expresión canónica en forma de producto de sumas

$$f = \prod_3 (2,5,7)$$

La forma algebraica de la expresión canónica producto de sumas canónicas se puede también obtener directamente de la tabla de verdad observando las combinaciones para las cuales la función f toma el valor 0 y sustituyendo para cada una de ellas el valor 0 de una variable por su expresión directa y el valor 1 por su expresión inversa. Por ejemplo, en la función que ahora estudiamos correspondiente a la tabla 2.2, por ser $f = 0$ para la combinación $c = b = a = 0$ la suma canónica $a + b + c$ forma parte de la función. Por tanto, la expresión de la misma será:

$$f = (a + b + c)(\bar{a} + b + \bar{c})(a + \bar{b} + c)$$

que, como podemos comprobar, coincide con la expresión numérica antes indicada.

2.5 FUNCIONES IMPORTANTES DE UN ALGEBRA DE BOOLE

En apartados anteriores de este capítulo hemos estudiado, en primer lugar, las funciones básicas de un álgebra de Boole producto lógico, suma lógica, e inversión y posteriormente las funciones NO-O (NOR) y NO-Y (NAND).

Además de estas funciones existen otras cuya importancia se deriva de que pueden ser utilizadas para la realización de las demás funciones lógicas.

2.5.1 Función O-exclusiva

La función O-exclusiva de dos variables a y b es aquella que toma el valor

b	a	f_a
0	0	0
0	1	1
1	0	1
1	1	0

TABLA 2.3

uno cuando una de las variables toma el valor uno y la otra el valor cero o viceversa. La función O-exclusiva se representa mediante el símbolo $\oplus$.

En la tabla 2.3 se representa la tabla de verdad de la función O-exclusiva de dos variables:

$$f_0 = a \oplus b$$

En las figuras 2.10 y 2.11 se representan respectivamente los símbolos antiguo y nuevo normalizado de esta función.

FIGURA 2.10.—**Símbolo lógico no normalizado de la función O-exclusiva.**

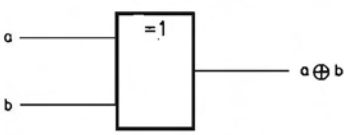


FIGURA 2.11.—**Símbolo lógico normalizado de la función O-exclusiva.**

De la tabla 2.3 se deducen las expresiones canónicas de la función O-exclusiva:

$$f_0 = a\bar{b} + \bar{a}b = (a + b)(\bar{a} + \bar{b})$$

que no son simplificables.

La función O-exclusiva se puede realizar con puertas NO-Y o NO-O, pero el perfeccionamiento de la tecnología de fabricación de los circuitos ha permitido fabricar bloques funcionales en circuito integrado que realizan esta función.

Las propiedades de la función O-exclusiva de n variables se deducen aplicándola primero a dos variables, seguidamente al resultado obtenido y una tercera variable y así sucesivamente.

Se comprueba fácilmente que la función O-exclusiva de n variables toma el valor lógico uno si se encuentra un número impar de ellas en estado uno, y el valor lógico cero si es un número par de ellas el que posee el valor lógico uno:

$$f_0 = a \oplus b \oplus c \oplus d... \oplus n$$

$f_0 = 1$ si un número impar de variables está en uno
$f_0 = 0$ si un número par de variables está en uno
(se considera el cero un número par)

La función O-exclusiva presenta las propiedades siguientes, que el lector puede demostrar a partir de los postulados y teoremas estudiados en los apartados anteriores de este capítulo:

$$f_0 = a \oplus b = \overline{a \oplus \bar{b}} = \overline{\bar{a} \oplus b} = \bar{a} \oplus \bar{b}$$

2.5.2 Función equivalencia o comparación

La función equivalencia de dos o más variables es aquella que toma el valor uno cuando todas las variables de entrada se encuentran en el mismo estado lógico y el valor cero en caso contrario.

La tabla de verdad de la función equivalencia de dos variables se representa en la tabla 2.4. Se observa que esta función es el inverso de la función O-exclusiva de dos variables.

$$f_e = \overline{a \oplus b}$$

Las expresiones canónicas de la función equivalencia se deducen de la tabla 2.4.

b	a	f_e
0	0	1
0	1	0
1	0	0
1	1	1

TABLA 2.4

Aplicando la propiedad de la función O-exclusiva estudiada en el apartado anterior resulta:

$$f_e = \overline{a \oplus b} = a \oplus \bar{b} = \bar{a} \oplus b$$

y por tanto la realización física más sencilla de esta función se logra mediante una puerta O-exclusiva de dos entradas a la cual se le aplica una variable en forma directa y la otra invertida tal como se representa en la figura 2.12 con el símbolo antiguo y en la figura 2.13 con el símbolo nuevo. En el nuevo sistema de representación se ha reservado el indicativo « = » para la función equivalencia que, por tanto, puede representarse mediante el símbolo de la figura 2.14.

FIGURA 2.12.—**Símbolo lógico no normalizado de la función equivalencia.**

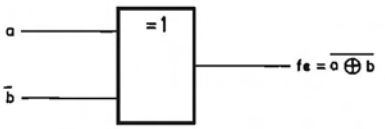


FIGURA 2.13.—**Representación de la función equivalencia utilizando el símbolo lógico normalizado de la función O-exclusiva.**

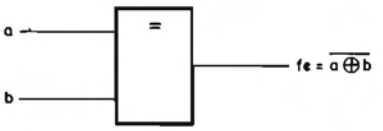


FIGURA 2.14.—**Símbolo lógico normalizado de la función equivalencia.**

PROBLEMAS

1. *a*) Hallar las expresiones canónicas suma de productos y producto de sumas de las funciones:

$$f_1 = ab + a\bar{b}c + \bar{a}\bar{b}$$
$$f_2 = abc\bar{d} + ab\bar{c} + \bar{a}bd$$

b) Representar la tabla de verdad de las dos funciones anteriores.

2. Dada la función f_1 representada mediante la expresión canónica de suma de productos:

$$f_1 = \sum_4 (0, 1, 2, 3, 12, 15)$$

a) Obtener la expresión canónica de producto de sumas.
b) Obtener las dos expresiones canónicas algebraicas de esta función.
c) Representar la tabla de verdad de esta función.

3. La función $f(a, b, c, d)$ cumple la siguiente tabla de verdad.

d	*c*	*b*	*a*	*f*
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

a) Obtener las expresiones numéricas canónicas de suma de productos y de productos de sumas.
b) Obtener las expresiones algebraicas canónicas de esta función.
c) Simplificar las expresiones anteriores por procedimientos algebraicos.

4. Una función de tres variables $f(a, b, c)$ ha de tomar el valor cero cuando la variable b se encuentre en estado uno y la variable a no esté en estado uno. En los demás casos posibles ha de estar en estado uno.

a) Realizar la tabla de verdad de esta función.
b) Obtener las formas canónicas de suma de productos y producto de sumas.

5. Realizar la tabla de verdad de un convertidor de código BCD exceso tres a BCD natural. Indicar mediante el símbolo X el valor de las variables correspondientes a aquellas combinaciones de entrada que no pueden existir.

6. Demostrar las siguientes propiedades de la función O-exclusiva:
a) $a \oplus b = \bar{a} \oplus \bar{b}$
b) $a(b \oplus c) = ab \oplus ac$.

7. Dada la función:

$$f = \overline{a + b} + \bar{a}bc + \overline{a(b + c)}$$

a) Obtener las expresiones canónicas en forma de producto de sumas y suma de productos.
b) Representar la tabla de verdad de esta función.

Capítulo

3

Sistemas combinacionales

3.1 GENERALIDADES

Sistemas lógicos combinacionales son aquellos en los que en cada instante, el estado lógico de sus salidas depende únicamente del estado de sus entradas. Por lo tanto, en ellos no es necesario tener en cuenta la noción de tiempo. Si analizamos esta definición, observamos que un sistema combinacional es realmente una función lógica tal como se ha definido en el capítulo anterior. Por tanto, los sistemas combinacionales pueden ser representados mediante una tabla de verdad o mediante las expresiones numéricas estudiadas en el capítulo anterior correspondiente a una suma de productos canónicos y un producto de sumas canónicas.

En el capítulo anterior se ha estudiado tambien que las funciones lógicas podían simplificarse por métodos algebraicos aplicando los postulados y teoremas de álgebra de Boole. En la práctica, en especial si la función depende de más de tres variables, los métodos algebraicos, aunque son utilizables, no representan una forma sistemática de minimizar las funciones lógicas. Por ello en este capítulo se desarrollan diversos métodos sistemáticos de minimizar las funciones lógicas.

El diseño de todo sistema combinacional se inicia mediante la obtención de una tabla de verdad a partir de las especificaciones que indican los valores que debe tomar la función para cada una de las combinaciones de las variables de entrada de las cuales depende. De la tabla de verdad se deducen las expresiones canónicas a partir de las cuales se realizará la simplificación. Un ejemplo ayudará a aclarar el procedimiento.

Ejemplo 3.1: Diseñar un sistema combinacional de tres variables de entrada a, b y c cuya salida debe tomar el valor uno solamente cuando dos variables de entrada toman el valor uno.

En primer lugar se realiza la tabla de verdad de esta función, para lo cual se indican en orden creciente en el sistema binario natural todas las combinaciones posibles de las variables de entrada a, b y c. (tabla 3.1)

Seguidamente y en una nueva columna se indica el valor que toma la salida f para cada combinación de entrada, de acuerdo con las especificaciones del enun-

ciado. Para cada combinación de entrada en que el número de unos sea distinto de dos, la función ha de tomar el valor cero y, por el contrario, cuando dicho número sea igual a dos, la función f ha de tomar el valor uno. En la tabla 3.1 se representa la tabla de verdad obtenida.

A partir de la tabla de verdad se obtienen las expresiones canónicas por el procedimiento indicado en el apartado 2.4.

Dichas expresiones son:

$$f = \sum_3 (3,5,6)$$

$$f = \prod_3 (0,3,5,6,7)$$

En sucesivos apartados se completará el diseño de este sistema combinacional.

c	b	a	f
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

TABLA 3.1

3.2 SIMPLIFICACION DE LAS FUNCIONES LOGICAS

Los criterios de minimización de la expresión de una función lógica han tenido una gran importancia cuando sólo existían circuitos de pequeña escala de integración. Posteriormente la aparición de los circuitos de escala de integración media disminuyó su interés para el diseñador de sistemas electrónicos. El progreso de la microelectrónica con el desarrollo de los circuitos combinacionales programables (que se estudian en el apartado 3.8 al final de este capítulo) ha hecho que los métodos de minimización se lleven a cabo mediante programas de computador, incorporándolos así a las técnicas de diseño microelectrónico asistido por computador.

El criterio de minimización más utilizado es el de obtener una expresión en forma de suma de productos o producto de sumas que tenga un número mínimo de términos con el menor número de variables posible en cada uno de ellos.

Para obtener una expresión mínima de suma de productos o producto de sumas partiremos de la forma canónica correspondiente. En general se han de obtener ambas expresiones y utilizar la más sencilla de ellas para realizar prácticamente la función.

En los métodos que estudiaremos en los apartados siguientes aplicaremos de una forma sistemática y adecuada la siguiente propiedad del álgebra de Boole:

$$abc... + \bar{a}bc... = bc... \qquad [1]$$

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

La demostración de estas ecuaciones es inmediata aplicando los postulados del álgebra de Boole indicados en el capítulo anterior. La primera de las dos expresiones indican que la suma de dos productos canónicos adyacentes lógicamente, es decir que difieren solamente por el estado de una de las variables, se reduce a un único producto en el cual se ha suprimido dicha variable. La segunda expresión es la dual de la primera y por lo tanto indica lo mismo para las sumas canónicas.

Esta propiedad algebraica de simplificación de las funciones lógicas se puede indicar también de una forma numérica utilizando los equivalentes numéricos de las combinaciones binarias tal como se indicó en el capítulo anterior.

Supongamos una expresión suma de productos canónicos de cuatro variables:

$$abcd + abc\bar{d} + ab\bar{c}d + ab\bar{c}\bar{d}$$

De acuerdo con la ecuación [1] se tiene:

$$abcd + abc\bar{d} = abc(d+\bar{d}) = abc$$
$$\text{y } ab\bar{c}d + ab\bar{c}\bar{d} = ab\bar{c}(d+\bar{d}) = ab\bar{c}$$

Estas ecuaciones se pueden representar en forma numérica

$$\begin{array}{ccc} abcd + abc\bar{d} = abc \\ 15 \qquad 7 \rightarrow 7-15 \\ ab\bar{c}d + ab\bar{c}\bar{d} = ab\bar{c} \\ 11 \qquad 3 \rightarrow 3-11 \end{array}$$

Se observa que los números difieren en una potencia de dos igual al peso de la variable binaria que desaparece.

Los dos términos abc y $ab\bar{c}$ pueden a su vez agruparse:

$$\begin{array}{cc} abc + ab\bar{c} = ab(c+\bar{c}) = ab \\ 7-15 \quad 3-11 \rightarrow 3-7-11-15 \end{array}$$

La expresión resultante es, pues:

$$ab \equiv 3 - 7 - 11 - 15.$$

Otro ejemplo es:

$$\begin{array}{ccc} abc + ab\bar{c} = ab \\ 7 \qquad 3 \rightarrow 3-7 \\ a\bar{b}c + a\bar{b}\bar{c} = a\bar{b} \\ 5 \qquad 1 \rightarrow 1-5 \\ a\bar{b} \quad + \quad ab \quad = \quad a \\ 1-5 \quad 3-7 \rightarrow 1-3-5-7 \end{array}$$

Se observa que el número de términos canónicos que quedan comprendidos en un término reducido es una potencia de dos.

Aplicando estas propiedades se logra por lo tanto reducir al mínimo cualquier expresión lógica en forma de producto de sumas o de suma de productos. La expresión final, en la que no se puede suprimir ningún término ni eliminar variables de ellos, se denomina expresión irreducible.

Combinando los términos para su simplificación de diferentes maneras, se

obtienen a veces varias expresiones irreducibles. De todas ellas es necesario elegir la menos compleja para su realización.

3.2.1 Métodos tabulares de Karnaugh y Veitch de simplificación de las funciones lógicas

Aunque en cierto modo es sistemática la aplicación directa del método algebraico, no lo es totalmente porque, en general, existen diversas formas de agrupar los términos para su reducción y por tanto, varias expresiones irreducibles. Por ello se idearon los métodos tabulares que constituyen una forma gráfica de representar la tabla de verdad de una función lógica.

Se ha visto en el apartado anterior que los términos canónicos adyacentes pue-

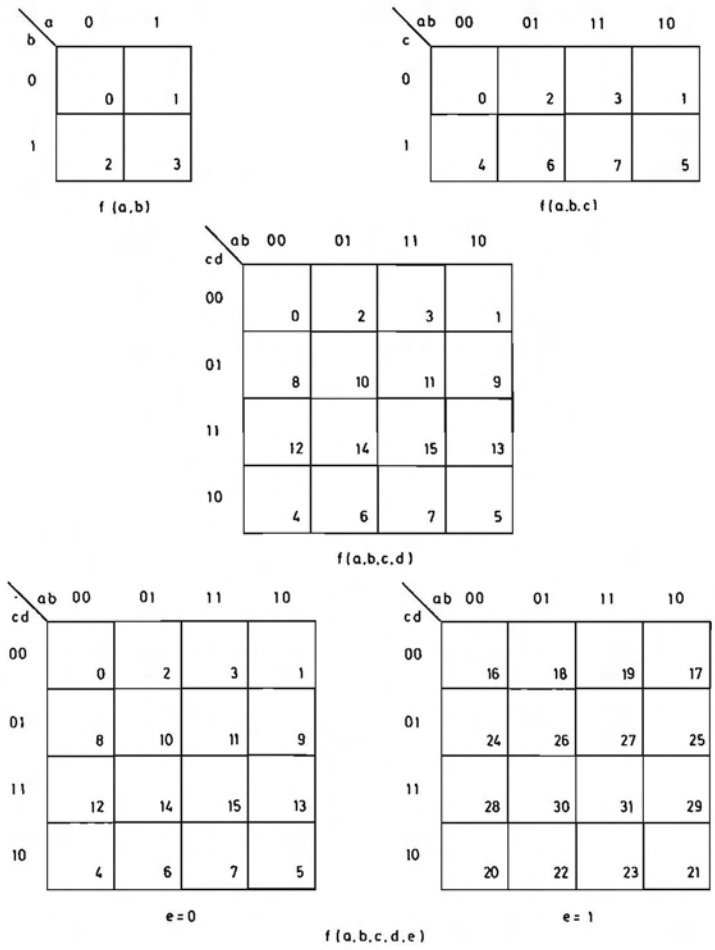


FIGURA 3.1.—**Tablas de Karnaugh.**

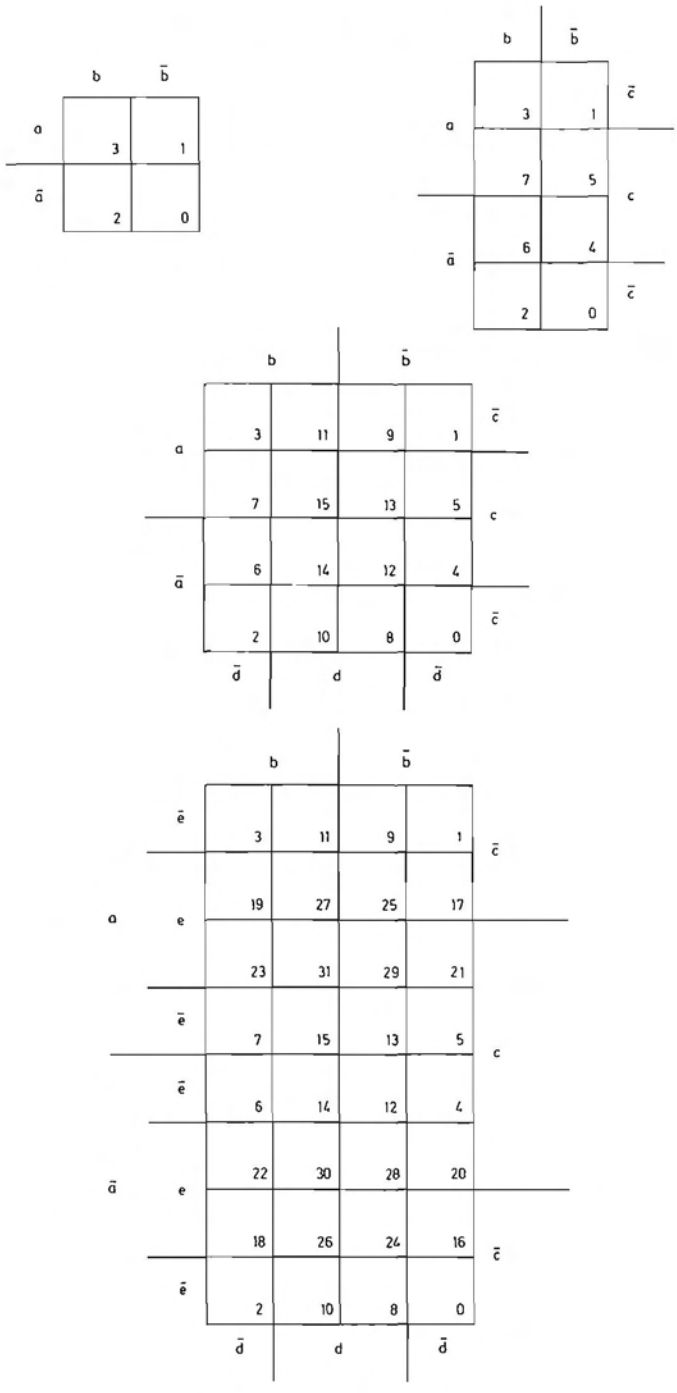


FIGURA 3.2.—**Tablas de Veitch.**

den reducirse a un solo término en el cual se ha suprimido la variable cuyo estado es diferente en ambos.

En los métodos tabulares los términos canónicos adyacentes se agrupan en una tabla de tal manera que estén físicamente contiguos y por tanto sea muy sencillo realizar las agrupaciones que permiten reducir al mínimo la expresión de la función.

Se han adoptado dos formas diferentes de realizar las tablas, que reciben los nombres de los primeros matemáticos que las realizaron, Karnaugh y Veitch.

En la figura 3.1 se representan las tablas de Karnaugh de funciones de dos a cinco variables y en la figura 3.2 las de Veitch de igual número de variables.

Tal como se observa en ambas figuras, cada cuadrado corresponde a un término (producto o suma) canónico cuyo número se indica en el vértice inferior. Los cuadros que tienen un lado común, es decir, que son físicamente adyacentes, corresponden a términos canónicos que son lógicamente adyacentes y los números decimales que les corresponden se diferencian en una potencia de dos. Además, los cuadrados de la fila superior son adyacentes a los respectivos de la fila inferior y los de la columna de la izquierda a los de la derecha.

En la tabla de Karnaugh de cinco variables, los de la tabla de la izquierda son adyacentes a los correspondientes de la tabla de la derecha (p. ej., el 0 y el 16) y en la tabla de Veitch de cinco variables existe un eje de adyacencia horizontal que divide a la tabla en dos partes cuyos cuadrados simétricos son adyacentes dos a dos.

En este estudio se utilizan exclusivamente las tablas de Karnaugh. Los cuadrados correspondientes a los términos canónicos que forman parte de la función se indican mediante un uno y los correspondientes a los términos que no forman parte de la función se dejan en blanco.

Para obtener la expresión algebraica más sencilla de la función es necesario realizar en la tabla el mínimo número de agrupaciones de términos de la máxima complejidad, de modo que cada uno cubra todos los unos de la tabla. De acuerdo con lo indicado en el apartado 3.2, el número de términos canónicos adyacentes que pueden agruparse es una potencia de dos.

El procedimiento sistemático de obtener la expresión más simple es el siguiente:

1. Se toman todos los «unos» que no se pueden combinar con ningún otro.
2. Se forman los grupos de dos «unos» que no pueden formar un grupo de cuatro.
3. Se forman los grupos de cuatro «unos» que no pueden formar un grupo de ocho.
4. Cuando se hayan cubierto todos los unos se detiene el proceso.

En la práctica es necesario realizar este proceso para ambas expresiones canónicas y elegir la más sencilla de las resultantes.

Varios ejemplos aclararán lo expuesto.

Ejemplo 3.2

Sea la función

$$f_1 = \sum_4 (2, 3, 5, 7, 10, 11, 15) = \prod_4 (1, 2, 3, 6, 7, 9, 11, 14, 15)$$

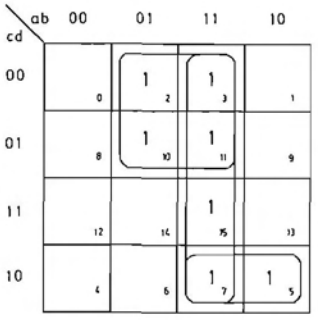


FIGURA 3.3.—**Tabla de Karnaugh de la expresión canónica de la función f_1 en forma de suma de productos.**

En la figura 3.3 se representa la tabla de Karnaugh de la expresión canónica de f_1 en forma de suma de productos. La expresión mínima es única y está compuesta por dos agrupaciones de cuatro términos y una de dos términos. El producto lógico que corresponde a cada grupo se obtiene eliminando las variables que toman el valor 0 en la mitad de las células y el valor 1 en la otra mitad y asignando la forma directa a la variable que toma el valor 1 y la forma inversa a la que toma el valor 0, de acuerdo con el convenio indicado en el capítulo 2. Por ejemplo, al agrupamiento formado por las casillas 2, 3, 10 y 11 le corresponde el producto $b\bar{c}$.

La expresión algebraica resultante de f_1 es, pues:

$$f_1 = b\bar{c} + ab + ac\bar{d}$$

Simplificando la expresión canónica en forma de producto de sumas cuya tabla de Karnaugh se representa en la figura 3.4 se tiene:

$$f_1 = (a + \bar{c})(b + c)(b + \bar{d})$$

Se observa que esta segunda expresión es más sencilla que la primera porque contiene el mismo número de términos y una variable menos en uno de ellos. Por tanto deberá ser utilizada para realizar en la práctica la función.

Al realizar la minimización de una función puede suceder que exista más de

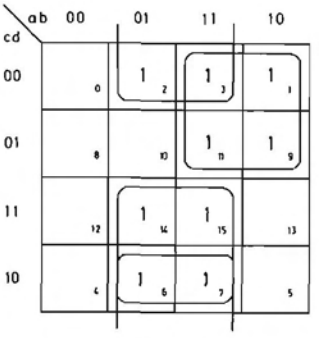


FIGURA 3.4.—**Tabla de Karnaugh de la expresión canónica de la función f_1 en forma de producto de sumas.**

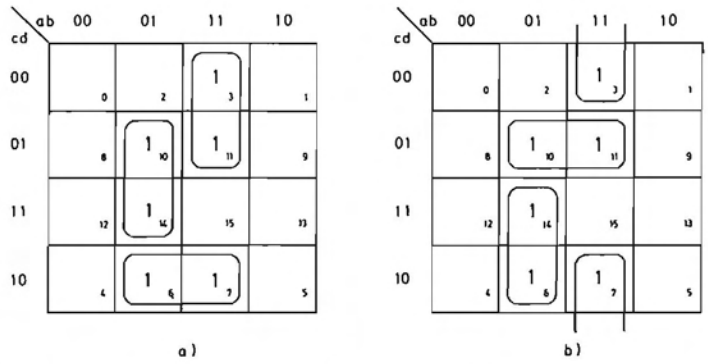


FIGURA 3.5.—**Tablas de Karnaugh de la expresión canónica de la función f_2 en forma de producto de sumas.**

una forma mínima irreducible. En este caso se puede elegir cualquiera de las dos para realizar la función tal como se indica en el ejemplo siguiente.

Ejemplo 3.3: Sea la función:

$$f_2 = \sum_4 (3, 6, 7, 10, 11, 14)$$

En la figura 3.5 se presentan dos tablas de Karnaugh de esta función correspondientes a las dos formas posibles de agrupar los términos canónicos. Por ejemplo el término 3 se puede agrupar con el 11 o con el 7. Las dos expresiones mínimas que corresponden respectivamente a las tablas *a* y *b* de la figura 3.5 son:

$$f_2 = ab\bar{c} + \bar{a}bd + bc\bar{d}$$
$$f_2 = ab\bar{d} + b\bar{c}d + \bar{a}bc$$

Ejemplo 3.4: Se continúa el diseño del sistema del ejemplo 3.1, simplificando las expresiones canónicas obtenidas, que fueron las siguientes:

$$f = \sum_3 (3, 5, 6) = \prod_3 (0, 3, 5, 6, 7)$$

Las tablas de Karnaugh de ambas expresiones se presentan en las figuras 3.6 y 3.7.

En la tabla de la figura 3.6 de la suma de productos canónicos se observa que no existen términos adyacentes y por tanto la expresión mínima es equivalente a la canónica:

$$f = ab\bar{c} + \bar{a}bc + a\bar{b}c$$

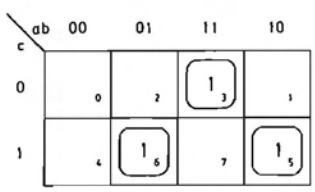


FIGURA 3.6.—**Tabla de Karnaugh de la función $f = \sum_3 (3, 5, 6)$.**

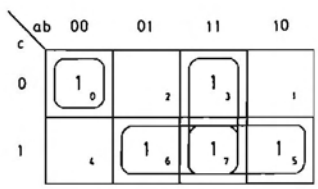


FIGURA 3.7.—**Tabla de Karnaugh de la función $f = \prod_3 (0, 3, 5, 6, 7)$.**

En la tabla de la figura 3.7 del producto de sumas canónicas se observa que la suma canónica 3 se puede agrupar con la 7. De igual forma la 6 y la 5 se pueden agrupar con la 7. No existe ninguna suma canónica adyacente a la 0 que tome el valor lógico uno y por tanto ésta se ha de realizar por sí sola. La expresión resultante obtenida es:

$$f = (\bar{a} + \bar{b} + \bar{c})(a + b)(b + c)(a + c)$$

La aplicación del método tabular de Karnaugh, tal como se acaba de exponer, a las funciones de más de cinco variables no resulta realizable en la práctica porque es difícil observar todos los términos adyacentes. Por ello tiene gran interés el método que se expone seguidamente, que permite tratar funciones de $n + 1$, $n + 2$, etc. variables con una tabla de Karnaugh de n variables.

En el procedimiento que se acaba de estudiar se coloca un uno en los cuadrados de la tabla de Karnaugh de suma de productos correspondientes a aquellos productos canónicos que forman parte de la función, y un cero en todos los demás.

Con una tabla de n variables es posible representar una función de $n + 1$ asignando a cada cuadrado dos términos canónicos en lugar de uno solo. De esta forma cada cuadrado puede contener un uno, un cero, una de las $n + 1$ variables en forma directa o en forma inversa. La elección de la variable es arbitraria.

La mejor manera de desarrollar lo que se acaba de exponer es mediante un ejemplo.

Ejemplo 3.5:

Sea la función de cinco variables:

$$f(a, b, c, d, e) = \sum (0, 1, 2, 3, 8, 9, 10, 16, 17, 18, 19, 24, 25, 27)$$

Esta función se puede simplificar mediante el procedimiento general realizando las tablas indicadas en la figura 3.8*a* y teniendo en cuenta que los términos simétricos de ambas tablas son adyacentes porque se diferencian solamente en el estado de la variable *e*.

Pero se puede realizar más fácilmente la simplificación si ambas tablas se reúnen en una sola asignando a cada cuadrado dos términos canónicos. Se obtiene así la tabla de la figura 3.8*b*. A todos aquellos cuadrados que tienen un uno en ambas tablas de la figura 3.8*a* se les asigna un uno en la tabla de la figura 3.8*b*. A aquellos cuadrados que poseen un uno en una tabla y un cero en la otra se les asigna el estado de la variable *e* correspondiente a la tabla del cuadrado al que le corresponde el uno. Por ejemplo, al cuadrado 10 le corresponde un uno y al 26 un cero en las tablas de la figura 3.8*a*, y por ello al cuadrado 10-26 de la figura 3.8*b* se le asigna la variable $\bar{e}$. Por la misma razón al cuadrado 11-27 se le asigna la variable *e*.

El agrupamiento de términos en la tabla de la figura 3.8*b* se realiza de la forma siguiente:

a) Se agrupan entre sí los términos que poseen la variable *e* y con los que poseen uno, de tal manera que se cubran de la forma más sencilla posible todos los cuadra-

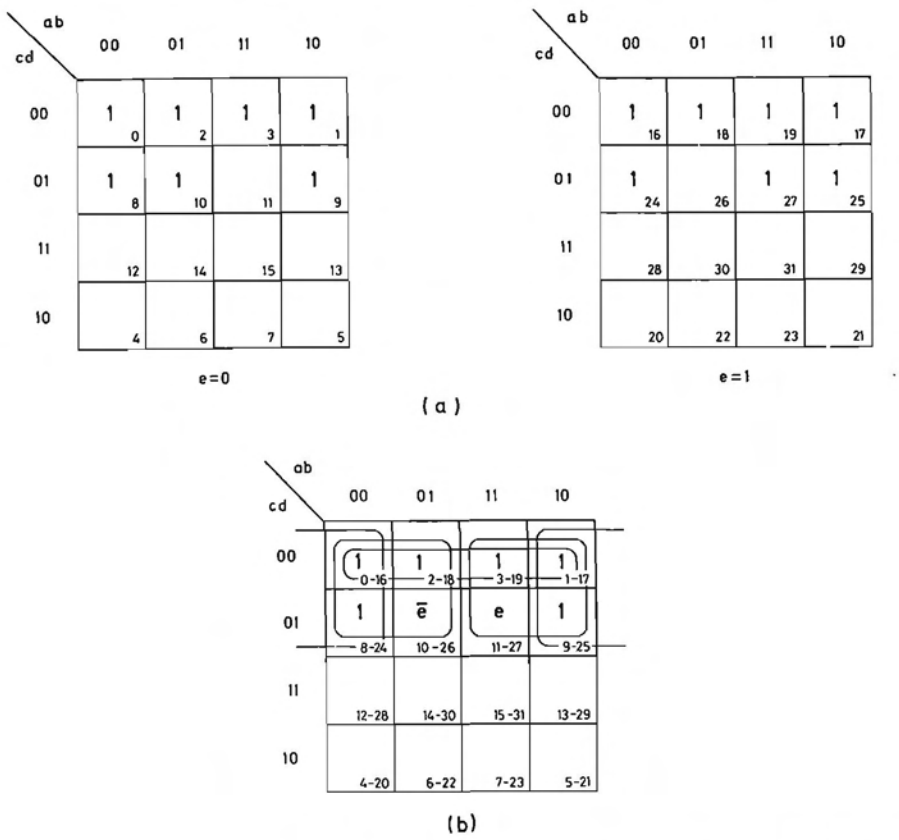


FIGURA 3.8.—**Tablas de Karnaugh de la función** $f(a,b,c,d,e) = \sum^{3}(0, 1, 2, 3, 8, 9, 10, 16, 17, 18, 19, 24, 25, 27)$: *a*) **Tablas independientes para** $e = 0$ **y** $e = 1$; *b*) **Tabla única.**

dos que poseen la variable *e*. En la expresión del término obtenido aparece la variable *e* en forma directa.

Se obtiene así el término:

$$3 - 19 - 1 - 17 - 11 - 27 - 9 - 25 \equiv a\bar{c}e$$

b) Se agrupan entre sí todos los cuadros que poseen $\bar{e}$ y con los que poseen uno, de tal manera que se cubran de la forma más sencilla posible todos los cuadrados que poseen la variable $\bar{e}$. En la expresión del término obtenido aparece la variable *e* en forma inversa.

Se obtiene así el término:

$$0 - 16 - 2 - 18 - 8 - 24 - 10 - 26 \equiv \bar{a}\bar{c}\bar{e}$$

c) Se agrupan de la forma más sencilla posible todos los términos que poseen un uno y que no han sido realizados conjuntamente con *e* y con $\bar{e}$ en los pasos anteriores.

Como en la figura 3.8*b* no existe ningún uno realizado conjuntamente con e y con $\bar{e}$ se unen entre sí los unos de la fila superior y se obtiene así el término:

$$0-16-2-18-3-19-1-17 \equiv \bar{c}\bar{d}$$

De igual forma se obtiene el término:

$$0-16-1-17-8-24-9-25 \equiv \bar{b}\bar{c}$$

Por lo tanto, la expresión resultante de la función f es:

$$f = \bar{c}d + \bar{b}\bar{c} + a\bar{c}e + \bar{a}\bar{c}\bar{e}$$

Si, por el contrario, a la función f le corresponde la tabla de Karnaugh de la figura 3.9 su expresión numérica es:

$$f = ebd + \bar{e}ad$$

Los términos que valen uno no hay que realizarlos de forma independiente porque ya están comprendidos tanto en el término que cubre la variable e como en el que cubre la variable $\bar{e}$.

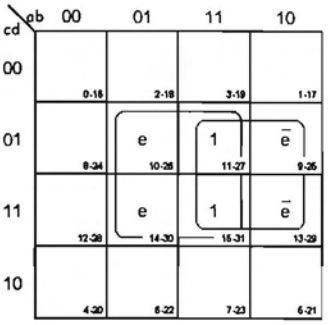


FIGURA 3.9

Para facilitar la comprensión del procedimiento se realizaron las tablas de la figura 3.8*a*, pero en la práctica se obtiene la tabla de la figura 3.8*b* directamente.

En una tabla de Karnaugh de cuatro variables se pueden representar funciones de 6 y más variables, en especial en el caso de que las funciones no estén definidas para todas las combinaciones posibles de las variables de entrada tal como se explica en el apartado 3.3.

3.2.2 Método numérico de Quine-McCluskey de simplificación de los sistemas lógicos combinacionales

Los métodos tabulares estudiados anteriormente son de aplicación práctica para simplificar funciones de un máximo de cuatro a cinco variables, pero, cuando dicho

número es mayor y también cuando se trata de una multifunción cuya simplificación se estudiará en un apartado posterior, es necesario recurrir a otros métodos, de los cuales el numérico es el de uso más corriente.

Por otra parte, el método numérico es adecuado para ser realizado mediante un algoritmo ejecutable por un programa de computador. Resulta, por lo tanto, un método idóneo en las técnicas de diseño asistido por computador.

Tabla 1

N.° de unos		
0	0	x
	1	x
1	4	x
	8	x
	3	x
2	12	x
	7	x
3	11	x
	13	x
4	15	x

Tabla 2

		Dif.
0-1		1
0-4	x	4
0-8	x	8
1-3		2
4-12	x	8
8-12	x	4
3-7	x	4
3-11	x	8
12-13		1
7-15	x	8
11-15	x	4
13-15		2

Tabla 3

	Dif.
0-4-8-12	(4,8)
0-8-4-12	(8,4)
3-7-11-15	(4,8)
3-11-7-15	(8,4)

Tabla de Quine-McCluskey de la función

$f(a, b, c, d) = \sum_4(0, 1, 3, 4, 7, 8, 11, 12, 13, 15)$

TABLA 3.2

Este método está basado en el convenio estudiado en el capítulo 2 de asignar un número decimal a cada término canónico. Si dos términos canónicos difieren en una sola variable, se ha visto anteriormente que sus números correspondientes difieren en una potencia de dos y que pueden agruparse en un solo término que ya no es canónico y al cual le falta dicha variable. A su vez, si dos términos a los cuales les falte la misma variable difieren en una misma potencia de dos, pueden ser agrupados en un nuevo término al cual le falte la variable correspondiente a dicha diferencia.

Repitiendo este proceso se logra obtener todos los términos primos, que son aquellos que contienen el máximo número de términos canónicos de la función y que, por otra parte, no existe ningún término de menor complejidad que los contenga.

En la tabla 3.2 se representan las tablas numéricas de la función de cuatro variables $f(a, b, c, d) = \sum_4 (0, 1, 3, 4, 7, 8, 11, 12, 13, 15)$.

Primero se forma una tabla en la que los términos canónicos se ordenan en grupos de acuerdo con el número de unos que posee la combinación binaria que les corresponde. De esta forma, para realizar las agrupaciones es necesario comparar solamente los términos de cada grupo con los del siguiente.

Partiendo de esta primera tabla se forma una segunda comparando los términos canónicos que pertenecen a grupos adyacentes y agrupando en un solo término

aquellos cuya diferencia (se toma como minuendo el término que contiene mayor número de unos) sea una potencia de dos positiva. Por ejemplo, en la función de cuatro variables representada en la tabla 3.2 se pueden agrupar los términos canónicos 4 y 12 porque $12 - 4 = 8 = 2^3$ y además se puede comprobar que sus equivalentes binarios respectivos 0100 y 1100 son adyacentes. Por el contrario, 4 y 3 no se pueden agrupar porque $3 - 4 = -1$ y, en efecto, sus equivalentes binarios 0100 y 0011 no son adyacentes. Todos los términos de la primera tabla que han sido utilizados para realizar la segunda se marcan con una cruz, lo cual indica que no son términos primos.

En esta segunda tabla existe una columna en la cual se indica la diferencia entre los términos canónicos que forman parte de cada elemento de la misma. A partir de ella se forma una tercera tabla agrupando los términos pertenecientes a grupos adyacentes cuya diferencia es igual y que además difieren entre sí en una potencia de dos. Por ejemplo, los términos 0 — 4 y 8 — 12 de la segunda tabla de la tabla 3.2 se pueden agrupar entre sí porque su diferencia es la misma (4) y además difieren en una potencia de dos positiva (8 — 0 = 12 — 4 = 8).

En la tercera tabla se indican en una segunda columna ambas diferencias, la interna de cada grupo y la que existe entre los grupos que se unen. Por ejemplo, la diferencia del grupo 0 — 4 — 8 — 12 es 4,8. El proceso se continúa realizando tablas sucesivas hasta que no es posible realizar más agrupaciones.

Debido a que una expresión formada por el agrupamiento de cuatro términos canónicos adyacentes puede obtenerse de dos formas diferentes, tal como se indica en la tabla de Karnaugh de la figura 3.10 se obtienen en esta tabla todos los términos por duplicado con diferente ordenación. De ellos solamente es necesario considerar uno, por ejemplo, aquel en que los términos están ordenados en orden creciente.

La cuarta tabla, en caso de que sea posible, se forma agrupando los términos de grupos adyacentes de la tercera tabla cuyas diferencias coinciden y que además difieren en una potencia de dos.

Una vez obtenidas todas las tablas se tienen todos los términos primos que pueden utilizarse para realizar la función, que son aquellos que no han sido marcados con una cruz.

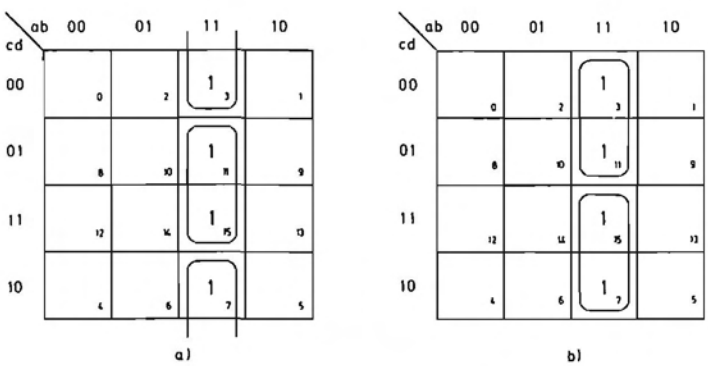


FIGURA 3.10.—**Diferentes formas de agrupar los términos adyacentes en una tabla de Karnaugh.**

	Términos primos	0	1	3	4	7	8	11	12	13	15
A	0 - 1	X	X								
B	1 - 3		X	X							
C	12 - 13								X	X	
D	13 - 15									X	X
E	0 - 4 - 8 - 12	X			⊗		⊗		X		
F	3 - 7 - 11 - 15			X		⊗		⊗			X

TABLA 3.3.—**Tabla de los términos primos de la función** $f = \sum_{d} (0, 1, 3, 4, 7, 8, 11, 12, 13, 15)$.

En el ejemplo de la tabla 3.2 los términos primos son:

0-1, 1-3, 12-13, 13-15, 0-4-8-12 y 3-7-11-15.

Ahora es necesario elegir la mínima combinación de estos términos primos que cubra la función. Para ello se puede utilizar una representación como la indicada en la tabla 3.3 para la función que hemos estudiado en párrafos anteriores. Esta tabla contiene una columna por cada término canónico que forma parte de la función y una fila por cada término primo.

En la fila correspondiente a un determinado término primo, se coloca el símbolo X en las columnas cuyo número está contenido en el término.

En primer lugar se observa en la tabla si existe alguna columna que contenga un solo símbolo X, y se indican mediante una flecha. El término primo que realice esa X es esencial, es decir, ineludiblemente ha de formar parte de la función. Por tanto, todos los términos canónicos incluidos en ese término primo esencial quedan realizados por él. En la tabla 3.3, los términos 0-4-8-12 y 3-7-11-15 son esenciales y por tanto, al formar parte de la expresión final, quedan realizados los términos 0, 3, 4, 7, 8, 11, 12 y 15 que se indican en un recuadro.

Ahora es necesario elegir la combinación más sencilla de los términos primos restantes que realiza el resto de los términos canónicos.

Esta elección se puede hacer de dos formas diferentes. La primera es realizar una tabla reducida tal como se indica en la tabla 3.4 cuyas columnas son los

Términos primos no esenciales	1	13
0 - 1	X	
1 - 3	X	
12 - 13		X
13 - 15		X

TABLA 3.4.—**Tabla reducida**

términos canónicos que todavía no han sido realizados y cuyas filas corresponden a los términos primos no esenciales. Se observa que, para realizar el producto canónico 1, se pueden utilizar el término 0-1 o el 1-3 y para realizar el 13, se pueden utilizar el 12-13 o, el 13-15. Por lo tanto existen cuatro formas de combinar los términos primos para realizar el 1 y 13 que son:

0-1 y 12-13; 0-1 y 13-15; 1-3 y 12-13; 1-3 y 13-15.

La segunda forma de realizarlo, que es más sistemática, consiste en la obtención de una expresión algebraica igual al producto lógico de las sumas lógicas de los términos primos disponibles para realizar cada término canónico. Para facilitar la escritura, le asignaremos a cada término primo una letra (tabla 3.3).

Para realizar el término 1 en la tabla 3.3 se dispone del 0-1 o el 1-3 y para realizar el 13 se dispone de los términos 12-13 o 13-15.

Por lo tanto la expresión algebraica será

$$(A + B)\,(C + D)$$

y aplicando la propiedad distributiva del producto con respecto a la suma lógica se tiene:

$$(A + B)\,(C + D) = AC + AD + BC + BD$$

Por lo tanto, se puede coger cualquiera de las cuatro combinaciones A y C, A y D, B y C o B y D para realizar los términos 1 y 13 y el resultado es el mismo que el obtenido anteriormente.

Las expresiones más reducidas de f son las cuatro siguientes:

$$\begin{aligned} f &= (0\text{-}4\text{-}8\text{-}12) + (3\text{-}7\text{-}11\text{-}15) + (0\text{-}1) + (12\text{-}13) \\ f &= (0\text{-}4\text{-}8\text{-}12) + (3\text{-}7\text{-}11\text{-}15) + (0\text{-}1) + (13\text{-}15) \\ f &= (0\text{-}4\text{-}8\text{-}12) + (3\text{-}7\text{-}11\text{-}15) + (1\text{-}3) + (12\text{-}13) \\ f &= (0\text{-}4\text{-}8\text{-}12) + (3\text{-}7\text{-}11\text{-}15) + (1\text{-}3) + (13\text{-}15) \end{aligned}$$

Obtener las expresiones algebraicas de f resulta muy sencillo partiendo de las expresiones numéricas y recordando lo explicado en el apartado 3.2.

En efecto, la expresión algebraica de 0-4-8-12 se obtiene teniendo en cuenta que las variables que desaparecen en la expresión $\bar{a}\bar{b}\bar{c}\bar{d}$ del término canónico 0 son la de peso 4 ($4 - 0 = 4$), que es la c, y la de peso 8 ($8 - 0 = 8$), que es la d. Por tanto

$$0\text{-}4\text{-}8\text{-}12 \equiv \bar{a}\bar{b}$$

Las expresiones algebraicas de los restantes términos son:

$$\begin{aligned} 3\text{-}7\text{-}11\text{-}15 &\equiv a\,b \\ 0\text{-}1 &\equiv \bar{b}\,\bar{c}\,\bar{d} \\ 1\text{-}3 &\equiv a\,\bar{c}\,\bar{d} \\ 12\text{-}13 &\equiv \bar{b}\,c\,d \\ 13\text{-}15 &\equiv a\,c\,d \end{aligned}$$

y por tanto las cuatro expresiones algebraicas mínimas de la función f son:

$$f = \bar{a}\,\bar{b} + a\,b + \bar{b}\,\bar{c}\,\bar{d} + \bar{b}\,c\,d$$
$$f = \bar{a}\,\bar{b} + a\,b + \bar{b}\,\bar{c}\,\bar{d} + a\,c\,d$$
$$f = \bar{a}\,\bar{b} + a\,b + a\,\bar{c}\,\bar{d} + \bar{b}\,c\,d$$
$$f = \bar{a}\,\bar{b} + a\,b + a\,\bar{c}\,\bar{d} + a\,c\,d$$

Cuando la tabla de términos primos no esenciales contiene un número elevado de aquéllos es conveniente observar si alguna fila está cubierta por otra. Si todos los términos canónicos realizados por una fila son a su vez realizados por otra a la cual le corresponde un término primo de igual o menor complicación que aquélla, se dice que la primera está cubierta por la segunda y puede ser suprimida porque no formará parte de la solución mínima de la función.

En las tablas 3.5 y 3.6 se indica un ejemplo que aclara lo dicho.

La tabla 3.5 representa una tabla de términos primos no esenciales obtenida partiendo de la tabla general de términos primos, suprimiendo las filas correspondientes a los términos esenciales y las columnas cubiertas por ellos. Observando dicha tabla se comprueba que la línea *A* está cubierta por la *D* porque ésta realiza todos los términos de aquélla y a ambas le corresponden términos primos de igual complejidad. Igualmente la línea *B* cubre a la *C*. Por tanto pueden suprimirse las líneas *A* y *C* y la tabla resultante se indica en la tabla 3.6.

Términos primos no esenciales		2	3	8	10
A	0-1-2-3	X	X		
B	0-2-8-10	X		X	X
C	0-4-8-12			X	
D	2-3-10-11	X	X		X

TABLA 3.5

Términos primos no esenciales		2	3	8	10
B	0-2-8-10	X		X	X
D	2-3-10-11	X	X		X

TABLA 3.6

3.3 FUNCIONES INCOMPLETAS: DEFINICION Y APLICACION DE LOS METODOS DE SIMPLIFICACION

Hasta ahora se han estudiado funciones lógicas en las que para cada combinación de las entradas se define un valor lógico uno o cero de la función. Estas funciones se denominan totalmente definidas.

También existen funciones no totalmente definidas llamadas funciones incompletas, que son aquellas en las que, para una o más combinaciones de entrada, a la salida se le puede asignar el valor cero o el uno indistintamente. Esta situación puede presentarse por las dos causas siguientes:

1. No pueden existir una o más combinaciones de entrada. Por tanto, a la salida correspondiente a esas combinaciones se le puede asignar el valor cero o el valor uno.
2. Cuando aparecen una o más combinaciones de entrada, la acción de la salida del sistema lógico está inhibida. En consecuencia el valor de la salida para esas combinaciones de entrada también es indiferente.

La forma canónica de una función incompleta se representa indicando separadamente los términos canónicos para los cuales la función vale uno y los términos canónicos para los cuales es indiferente (lo cual se indica mediante el símbolo ϕ).

Sea, por ejemplo, la función incompleta

$$f(a, b, c, d) = \sum_{4} (1, 3, 6, 8, 10, 11) + \sum_{\phi} (0, 2, 4, 12, 13)$$

En la tabla de verdad de esta función que se representa en la tabla 3.7 se coloca un signo X en las posiciones de f correspondientes a las combinaciones de entrada para las cuales no está definido el estado de la salida.

La obtención de una de las expresiones canónicas de una función incompleta partiendo de la otra se realiza sin gran dificultad. La función inversa $\bar{f}$ será indiferente para las mismas combinaciones de entrada. Su expresión canónica en forma de suma de productos estará formada por una parte de términos indiferentes que coincide con la de f, y además serán uno los términos para los cuales f es cero.

Aplicando lo dicho al ejemplo de la tabla 3.7 resulta:

$$\bar{f} = \sum_{4} (5, 7, 9, 14, 15) + \sum_{\phi} (0, 2, 4, 12, 13)$$

Invirtiendo f y aplicando lo estudiado en el capítulo 2 resulta

$$\bar{\bar{f}} = f = \prod_{4} (0, 1, 6, 8, 10) \prod_{\phi} (2, 3, 11, 13, 15)$$

La minimización de estas funciones se puede realizar mediante cualquiera de los dos métodos que acabamos de estudiar.

La única diferencia con respecto a las funciones totalmente definidas es que a los términos indiferentes se les puede asignar a voluntad el valor cero o el valor uno. Por tanto el método tabular de Karnaugh se aplica de la misma forma que en el apartado 3.3 y, para realizar cada término de la función que toma el valor uno, se agrupan con él el máximo número de términos posible. Para lograrlo se asigna el valor uno a aquellos términos indiferentes que permitan simplificar la función y el valor cero a los restantes.

En la figura 3.11 se representan las tablas de Karnaugh de la expresión canónica en forma de suma de productos canónicos de la función cuya tabla de verdad se

d	*c*	*b*	*a*	*f*
0	0	0	0	X
0	0	0	1	1
0	0	1	0	X
0	0	1	1	1
0	1	0	0	X
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	X
1	1	0	1	X
1	1	1	0	0
1	1	1	1	0

TABLA 3.7

representa en la tabla 3.7. Se observa que existen dos expresiones mínimas de suma de productos representadas en las tablas *a* y *b* de la figura 3.11. La primera se obtiene realizando el producto canónico 8 mediante el término 0-2-8-10, y la segunda, realizándolo mediante el término 0-4-8-12. En la tabla *a* se asigna el valor cero a los productos canónicos indiferentes 12 y 13 y el valor uno a los productos canónicos 0, 2 y 4. En la tabla *b* se asigna el valor cero al producto canónico 13 y el valor uno a los productos canónicos 0, 2, 4 y 12.

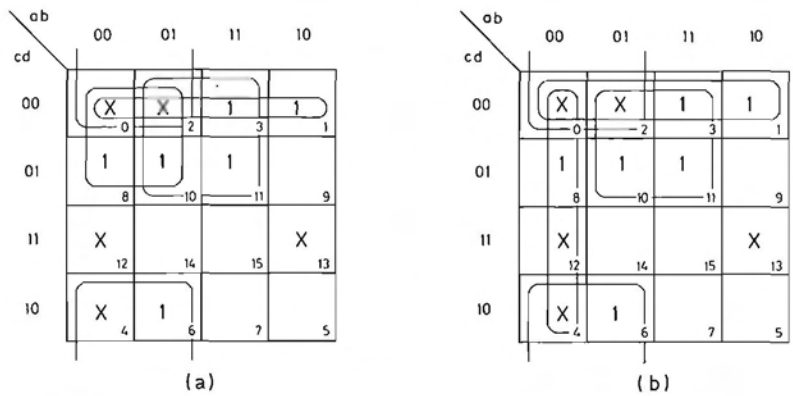


FIGURA 3.11. —**Tablas de Karnaugh de la función** $f = \sum_{4} (1, 3, 6, 8, 10, 11) + \sum_{\delta} (0, 2, 4, 12, 13)$.

Las expresiones algebraicas correspondientes a ambas tablas de la figura 3.11*a* y *b* son respectivamente:

$$f = \bar{a}\bar{c} + b\bar{c} + \bar{c}\bar{d} + \bar{a}\bar{d}$$
$$f = \bar{a}\bar{b} + b\bar{c} + \bar{c}\bar{d} + \bar{a}\bar{d}$$

La minimización de esta misma función por el método numérico se representa en las tablas 3.8 y 3.9. En la tabla 3.8 se representan las tablas de agrupación de los productos canónicos obtenidos de la forma explicada en el apartado 3.2.2. En la tabla 1 se incluyen los productos canónicos que toman el valor uno y los productos canónicos indiferentes y con ellos se obtienen los términos primos.

Tabla 1			Tabla 2			Tabla 3		Tabla 3 *reducida*	
N.° de unos					Dif.		Dif.		Dif.
0	0	X	0 - 1	X	1	0 - 1 - 2 - 3	1,2	0 - 1 - 2 - 3	1,2
	1	X	0 - 2	X	2	0 - 2 - 1 - 3	2,1	0 - 2 - 4 - 6	2,4
	2	X	0 - 4	X	4	0 - 2 - 4 - 6	2,4	0 - 2 - 8 -10	2,8
1	4	X	0 - 8	X	8	0 - 2 - 8 -10	2,8	0 - 4 - 8 -12	4,8
	8	X	1 - 3	X	2	0 - 4 - 2 - 6	4,2	2 - 3 -10 -11	1,8
	3	X	2 - 3	X	1	0 - 4 - 8 -12	4,8		
	6	X	2 - 6	X	4	0 - 8 - 2 -10	8,2		
2	10	X	2 -10	X	8	0 - 8 - 4 -12	8,4		
	12	X	4 - 6	X	2	2 - 3 -10 -11	1,8		
	11	X	4 -12	X	8	2 -10 - 3 -11	8,1		
3	13	X	8 -10	X	2				
			8 -12	X	4				
			3 -11	X	8				
			10 -11	X	1				
			12 -13		1				

TABLA 3.8

Términos primos	1	3	6	8	10	11
0 - 1 - 2 - 3	⊗	X				
0 - 2 - 4 - 6			⊗			
0 - 2 - 8 -10				X	X	
0 - 4 - 8 -12				X		
2 - 3 -10 -11		X			X	⊗

TABLA 3.9

Las columnas de la tabla de términos primos de la tabla 3.9 corresponden a los productos canónicos que es necesario realizar y que son únicamente los que toman el valor uno. De acuerdo con lo explicado en el apartado 3.2.2, se observa que los términos 0-1-2-3, 0-2-4-6 y 2-3-10-11 son esenciales. Sin necesidad de hacer otra tabla o utilizar la ecuación algebraica correspondiente se observa que el único producto canónico que queda por realizar es el 8 y que para ello se pueden utilizar los términos 0-2-8-10 o 0-4-8-12. Por tanto existen las soluciones mínimas.

$$f = (0\text{-}1\text{-}2\text{-}3) + (0\text{-}2\text{-}4\text{-}6) + (2\text{-}3\text{-}10\text{-}11) + (0\text{-}2\text{-}8\text{-}10)$$
$$f = (0\text{-}1\text{-}2\text{-}3) + (0\text{-}2\text{-}4\text{-}6) + (2\text{-}3\text{-}10\text{-}11) + (0\text{-}4\text{-}8\text{-}12)$$

Observando las tablas de la figura 3.11 se comprueba que estas dos soluciones coinciden con las obtenidas mediante el método tabular de Karnaugh.

Los ejemplos se han realizado utilizando la suma de productos canónicos, pero los métodos que se han empleado son también aplicables al producto de sumas canónicas.

En la tabla 3.7 se representa la tabla de verdad de una función incompleta y en ella se indican todas las combinaciones posibles de las variables de entrada. Cuando una función f no está definida para un número elevado de las combinaciones de entrada no hay que representar la tabla de verdad completa. No es necesario indicar las combinaciones de entrada para las que la función no está definida. Y además cuando la función f toma un cierto valor para una combinación de determinadas variables de entrada independientemente del estado de las demás, esta situación se puede representar mediante una sola línea en la tabla de verdad.

La tabla 3.10 es un ejemplo de lo que se acaba de exponer. La función f adopta el estado cero cuando $a = b = c = d = 0$ independientemente del estado de e y g. Esto se indica mediante la primera línea de la tabla 3.10 en la que se coloca el símbolo de indiferencia X en las columnas de las variables e y g. En esta tabla no se indican todos los términos correspondientes a las combinaciones de las variables

g	e	d	c	b	a	f
X	X	0	0	0	0	0
X	0	0	0	0	1	1
X	1	0	0	1	0	0
X	X	0	0	1	1	0
X	X	0	1	0	0	1
X	X	0	1	0	1	1
X	1	0	1	1	0	1
1	X	0	1	1	1	0
0	X	1	0	0	0	1
X	X	1	0	0	1	0

TABLA 3.10

a, b, c y d superiores a 1001 porque se supone que son indiferentes. De esta forma se reduce considerablemente el tamaño de la tabla de verdad.

El método simplificado de Karnaugh descrito en el apartado 3.2.1 es aplicable también a las funciones incompletas. Para ello se deben asignar a las filas y columnas de la tabla de Karnaugh aquellas variables que están definidas en la mayoría de las combinaciones de la tabla. Por ejemplo, si se trata de obtener la expresión más sencilla de la función cuya tabla de verdad se representa en la tabla 3.10 mediante una tabla de Karnaugh de 4 variables, se deben asignar a las filas y columnas las variables a, b, c y d que son las que están definidas en todas las líneas de la tabla.

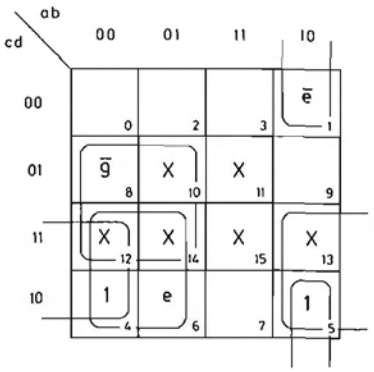


FIGURA 3.12.—**Tabla de Karnaugh única de la función representada en la tabla 3.10.**

En la figura 3.12 se representa la tabla de Karnaugh de suma de productos de cuatro variables. A cada cuadrado se le asignan los números 0 a 15 correspondientes a los dieciséis primeros productos canónicos de la función f. En los cuadrados 10 a 15 se coloca el símbolo X porque la función f no está definida para las combinaciones de entrada correspondientes. En los cuadrados 4 y 5 se coloca un uno porque la función f adopta el estado uno para las combinaciones 0100 y 0101 de las variables d, c, b y a independientemente del estado de e y g. En el cuadrado 1 se coloca la variable $\bar{e}$ porque para la combinación 0001 de las variables d, c, b y a la función adopta el estado uno cuando $e = 0$. De igual forma en el cuadrado 8 se coloca la variable $\bar{g}$ y en el cuadrado 6 la variable e.

A continuación se realizan los grupos necesarios para cubrir todos los unos de la manera más sencilla posible. Se obtiene así el grupo $4 - 5 - 12 - 13$ que corresponde al producto $\bar{b}c$. Seguidamente se realizan las variables $\bar{e}, \bar{g}$ y e combinándolas con las X y los unos.

Se obtienen así los grupos:

$$\begin{aligned} 8 - 10 - 12 - 14 &\equiv \bar{a}d\bar{g} \\ 4 - 6 - 12 - 14 &\equiv \bar{a}ce \\ 1 - 5 &\equiv a\bar{b}\bar{d}\bar{e} \end{aligned}$$

Por lo tanto la expresión mínima de sumas de productos de la función f resulta:

$$f = \overline{b}c + \overline{a}d\overline{g} + \overline{a}ce + a\,\overline{b}\,\overline{d}\,\overline{e}$$

3.4 MULTIFUNCIONES: DEFINICION Y APLICACION DE LOS METODOS DE SIMPLIFICACION

En apartados anteriores se ha estudiado la simplificación de los circuitos combinacionales equivalentes a una sola función lógica y realizado diversos ejemplos. Con frecuencia, los circuitos combinacionales poseen más de una salida, es decir están constituidos por varias funciones lógicas que dependen de las mismas variables de entrada y han de ser realizadas simultáneamente. Estos circuitos combinacionales son por tanto equivalentes a una multifunción.

La minimización de una multifunción se puede realizar tratando cada una de las funciones independientemente por los métodos indicados en los apartados anteriores, pero con ello no se tiene la seguridad de obtener el circuito más sencillo y por lo tanto más económico. Puede ser conveniente complicar la expresión lógica de las funciones independientes a fin de que tengan productos o sumas comunes entre sí y de que el conjunto implique un menor número de elementos en su realización.

Las multifunciones pueden ser totalmente definidas o no totalmente definidas. El tratamiento de ambos casos se diferencia en lo que ya se ha indicado en el apartado 3.3 para las funciones sencillas y, por tanto, no es preciso volver a analizarlo.

También pueden simplificarse las multifunciones por el método tabular de Karnaugh o el método numérico. Tanto en uno como en otro es necesario tener en cuenta los términos comunes a más de una función.

Por el método de Karnaugh se han de realizar las siguientes etapas en la minimización:

a) Realización de tablas de todas las funciones y sus productos lógicos.

b) Todos aquellos términos canónicos que sólo toman el valor uno para una función se realizan de la forma más sencilla posible en la tabla correspondiente a dicha función.

c) Todos aquellos términos que son comunes a dos funciones, es decir que forman parte del producto lógico de ambas, que no son comunes a tres funciones y que no han sido realizados en el apartado anterior, se realizan de la forma más sencilla posible en la tabla correspondiente.

d) El proceso se repite para los términos comunes a tres funciones que todavía no han sido realizados y así sucesivamente.

De lo dicho se deduce que el método tabular de Karnaugh no resulta aplicable en la práctica a multifunciones que incluyan más de tres funciones simples.

Un ejemplo aclarará todo lo expuesto. En la figura 3.13 se representan las tablas de Karnaugh de la multifunción cuya tabla de verdad se representa en la tabla 3.11.

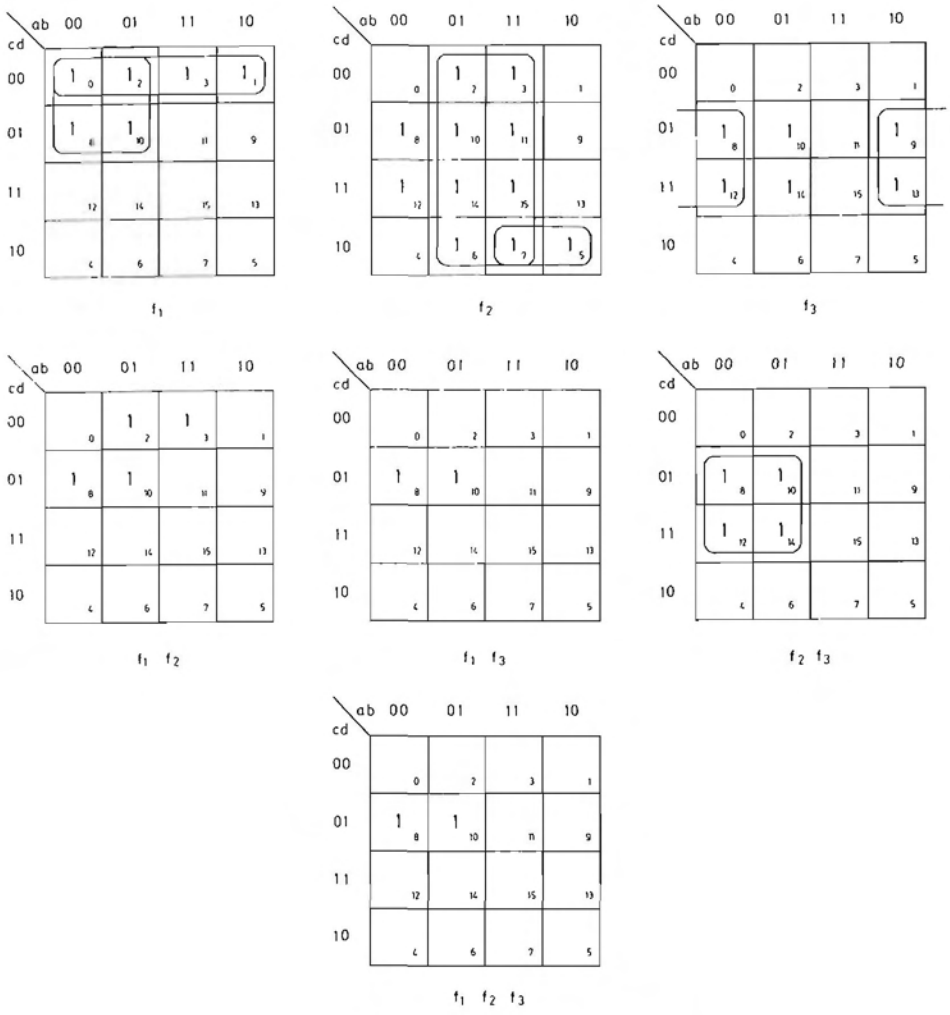


FIGURA 3.13.—**Tablas de Karnaugh de las funciones de la tabla 3.11.**

Tabla de verdad de la multifunción:

$$f_1(abcd) = \sum_4 (0, 1, 2, 3, 8, 10)$$

$$f_2(abcd) = \sum_4 (2, 3, 5, 6, 7, 8, 10, 11, 12, 14, 15)$$

$$f_3(abcd) = \sum_4 (8, 9, 10, 12, 13, 14)$$

Analizando la tabla de cada función se observa, por ejemplo, que los productos canónicos 0 y 1 sólo toman parte de la función f_1 y se realizan de la forma más sencilla posible mediante el término 0-1-2-3. Igualmente los productos canónicos

d	c	b	a	f_1	f_2	f_3
0	0	0	0	1	0	0
0	0	0	1	1	0	0
0	0	1	0	1	1	0
0	0	1	1	1	1	0
0	1	0	0	0	0	0
0	1	0	1	0	1	0
0	1	1	0	0	1	0
0	1	1	1	0	1	0
1	0	0	0	1	1	1
1	0	0	1	0	0	1
1	0	1	0	1	1	1
1	0	1	1	0	1	0
1	1	0	0	0	1	1
1	1	0	1	0	0	1
1	1	1	0	0	1	1
1	1	1	1	0	1	0

TABLA 3.11

6, 7, 11 y 15 sólo forman parte, al igual que el 5, de la función f_2 y se realizan respectivamente mediante los términos 2-3-6-7-10-11-14-15 y 5-7. De igual manera, los términos 9 y 13 sólo pertenecen a f_3 y se realizan mediante el término 8-9-12-13.

Una vez realizado lo anterior se observan las tablas correspondientes al producto de dos funciones. Los productos canónicos 12 y 14 que todavía no han sido realizados y pertenecen a $f_2 f_3$ no están en $f_1 f_2 f_3$ y por tanto se realizan mediante el término 8-10-12-14.

Sólo quedan por realizar, por tanto, los productos canónicos 8 y 10 de f_1 para lo cual se utilizará el producto 8-10 que está en $f_1 f_2 f_3$. Pero como este término sólo formará parte de f_1 porque ya ha sido realizado en f_2 y f_3 mediante 8-10-12-14, se puede simplificar uniéndolo al 0-2 para formar el término 0-2-8-10.

Por tanto, las funciones mínimas resultantes son:

$$f_1 = \bar{c}\bar{d} + \bar{a}\bar{c}$$
$$f_2 = b + ac\bar{d} + \bar{a}d$$
$$f_3 = \bar{b}d + \bar{a}d$$

La minimización de las multifunciones por el método numérico presenta la enorme ventaja de ser aplicable en cualquier caso independientemente de su complejidad, en especial mediante un programa de computador.

En la tabla de términos (productos o sumas) canónicos ordenados por el número de unos se indica, a la derecha del término correspondiente, a qué funciones pertenece. Mediante un ejemplo se aclaran mejor los conceptos. Se utiliza el mismo al que se le aplicó el método tabular de Karnaugh y cuya tabla de verdad

Tabla 1

N.º de unos			
0	0	X	f_1
	1	X	f_1
1	2	X	f_1f_2
	8	X	$f_1f_2f_3$
	3	X	f_1f_2
	5	X	f_2
2	6	X	f_2
	9	X	f_3
	10	X	$f_1f_2f_3$
	12	X	f_2f_3
	7	X	f_2
	11	X	f_2
3	13	X	f_3
	14	X	f_2f_3
4	15	X	f_2

Tabla 2

			Dif.
0-1	X	f_1	1
0-2	X	f_1	2
0-8	X	f_1	8
1-3	X	f_1	2
~~1-5~~		-	
~~1-9~~		-	
2-3		f_1f_2	1
2-6	X	f_2	4
2-10		f_1f_2	8
8-9	X	f_3	1
8-10		$f_1f_2f_3$	2
8-12	X	f_2f_3	4
3-7	X	f_2	4
3-11	X	f_2	8
5-7		f_2	2
~~5-13~~		-	
6-7	X	f_2	1
6-14	X	f_2	8
~~9-11~~		-	
9-13	X	f_3	4
10-11	X	f_2	1
10-14	X	f_2f_3	4
12-13	X	f_3	1
12-14	X	f_2f_3	2
7-15	X	f_2	8
11-15	X	f_2	4
~~13-15~~		-	
14-15	X	f_2	1

Tabla 3

		Dif.
0-1-2-3	f_1	1,2
~~0-1-8-9~~	-	
0-2-1-3	f_1	2,1
0-2-8-10	f_1	2,8
0-8-2-10	f_1	8,2
~~1-3-5-7~~	-	
2-3-6-7	f_2	1,4
2-3-10-11	f_2	1,8
2-6-3-7	f_2	4,1
2-6-10-14	f_2	4,8
2-10-3-11	f_2	8,1
2-10-6-14	f_2	8,4
~~8-9-10-11~~	-	
8-9-12-13	f_3	1,4
8-10-12-14	f_2f_3	2,4
8-12-10-14	f_2f_3	4,2
8-12-9-13	f_3	4,1
3-7-11-15	f_2	4,8
3-11-7-15	f_2	8,4
6-7-14-15	f_2	1,8
6-14-7-15	f_2	8,1
~~9-13-11-15~~	-	
10-11-14-15	f_2	1,4
10-14-11-15	f_2	4,1
~~12-13-14-15~~	-	

Tabla 3 *reducida*

			Dif.
0-1-2-3		f_1	1,2
0-2-8-10		f_1	2,8
2-3-6-7	X	f_2	1,4
2-3-10-11	X	f_2	1,8
2-6-10-14	X	f_2	4,8
8-9-12-13		f_3	1,4
8-10-12-14		f_2f_3	2,4
3-7-11-15	X	f_2	4,8
6-7-14-15	X	f_2	1,8
10-11-14-15	X	f_2	1,4

Tabla 4

		Dif.
2-3-6-7-10-11-14-15	f_2	1,4,8
2-3-10-11-6-7-14-15	f_2	1,8,4
2-6-10-14-3-7-11-15	f_2	4,8,1
~~8-9-12-13-10-11-14-15~~	-	

Tabla 4 *reducida*

		Dif.
2-3-6-7-10-11-14-15	f_2	1,4,8

TABLA 3.12

se representa en la tabla 3.11. En la tabla 3.12 se indican todas las tablas obtenidas. Las tablas 2 y 3 se realizan por el método ya explicado en el apartado 3.2.2 indicando las funciones a las que pertenece cada término. Por ejemplo, el término 12-13 pertenece a f_3 porque el 12 pertenece a f_2 y f_3 y el 13 sólo a f_3. El término 1-5, por

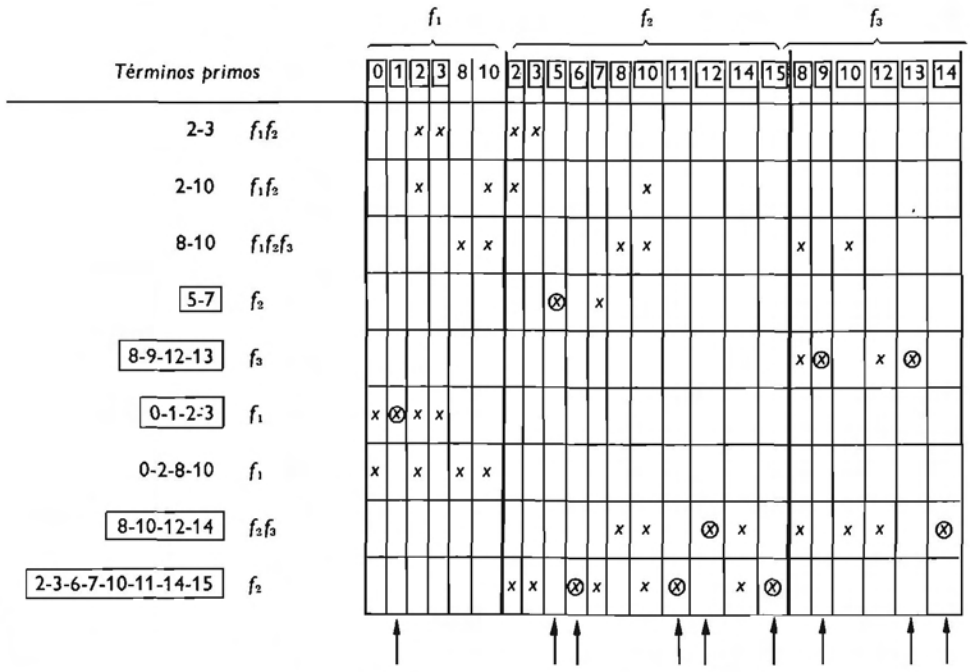


TABLA 3.13.—**Tabla de términos primos.**

ejemplo, se suprime porque el 1 pertenece a f_1 y el 5 a f_2. Los términos suprimidos se cubren con un aspa. Para que un término de cualquiera de ellas sea primo y por tanto utilizable en la realización de la función, es necesario que no exista ninguno que lo contenga y que pertenezca a las mismas funciones que él. Por ejemplo, el término 8-10 es primo porque pertenece a f_1, f_2 y f_3, mientras el 0-2-8-10 sólo pertenece a f_1 y el 8-10-12-14 sólo a f_2 y f_3.

La tabla de los términos primos se presenta en la tabla 3.13 cuyas columnas corresponden a todos los términos canónicos de las tres funciones. En esta tabla se observa que los términos 1 de f_1, 5, 6, 11, 12, 15 de f_2 y 9, 13, 14 de f_3 son realizados por un solo término primo, el cual es pues esencial. El lector puede comprobar que los términos primos esenciales son 5-7(f_2), 8-9-12-13(f_3) 0-1-2-3(f_1), 8-10-12-14(f_2f_3) y 2-3-6-7-10-11-14-15(f_2) y que los únicos términos canónicos no realizados por ellos son el 8 y el 10 de f_1. Estos dos términos canónicos están cubiertos por los términos primos 2-10, 8-10 y 0-2-8-10, obteniéndose la tabla 3.14 en la cual se observa que la línea *C* cubre a las líneas *A* y *B*. Por tanto, a los términos esenciales hay que añadir el 0-2-8-10 (f_1).

Las expresiones resultantes de las tres funciones son:

$$f_1 = (0\text{-}1\text{-}2\text{-}3) + (0\text{-}2\text{-}8\text{-}10)$$
$$f_2 = (5\text{-}7) + (8\text{-}10\text{-}12\text{-}14) + (2\text{-}3\text{-}6\text{-}7\text{-}10\text{-}11\text{-}14\text{-}15)$$
$$f_3 = (8\text{-}9\text{-}12\text{-}13) + (8\text{-}10\text{-}12\text{-}14)$$

	Términos primos no esenciales		f_1 8	f_1 10
A	2 - 10	$f_1 f_2$	X	
B	8 - 10	$f_1 f_2 f_3$	X	X
C	0 - 2 - 8 - 10	f_1	X	X

TABLA 3.14

cuyo equivalente algebraico es:

$$f_1 = \bar{c}\bar{d} + \bar{a}\bar{c}$$
$$f_2 = b + ac\bar{d} + \bar{a}d$$
$$f_3 = \bar{b}d + \bar{a}d$$

Se observa que estas expresiones coinciden con las obtenidas mediante el método tabular de Karnaugh.

3.5 REALIZACION DE LAS FUNCIONES LOGICAS

3.5.1 Realización con puertas NO-Y (NAND) y NO-O (NOR)

Una vez que se ha obtenido la expresión mínima de una función es necesario realizarla en la práctica mediante elementos físicos. El diseño de puertas lógicas con transistores en un principio y la posterior aparición de los circuitos integrados ha hecho que las puertas NO-Y y NO-O sean las más utilizadas en la realización de las funciones lógicas. En el capítulo 2 se ha demostrado que las funciones NO-Y y NO-O pueden realizar cualquiera de las tres funciones elementales suma, producto, e inversión.

Para realizar con puertas NO-Y (NO-O) la expresión mínima de la función obtenida por el método tabular o el método numérico, se aplicarán las siguientes reglas cuya validez se deduce de los postulados y teoremas estudiados en el capítulo 2.

a) Se aplican a la expresión global de la función dos inversiones con lo cual la misma queda invariable.

b) Si la operación más externa es una suma (producto) lógica, se opera una de las inversiones aplicando el teorema de De Morgan y si es producto (suma) no se operan ninguna de las dos.

c) Si en el interior de la expresión existen sumas (productos) lógicas, se aplican a cada una de ellas dos inversiones y se opera una de ellas para convertirla en el inverso de un producto (suma).

d) Se continúa realizando esta operación hasta que todas las sumas (productos) hayan quedado convertidas en inversos de productos (sumas).

Las reglas para realizar cualquier expresión con puertas NO-O (NOR) son iguales a las de la puerta NO-Y (NAND) sustituyendo la palabra suma por producto, lo cual se ha indicado incluyendo la palabra suma entre paréntesis en las reglas que acabamos de indicar.

Unos cuantos ejemplos aclararán estas reglas. Se utilizarán para ello las expresiones de las funciones simplificadas en apartados anteriores.

Las expresiones obtenidas en el ejemplo 3.2 fueron:

$$f_1 = b\bar{c} + ab + ac\bar{d}$$
$$f_1 = (a + \bar{c})\,(b + c)\,(b + \bar{d})$$

Se realizan primero con NO-Y y después con NO-O aplicando las reglas anteriores:

$$f_1 = b\bar{c} + ab + ac\bar{d} = \overline{\overline{b\bar{c} + ab + ac\bar{d}}} = \overline{\overline{b\bar{c}}\ \ \overline{ab}\ \ \overline{ac\bar{d}}}$$

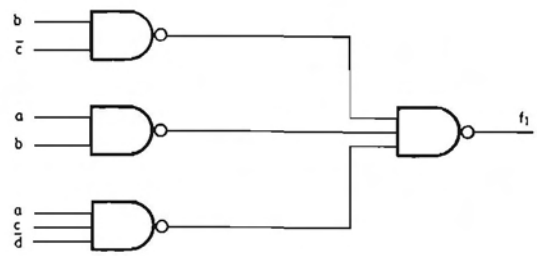


FIGURA 3.14.—**Esquema del circuito con puertas NO-Y (NAND) de la función $f_1 = b\,\bar{c} + a\,b + a\,c\,\bar{d}$, realizado con símbolos no normalizados.**

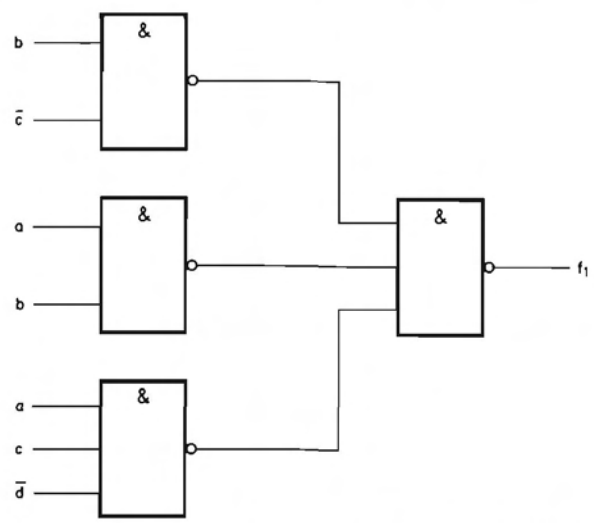


FIGURA 3.15.—**Esquema del circuito con puertas NO-Y (NAND) de la función $f_1 = b\,c + a\,b + a\,c\,\bar{d}$, realizado con símbolos normalizados.**

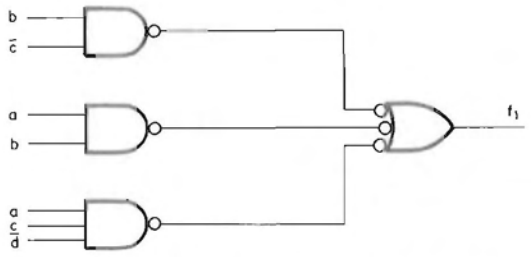


FIGURA 3.16.—**Esquema equivalente al de la figura 3.14 realizado con símbolos no normalizados.**

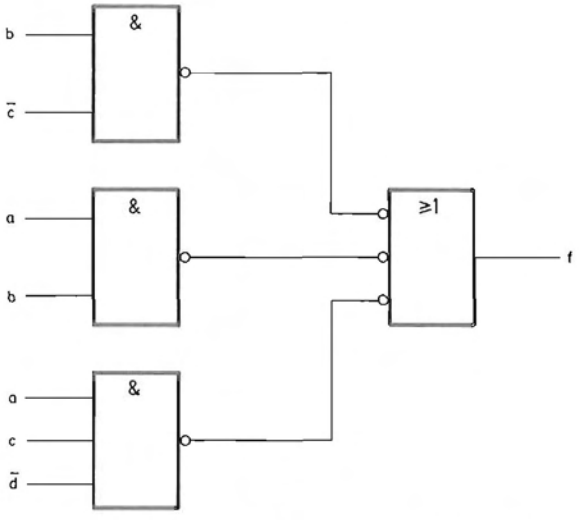


FIGURA 3.17.—**Esquema equivalente al de la figura 3.15 realizado con símbolos normalizados.**

En la figura 3.14 se representa el esquema lógico correspondiente con los símbolos utilizados hasta ahora y en la figura 3.15 con los nuevos normalizados. Antes de continuar el estudio de este capítulo se recomienda al lector que analice los apartados A1.1, A1.2, A1.3, A1.4.1 y A1.4.2 del apéndice 1 dedicado a los símbolos lógicos normalizados.

Se supone en este ejemplo y en lo sucesivo que se dispone tanto de las variables inversas como de las directas.

En el apartado 2.2 se indican los dos símbolos posibles de la función NO-Y (NAND). La combinación de ambos ayuda a interpretar mejor los esquemas. Como regla general se puede decir que las puertas NO-Y mediante las que se realizan los productos lógicos se deben representar con el símbolo de la puerta Y seguida de una inversión. Las puertas NO-Y que realizan las sumas lógicas se deben representar por medio de la puerta O precedida de inversiones en sus entradas.

Se obtienen de esta forma los esquemas de las figuras 3.16 y 3.17 que son equivalentes a los de las figuras 3.14 y 3.15.

Para realizar la expresión de producto de sumas con puertas NO-Y (NAND) es necesario también transformarla:

$$f_1 = (a + \bar{c})\,(b + c)\,(b + \bar{d}) = \overline{(a + \bar{c})\,(b + c)\,(b + \bar{d})} = \overline{\overline{a + \bar{c}}\;\;\overline{b + c}\;\;\overline{b + \bar{d}}} =$$

$$= \overline{\bar{a}c\;\;\bar{b}\bar{c}\;\;\bar{b}d}$$

Esta expresión se representa en las figuras 3.18 y 3.19 y, se observa que son más complejas que la de las figuras 3.14 y 3.15. Combinando ambos tipos de símbolos se obtienen los esquemas de las figuras 3.20 y 3.21 que son equivalentes a los de las figuras 3.18 y 3.19.

De igual forma, ambas expresiones se realizan con puertas NO-O.

$$f_1 = b\bar{c} + ab + ac\bar{d} = \overline{\overline{b\bar{c}} + \overline{ab} + \overline{ac\bar{d}}} =$$

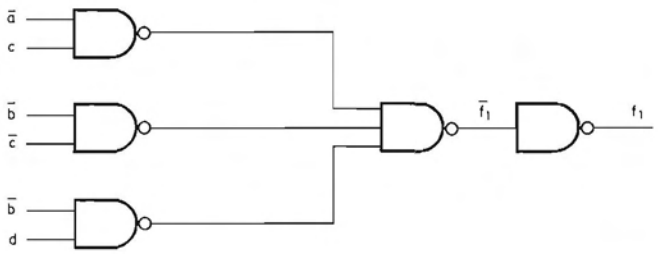


FIGURA 3.18. —**Esquema del circuito con puertas NO-Y (NAND) de la expresión de la función $f_1 = (\bar{a} + c)\,(\bar{b} + \bar{c})\,(\bar{b} + d)$, realizado con símbolos no normalizados.**

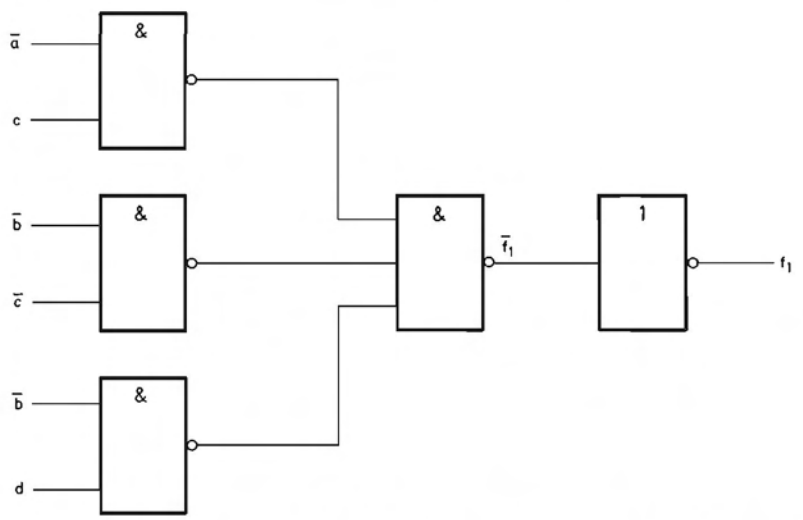


FIGURA 3.19. —**Esquema del circuito con puertas NO-Y (NAND) de la expresión de la función $f_1 = (\bar{a} + c)\,(\bar{b} + \bar{c})\,(\bar{b} + d)$, realizado con símbolos normalizados.**

$$= \overline{\overline{b\bar{c}} + \overline{ab} + \overline{acd}} = \overline{\overline{b} + c + \bar{a} + \bar{b} + \bar{a} + \bar{c} + d}$$

$$f_1 = (a + \bar{c})\,(b + c)\,(b + \bar{d}) = \overline{\overline{(a + \bar{c})\,(b + c)\,(b + \bar{d})}} =$$

$$= \overline{\overline{a + \bar{c}} + \overline{b + c} + \overline{b + \bar{d}}}$$

En las figuras 3.22 y 3.23 se representa la primera expresión y en las figuras 3.24 y 3.25 la segunda.

Al igual que la función NO-Y (NAND), la función NO-O (NOR) se puede representar gráficamente mediante dos símbolos diferentes indicados en las figuras 2.4 y 2.5, y la combinación de ambos también ayuda a interpretar mejor los esquemas.

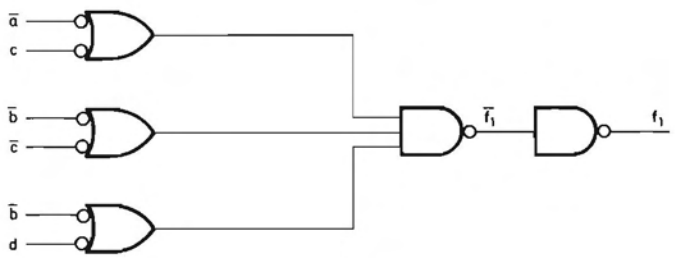


FIGURA 3.20.—**Esquema equivalente al de la figura 3.18 realizado con símbolos no normalizados.**

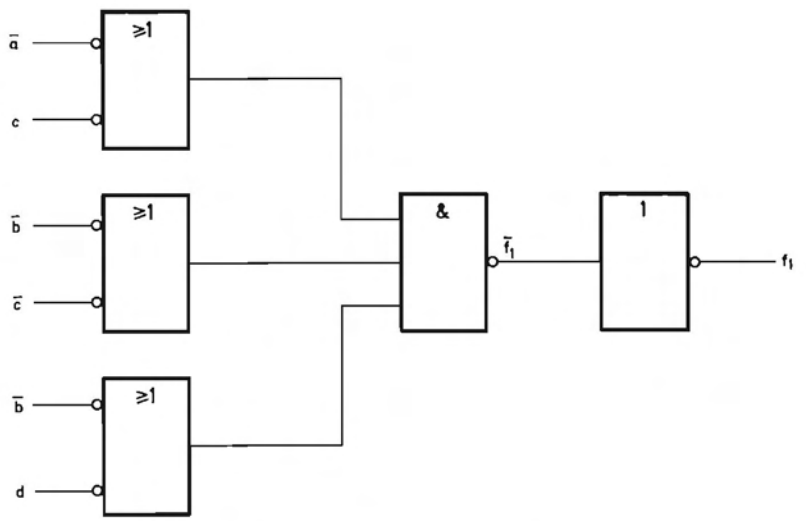


FIGURA 3.21.—**Esquema equivalente al de la figura 3.19 realizado con símbolos normalizados.**

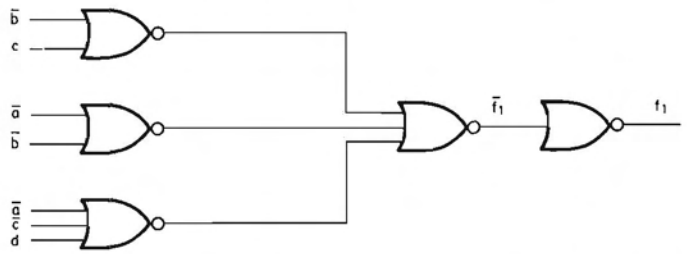


FIGURA 3.22. —**Esquema con puertas NO-O (NOR) de la expresión de la función** $f_1 = \overline{b}\,c + \overline{a}\,\overline{b} + \overline{a}\,\overline{c}\,d$, **realizado con símbolos no normalizados.**

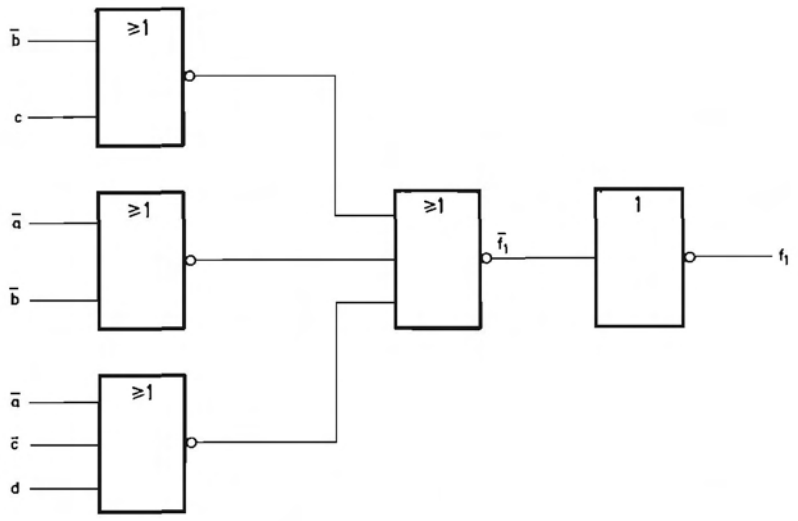


FIGURA 3.23. —**Esquema con puertas NO-O (NOR) de la expresión de la función** $f_1 = \overline{b}\,c + \overline{a}\,\overline{b} + \overline{a}\,\overline{c}\,d$, **realizado con símbolos normalizados.**

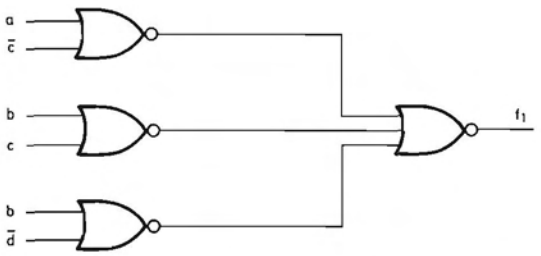


FIGURA 3.24. —**Esquema con puertas NO-O (NOR) de la expresión de la función** $f_1 = (a + \overline{c})\,(b + c)\,(b + \overline{d})$, **realizado con símbolos no normalizados.**

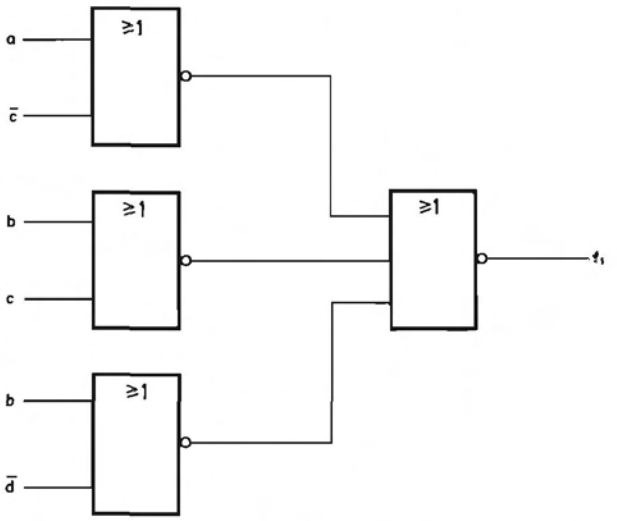


FIGURA 3.25.—**Esquema con puertas NO-O (NOR) de la expresión de la función $f_1 = (a + \bar{c})\ (b + c)\ (b + \bar{d})$, realizado con símbolos normalizados.**

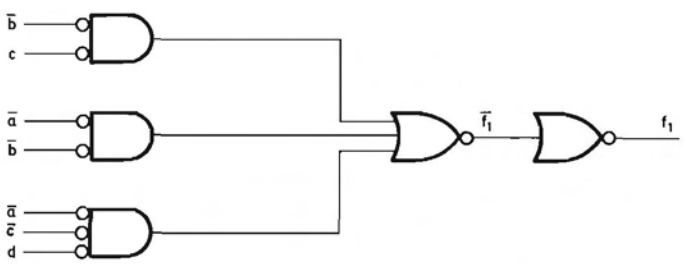


FIGURA 3.26.—**Esquema equivalente al de la figura 3.22 realizado con símbolos no normalizados.**

Es preferible representar con el símbolo de la puerta Y precedida de inversiones aquellas puertas que realizan los productos y mediante la puerta O seguida de una inversión las que realizan las sumas. Se obtienen de esta forma las figuras 3.26 y 3.27 equivalentes a las 3.22 y 3.23 respectivamente y las 3.28 y 3.29 equivalentes a las 3.24 y 3.25.

No teniendo en cuenta el inversor final cuando existe, en las figuras 3.14 a 3.29 se observa que para que el cambio de cualquier entrada actúe sobre una salida es necesario que conmuten como máximo dos puertas. Cada puerta representa un retraso a la conmutación y se denomina nivel de un circuito el máximo número de puertas que ha de atravesar la información del cambio de una cualquiera de las entradas para actuar sobre la salida. Se observa que, realizando las expresiones mínimas que se han obtenido por el método tabular o el numérico, se obtienen circuitos de nivel 2.

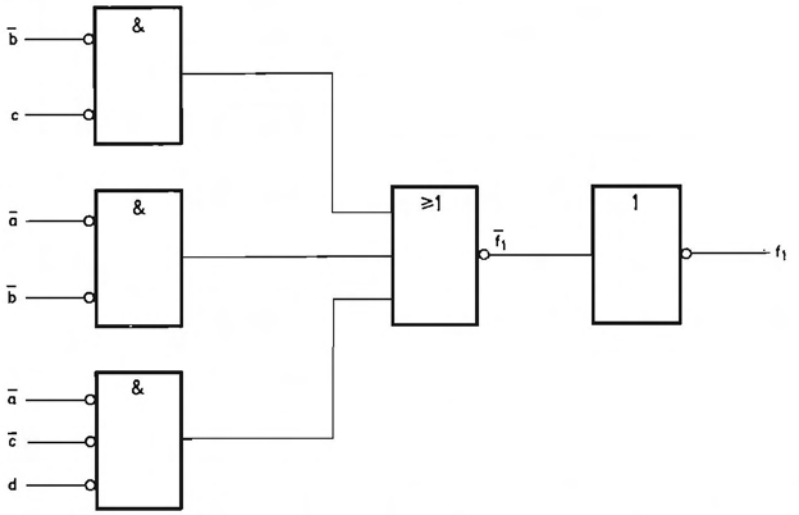


FIGURA 3.27.—**Esquema equivalente al de la figura 3.23 realizado con símbolos normalizados.**

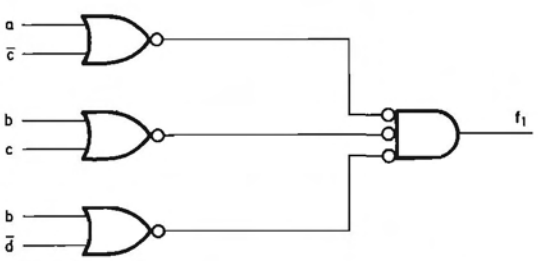


FIGURA 3.28.—**Esquema equivalente al de la figura 3.24 realizado con símbolos no normalizados.**

A dichas expresiones se les puede aplicar las propiedades distributivas de la suma con respecto al producto y viceversa. De esta forma se puede lograr una expresión más sencilla que se realizará con un menor número de puertas o el mismo número de puertas con menos entradas, pero que dará, en general, lugar a un circuito cuyo nivel será superior a 2. Naturalmente, existirá más de una forma de aplicar dichas propiedades, y por tanto, se obtendrá más de una expresión. Se utilizará el ejemplo analizado en párrafos anteriores para aclarar lo expuesto.

En la expresión de f_1 en forma de suma de productos:

$$f_1 = b\bar{c} + ab + ac\bar{d}$$

se observa que b es común a los dos primeros y a lo es a los dos segundos. Sacando factor común «b» resulta la expresión:

$$f_1 = b(a + \bar{c}) + ac\bar{d}$$

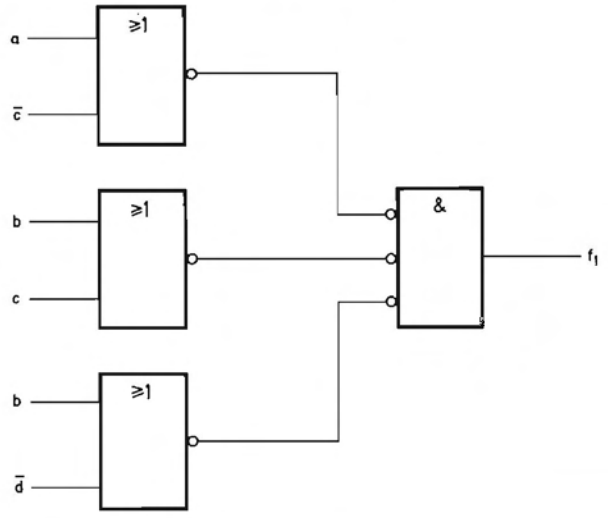


FIGURA 3.29.—**Esquema equivalente al de la figura 3.25 realizado con símbolos normalizados.**

y sacando factor común «a» resulta:

$$f_1 = b\bar{c} + a(b + c\bar{d})$$

Aplicando las reglas antes indicadas se pueden convertir estas expresiones para ser realizadas con puertas NO-Y y NO-O. Con puertas NO-Y resulta:

$$f_1 = b(a + \bar{c}) + ac\bar{d} = \overline{\overline{b(a + \bar{c}) + ac\bar{d}}} = \overline{\overline{b(a + \bar{c})}\ \overline{ac\bar{d}}} = \overline{b\ \overline{\overline{a + \bar{c}}}\ \overline{ac\bar{d}}} =$$

$$= \overline{\overline{b\bar{a}c}\ \overline{ac\bar{d}}}$$

$$f_1 = b\bar{c} + a(b + c\bar{d}) = \overline{\overline{b\bar{c} + a(b + c\bar{d})}} = \overline{\overline{b\bar{c}}\ \overline{a(b + c\bar{d})}} = \overline{\overline{b\bar{c}}\ \overline{a\ \overline{b + c\bar{d}}}} =$$

$$= \overline{\overline{b\bar{c}}\ \overline{a\ \overline{\bar{b}\ \overline{c\bar{d}}}}}$$

En lo sucesivo se utilizan exclusivamente en los esquemas los nuevos símbolos normalizados por la Comisión Electrotécnica Internacional.

En las figuras 3.30 y 3.31 se representan los esquemas correspondientes a las dos últimas expresiones de f_1.

De igual forma, representando las puertas NO-Y que realizan las sumas de las expresiones anteriores mediante el símbolo de la puerta O precedido de inversiones, se obtienen los esquemas de las figuras 3.32 y 3.33 equivalentes respectivamente a los de las figuras 3.30 y 3.31.

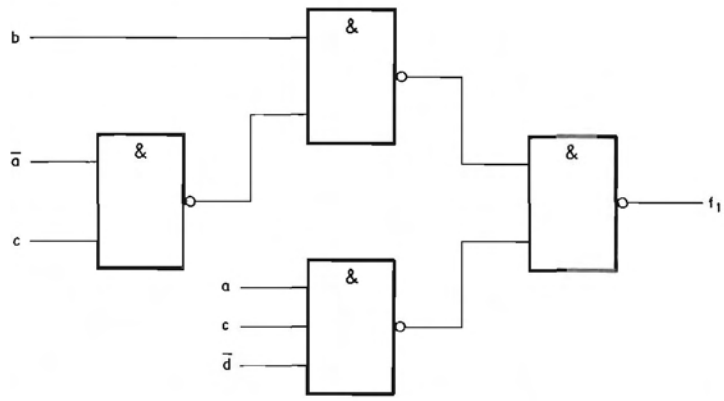


FIGURA 3.30. —**Esquema con puertas NO-Y (NAND) de la expresión de la función** $f_1 = b\ (a + \bar{c}) + a\ c\ \bar{d}$.

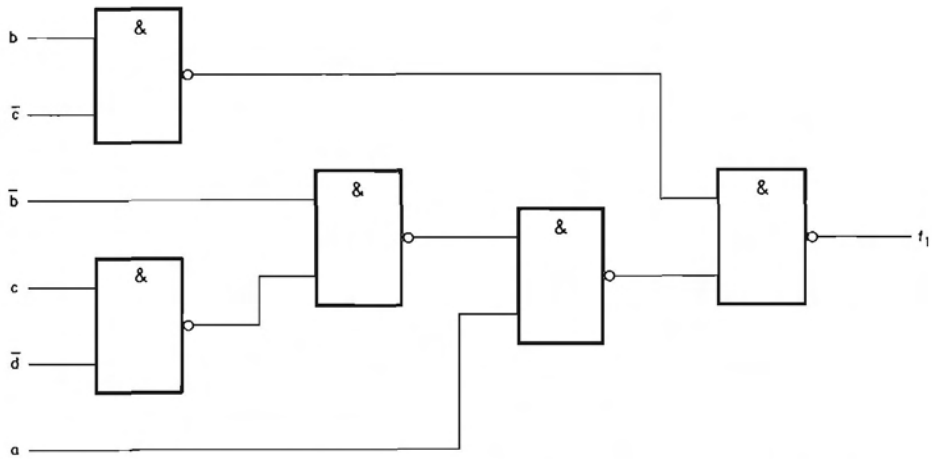


FIGURA 3.31. —**Esquema con puertas NO-Y (NAND) de la expresión de la función** $f_1 = b\ \bar{c} + a\ (b + c\ \bar{d})$.

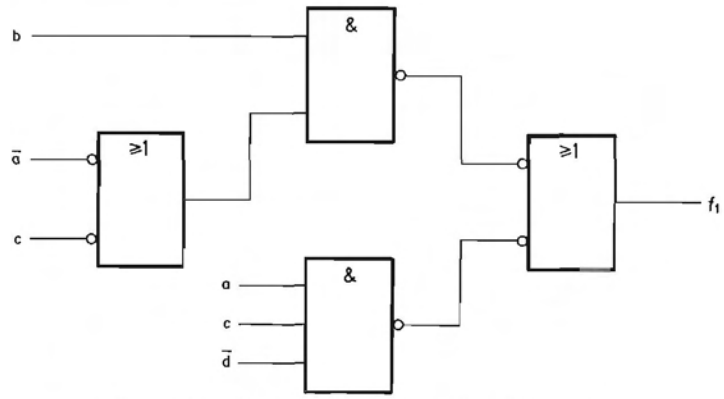


FIGURA 3.32.—**Esquema equivalente al de la figura 3.30.**

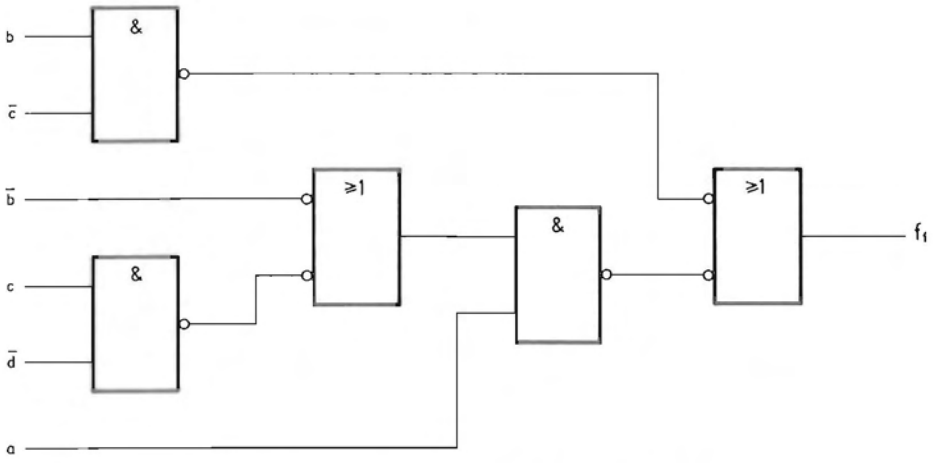


FIGURA 3.33.—**Esquema equivalente al de la figura 3.31.**

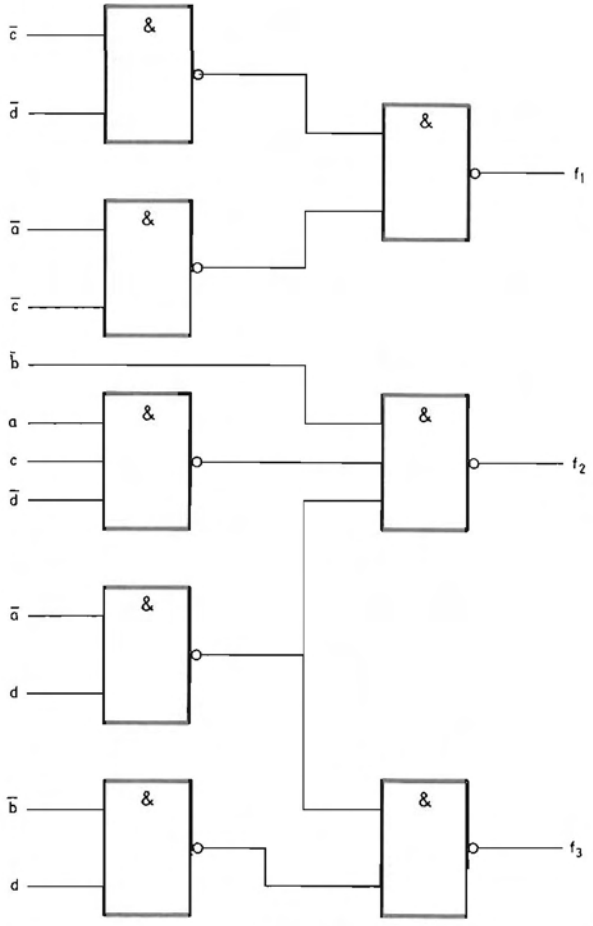


FIGURA 3.34. —**Esquema con puertas NO-Y (NAND) de la multifunción** $f_1 = \bar{c}\,\bar{d} + \bar{a}\,\bar{c}$, $f_2 = b + a\,c\,\bar{d} + \bar{a}\,d$ **y** $f_3 = \bar{b}\,d + \bar{a}\,d$.

Comparando la figura 3.15 con la 3.30 se observa que en esta última se utiliza el mismo número de puertas que en aquélla, con una entrada menos, pero que su nivel es 3 en lugar de 2.

Las multifunciones se realizan de la misma forma con puertas NO-Y y NO-O pero realizando una sola vez los términos comunes. Como ejemplo se realiza con puertas NO-Y la multifunción diseñada en el apartado 3.4.

Aplicando las reglas antes indicadas resulta:

$$f_1 = \bar{c}\bar{d} + \bar{a}\bar{c} = \overline{\overline{\bar{c}\bar{d} + \bar{a}\bar{c}}} = \overline{\overline{\bar{c}\bar{d}}\ \overline{\bar{a}\bar{c}}}$$

$$f_2 = b + ac\bar{d} + \bar{a}d = \overline{\overline{b + ac\bar{d} + \bar{a}d}} = \overline{\bar{b}\ \overline{ac\bar{d}}\ \overline{\bar{a}d}}$$

$$f_3 = \bar{b}d + \bar{a}d = \overline{\overline{\bar{b}d + \bar{a}d}} = \overline{\overline{\bar{b}d}\ \overline{\bar{a}d}}$$

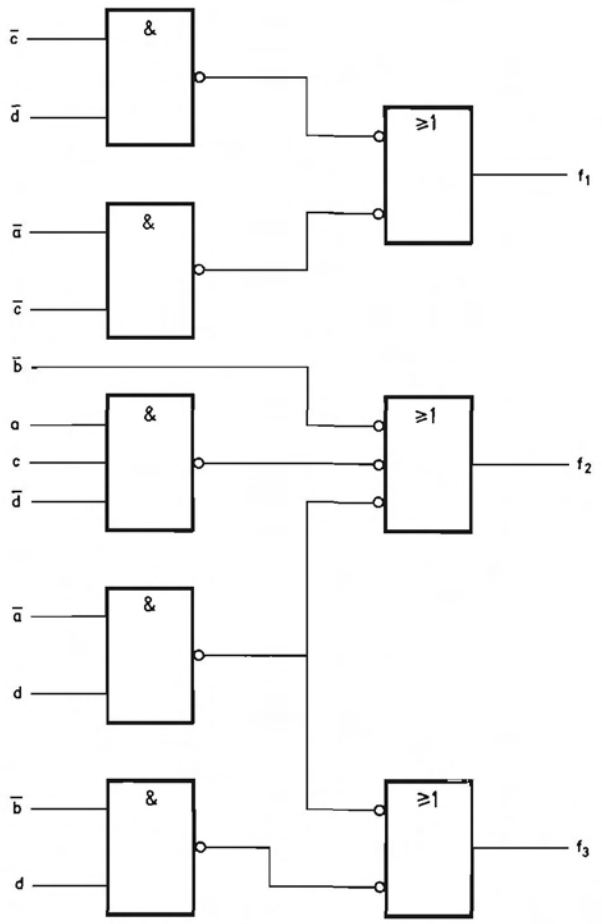


FIGURA 3.35.—**Esquema equivalente al de la figura 3.34.**

En la figura 3.34 se representa el circuito correspondiente. Aplicando el criterio de representación explicado en párrafos anteriores se obtiene el esquema de la figura 3.35, equivalente al de la 3.34.

3.5.2 Realización de funciones mediante el montaje «Y por conexión»

En las tecnologías en las que el transistor de salida posee como carga una resistencia [tecnología DTL y algunos circuitos de la TTL (ver apartado 5.4.4.3.2 del capítulo 5)] se pueden unir directamente las salidas de las puertas NO-Y (NAND) obteniéndose la función denominada «Y por conexión» (en inglés «Wired And»). La figura 3.36 representa este montaje con los símbolos antiguos y la 3.37 con los nuevos nor-

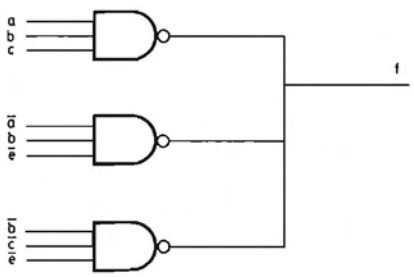


FIGURA 3.36.—**Ejemplo del montaje «Y por conexión» con símbolos no normalizados.**

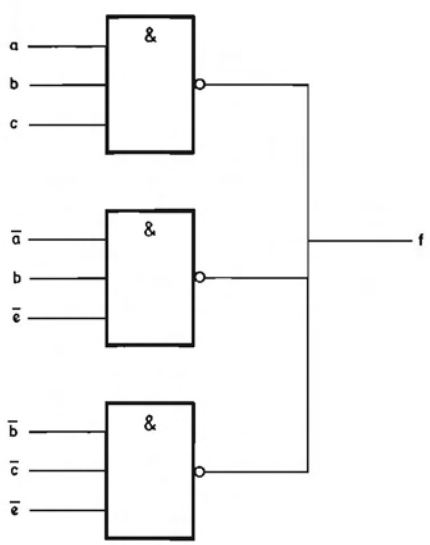


FIGURA 3.37.—**Ejemplo del montaje «Y por conexión» con símbolos normalizados.**

malizados. La función f toma el valor cero cuando todas las entradas de cualquiera de las puertas NO-Y se encuentran en estado uno simultáneamente. Por lo tanto:

$$f = \overline{abc}\ \overline{\bar{a}b\bar{e}}\ \overline{\bar{b}ce}$$

Una de las ventajas del nuevo sistema de representación gráfica de las funciones lógicas es la adopción de símbolos para las distintas configuraciones de las etapas de salida descritas en el apartado A1.3 del apéndice 1. De acuerdo con la nueva simbología el esquema de la figura 3.37 debe ser sustituido por el de la figura 3.38.

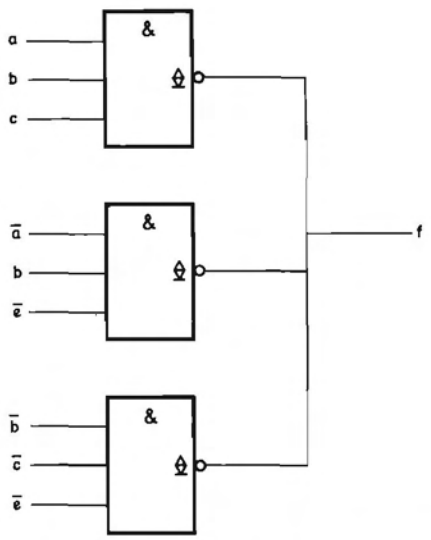


FIGURA 3.38. —**Ejemplo del montaje «Y por conexión» con símbolos normalizados con indicativo de salida con carga pasiva en el colector.**

El montaje «Y por conexión» se puede realizar también mediante puertas de colector abierto, que reciben esta denominación porque el transistor de salida carece de carga (ver la figura 5.36 del capítulo 5). El esquema, en este caso, del circuito que realiza la función f se representa en la figura 3.39. Cualquiera de las expresiones mínimas de producto de sumas o suma de productos puede ser realizada mediante el montaje «Y por conexión». En algunos casos, la realización de las funciones se simplifica mucho mediante la utilización de este montaje.

El progreso de las técnicas de integración ha disminuido el interés de la utilización del montaje «Y por conexión» en la generación de funciones por obtenerse tiempos de propagación y consumos superiores a los de las puertas con carga activa (ver apartado 5.4.4.3.2 del capítulo 5). No obstante, este montaje se utiliza en circuitos de control de entrada-salida de los microprocesadores y por ello para el lector es interesante su comprensión.

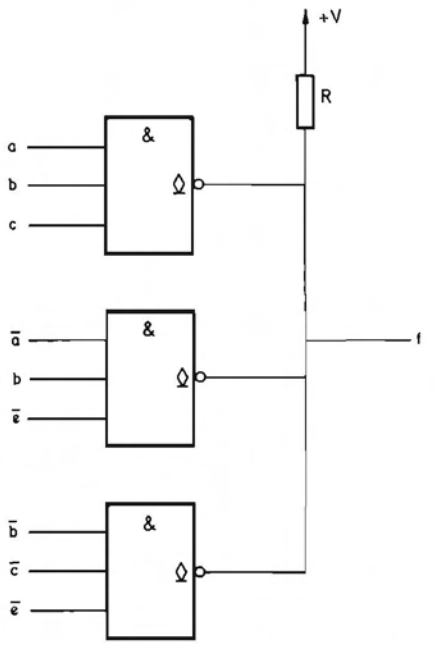


FIGURA 3.39.—**Ejemplo del montaje «Y por conexión» realizado con puertas de colector abierto.**

3.5.3 Realización de las funciones lógicas con puertas O-exclusiva

La utilización de puertas O-exclusiva de dos entradas disponibles en circuito integrado de pequeña escala de integración, en combinación con puertas NO-Y y NO-O, permite reducir el número total de puertas necesario para la realización de ciertas funciones lógicas. Este hecho tenía gran interés cuando la microelectrónica no había alcanzado el nivel de gran escala de integración (ver apartado 5.4.4.2.1.2 del capítulo 5). En esa época (1965-1975) se desarrolló un método tabular de síntesis de funciones lógicas mediante la función O-exclusiva que constituyó una importante herramienta de trabajo en el diseño de sistemas digitales. Al alcanzarse los niveles de gran y muy gran escala de integración y no ser necesaria la simplificación máxima de los circuitos combinacionales, este método ha perdido una gran parte de su utilidad y por ello no se incluye en esta edición. Al lector interesado en conocerlo se le sugiere que se ponga en contacto con el autor que le facilitará un ejemplar del mismo.

3.6 FENOMENOS ALEATORIOS EN LOS SISTEMAS COMBINACIONALES

En el estudio que se ha realizado en los apartados anteriores no se ha tenido en cuenta el retardo inherente a la propagación de la señal a través de las puertas lógicas. Este retardo se presenta en todas las tecnologías y su magnitud depende en gran medida de aquélla.

A continuación se estudian los tipos más importantes de fenómenos aleatorios en los circuitos combinacionales y la forma de diseñar éstos para evitar su aparición.

3.6.1 Fenómenos aleatorios estáticos

Se presenta un fenómeno aleatorio estático, si para dos estados de las variables de entrada adyacentes en los que la salida debe ser constante (0 o 1), existe un régimen transitorio durante el cual la salida cambia eventualmente de estado. Existen por lo tanto un fenómeno aleatorio de tipo cero y un fenómeno aleatorio de tipo uno.

Los fenómenos aleatorios de tipo cero se producen cuando, al cambiar de estado una variable, la función debe de permanecer en estado uno, pero pasa transitoriamente por cero. Veamos cómo puede producirse un fenómeno de este tipo. Sea una función f que depende de una serie de variables a, b, c, d,... y designemos por X al conjunto de todas las variables excepto la a. En el capítulo 2 se demuestra que f puede expresarse algebraicamente de la forma siguiente:

$$f = f(a, b, c...) = f(a, X) = a\,f(1, X) + \bar{a}\,f(0, X)$$

Para la combinación de estados X_0 de todas las variables excepto la a en que se verifica:

$$f(1, X_0) = f(0, X_0) = 1$$

resulta

$$f = a + \bar{a}$$

En régimen estático, el estado de la función f es uno lógico, pero debido al retardo en la conmutación, al producirse un cambio de estado de a puede suceder que a y $\bar{a}$ se encuentren en estado cero simultáneamente y que durante un cierto tiempo f tome el valor lógico cero.

En la figura 3.40 se representa esta situación cuando $\bar{a}$ sufre un retraso en la conmutación con respecto a a.

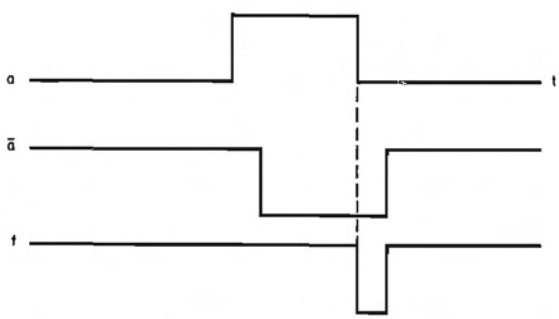


FIGURA 3.40. —**Representación gráfica de un fenómeno aleatorio estático de tipo cero.**

La forma de eliminar este fenómeno es añadir a la expresión de f un término que tome el valor lógico uno para la combinación X_0 que es precisamente el producto lógico $f(1, X)\, f(0, X)$. Aplicando los postulados del álgebra de Boole se demuestra que la expresión de la función f no varía al añadir dicho término, que cubre la adyacencia existente entre los términos $af(1, X)$ y $\bar{a}f(0, X)$. Un ejemplo sencillo lo aclarará.

Sea la función de tres variables

$$f = \sum_3 (3, 4, 6, 7)$$

cuya tabla de Karnaugh se representa en la figura 3.41.

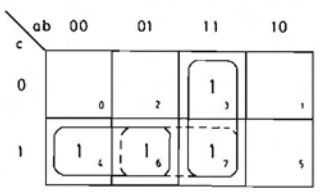


FIGURA 3.41. —**Tabla de Karnaugh de la función** $f = \sum_3 (3,4,6,7)$.

Minimizando la función por el método indicado en el apartado 3.2.1 resulta la expresión:

$$f = ab + \bar{a}c$$

que presenta la posibilidad de la aparición de un impulso aleatorio cuando $b = c = 1$.

En la figura 3.42*a* se representa la realización física de esta función con relés y se observa que, si b y c están cerrados, se produce una apertura transitoria del circuito al actuar el conmutador a si éste abre un contacto antes de cerrar el otro.

Esta función se puede realizar también mediante puertas NO-Y (NAND) transformando adecuadamente su expresión.

$$f = ab + \bar{a}c = \overline{\overline{ab}\ \ \overline{\bar{a}c}}$$

En la figura 3.42*b* se representa esta realización. Si la puerta N1 posee un tiempo de retraso a la propagación superior a N2, se producirá un impulso aleatorio cuando la variable a conmute del estado cero al uno siendo $b = c = 1$.

Para evitar la aparición de este fenómeno aleatorio se ha de sumar lógicamente a la expresión de f el término bc indicado en línea de puntos en la tabla de la figura 3.41 y que cubre la adyacencia entre los otros dos términos. Por tanto, para eliminar los fenómenos aleatorios estáticos de tipo cero al realizar la minimización de una función lógica mediante la tabla de Karnaugh de la suma de productos canónicos es necesario, no sólo cubrir todos los unos de la función, sino también cubrir todas las adyacencias, solapando todos los términos primos.

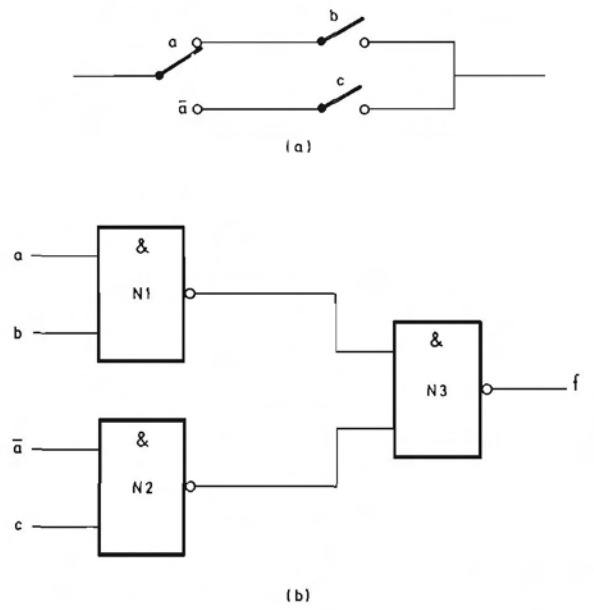


FIGURA 3.42.—**Realización física de la expresión mínima de la función *f* de la figura 3.41 que presenta un fenómeno aleatorio estático de tipo cero.**

La expresión resultante de f es pues

$$f = ab + \bar{a}c + bc$$

cuya realización física se representa en las figuras 3.43*a* y 3.43*b*.

El lector puede comprobar la ausencia en estos circuitos del fenómeno aleatorio anteriormente descrito.

En efecto, en la figura 3.43*b* se observa que, cuando las variables b y c se encuentran en estado uno simultáneamente, la salida de la puerta NO-Y N3 toma el valor lógico cero y, por tanto, la salida de N4 adopta el estado lógico uno independientemente del estado lógico de las restantes entradas.

Los fenómenos aleatorios de tipo uno se producen cuando, al cambiar de estado una variable, la función debe de permanecer en estado cero, pero pasa transitoriamente por el estado uno. Este fenómeno puede producirse cuando una función ha sido expresada mediante un producto de sumas. Sea una función f de las variables a, b, c, d, ..., que puede expresarse, tal como vimos en el capítulo 2, de la forma siguiente:

$$f(a, b, c, d...) = f(a, X) =$$
$$= [a + f(0, X)]\,[\bar{a} + f(1, X)]$$

Si para la combinación de estados X_0 de todas las variables excepto la a se verifica:

$$f(0, X_0) = f(1, X_0) = 0$$

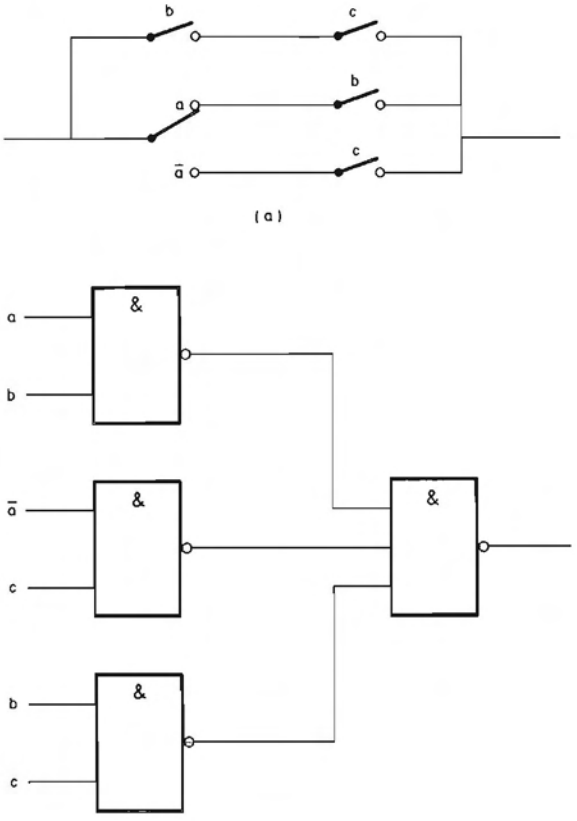


FIGURA 3.43.—**Realización física de la función *f* de la figura 3.41 en la que se ha suprimido el fenómeno aleatorio estático de tipo cero.**

resulta:

$$f = a\bar{a}$$

En régimen estático la función f se encuentra en estado lógico cero, pero al producirse un cambio en la variable a debido al retardo en la conmutación, puede suceder que a y $\bar{a}$ tomen el valor uno simultáneamente y que durante un cierto tiempo f tome por tanto el valor lógico uno. En la figura 3.44 se representa gráficamente este fenómeno.

La forma de eliminar este fenómeno es similar a la descrita para el tipo cero. La expresión de f ha de multiplicarse por $f(0, X) + f(1, X)$ que toma el valor lógico cero para la combinación X_0 de entrada. También se demuestra que la expresión de la función f no varía al añadir dicho término que cubre la adyacencia existente entre $a + f(0, X)$ y $\bar{a} + f(1, X)$.

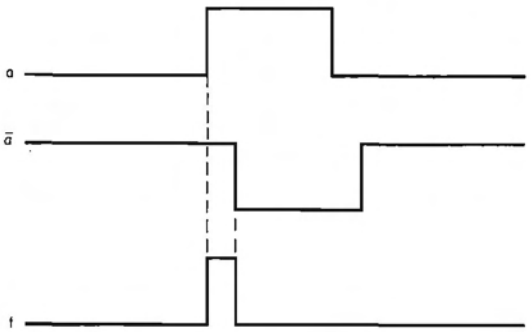


FIGURA 3.44.—**Representación gráfica de un fenómeno aleatorio estático de tipo uno.**

Veremos también un ejemplo sencillo. Sea la función f de tres variables,

$$f = \prod_3 (2, 3, 5, 7)$$

cuya tabla de Karnaugh se representa en la figura 3.45.

La expresión mínima deducida de dicha tabla es,

$$f = (b + \bar{c})\,(a + c)$$

Si $b = a = 0$ resulta:

$$f = c\,\bar{c}$$

y se puede presentar un fenómeno aleatorio de tipo uno como puede verse en los esquemas de realización física con relés y puertas NO-O de la figura 3.46.

Para eliminar este fenómeno aleatorio se ha de multiplicar lógicamente la expresión anterior por el término $a + b$ que se ha indicado en línea de puntos en la tabla de la figura 3.45. Se deduce como regla que, para minimizar mediante el método tabular de Karnaugh una función lógica expresada como producto de sumas canónicas, se han de cubrir todas las adyacencias, solapando de esta forma todos los términos primos.

La expresión resultante de f es:

$$f = (b + \bar{c})\,(a + c)\,(a + b)$$

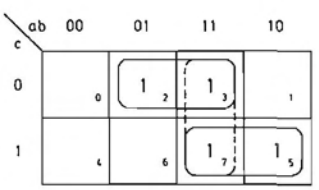


FIGURA 3.45.—**Tabla de Karnaugh de la función** $f = \prod_3 (2, 3, 5, 7)$.

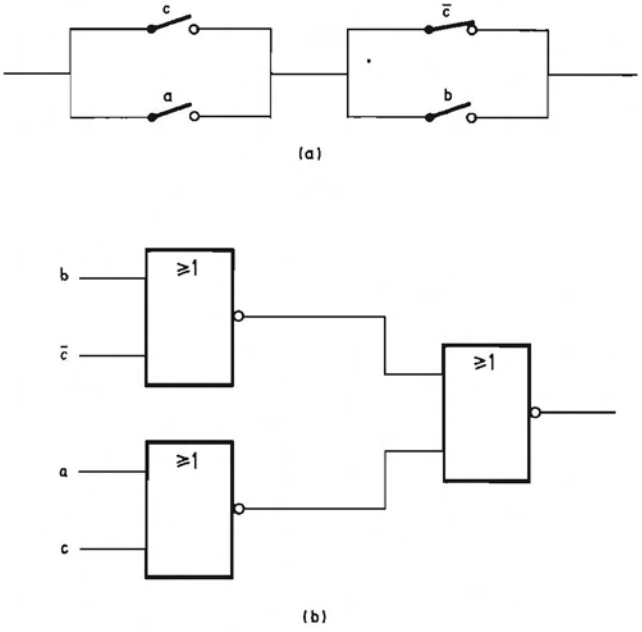


FIGURA 3.46.—**Realización de la expresión mínima de la función f de la figura 3.45 que presenta un fenómeno aleatorio estático de tipo uno.**

que, como puede observarse en la figura 3.47 que representa su realización física, está exenta de fenómenos aleatorios del tipo descrito.

Si se desea realizar la minimización de una función lógica por el método numérico, se han de cubrir igualmente todas las adyacencias a fin de evitar la aparición de fenómenos aleatorios estáticos de tipo cero o uno.

Esto se logra haciendo que los términos primos obtenidos por el procedimiento explicado en el apartado 3.2.2 realicen todos los términos canónicos que no tienen ningún otro adyacente y las agrupaciones de dos términos canónicos.

Como ejemplo minimizaremos la función

$$f = \sum_{4} (1, 2, 3, 5, 6, 7, 9, 10, 11, 14)$$

En la tabla 3.15 se representan las tablas de agrupación de los términos canónicos obtenidas por el procedimiento indicado en el apartado 3.2.2.

En la tabla 3.16 se presenta la relación de términos primos cuyas columnas corresponden a los términos de la tabla 2 de la 3.15. Se comprueba que todos los términos primos son esenciales y, por tanto, la expresión mínima de f en forma de suma de productos es:

$$f = a\bar{d} + a\bar{c} + b\bar{d} + b\bar{c} + \bar{a}b$$

El lector puede comprobar que, sin tener en cuenta los fenómenos aleatorios, serían suficientes los términos $a\bar{c}$, $\bar{a}b$ y $a\bar{d}$.

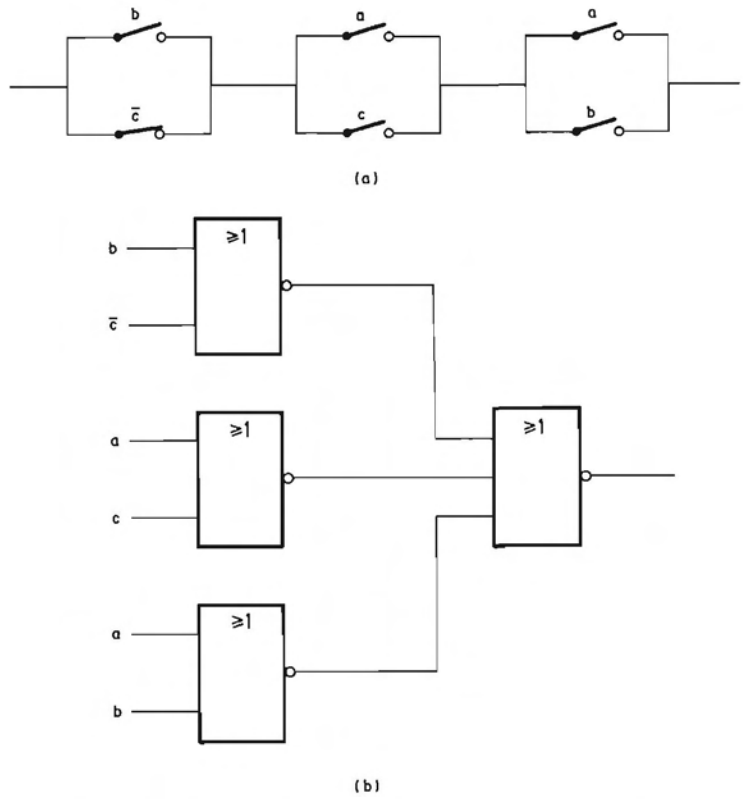


FIGURA 3.47.—**Realización física de la función *f* de la figura 3.45 en la que se ha suprimido el fenómeno aleatorio estático de tipo uno.**

3.6.2 Fenómenos aleatorios dinámicos

Se produce un fenómeno aleatorio dinámico cuando, al conmutar una variable de entrada, la salida debe cambiar de estado y lo hace pasando por un régimen transitorio tal como se indica en la figura 3.48.

Se analizará a continuación este fenómeno para las expresiones en forma de suma de productos. Para que un fenómeno aleatorio de tipo dinámico pueda producirse es necesario que, para una combinación determinada de valores lógicos de todas las variables de entrada, excepto la a, la salida del circuito adopte la expresión mínima:

$$f = a + a \quad \bar{a}$$

En régimen estático $f = a$, pero veamos lo que puede ocurrir al cambiar a de 0 a 1. Supondremos que el cambio de estado de todas las variables a no es simultáneo y las denominaremos a_1, a_2 y a_3 y por tanto:

$$f = a_1 + a_2 \quad \bar{a}_3$$

Tabla 1

N.º de unos		
1	1	X
	2	X
	3	X
	5	X
2	6	X
	9	X
	10	X
	7	X
3	11	X
	14	X

Tabla 2

			Dif.
1 - 3	X	2	
1 - 5	X	4	
1 - 9	X	8	
2 - 3	X	1	
2 - 6	X	4	
2 -10	X	8	
3 - 7	X	4	
3 -11	X	8	
5 - 7	X	2	
6 - 7	X	1	
6 -14	X	8	
9 -11	X	2	
10 -11	X	1	
10 -14	X	4	

Tabla 3

	Dif.
1-3-5-7	2,4
1-3-9-11	2,8
1-5-3-7	4,2
1-9-3-11	8,2
2-3-6-7	1,4
2-3-10-11	1,8
2-6-3-7	4,1
2-6-10-14	4,8
2-10-3-11	8,1
2-10-6-14	8,4

Tabla 3 *reducida*

	Dif.
1-3-5-7	2,4
1-3-9-11	2,8
2-3-6-7	1,4
2-3-10-11	1,8
2-6-10-14	4,8

TABLA 3.15

Términos primos	1-3	1-5	1-9	2-3	2-6	2-10	3-7	3-11	5-7	6-7	6-14	9-11	10-11	10-14
1-3-5-7	X	X					X		⊗					
1-3-9-11	X		⊗					X				⊗		
2-3-6-7				X	X		X			⊗				
2-3-10-11				X		X		X					⊗	
2-6-10-14					X	X					⊗			⊗

TABLA 3.16

Supongamos también que $a_1 = a_2 = 1$, $\bar{a}_3 = 0$ y por tanto $f = 1$. Si a cambia del estado uno al cero, se produce un fenómeno aleatorio dinámico cuando los cambios de a_1, a_2 y $\bar{a}_3$ se realizan de acuerdo con la siguiente tabla

	a_1	a_2	$\bar{a}_3$	f
Estado inicial	1	1	0	1
Estados	0	1	0	0
Transitorios	0	1	1	1
Estado final	0	0	1	0

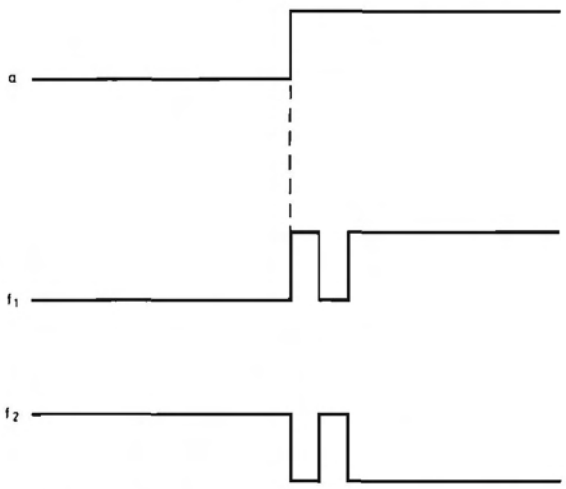


FIGURA 3.48.—**Representación gráfica de un fenómeno aleatorio dinámico.**

Se comprueba la igualdad de este cambio con el descrito para la función f_2 de la figura 3.48.

Un ejemplo práctico es el circuito de la figura 3.49 que corresponde a la función:

$$f = \overline{\overline{\bar{a}b} \quad \overline{\bar{a}\bar{c}} \quad \overline{ad}} = \bar{a}b \quad \bar{a}\bar{c} + ad = [(\bar{a} + \bar{b})\,(a + c)] + ad$$

Si

$$b = d = 1 \text{ y } c = 0$$

Se verifica

$$f = \bar{a}a + a$$

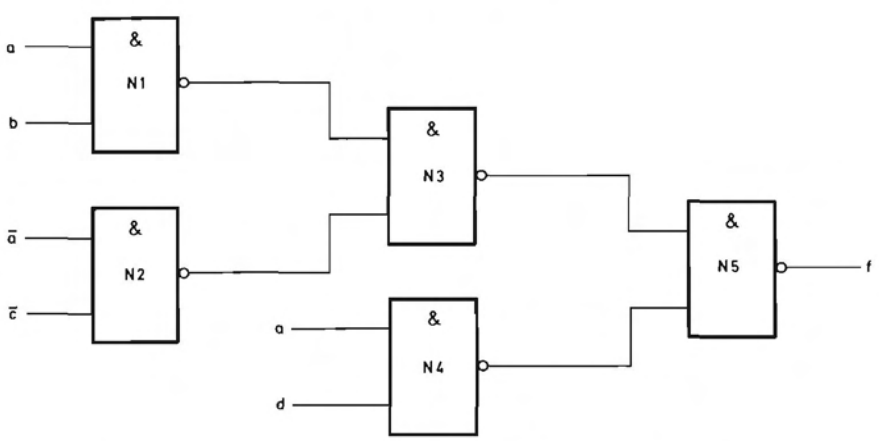


FIGURA 3.49.—**Ejemplo de circuito que presenta un fenómeno aleatorio dinámico.**

y observando la figura 3.49 se deduce que para que se produzca un fenómeno aleatorio dinámico al cambiar a de uno a cero es necesario que el tiempo de propagación de la señal a través de N_4 sea inferior al de N_1 más el de N_3 y al de N_2 más el de N_3 y además que a su vez el tiempo de propagación de N_2 sea menor al de N_1.

La eliminación de la posibilidad de fenómenos aleatorios dinámicos se realiza al minimizar una función por el método de Karnaugh o el numérico indicados en los apartados 3.2.1 y 3.2.2 porque la expresión de suma de productos obtenida nunca contendrá el producto $a\bar{a}$ (o la suma $a + \bar{a}$ si se trata de una expresión de producto de sumas). Por tanto, si además se realiza la minimización anulando la posibilidad de fenómenos aleatorios estáticos, se tiene la seguridad de que no existirán fenómenos aleatorios dinámicos.

3.7 BLOQUES FUNCIONALES COMBINACIONALES

En apartados anteriores se han estudiado las funciones lógicas y su realización mediante puertas NO-Y (NAND) y NO-O (NOR). Existen funciones lógicas y, en especial, multifunciones que presentan la característica de ser de aplicación general, es decir, que pueden ser utilizadas como bloques funcionales para la construcción de sistemas digitales.

El progreso de las técnicas de integración permitió (en la década de los sesenta) la realización de dichas funciones y multifunciones en circuito integrado, constituyendo lo que se ha denominado la escala de integración media (MSI). La combinación de estos circuitos con los secuenciales estudiados en el capítulo 6 generó bloques funcionales complejos que al ser integrados han dado lugar a los circuitos de gran (LSI), muy gran (VLSI) y ultra gran (ULSI) escala de integración.

Los bloques funcionales combinacionales responden, en general, al diagrama de bloques de la figura 3.50 en el que las variables de entrada se dividen en dos grupos:

a) Variables de entrada con las que el circuito realiza una determinada función u operación y a las que se puede denominar operativas.
b) Variables de entrada que influyen en la forma en que el circuito actúa sobre las operativas o que modifican el resultado o variables de salida del propio circuito. Reciben el nombre de variables de control.

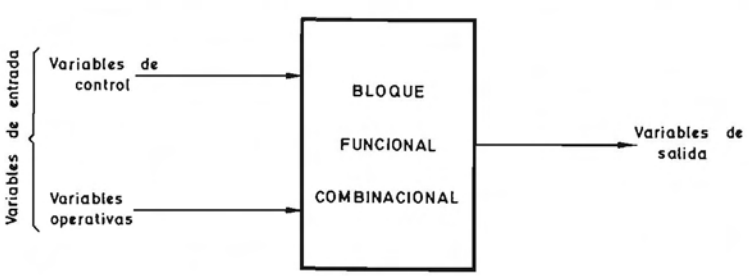


FIGURA 3.50.—**Diagrama de bloques de un bloque funcional combinacional.**

Se dice que existe entre ambos tipos de entradas una relación de dependencia. Hay varios tipos de dependencias que se distinguen por la forma en que se realiza la citada modificación, pero existe una regla general, común a todas, que es la siguiente:

La entrada que modifica la forma en que el circuito actúa sobre las de información u operativas o que modifica el resultado se indica con una letra correspondiente al tipo de dependencia seguida de un número. Cada una de las entradas afectadas tiene asignado el mismo número.

Las principales variables de control de los bloques funcionales combinacionales y las relaciones de dependencia asociadas a ellas son:

— Entrada o entradas de desinhibición/inhibición (enable/disable) que como su nombre indica inhiben o desinhiben la acción del circuito sobre las demás entradas y ponen las salidas en un determinado nivel lógico. La relación de dependencia correspondiente puede ser de tipo Y [G (AND)] o de tipo O [V (OR)]. En el primer caso, la colocación en cero de la entrada de control pone a cero las variables de salida y en el segundo, la colocación en uno de la entrada de control pone a uno las salidas.
— Entrada o entradas de control del tercer estado de las variables de salida. Cuando estas variables se encuentran en un determinado estado lógico, la salida se pone en tercer estado, que recibe esta denominación porque en él la salida presenta una gran impedancia a ambos polos de la tensión de alimentación. En el capítulo 5 se estudian las puertas de tres estados en diferentes tecnologías. Esta relación de dependencia recibe el nombre de desinhibición [EN (Enable)].
— Entradas de control de inversión de las variables operativas, que según cual sea su estado lógico hacen que el circuito actúe directamente sobre las variables operativas o sobre sus inversas. Esta relación de dependencia recibe el nombre de inversión [N (Negate)].
— Entradas de control de inversión de las variables de salida, que según cual sea su estado lógico hacen que a la salida aparezcan las variables en forma directa o invertida. Esta relación de dependencia recibe también el nombre de inversión [N (Negate)].
— Entradas de selección de operación o de modo de operación que como su nombre indica seleccionan la operación que realiza el circuito con las entradas operativas. La relación de dependencia correspondiente recibe el nombre de «modo de operación» [M (Mode)].

En este apartado se estudian los bloques funcionales combinacionales cuyo conocimiento resulta imprescindible para diseñar sistemas digitales y comprender el funcionamiento de los bloques más complejos. En su representación gráfica se utilizan los símbolos lógicos normalizados que se describen en el apartado A1.4.3. del apéndice 1.

3.7.1 Decodificadores. Demultiplexores

a) Decodificadores.

Los circuitos decodificadores son sistemas combinacionales que generan los

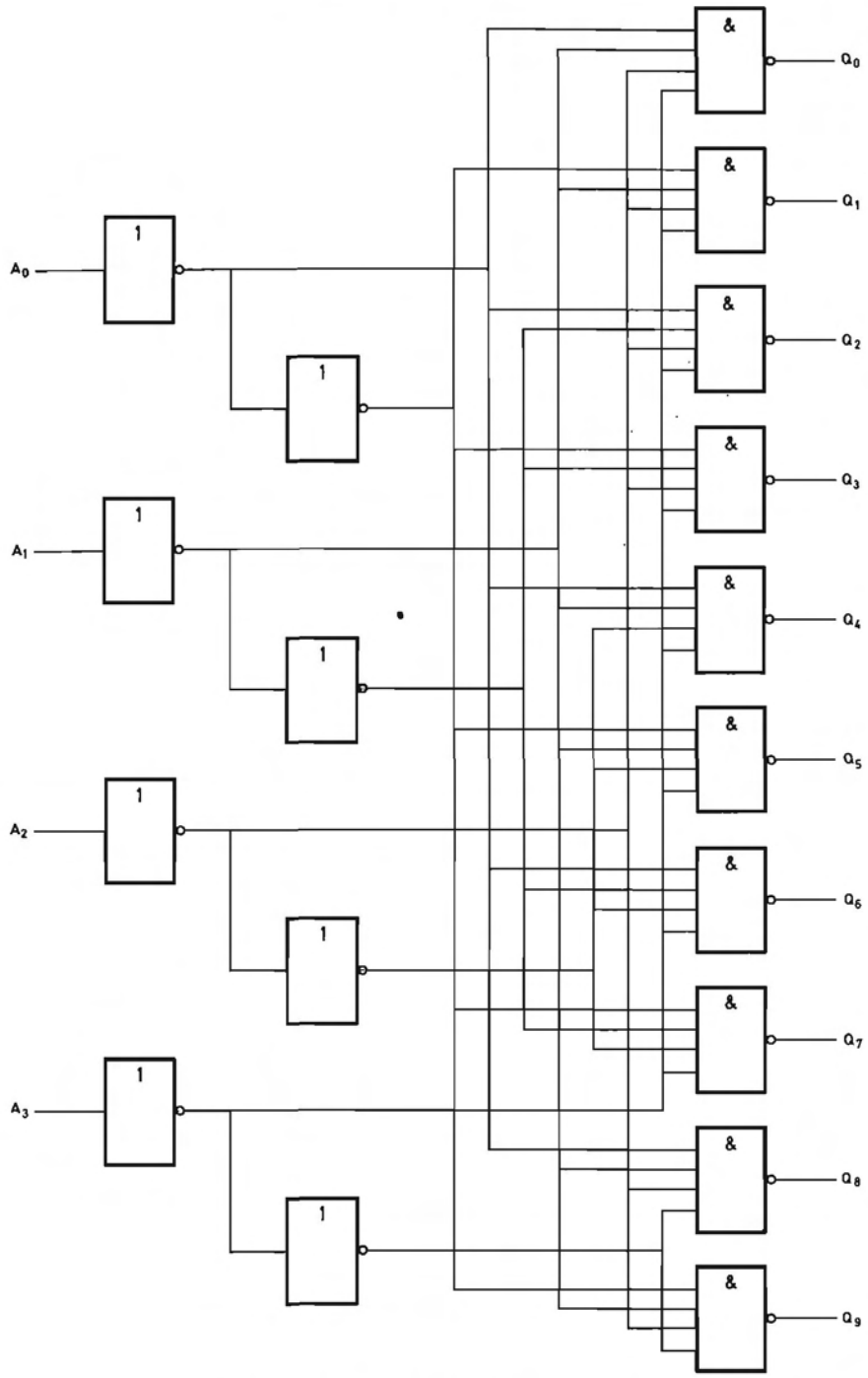


FIGURA 3.51.—**Esquema de un decodificador decimal.**

productos canónicos de un conjunto de variables binarias aplicadas a sus entradas y se dividen en excitadores (drivers) y no excitadores, según que sus salidas puedan o no controlar respectivamente a un indicador numérico. También reciben el nombre de decodificadores excitadores algunos circuitos que son realmente convertidores de código (ver figura 3.65).

En primer lugar se estudian los decodificadores no excitadores que poseen innumerables aplicaciones.

En la figura 3.51 se representa el esquema de un decodificador decimal que genera los productos canónicos 0 a 9 de cuatro variables binarias. Una cualquiera de las salidas toma el valor cero lógico solamente cuando aparece una determinada combinación de entrada. En la tabla 3.17 se representa la tabla de verdad del decodificador decimal, que el lector puede comprobar analizando el funcionamiento del circuito de la figura 3.51. La doble inversión de las variables A_0 a A_3 tiene como finalidad hacer que el decodificador cargue al circuito que se conecte a su entrada con el equivalente a una puerta lógica.

En la figura 3.52 se representa el símbolo lógico de este circuito en versión antigua y en la nueva normalizada. La aplicación inmediata de este circuito es la conversión del código BCD natural al decimal. En efecto aplicando a las entradas 1, 2, 4 y 8 (A_0 a A_3 en el símbolo no normalizado) una combinación del código BCD natural, solamente una de las salidas toma el valor lógico cero y a dicha salida se le asigna el número decimal equivalente a dicha combinación.

Entradas				*Salidas*									
A_3	A_2	A_1	A_0	Q_0	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6	Q_7	Q_8	Q_9
0	0	0	0	0	1	1	1	1	1	1	1	1	1
0	0	0	1	1	0	1	1	1	1	1	1	1	1
0	0	1	0	1	1	0	1	1	1	1	1	1	1
0	0	1	1	1	1	1	0	1	1	1	1	1	1
0	1	0	0	1	1	1	1	0	1	1	1	1	1
0	1	0	1	1	1	1	1	1	0	1	1	1	1
0	1	1	0	1	1	1	1	1	1	0	1	1	1
0	1	1	1	1	1	1	1	1	1	1	0	1	1
1	0	0	0	1	1	1	1	1	1	1	1	0	1
1	0	0	1	1	1	1	1	1	1	1	1	1	0
1	0	1	0	1	1	1	1	1	1	1	1	1	1
1	0	1	1	1	1	1	1	1	1	1	1	1	1
1	1	0	0	1	1	1	1	1	1	1	1	1	1
1	1	0	1	1	1	1	1	1	1	1	1	1	1
1	1	1	0	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1

TABLA 3.17.—**Tabla de verdad del decodificador decimal.**

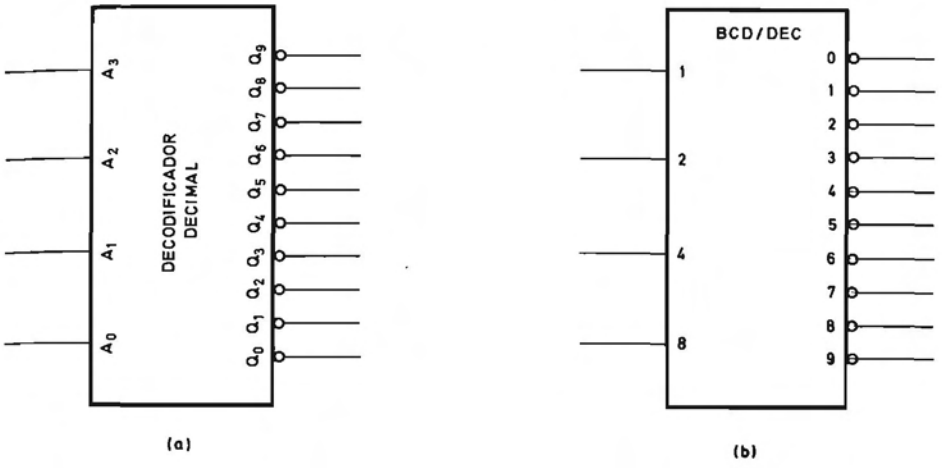


FIGURA 3.52.—**Símbolo lógico de un decodificador decimal: *a*) no normalizado; *b*) normalizado.**

I_1	I_2	A_3	A_2	A_1	A_0	Q_0	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6	Q_7	Q_8	Q_9	Q_{10}	Q_{11}	Q_{12}	Q_{13}	Q_{14}	Q_{15}
1	1	X	X	X	X	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	0	X	X	X	X	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	1	X	X	X	X	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	1	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	1	0	0	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1
0	0	0	1	0	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1
0	0	0	1	1	0	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1
0	0	0	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1
0	0	1	0	0	0	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1
0	0	1	0	0	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1
0	0	1	0	1	0	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1
0	0	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1
0	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
0	0	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1
0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

X Las entradas correspondientes pueden tomar los valores 0 o 1 lógicos.

TABLA 3.18.—**Tabla de verdad del decodificador hexadecimal.**

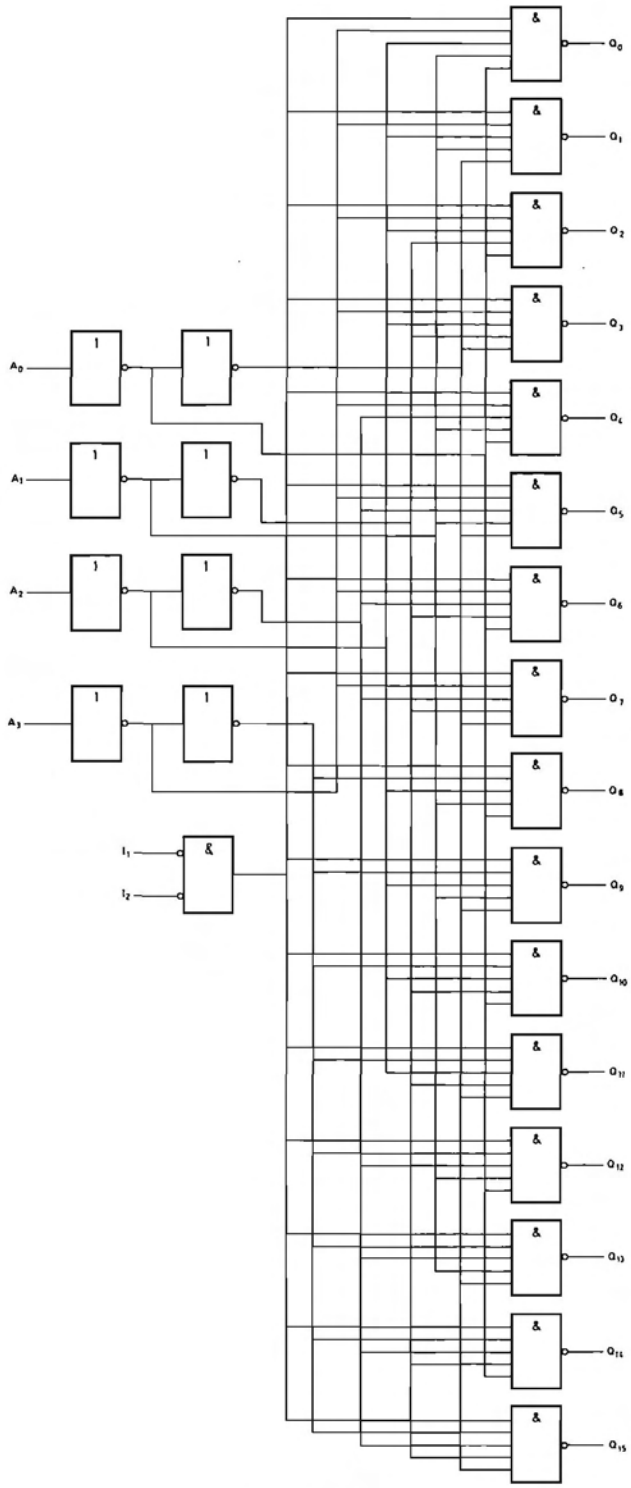


FIGURA 3.53. —**Esquema de un decodificador hexadecimal (uno entre dieciséis).**

En la figura 3.53 se representa con la simbología normalizada el esquema de un decodificador que genera los dieciséis productos canónicos de una función de cuatro variables aplicadas a sus entradas A_0 a A_3.

Al igual que el decodificador que se acaba de estudiar, una salida toma el valor cero solamente cuando se presenta una determinada combinación binaria en las entradas A_0 a A_3. Existen, además, dos entradas de inhibición I_1 e I_2; cuando cualquiera de ellas toma el valor lógico uno, todas las salidas toman el valor lógico uno. En la tabla 3.18 se presenta la tabla de verdad de este circuito.

En la figura 3.54 se representa el símbolo normalizado de este decodificador que presenta la ventaja, además, de que su tabla de verdad se deduce directamente de él. Las salidas 0 a 15 corresponden a los 16 productos canónicos de las variables de entrada 1, 2, 4 y 8. La inversión colocada en cada salida indica que en ellas aparece el inverso del producto lógico correspondiente. La denominación de las entradas de control con el apelativo de desinhibición (enable) en lugar de inhibición (disable) hace ver que sólo se produce la decodificación si $\overline{EN}_1$ (I_1) y $\overline{EN}_2$ (I_2) están en nivel cero simultáneamente (función &).

La aplicación más inmediata del decodificador hexadecimal es la generación de todos los productos canónicos de las cuatro variables A, B, C y D. Para ello se conectan éstas a las entradas 1, 2, 4 y 8 y se obtienen los productos canónicos invertidos P_0 a P_{15} en las salidas 0 a 15 (fig. 3.54).

En la figura 3.55 se representa un decodificador de uno entre 32 estados realizado mediante dos decodificadores hexadecimales. Este circuito genera todos los

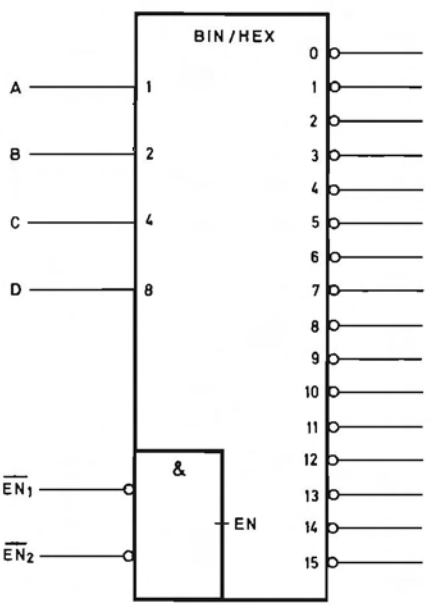


FIGURA 3.54.—**Símbolo lógico normalizado del decodificador hexadecimal de la figura 3.53.**

productos canónicos de 5 variables binarias. Se deja al lector que realice su análisis y compruebe su correcto funcionamiento.

En la figura 3.56 se representa el símbolo lógico de un decodificador de binario

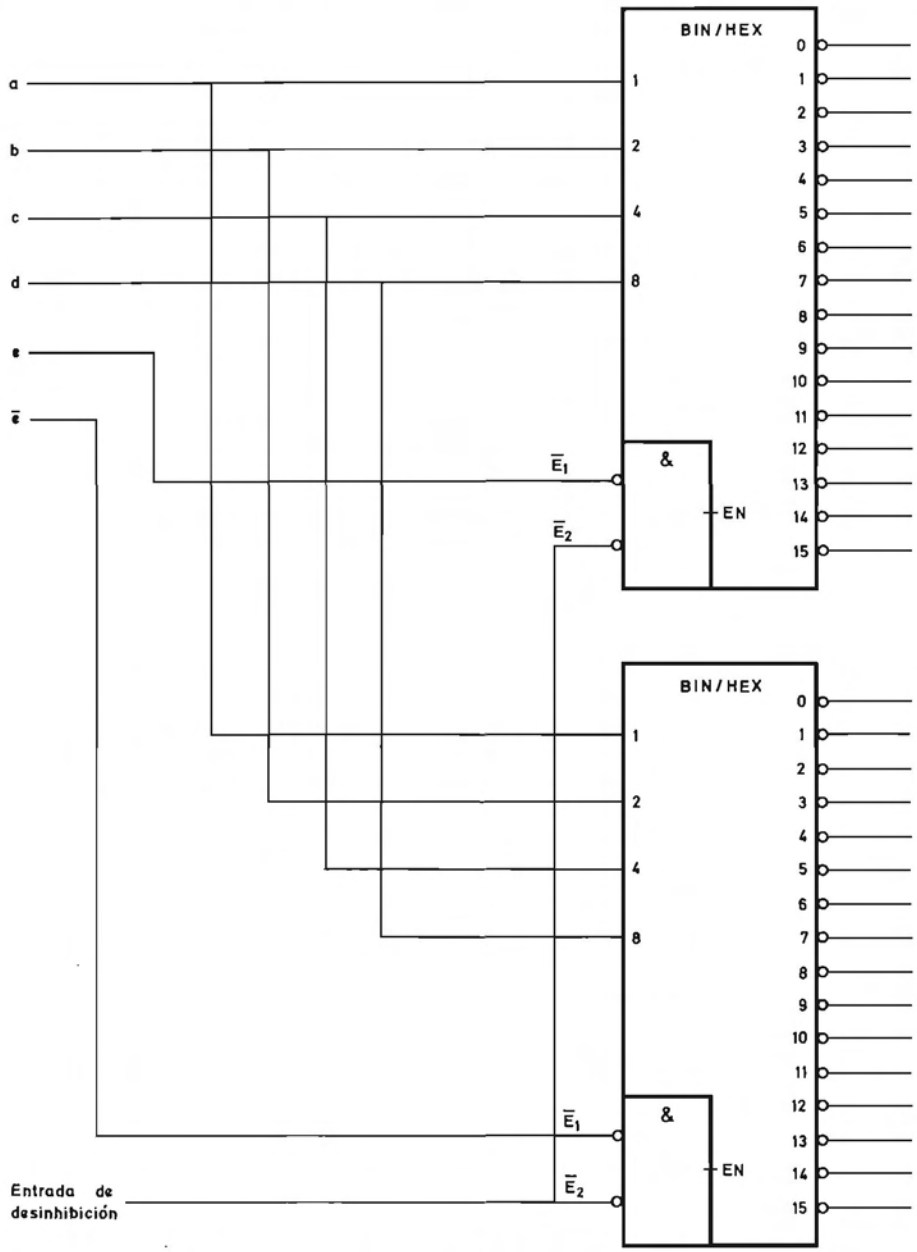


FIGURA 3.55.—**Decodificador de uno entre treinta y dos realizado con dos decodificadores hexadecimales.**

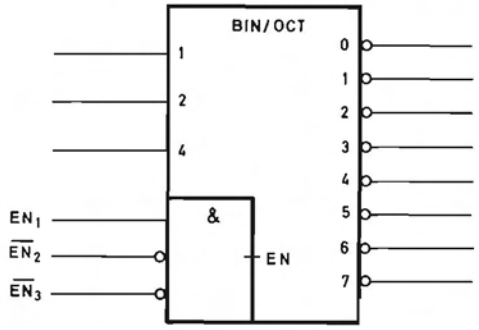


FIGURA 3.56. —**Símbolo lógico normalizado de un decodificador de binario a octal.**

a octal que posee tres entradas de desinhibición que generan una desinhibición interna *EN* por medio de la función:

$$EN = EN_1 \overline{EN_2}\,\overline{EN_3}$$

En la figura 3.57 se representa el esquema de un decodificador de uno entre cuatro que genera todos los productos canónicos de dos variables. La tabla de verdad de este decodificador que posee una entrada de inhibición se representa en la tabla 3.19.

I	A_1	A_0	Q_0	Q_1	Q_2	Q_3
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

X Las entradas correspondientes pueden tomar los valores 0 o 1 lógicos

TABLA 3.19

En la figura 3.58 se representa el símbolo lógico normalizado de un circuito de escala de integración media que contiene dos decodificadores de uno entre cuatro.

La utilización conjunta del decodificador uno entre cuatro y los decodificadores decimal y hexadecimal permite realizar fácilmente decodificadores de un número elevado de variables.

En la figura 3.59 se indica el esquema de un decodificador de seis variables realizado por un decodificador de uno entre cuatro en combinación con cuatro decodificadores hexadecimales.

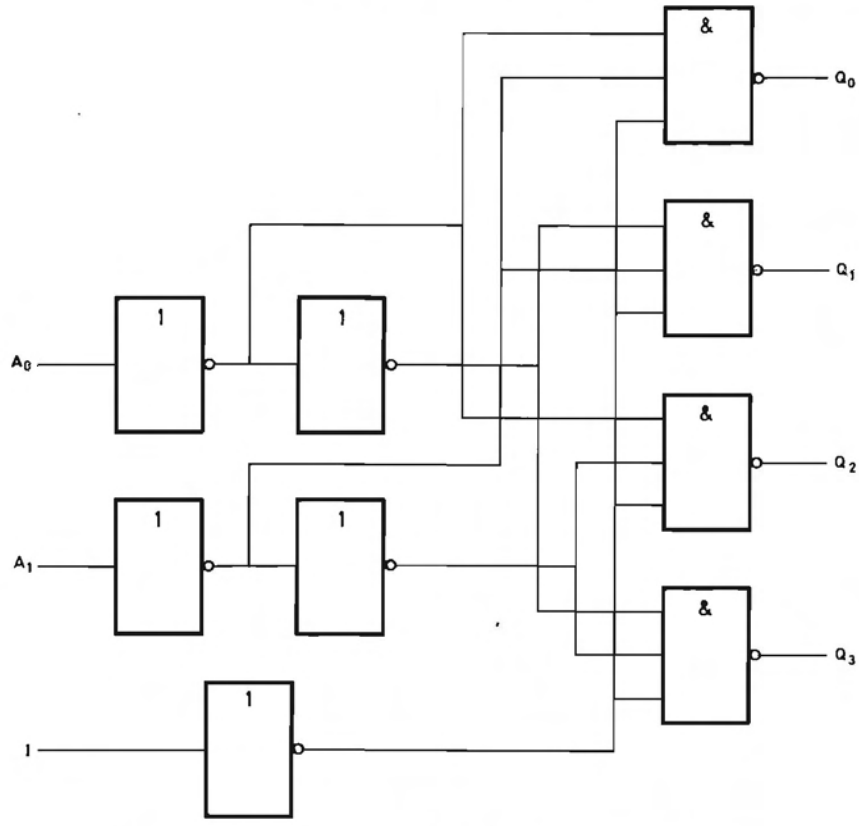


FIGURA 3.57.—**Esquema de un decodificador de uno entre cuatro.**

Los decodificadores admiten diferentes variables de control. Como ejemplo en la figura 3.60 se representa un decodificador de uno entre cuatro que posee tres variables de control:

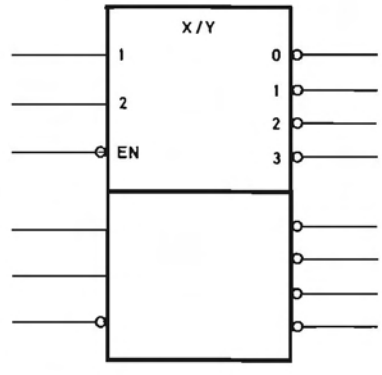


FIGURA 3.58.—**Símbolo lógico normalizado de un doble decodificador de uno entre cuatro.**

— Una variable de desinhibición $\overline{G}$ que desinhibe la decodificación cuando se encuentra en nivel lógico cero. En efecto, si G se encuentra en nivel uno, se ponen a nivel uno las salidas de todas las puertas NO-Y (NAND).

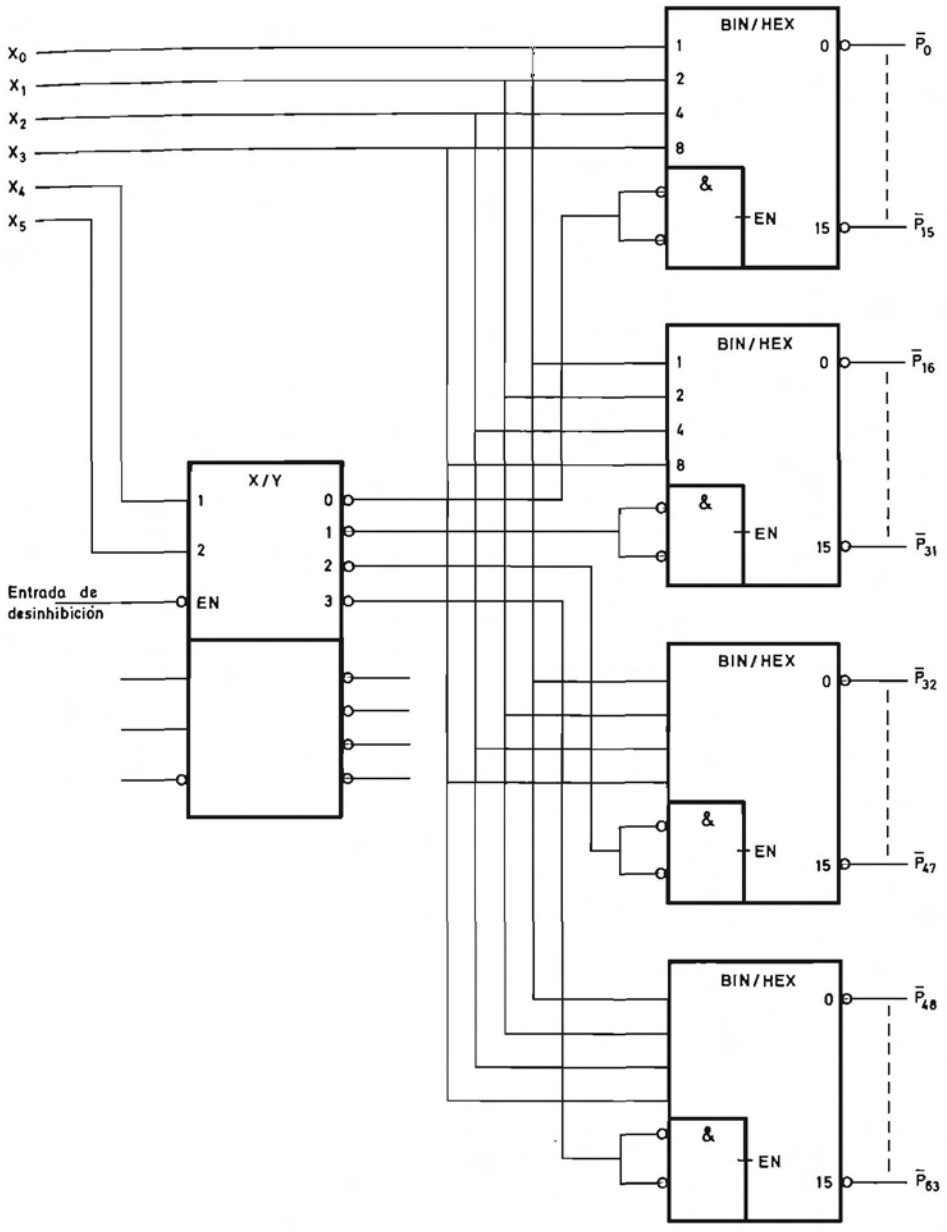


FIGURA 3.59.—**Esquema de un decodificador de seis variables binarias realizado con un decodificador de uno entre cuatro y cuatro decodificadores hexadecimales.**

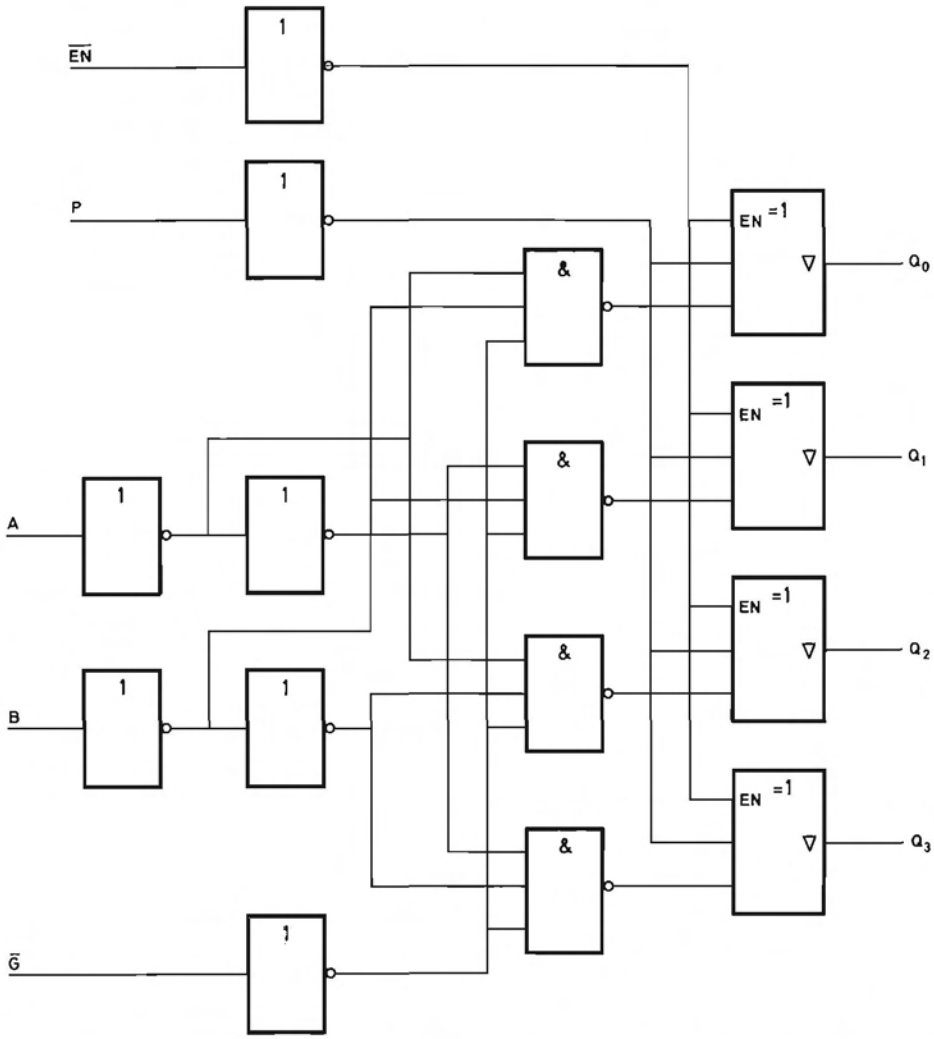


FIGURA 3.60.—**Esquema de un decodificador de uno entre cuatro con variable de desinhibición, de inversión y de control del tercer estado de las salidas.**

— Una variable de control de inversión o de polaridad de salida P que según se encuentre en nivel cero o uno hace que las salidas sean activas con uno o cero respectivamente.
— Una variable $\overline{EN}$ de control del tercer estado de las salidas que hace que todas se pongan en dicho estado cuando ella se encuentra en nivel uno.

En la tabla 3.20 se representa la tabla de verdad del decodificador de la figura 3.60. Se recomienda al lector que la analice para comprobar la veracidad de lo indicado en los párrafos anteriores. En la figura 3.61 se representa el símbolo nor-

$\overline{EN}$	P	$\overline{G}$	B	A	Q_0	Q_1	Q_2	Q_3
1	X	X	X	X	Tercer Estado			
0	0	1	X	X	0	0	0	0
0	1	1	X	X	1	1	1	1
0	0	0	0	0	1	0	0	0
0	0	0	0	1	0	1	0	0
0	0	0	1	0	0	0	1	0
0	0	0	1	1	0	0	0	1
0	1	0	0	0	0	1	1	1
0	1	0	0	1	1	0	1	1
0	1	0	1	0	1	1	0	1
0	1	0	1	1	1	1	1	0

TABLA 3.20.—**Tabla de verdad del decodificador uno entre cuatro de la figura 3.60.**

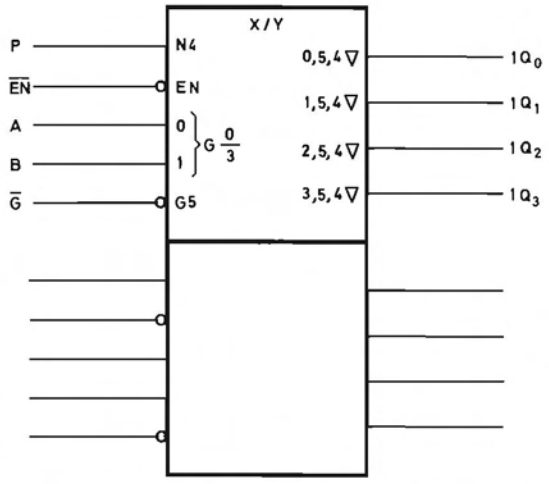


FIGURA 3.61.—**Símbolo lógico normalizado de un doble decodificador como el de la figura 3.60.**

malizado de un doble decodificador como el descrito que constituye un ejemplo de la combinación de simultaneidad de relaciones de dependencia descrita en el apartado A1.4.3.8 del apéndice 1. Al lector que no conozca la simbología normalizada se le recomienda nuevamente la lectura del citado apéncice.

Una aplicación muy interesante de los decodificadores no excitadores en combinación con una puerta NO-Y es la generación de funciones lógicas (fig. 3.62). La función se genera conectando a las entradas de la puerta NO-Y las salidas correspondientes a los productos canónicos que toman el valor uno.

Mediante el decodificador decimal se puede generar cualquier función de tres variables conectando éstas a las entradsa 1, 2 y 4. Las salidas 0 a 7 toman el valor

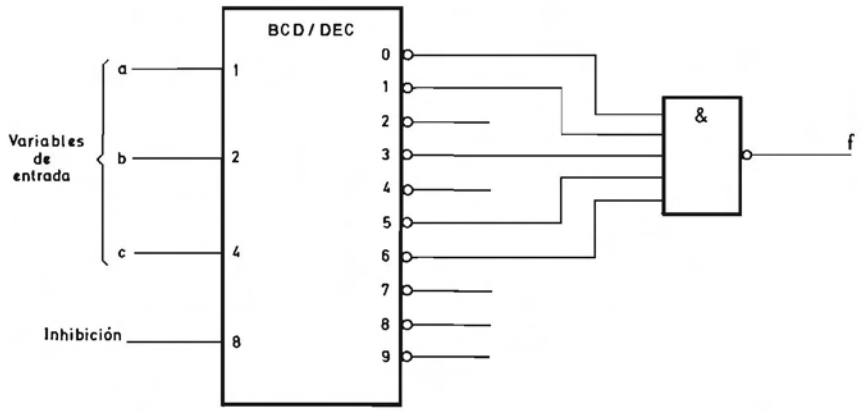


FIGURA 3.62.—**Generación de funciones lógicas mediante un decodificador decimal.**

cero cuando en aquéllas se presenta la combinación correspondiente y, por tanto, cada una de ellas realiza un producto canónico de las variables de entrada. La entrada 8 se puede utilizar como inhibidora, manteniéndola en estado lógico cero y haciéndola subir al estado uno cuando se desea inhibir la generación de la función. En la figura 3.62 se representa el circuito que genera la función de tres variables $f = \sum_{3}(0, 1, 3, 5, 6)$.

En la figura 3.63 se indica la utilización del decodificador hexadecimal como generador de funciones de cuatro variables a, b, c y d en combinación con una puerta NO-Y.

En dicha figura se generan las funciones:

$$f_1\ (a, b, c, d) = \sum_{4}\ (0, 9, 15)$$

$$f_2\ (a, b, c, d) = \sum_{4}\ (1, 2, 5, 10)$$

De igual forma con el decodificador uno entre cuatro se puede generar cualquier función de dos variables.

Los decodificadores excitadores tal como se indicó al principio de este apartado se diseñan para activar cargas externas como por ejemplo indicadores numéricos, relés electromagnéticos u optoacopladores. Sus salidas admiten unas tensiones mayores y poseen una cargabilidad más elevada que los demás bloques funcionales combinacionales de la misma tecnología.

En el diseño de un decodificador excitador surgen diversas variantes que hacen que existan diferentes circuitos en la práctica.

En la figura 3.64 se representa el símbolo normalizado de un decodificador excitador de decimal codificado en binario natural (BCD) a decimal en el que las salidas son de colector abierto y se indican con el símbolo correspondiente (ver apartado A1.3 del apéndice 1). Debajo del indicativo se coloca el símbolo ▷ que especifica que el circuito posee amplificadores (drivers) a la salida que le dan las características eléctricas anteriormente indicadas.

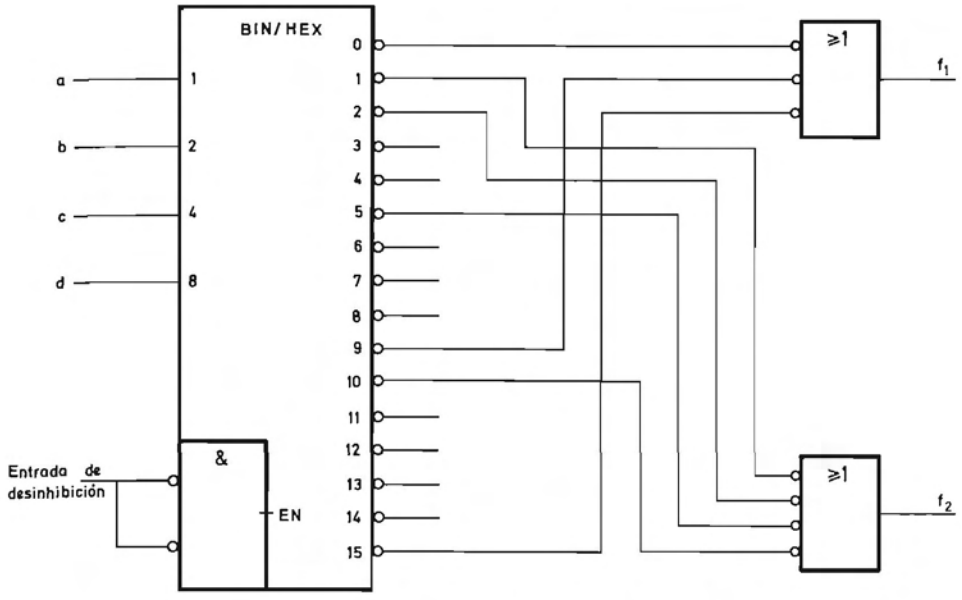


FIGURA 3.63.—**Generación de funciones lógicas con un decodificador hexadecimal.**

Aunque no es muy correcto, los fabricantes de circuitos integrados suelen llamar decodificadores excitadores de BCD a 7 segmentos (BCD to seven segment decoder/driver) a los convertidores de código que cambian la información de un código binario a otro distinto del decimal y que además poseen amplificadores a la salida. Uno de los más utilizados es el que convierte del BCD natural al de 7 segmentos de los visualizadores (displays) de estado sólido.

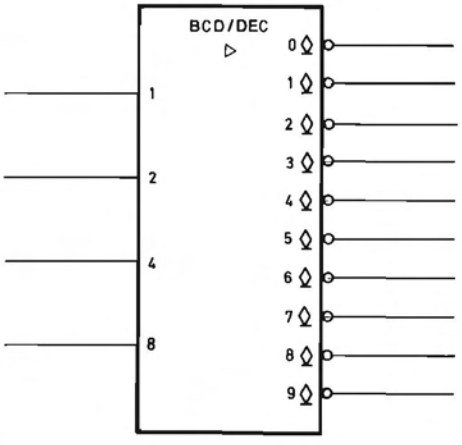


FIGURA 3.64.—**Símbolo lógico de un decodificador excitador (decoder driver) decimal.**

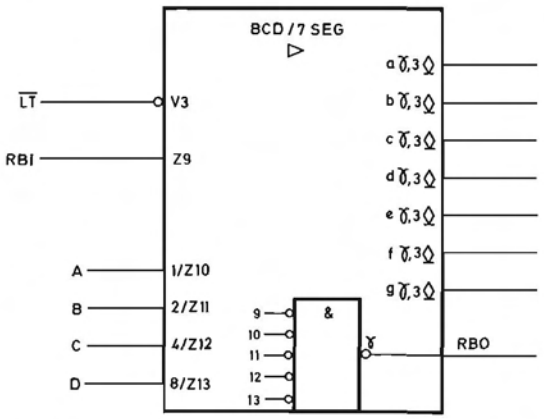


FIGURA 3.65.—**Símbolo lógico de un decodificador excitador de BCD a siete segmentos.**

En la figura 3.65 se representa el símbolo lógico de un decodificador excitador de decimal codificado en binario natural (BCD) a un código de 7 segmentos adecuado para gobernar visualizadores numéricos de diodos emisores de luz con el citado número de segmentos o barras. Las salidas del circuito son de colector abierto y por ello en ellas se coloca el símbolo ◇. La tabla de verdad del circuito se representa en la tabla 3.21. Las salidas reciben las denominaciones «*a*, *b*, *c*, *d*, *e*, *f*, *g*», que corresponden a las barras de un visualizador de 7 segmentos representado en la figura 3.66*b*.

Los estados de las salidas reciben la denominación «*S*» (correspondiente a transistor saturado) o «*C*» (correspondiente a transistor cortado). El circuito posee una entrada de prueba $\overline{LT}$ (Lamp Test) activa con un cero lógico que cuando se en-

$\overline{LT}$	*RBI*	*D*	*C*	*B*	*A*	*a*	*b*	*c*	*d*	*e*	*f*	*g*	*RBO*
1	1	0	0	0	0	S	S	S	S	S	S	C	1
1	X	0	0	0	1	C	S	S	C	C	C	C	1
1	X	0	0	1	0	S	S	C	S	S	C	S	1
1	X	0	0	1	1	S	S	S	S	C	C	S	1
1	X	0	1	0	0	C	S	S	C	C	S	S	1
1	X	0	1	0	1	S	C	S	S	C	S	S	1
1	X	0	1	1	0	C	C	S	S	S	S	S	1
1	X	0	1	1	1	S	S	S	C	C	C	C	1
1	X	1	0	0	0	S	S	S	S	S	S	S	1
1	X	1	0	0	1	S	S	S	C	C	S	S	1
1	0	0	0	0	0	C	C	C	C	C	C	C	0
0	X	X	X	X	X	S	S	S	S	S	S	S	1

TABLA 3.21

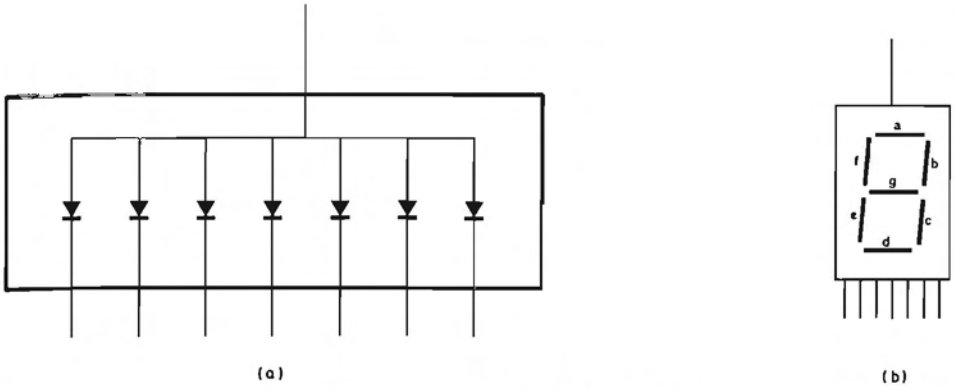


FIGURA 3.66.—**Visualizazor de siete segmentos: *a*) montaje de diodos emisores de luz (LED) con ánodo común; *b*) realización física.**

cuentra en este estado satura todos los transistores y por ello se indica con el símbolo V (correspondiente a la relación de dependencia 0 estudiada en el apartado A1.4.3.2 del apéndice 1). Además tiene una entrada de inhibición o de propagación de cero *RBI* (Ripple Blanking Input) que cuando está en nivel cero inhibe la visualización si el número a visualizar es el cero en binario. La entrada *RBI* está ligada con la salida de propagación de ceros [*RBO* (Ripple Blanking Output)], que está siempre en uno excepto en la situación que se acaba de describir. Para ello se genera la variable γ (figura 3.65) que constituye una función NO-Y del inverso de las entradas *A*, *B*, *C* y *D* y el inverso de *RBI*. Cada salida que se aplica al visualizador incorpora una letra de la «*a*» a la «*g*» [correspondiente al segmento del visualizador sobre el que actúa (figura 3.66 y tabla 3.21)] seguida de la letra γ. Ello significa que para que esta salida se active [transistor saturado (S)] ha de aplicarse a las entradas *A*, *B*, *C* y *D* una combinación binaria para la cual se deba encender la barra correspondiente y simultáneamente ha de estar γ en nivel uno. Por ello la relación entre el terminal γ y los terminales de salida de los segmentos es la relación *Y*.

Además todas las salidas que se aplican al visualizador deben activarse si se pone a cero el terminal $\overline{LT}$. Por ello cada terminal incorpora el número 3. La «,» entre γ y 3 indica que γ y *V*3 actúan simultáneamente.

Este decodificador permite visualizar números representados en BCD natural mediante visualizadores realizados con diodos emisores de luz conectados en un montaje de ánodo común (figura 3.66*a*), realizando el circuito de la figura 3.67 en el que las resistencias *R* limitan la corriente a través de los diodos. La utilidad de la entrada *RBI* y la salida *RBO* se comprueba con el montaje de la figura 3.68 que representa un visualizador de tres dígitos, en el que no se visualizan los ceros no significativos.

En efecto, la conexión en serie (ripple) de las entradas *RBI* y salidas *RBO* hace que un determinado dígito solamente se active si el número presente a la entrada de su visualizador es distinto de cero o bien si siendo igual a cero, es distinto de cero alguno de los que le preceden.

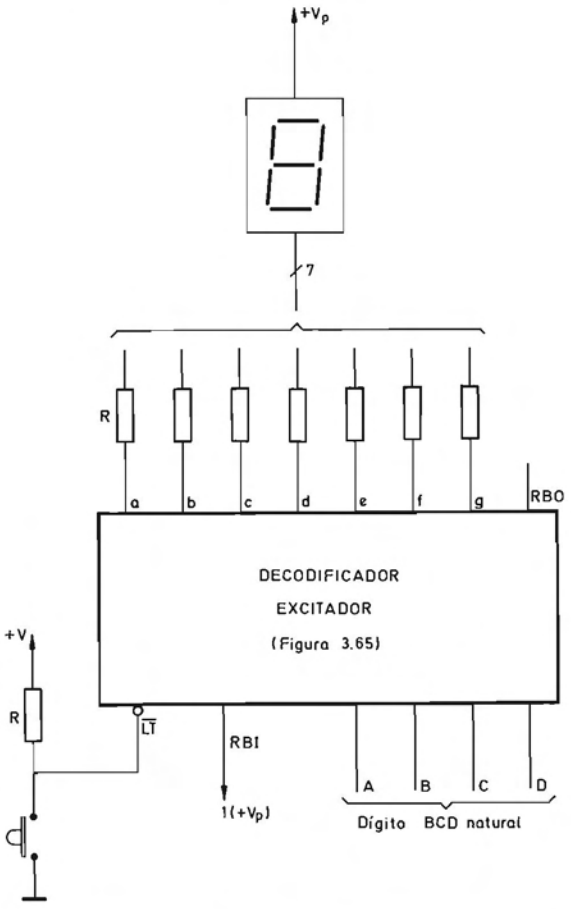


FIGURA 3.67.—**Esquema de un visualizador de un digito BCD natural realizado con el visualizador de la figura 3.66.**

b) Demultiplexores.

El circuito decodificador puede ser utilizado como demultiplexor. Un circuito demultiplexor es un sistema combinacional con una entrada de información D y m salidas, que posee además n entradas de selección tal que $2^n \geqslant m$. La información D se puede hacer aparecer en cualquiera de las salidas aplicando a las entradas de selección la combinación binaria adecuada.

En la figura 3.69 se representa una aplicación del decodificador decimal como demultiplexor de ocho canales. La combinación binaria presente en las entradas 0, 1 y 2 (equivalentes a 1, 2 y 4 respectivamente de la figura 3.52 *b*) hace que la información aplicada en la entrada 3 (equivalente a 8 en la figura 3.52*b*) aparezca en una de las salidas 0 a 7. Dicha combinación realiza, por lo tanto, la función *Y (G)* con las salidas 0 a 7 y, por ello, se le asigna el indicativo *G* 0/7. Por otra parte, las salidas 8 y 9 son el in-

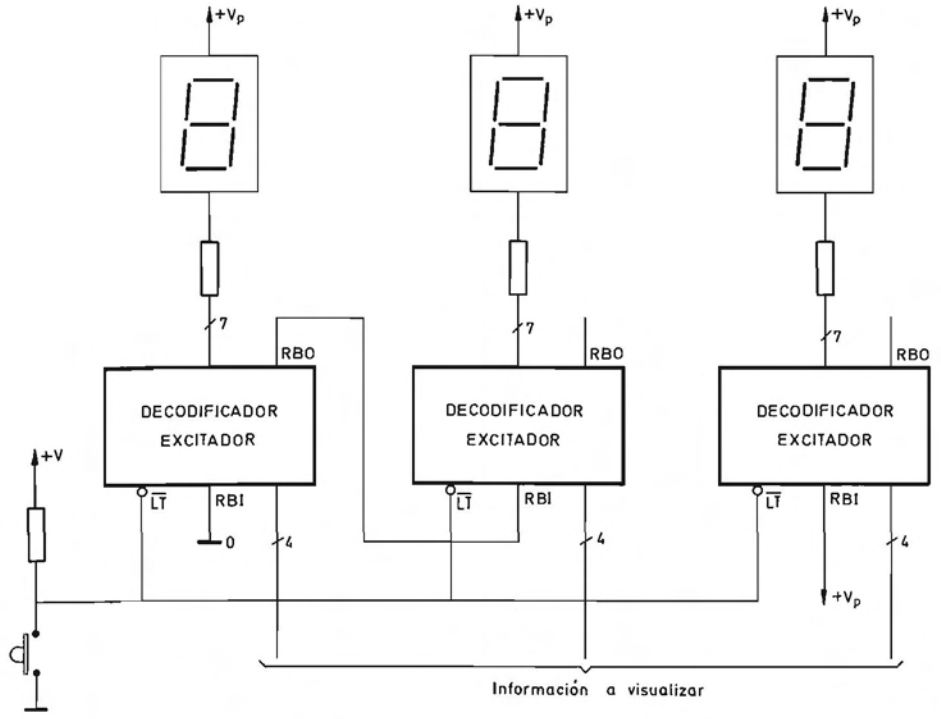


FIGURA 3.68.—**Esquema de tres visualizadores de siete segmentos con inhibición de la visualización de los ceros no significativos.**

verso de la 0 y de la 1 respectivamente. En la figura 3.69 se ha cambiado también la denominación (DMUX en lugar de BCD/DEC) del símbolo y la de los terminales de salida para adaptarlos a la nueva función del circuito.

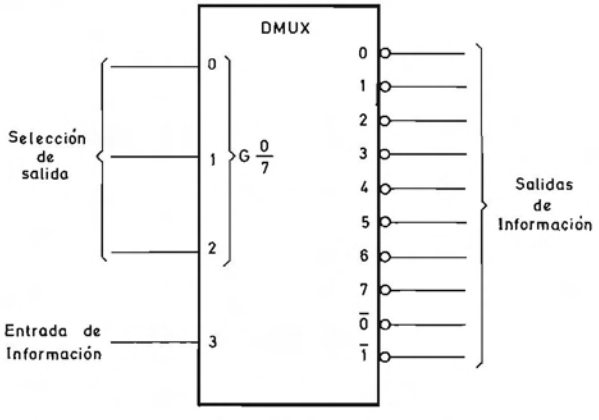


FIGURA 3.69.—**Decodificador decimal utilizado como demultiplexor.**

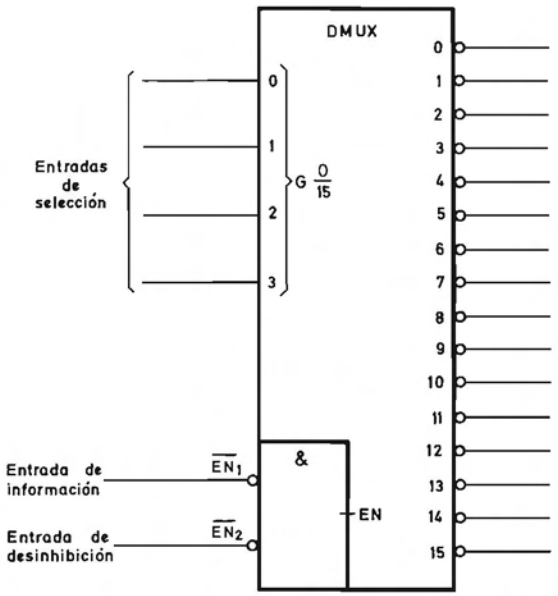


FIGURA 3.70.—**Decodificador hexadecimal utilizado como demultiplexor.**

El decodificador hexadecimal puede también ser utilizado como demultiplexor. En la figura 3.70 se representa el símbolo normalizado. Las entradas 1, 2, 4 y 8 de la figura 3.54 pasan a llamarse 0, 1, 2 y 3 respectivamente, y se les asignan los símbolos *G* 0/15 que indican que se realiza la función *Y (G)* de cada combinación con la entrada de información para obtener las salidas 0 a 15. Una de las entradas de desinhibición (en la figura 3.70 la $\overline{EN_1}$) es la entrada de información y la otra (en la figura 3.70 la $\overline{EN_2}$) es la de desinhibición propiamente dicha.

En la figura 3.71 se presenta una aplicación del demultiplexor de la figura 3.70. Las salidas del multiplexor se conectan a sendos contadores C0 a C15 que reciben los impulsos de una entrada común a todos ellos. Cada uno posee una entrada de inhibición G1 que según se encuentre en cero o uno permite o no que se realice el contaje de los impulsos. Cada entrada de inhibición se conecta a una salida del demultiplexor. Las entradas de selección *G* 0/15 de éste, seleccionan el canal de salida 0 a 15 en el que aparece el estado de la entrada de información. De esta forma se logra que el contador correspondiente realice o no el contaje de los impulsos según la entrada de información $\overline{EN_1}$ se encuentre en nivel cero o uno respectivamente (siempre y cuando $\overline{EN_2}$ se encuentre en nivel cero).

3.7.2 Codificadores

Los circuitos codificadores son sistemas combinacionales de 2^n entradas y n salidas realizados de tal forma que, cuando una sola de las entradas adopta un estado lógico determinado cero o uno, a la salida aparece la combinación binaria

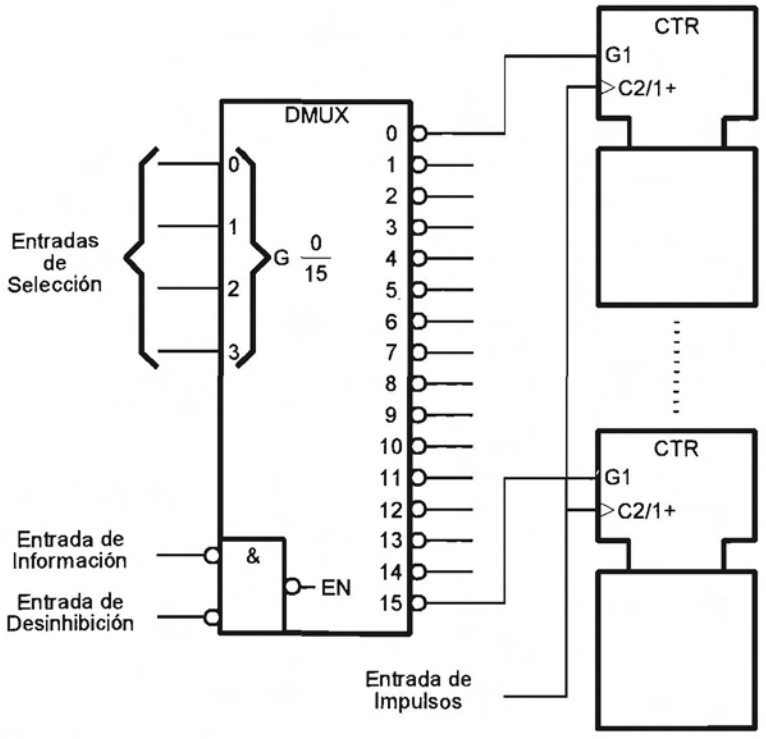


FIGURA 3.71.—**Ejemplo de aplicación del decodificador hexadecimal como demultiplexor.**

correspondiente al número decimal asignado a dicha entrada. Realizan por tanto la función inversa de los decodificadores.

Los circuitos codificadores pueden ser diseñados con prioridad o sin ella. En los codificadores sin prioridad, cuando más de una entrada toma el estado activo uno o cero, la combinación de salida posee tantos unos como hay en las correspondientes a la excitación de cada una de las entradas independientemente; por lo que en este codificador solamente debe de ser activa una entrada en cada instante.

Los codificadores con prioridad codifican la entrada activa de mayor valor decimal sin tener en cuenta las demás. Su tabla de verdad se representa en la tabla 3.22 para el caso en que $n = 3$ y las entradas son activas con un cero lógico.

En la tabla 3.22 se observa que, debido a que las tres variables binarias Q_2, Q_1 y Q_0 solamente tienen ocho combinaciones diferentes, no es posible discernir entre la situación en que ninguna de las entradas está activada y aquella en que es activa la de mayor peso. Para lograrlo se puede añadir al circuito una nueva salida que detecte alguna de ellas. Además, se puede disponer una entrada de inhibición que fuerce a todas las salidas a un estado determinado, independientemente del estado de las restantes entradas.

La tabla 3.23 representa la tabla de verdad de un codificador con prioridad de $n = 3$ que posee todas las salidas y entradas descritas. Cuando la entrada I de

A_0	A_1	A_2	A_3	A_4	A_5	A_6	A_7	Q_2	Q_1	Q_0
1	1	1	1	1	1	1	1	1	1	1
X	X	X	X	X	X	X	0	1	1	1
X	X	X	X	X	X	0	1	1	1	0
X	X	X	X	X	0	1	1	1	0	1
X	X	X	X	0	1	1	1	1	0	0
X	X	X	0	1	1	1	1	0	1	1
X	X	0	1	1	1	1	1	0	1	0
X	0	1	1	1	1	1	1	0	0	1
0	1	1	1	1	1	1	1	0	0	0

TABLA 3.22

I	A_0	A_1	A_2	A_3	A_4	A_5	A_6	A_7	Q_2	Q_1	Q_0	P_0	P_1
1	X	X	X	X	X	X	X	X	1	1	1	1	1
0	1	1	1	1	1	1	1	1	1	1	1	1	0
0	X	X	X	X	X	X	X	0	0	0	0	0	1
0	X	X	X	X	X	X	0	1	0	0	1	0	1
0	X	X	X	X	X	0	1	1	0	1	0	0	1
0	X	X	X	X	0	1	1	1	0	1	1	0	1
0	X	X	X	0	1	1	1	1	1	0	0	0	1
0	X	X	0	1	1	1	1	1	1	0	1	0	1
0	X	0	1	1	1	1	1	1	1	1	0	0	1
0	0	1	1	1	1	1	1	1	1	1	1	0	1

X Las entradas correspondientes pueden tomar los valores 0 o 1 lógicos.

TABLA 3.23

inhibición toma el valor lógico uno, todas las salidas adoptan dicho estado. Cuando I es cero y todas las demás entradas están en uno, sólo P_1 adopta el estado cero.

Cuando, estando I en cero, cualquiera de las restantes entradas se encuentra en dicho estado, la salida P_0 adopta el estado cero y en las salidas Q_2, Q_1 y Q_0 aparece la combinación binaria equivalente al complemento a siete de la entrada de mayor peso que se encuentra en estado cero.

En la figura 3.72 se representa el símbolo lógico de este circuito en sus versiones no normalizada (*a*) y normalizada (*b*). Este circuito constituye un ejemplo de cómo los símbolos normalizados permiten prescindir de las tablas de verdad. En primer lugar, el indicativo HPRI/BIN indica que se trata de un codificador con

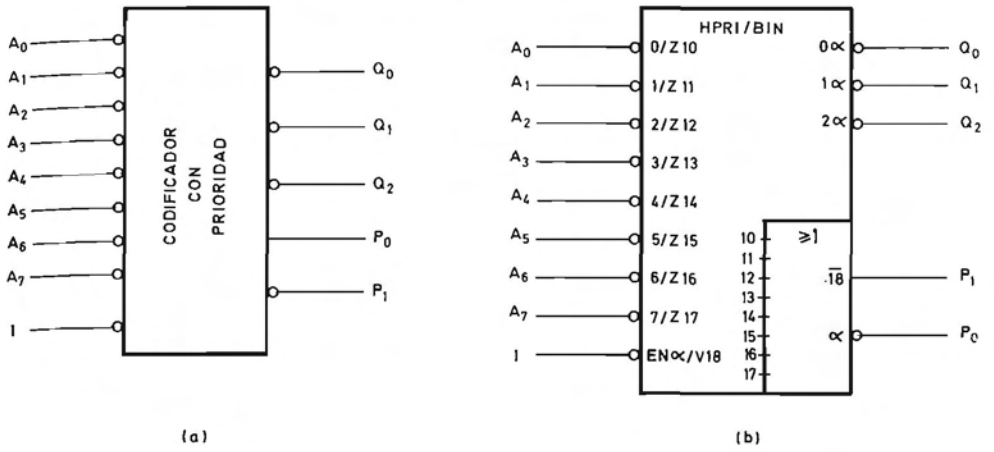


FIGURA 3.72.—**Símbolos lógicos de un codificador con prioridad: *a*) no normalizado; *b*) normalizado.**

prioridad con salida en binario natural. Las salidas codificadas Q_0, Q_1 y Q_2 reciben internamente las denominaciones 0α, 1α y 2α. Los números 0, 1 y 2 indican el peso de cada bit y el símbolo α establece que la salida correspondiente se pone a cero si ENα está en nivel cero. (El terminal externo se pone a uno debido a la existencia de una inversión).

Las entradas que se codifican se identifican con los números 0 a 7 correspondientes a su peso en decimal. Además, cada una de ellas posee el símbolo Z seguido de un número que indica que está unida a la correspondiente entrada de la puerta O que genera P_1 y P_0; por ejemplo, la entrada $Z14$ del circuito se une a la entrada 14 de la puerta O.

La entrada de inhibición se denomina ENα/V18 y posee un símbolo de inversión. Al mismo tiempo P_1 lleva el símbolo $\overline{18}$ y P_0 el símbolo α y está seguido de una inversión. Todo ello indica que P_1 y P_0 responden a las ecuaciones:

$$P_1 = \overline{A}_0 + \overline{A}_1 + \overline{A}_2 + \overline{A}_3 + \overline{A}_4 + \overline{A}_5 + \overline{A}_6 + \overline{A}_7 + I$$

$$P_0 = (\overline{A}_0 + \overline{A}_1 + \overline{A}_2 + \overline{A}_3 + \overline{A}_4 + \overline{A}_5 + \overline{A}_6 + \overline{A}_7)\,\overline{I}$$

La salida P_0 ($\overline{18}$) permite la realización con gran facilidad de un codificador de mayor número de entradas, porque adopta el estado cero cuando cualquiera de las entradas A_0 a A_7 es activa [ver Manual de prácticas de electrónica digital (diseño 4.6) de Editorial Marcombo].

Se sugiere al lector que realice el diseño de ese circuito con puertas como práctica.

En la figura 3.73 se representa el símbolo lógico normalizado de un codificador con prioridad de decimal a decimal codificado en binario natural (BCD natural). Se sugiere al lector que a partir de este símbolo deduzca la tabla de verdad de la tabla 3.24.

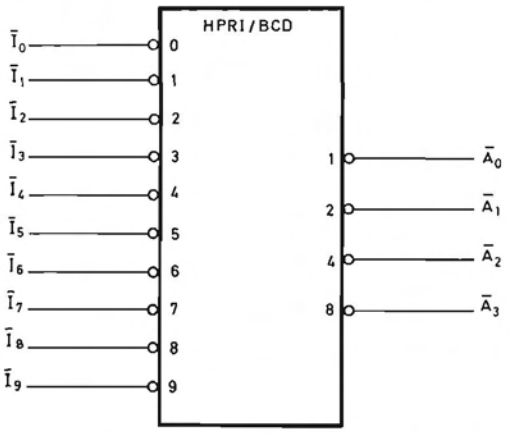


FIGURA 3.73.—**Símbolo lógico normalizado de un codificador con prioridad de decimal a decimal codificado en binario natural (BCD natural).**

$\overline{I_0}$	$\overline{I_1}$	$\overline{I_2}$	$\overline{I_3}$	$\overline{I_4}$	$\overline{I_5}$	$\overline{I_6}$	$\overline{I_7}$	$\overline{I_8}$	$\overline{I_9}$	$\overline{A_3}$	$\overline{A_2}$	$\overline{A_1}$	$\overline{A_0}$
1	1	1	1	1	1	1	1	1	1	1	1	1	1
X	X	X	X	X	X	X	X	X	0	0	1	1	0
X	X	X	X	X	X	X	X	0	1	0	1	1	1
X	X	X	X	X	X	X	0	1	1	1	0	0	0
X	X	X	X	X	X	0	1	1	1	1	0	0	1
X	X	X	X	1	0	1	1	1	1	1	0	1	0
X	X	X	1	0	1	1	1	1	1	1	0	1	1
X	X	X	0	1	1	1	1	1	1	1	1	0	0
X	X	0	1	1	1	1	1	1	1	1	1	0	1
X	0	1	1	1	1	1	1	1	1	1	1	1	0
0	1	1	1	1	1	1	1	1	1	1	1	1	1

TABLA 3.24.—**Tabla de verdad de un codificador decimal con prioridad.**

3.7.3 Multiplexores

Los multiplexores son circuitos combinacionales que poseen n canales de entrada, uno de salida y m entradas de selección, siendo $n = 2^m$ que permiten elegir cuál es el canal de entrada cuya información aparece en el de salida.

Existen diversas formas de realizar los multiplexores, que se estudian en apartados sucesivos.

3.7.3.1 Multiplexores realizados con puertas Y y puertas O. Una de las formas de realizar un multiplexor es mediante un número n de puertas Y conectadas a una única puerta O, que poseen m entradas de selección E (tal que $2^m = n$). Cada combinación binaria presente en las entradas de selección produce la aper-

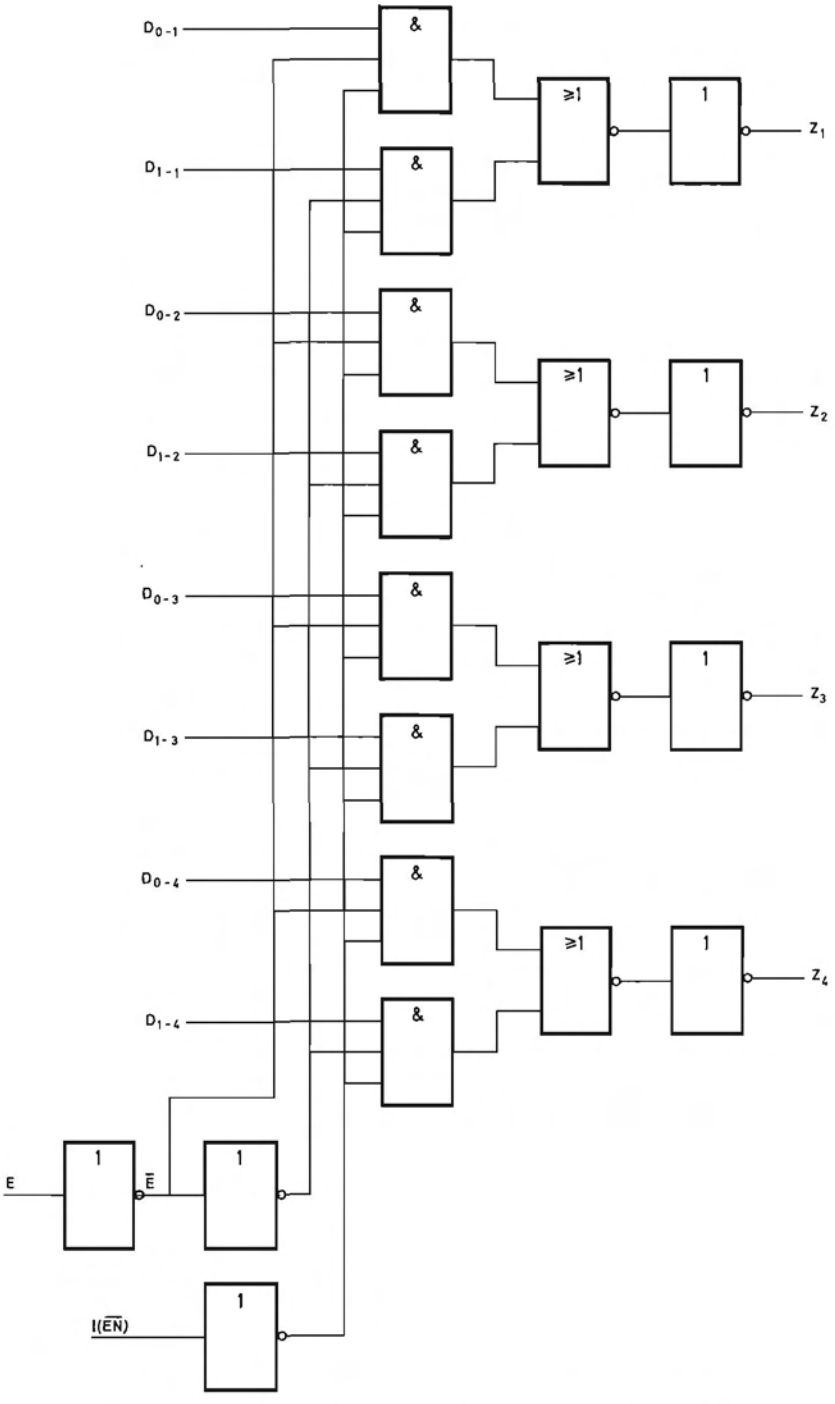


FIGURA 3.74.—**Esquema de un cuádruple multiplexor de dos canales.**

tura de una puerta Y únicamente, de tal forma que la información presente en la entrada D correspondiente puede pasar a la salida Z de la puerta O. Por ello, las entradas de selección reciben en la nueva simbología la denominación $G\frac{0}{2^m - 1}$. Eligiendo adecuadamente la combinación binaria presente en ellas, se logra que en la salida Z aparezca la información presente en cualquiera de las entradas D y el circuito actúa entonces como un conmutador electrónico.

Existen circuitos integrados que contienen cuatro multiplexores de dos canales, dos multiplexores de cuatro canales, un multiplexor de ocho canales y un multiplexor de dieciséis canales.

En la figura 3.74 se representa el esquema lógico de un cuádruple multiplexor de dos canales que posee una entrada de selección E ($G1$) común a todos ellos. Este circuito posee además una entrada de inhibición/desinhibición I ($\overline{EN}$) que cuando se encuentra en nivel uno pone a cero todas las salidas de las puertas Y y fuerza las salidas Z al estado lógico cero, independientemente de la información aplicada en la entrada de selección E y en las entradas de información D. En la figura 3.75 se representa el símbolo lógico no normalizado (a) y normalizado (b) de este multiplexor. Las entradas EN y $G1$ del símbolo normalizado se colocan en la parte superior porque son comunes a los cuatro multiplexores. En los canales de entrada se colocan los símbolos $\overline{1}$ y 1 que indican que la información del primero aparece en la salida si $G1$ está en nivel cero y por el contrario es la del segundo la que aparece si $G1$ está en nivel uno.

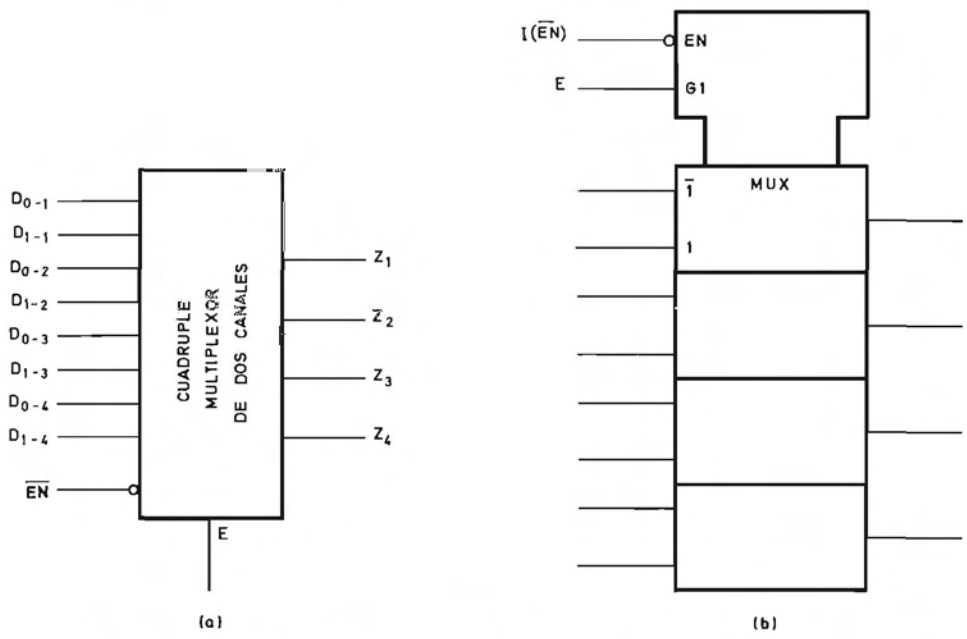


FIGURA 3.75.—**Símbolos lógicos del cuádruple multiplexor de dos canales de la figura 3.74: *a*) no normalizado; *b*) normalizado.**

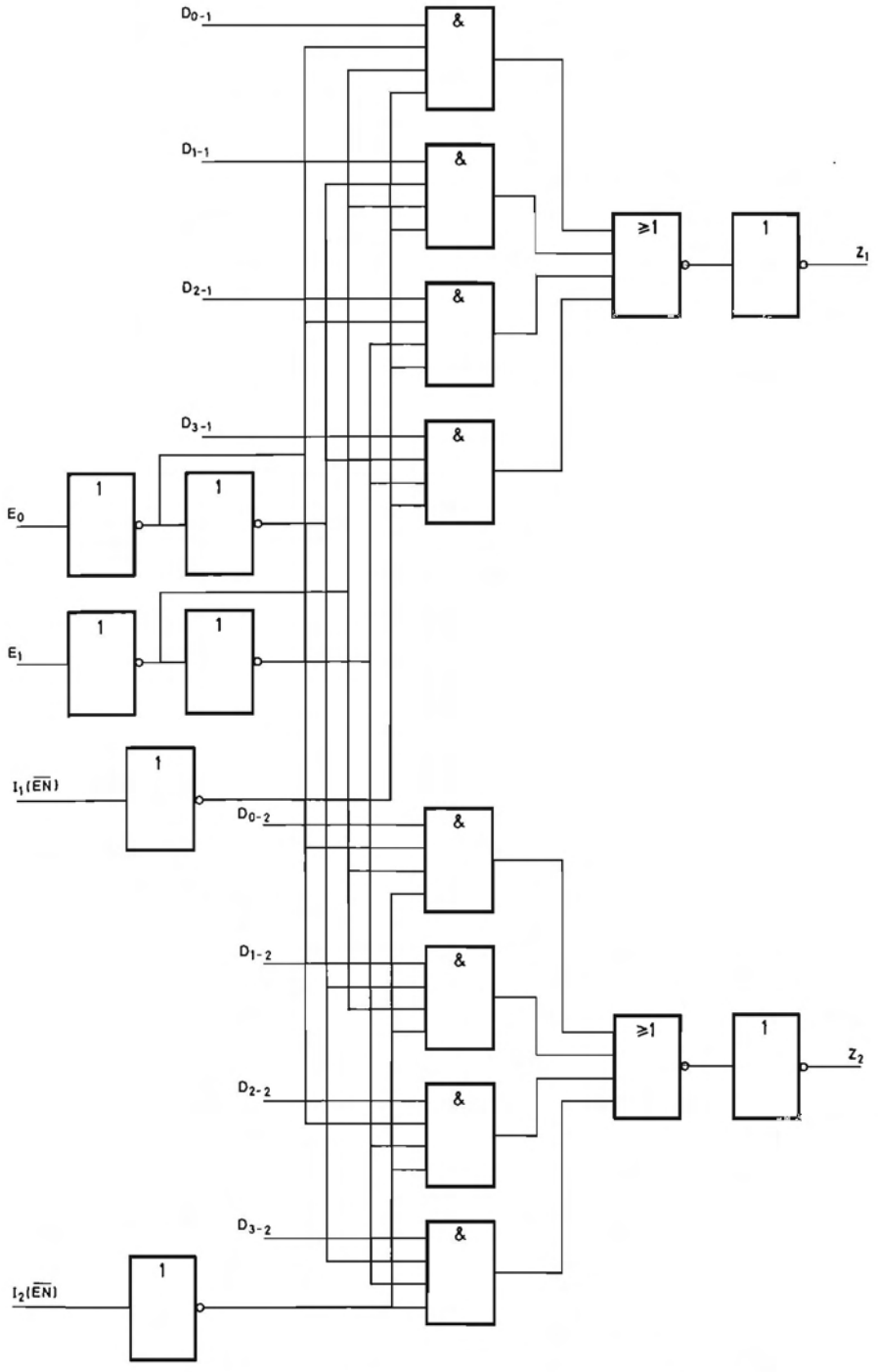


FIGURA 3.76.—**Esquema lógico de un doble multiplexor de cuatro canales.**

En la figura 3.76 se representa el esquema lógico de un doble multiplexor de cuatro canales con entradas de selección E_0 y E_1 (G 0/3) comunes a todos y entradas de inhibición I_1 e I_2 ($\overline{EN}$) independientes. En la figura 3.77 se representan los símbolos lógicos no normalizado y normalizado. En este último, en la parte superior se colocan solamente las entradas de selección E_1 (G 0/3) porque cada multiplexor posee una entrada de desinhibición (EN) independiente.

En la figura 3.78 se representa el esquema de un multiplexor de ocho canales con tres variables de selección E_0, E_1 y E_2 (G 0/7) y una variable de inhibición/desinhibición I ($\overline{EN}$), que si se encuentra en nivel uno pone a nivel cero la salida. En la figura 3.79 se representan los símbolos lógicos no normalizado (a) y normalizado (b) correspondientes a este multiplexor. En el caso de que la entrada EN ponga en tercer estado las salidas, se coloca el símbolo ▽ en éstas (figura 3.80).

La figura 3.81 representa un multiplexor de dieciséis canales con cuatro variables de selección E_0, E_1, E_2 y E_3 (G 0/15) que posee una entrada de inhibición I ($\overline{EN}$) que si está en estado uno, pone a cero la salida. En la figura 3.82 se representan los símbolos lógicos correspondientes.

En la figura 3.83 se representa el símbolo de un séxtuple multiplexor de dos canales de entrada A y B que puede presentar en su salida uno u otro o el producto de ambos en forma directa o inversa. Este circuito constituye un ejemplo claro de las ventajas de utilización de los símbolos normalizados. Las variables de selección de entrada E_0 y E_1 pueden encontrarse en los cuatro estados posibles, 0, 1, 2 y 3.

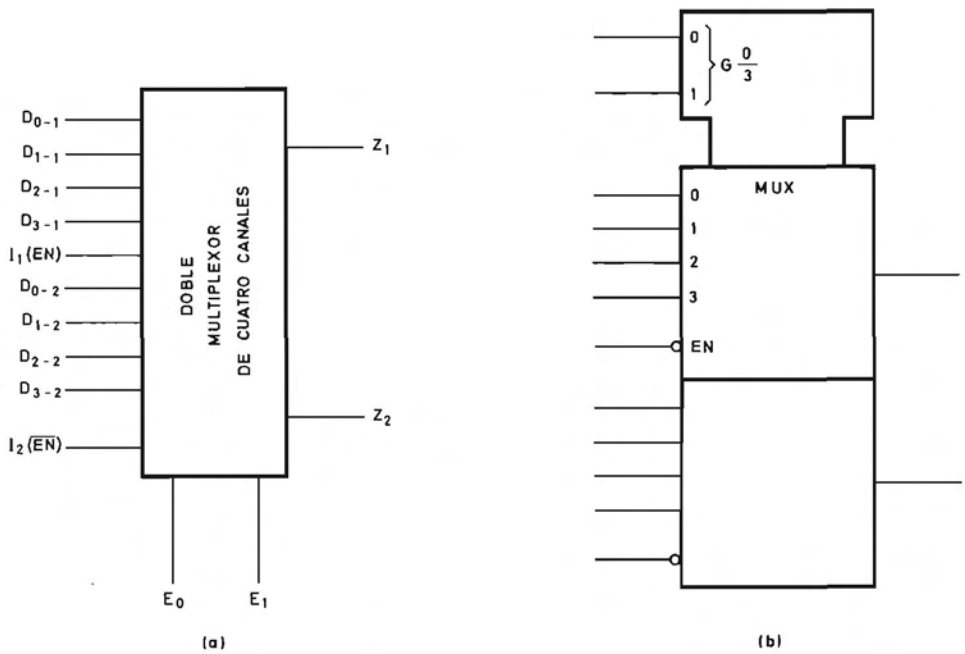


FIGURA 3.77.—**Símbolos lógicos del doble multiplexor de cuatro canales de la figura 3.76: *a*) no normalizado; *b*) normalizado.**

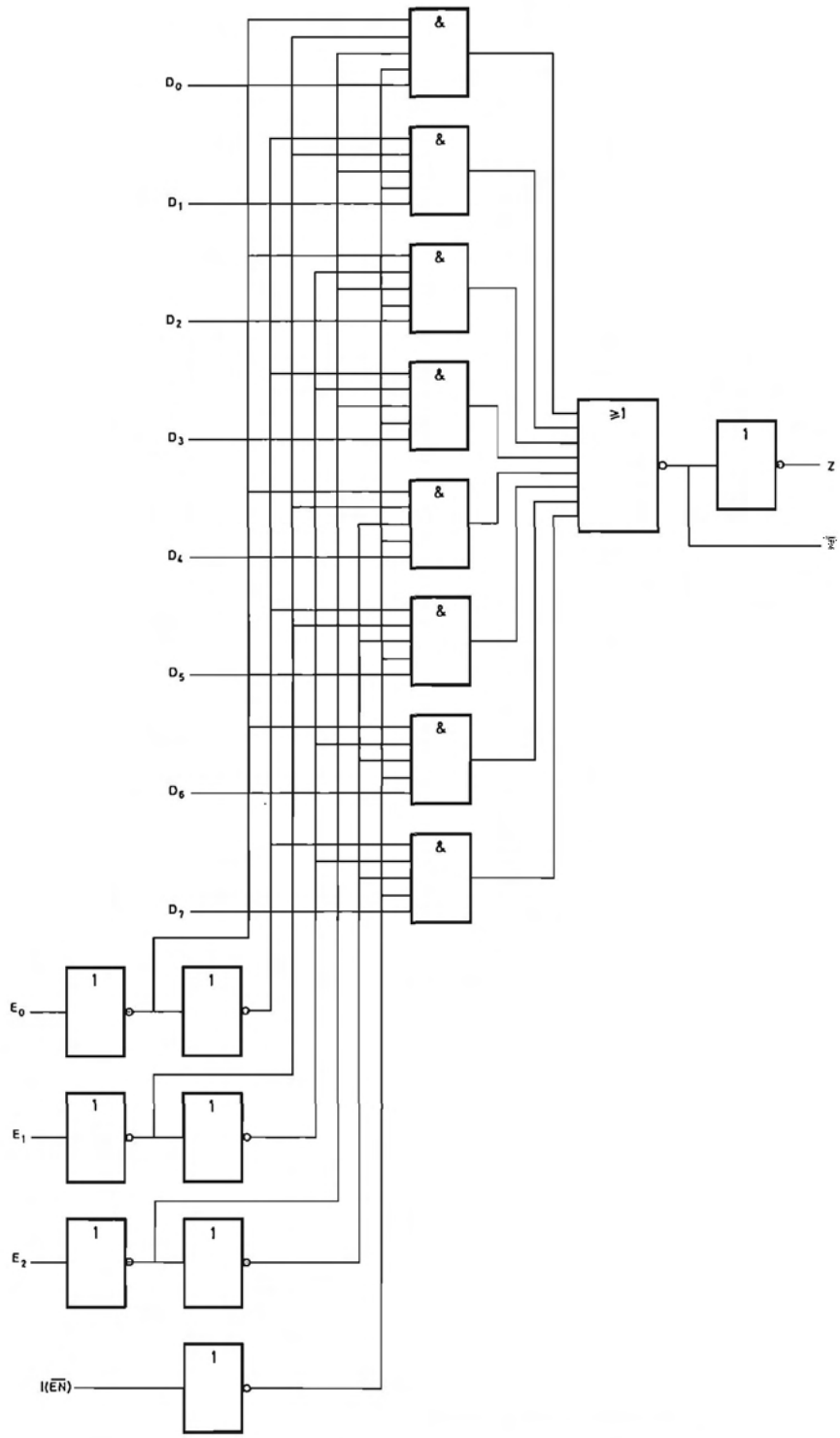


FIGURA 3.78.—**Esquema lógico de un multiplexor de ocho canales.**

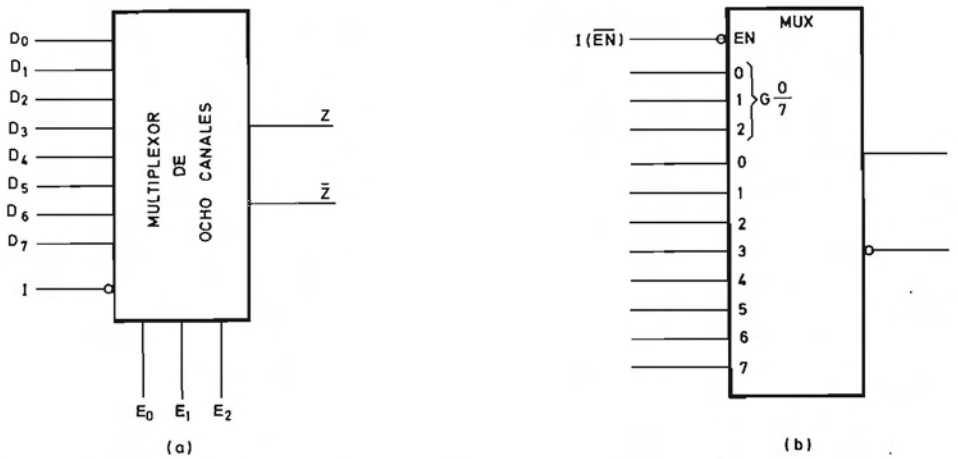


FIGURA 3.79.—**Símbolos lógicos del multiplexor de ocho canales de la figura 3.78:** *a*) **no normalizado;** *b*) **normalizado.**

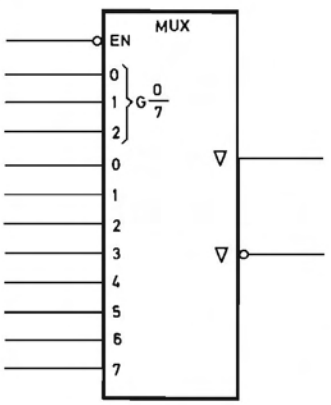


FIGURA 3.80.—**Símbolo lógico normalizado de un multiplexor de ocho canales con salidas de tres estados.**

Estas variables actúan mediante la relación de dependencia Y (G) con las entradas del multiplexor A y B y su producto. La asignación del 0 a la entrada A indica que este canal se selecciona cuando E_0 y E_1 se encuentran ambas en estado cero. La combinación E_1, E_0 = 11 (3) no selecciona ningún canal de entrada y hace que la salida se ponga a nivel cero.

La salida está afectada además simultáneamente por la entrada $N4$ y la variable interna $EN5$. Cuando la variable $N4$ se encuentra en nivel uno la salida se invierte siempre y cuando E_0 y E_1 se encuentren en 00, 01, 10.

La variable $EN5$ es igual a: $EN5 = \overline{COMP \cdot E_1 \cdot E_0}$. Por lo tanto, si E_1 E_0 = 11 (3) y $COMP$ ($N4$) se encuentra en nivel uno, resulta $EN5$ = 0 y la salida se pone en tercer estado.

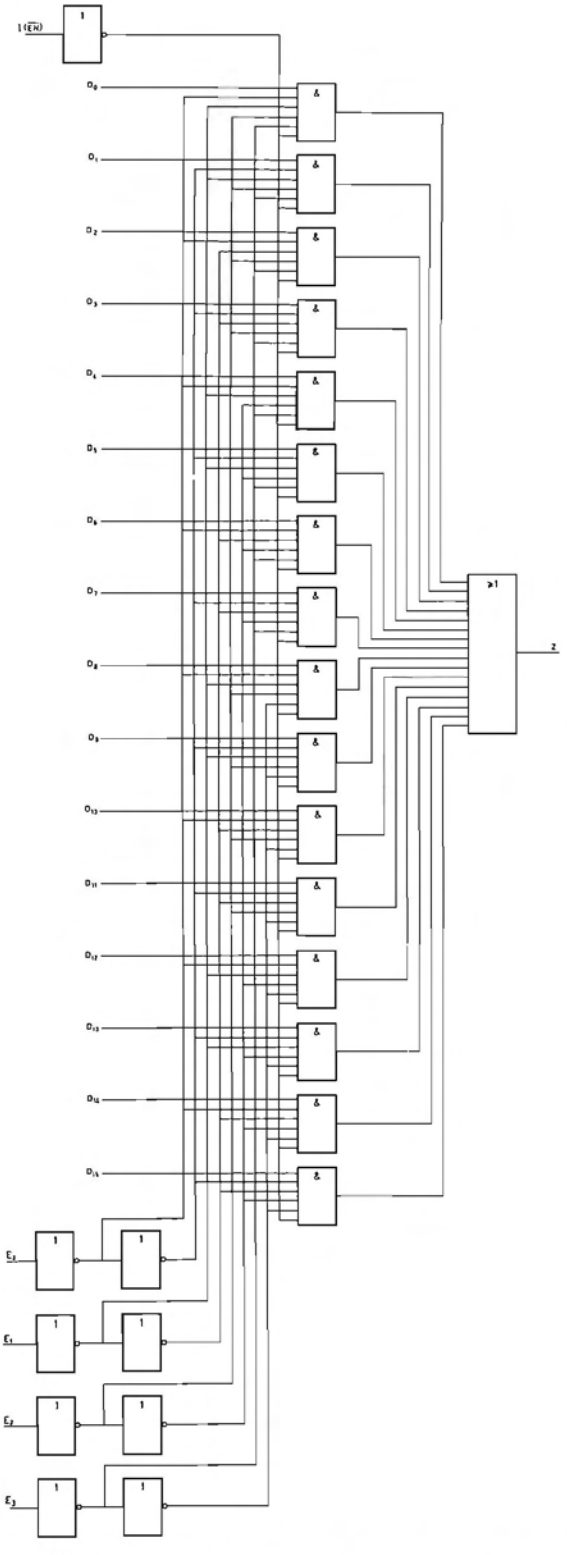


FIGURA 3.81.—**Esquema lógico de un multiplexor de dieciséis canales.**

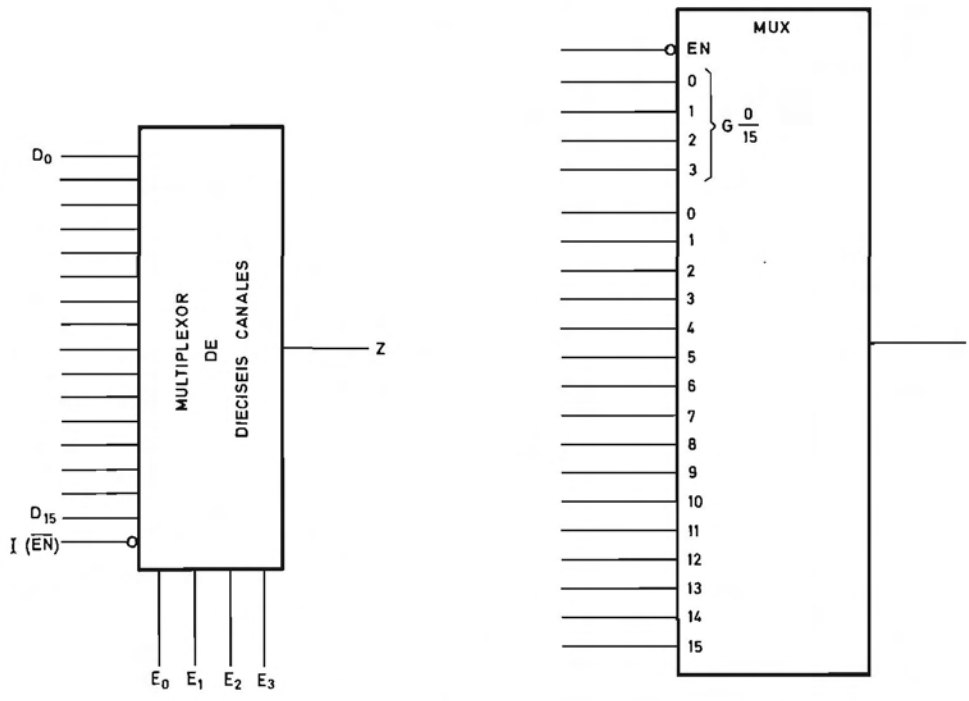


FIGURA 3.82.—**Símbolos lógicos del multiplexor de dieciséis canales de la figura 3.81:** *a*) **no normalizado;** *b*) **normalizado.**

El lector puede deducir la tabla de verdad de este multiplexor, que es la indicada en la tabla 3.25 y comprobar que la información contenida en ella se deduce directamente del símbolo normalizado de la figura 3.83, lo que la hace superflua.

Como ejemplo, en la tabla 3.26 se representa también la tabla de verdad del multiplexor de ocho canales de entrada de las figuras 3.78 y 3.79. El lector puede com-

COMP	E_1	E_0	Salida
0	0	0	A
0	0	1	B
0	1	0	AB
0	1	1	0
1	0	0	$\overline{A}$
1	0	1	$\overline{B}$
1	1	0	$\overline{AB}$
1	1	1	Tercer estado

TABLA 3.25

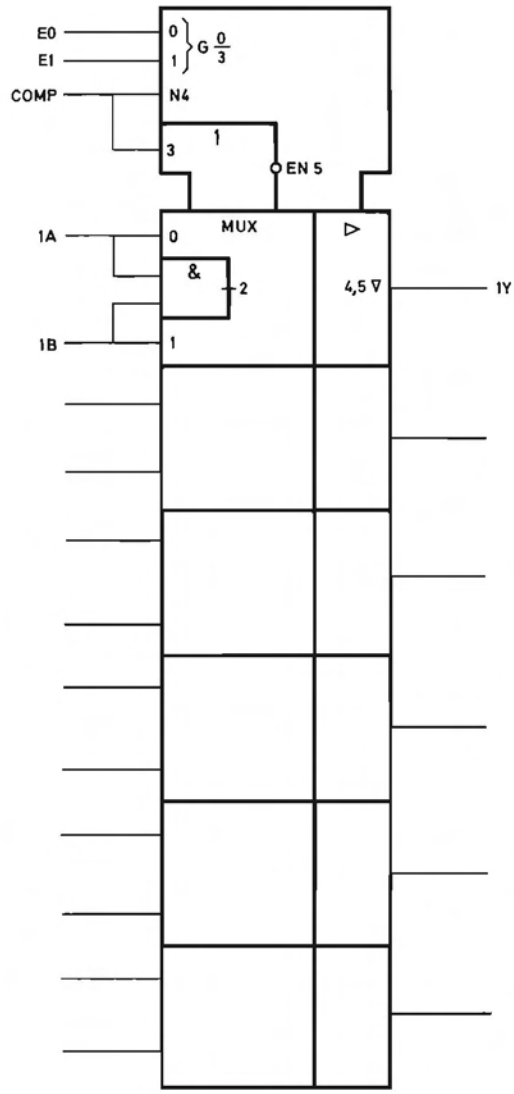


FIGURA 3.83.—**Símbolo de un séxtuple multiplexor de dos canales con función Y incorporada.**

probar que esta tabla se deduce directamente del símbolo lógico normalizado de la figura 3.79*b*.

3.7.3.2 Multiplexores realizados con puertas de tres estados. Las puertas de tres estados son aquellas cuya salida no solamente puede encontrarse en estado cero o uno sino que puede estarlo también en un tercer estado en el que la impe-

$\overline{EN}$	$G\frac{0}{7}$													
I_1	E_2	E_1	E_0	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	Z	$\overline{Z}$
1	X	X	X	X	X	X	X	X	X	X	X	0	1
0	0	0	0	0	X	X	X	X	X	X	X	0	1
0	0	0	0	1	X	X	X	X	X	X	X	1	0
0	0	0	1	X	0	X	X	X	X	X	X	0	1
0	0	0	1	X	1	X	X	X	X	X	X	1	0
0	0	1	0	X	X	0	X	X	X	X	X	0	1
0	0	1	0	X	X	1	X	X	X	X	X	1	0
0	0	1	1	X	X	X	0	X	X	X	X	0	1
0	0	1	1	X	X	X	1	X	X	X	X	1	0
0	1	0	0	X	X	X	X	0	X	X	X	0	1
0	1	0	0	X	X	X	X	1	X	X	X	1	0
0	1	0	1	X	X	X	X	X	0	X	X	0	1
0	1	0	1	X	X	X	X	X	1	X	X	1	0
0	1	1	0	X	X	X	X	X	X	0	X	0	1
0	1	1	0	X	X	X	X	X	X	1	X	1	0
0	1	1	1	X	X	X	X	X	X	X	0	0	1
0	1	1	1	X	X	X	X	X	X	X	1	1	0

X Las entradas correspondientes pueden tomar los valores 0 o 1 lógicos.

TABLA 3.26.—**Tabla de verdad del multiplexor de ocho canales.**

dancia entre ella y los dos terminales de la tensión de alimentación es muy elevada (de varios megohmios).

En el capítulo 5 se estudia la realización de las puertas de tres estados en diversas tecnologías.

Las puertas de tres estados más sencillas son las seguidoras que se representan en la figura 3.84. Estas puertas poseen una entrada de información *D*, una entrada de desinhibición (*EN*) del tercer estado de la salida y una salida. Cuando *EN* se encuentra en nivel uno, la salida sigue a la entrada *D*. Por el contrario cuando *EN* se encuentra en nivel cero, la salida se pone en el tercer estado de alta impedancia.

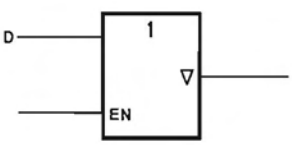


FIGURA 3.84.—**Símbolo lógico de una puerta seguidora de tres estados.**

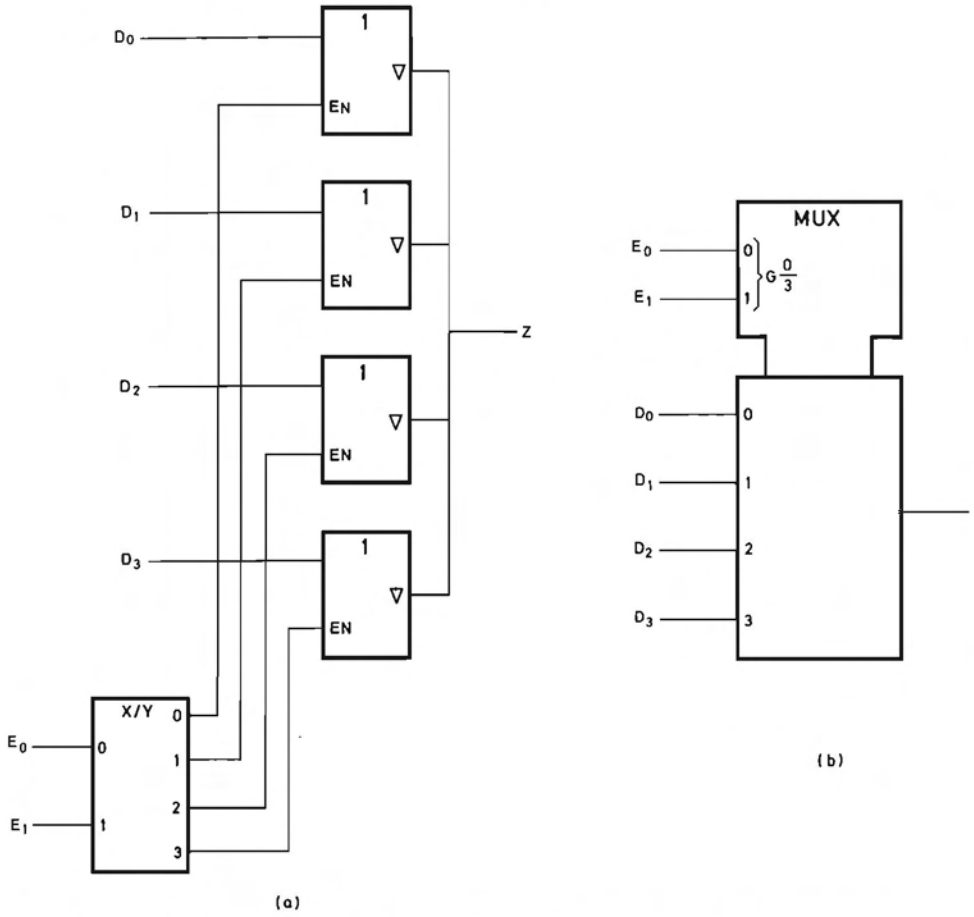


FIGURA 3.85.—**Multiplexor de cuatro canales sin entrada de desinhibición realizado con puertas de tres estados: *a*) Esquema lógico; *b*) Símbolo lógico normalizado.**

En la figura 3.85*a* se representa un multiplexor de cuatro canales realizado con otras tantas puertas seguidoras de tres estados, cuyas salidas se conectan entre sí para obtener la salida del multiplexor. Las entradas de selección E_0 y E_1, se conectan a las entradas de un decodificador de uno entre cuatro, cuyas cuatro salidas se conectan a su vez a las cuatro entradas de desinhibición (*EN*) de las cuatro puertas seguidoras. Se logra de esta forma que en cada instante sólo aparezca en la salida *Z* la información presente en la entrada *D* de la puerta cuya entrada *EN* está en nivel uno. En la figura 3.85*b* se representa el símbolo normalizado.

Dotando al decodificador de una entrada de desinhibición (*EN*) que ponga a cero todas las salidas cuando se encuentra en nivel cero, se obtiene un multiplexor con salida de tres estados. En la figura 3.86*a* y *b* se representan el esquema y el símbolo normalizado de un multiplexor realizado con puertas de tres estados y un decodificador que posee dos entradas de desinhibición I_1 e $\overline{I}_2$ cuyo producto lógi-

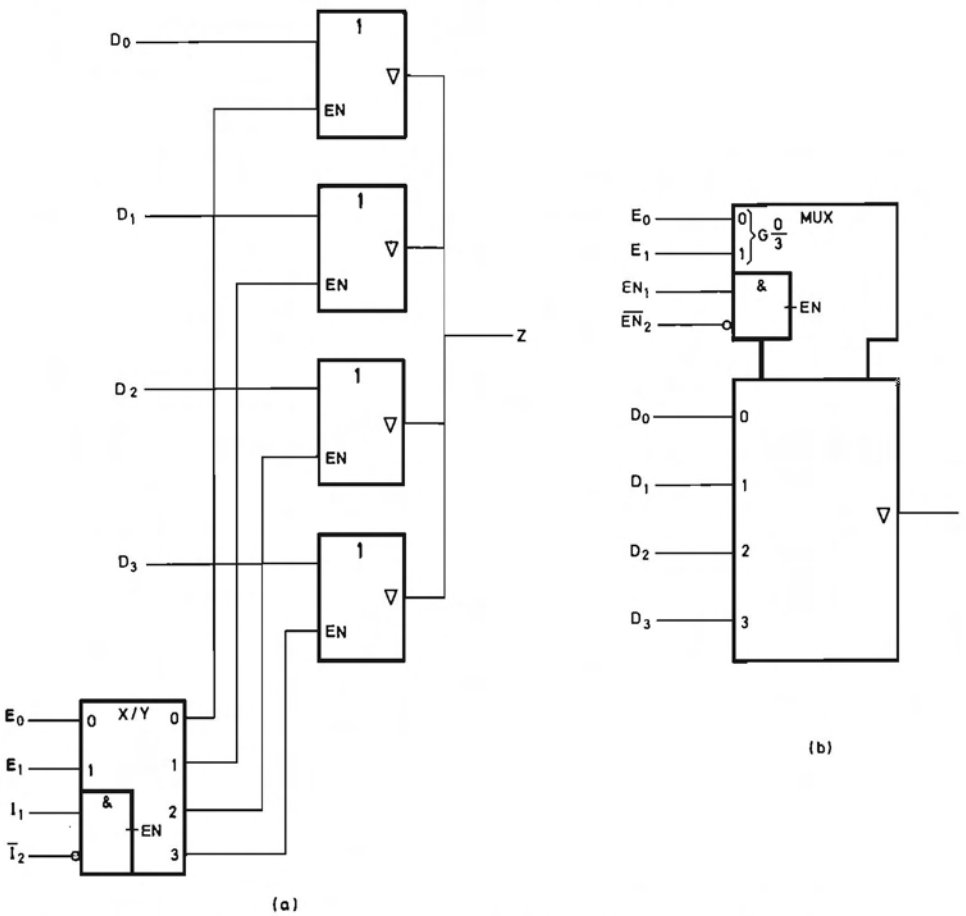


FIGURA 3.86.—**Multiplexor de cuatro canales con entradas de inhibición que controlan el tercer estado de la salida, realizado con puertas de tres estados: *a*) Esquema lógico; *b*) Símbolo lógico normalizado.**

co genera *EN*. Para que el multiplexor esté desinhibido es necesario que simultáneamente I_1 esté en uno e I_2 en cero. Mediante dos multiplexores como el de la figura 3.86 es posible realizar el multiplexor de ocho canales representado en la figura 3.87 que el lector puede comprender fácilmente analizándola con detenimiento.

3.7.3.3 Multiplexores realizados con interruptores. Otra forma de realizar los multiplexores es mediante la conexión de interruptores electrónicos. Un interruptor electrónico es un circuito que posee una entrada, una salida y una variable de control. Cuando la variable de control está en un cierto estado lógico, la resistencia entre el terminal de entrada y el de salida es muy baja (del orden de algunos ohmios) y cuando está en el estado contrario es muy alta (aproximadamente de algunos megohmios).

Existen diversas formas de representar los interruptores electrónicos de las que

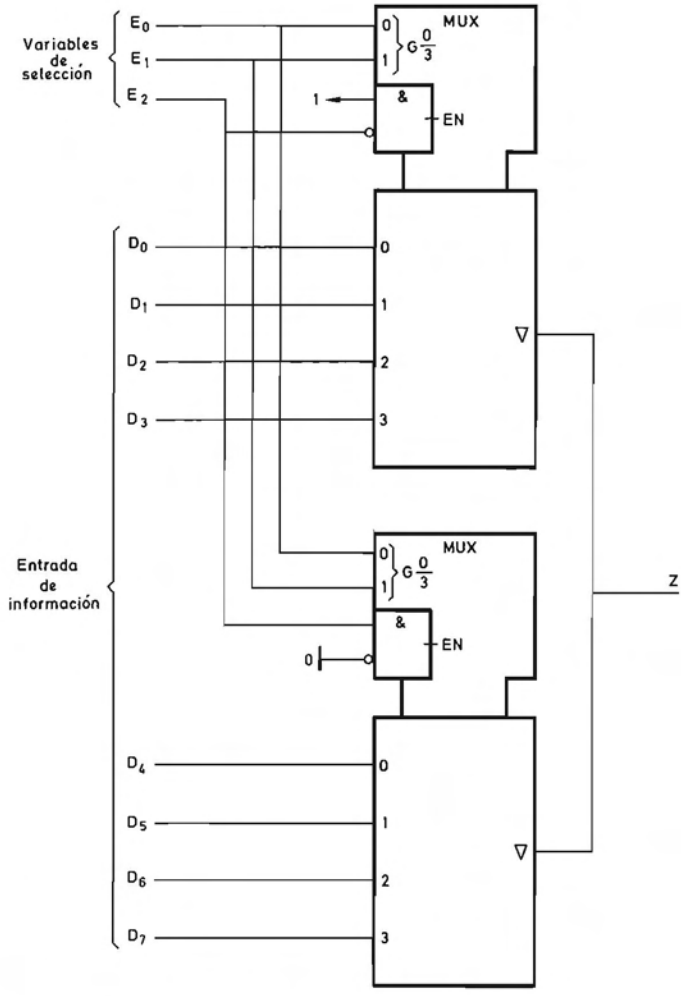


FIGURA 3.87.—**Multiplexor de ocho canales realizado con dos multiplexores como el de la figura 3.86.**

se indican tres en la figura 3.88. En particular es necesario destacar la representada en la figura 3.88*c* que corresponde a la versión normalizada por la Comisión Electrotécnica Internacional. La variable de control recibe la denominación de X que indica precisamente que controla la impedancia entre los otros dos terminales. Para indicar la relación existente se coloca en ambos terminales el dígito 1 y el símbolo X está seguido también de dicho dígito.

En la figura 3.89*a* se representa el esquema de un multiplexor de dos canales de entrada realizado con dos interruptores electrónicos. Los dos terminales de la derecha de cada interruptor se unen entre sí y constituyen la variable de salida del multiplexor. Los dos terminales de la izquierda constituyen las dos variables de entrada del multiplexor. La variable de selección se conecta de forma directa a la en-

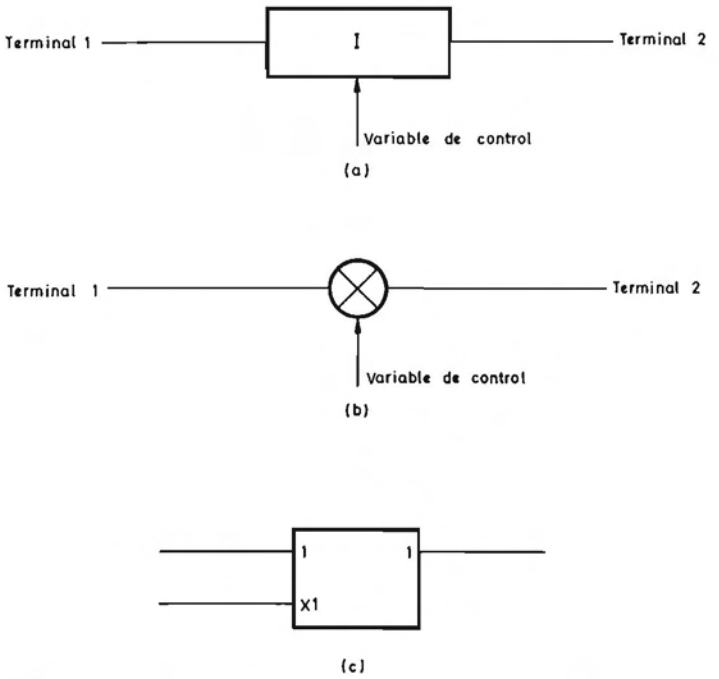


FIGURA 3.88.—**Símbolos lógicos de un interruptor electrónico bidireccional.**

trada $X1$ del interruptor inferior e invertida a la del superior. En la figura 3.89*b* se representa el símbolo normalizado en el que la entrada de selección se denomina $X\ \frac{0}{1}$ y los terminales de entrada 0 y 1. Ello indica que cuando $X\ \frac{0}{1}$ se pone a nivel cero se conecta el terminal de salida al de entrada 0 y por el contrario cuando se pone a uno se conecta al de entrada 1.

La figura 3.90*a* representa un demultiplexor de dos canales, cuyo funcionamiento se recomienda que analice el lector. En la figura 3.90*b* se representa el símbolo normalizado.

Precisamente el hecho de que los dos terminales de cada interruptor sean idénticos hace que los dos esquemas de las figuras 3.89*a* y 3.90*b* lo sean también y que, por lo tanto, constituyan un único circuito que puede ser multiplexor o demultiplexor según por donde se apliquen las señales de entrada y se obtengan las de salida. Se puede, por lo tanto, utilizar un único símbolo lógico para representar a ambos sustituyendo los indicativos MUX y DMUX por el de MDX (figura 3.91).

La principal característica de los circuitos que se acaban de describir es precisamente la de la reversibilidad unida a la de que los interruptores pueden dejar o impedir el paso tanto de señales digitales como analógicas. Por ello, estos circuitos se denominan multiplexores/demultiplexores analógicos.

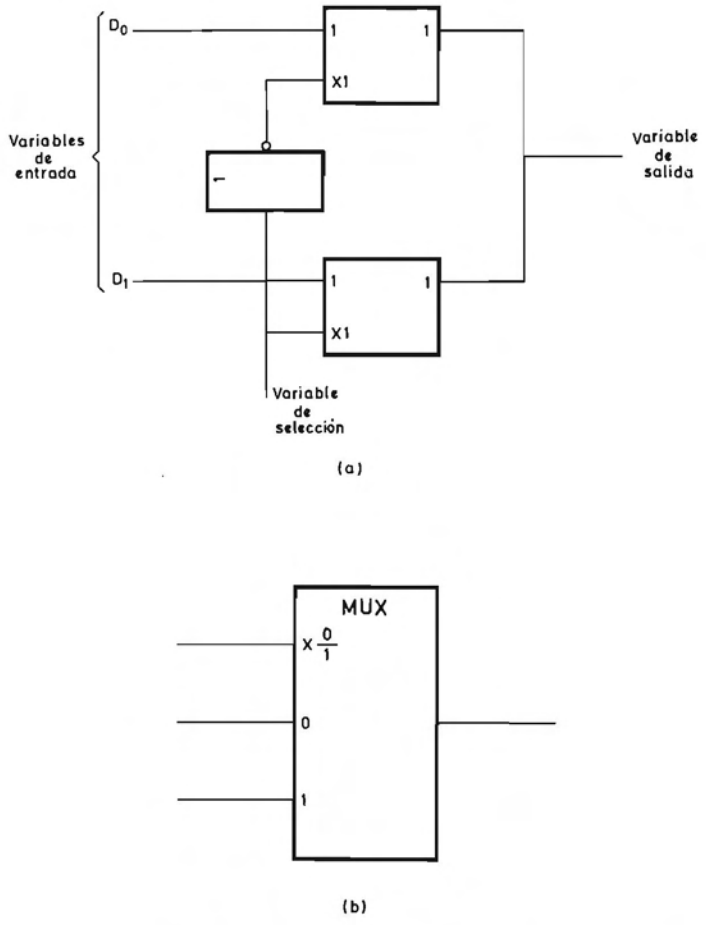


FIGURA 3.89.—**Multiplexor de dos canales realizado con sendos interruptores electrónicos: *a*) esquema lógico; *b*) símbolo normalizado.**

Los multiplexores/demultiplexores analógicos pueden poseer un número cualquiera de canales, en general potencia de dos, y estar dotados de una o más entradas de desinhibición que impidan el cierre del interruptor seleccionado mediante las variables de selección.

En la figura 3.92*a* se representa el esquema de un multiplexor/demultiplexor de cuatro canales. La selección de las variables de control de los interruptores se realiza mediante un decodificador que en este caso es de uno entre cuatro, cuya forma de funcionamiento se analizó en el apartado 3.7.1.

El decodificador de la figura 3.92*a* está dotado de una entrada de desinhibición (*EN*) que, si se encuentra en estado cero, inhibe la activación de la salida correspondiente a la combinación presente en las variables de selección.

En la figura 3.92*b* se representa el símbolo lógico normalizado, en el que se

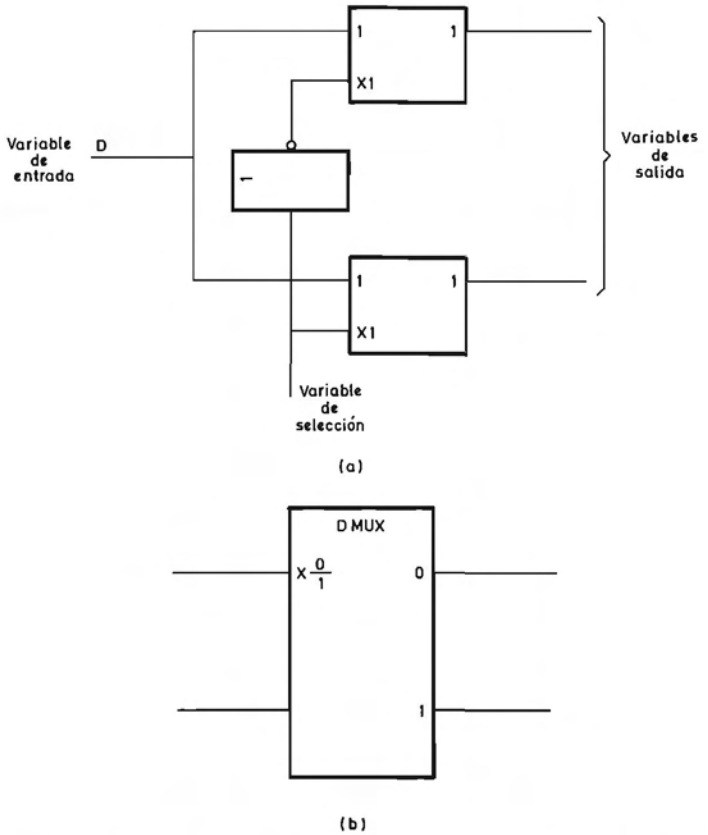


FIGURA 3.90.—**Demultiplexor de dos canales realizado con sendos interruptores electrónicos: *a*) esquema lógico; *b*) símbolo lógico normalizado.**

debe observar que la variable de desinhibición recibe la denominación G4 y las de selección 4 $X\ \frac{0}{3}$. Los digitos $\frac{0}{3}$ indican que las dos entradas de selección permiten elegir el canal de entrada/salida entre el 0, el 1, el 2 y el 3. El dígito 4 indica

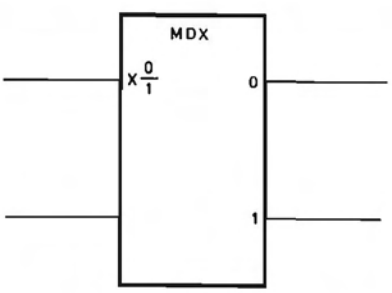


FIGURA 3.91.—**Símbolo lógico normalizado de un multiplexor/demultiplexor de dos canales.**

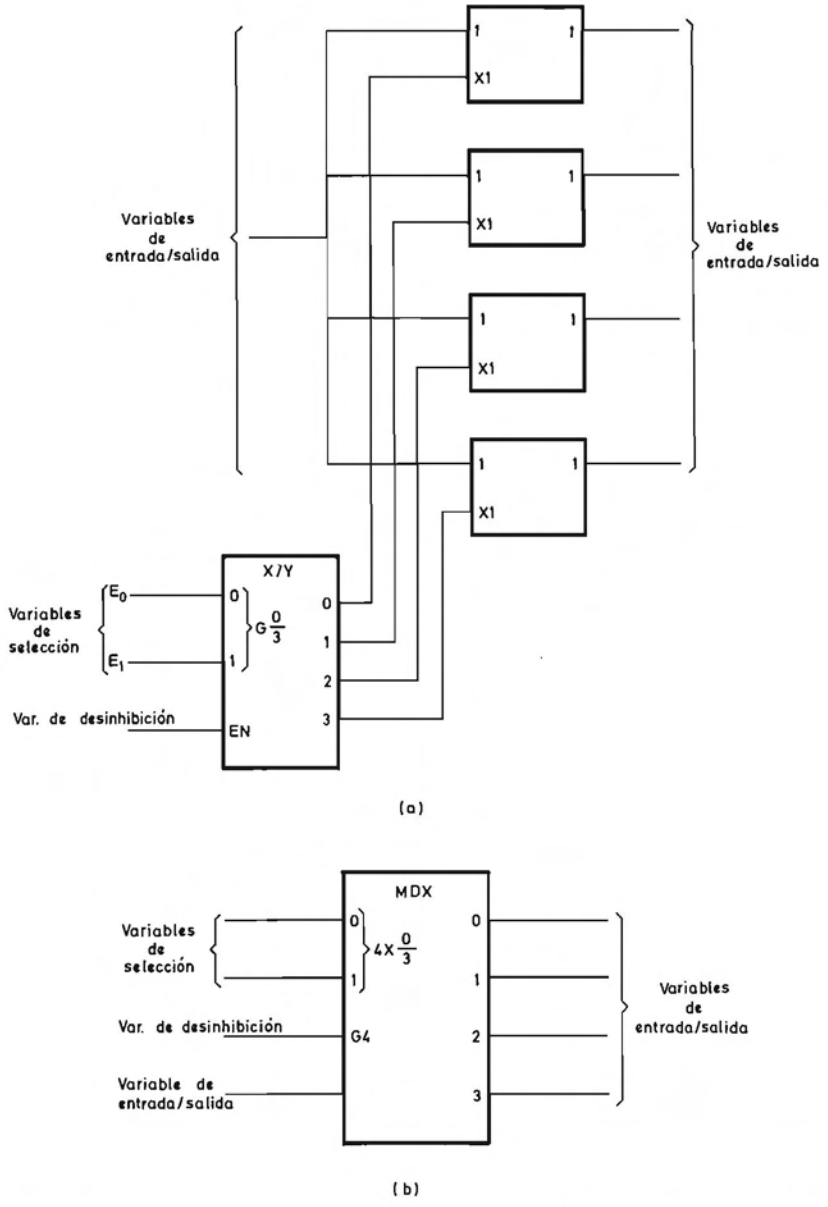


FIGURA 3.92.—**Multiplexor de cuatro canales con entrada de desinhibición:** *a*) **esquema lógico;** *b*) **símbolo lógico normalizado.**

que la entrada *G4* inhibe la acción realizada mediante las variables de selección.

En la práctica, la mejor manera de realizar los interruptores electrónicos es mediante transistores MOS y, por ello, los multiplexores/demultiplexores se realizan en tecnología CMOS estudiada en el apartado 5.4.4.4.2.3.

3.7.3.4 Aplicaciones de los multiplexores. Una de las aplicaciones más usuales de los circuitos multiplexores es la de enviar a un solo canal la información procedente de varios canales, seleccionando en cada instante el canal mediante la combinación binaria aplicada a las entradas E.

En la figura 3.93 se representa el esquema de un multiplexor de ocho canales con tres variables de selección E_0, E_1 y E_2. La variable de inhibición I pone a cero la salida cuando se encuentra en nivel uno ($EN = 0$).

La conexión en paralelo de multiplexores con salida de tres estados permite obtener un multiplexor de mayor número de canales. En la figura 3.94 se muestra el circuito correspondiente a un multiplexor de dieciséis canales realizado con dos multiplexores de ocho canales. Para seleccionar los dieciséis canales se necesitan cuatro variables de selección E_0 a E_3. Las tres primeras E_0 a E_2 son comunes a los dos multiplexores (figura 3.94) y la cuarta E_3 se conecta de forma directa a uno de ellos e invertida al otro. Si E_3 se encuentra en nivel cero se pone en tercer estado la salida del multiplexor superior y si se encuentra en nivel uno le sucede lo propio al multiplexor inferior.

Otra aplicación muy interesante de los multiplexores es la generación de funciones lógicas.

Mediante un multiplexor de n variables de selección (2^n canales) se puede generar cualquier función de $n + 1$ variables. Esta afirmación se demuestra con un ejemplo.

En la figura 3.95 se indica la realización de la función de la tabla 3.27 con un multiplexor de dos variables de selección.

Si las variables a y b se conectan a las entradas de selección, cada combinación de ellas hará que a la salida del multiplexor aparezca la información presente en la entrada D correspondiente. Por tanto, ha de seleccionarse la información que debe aparecer en las entradas D del multiplexor, para lo cual se puede hacer uso de la representación tabular de Karnaugh.

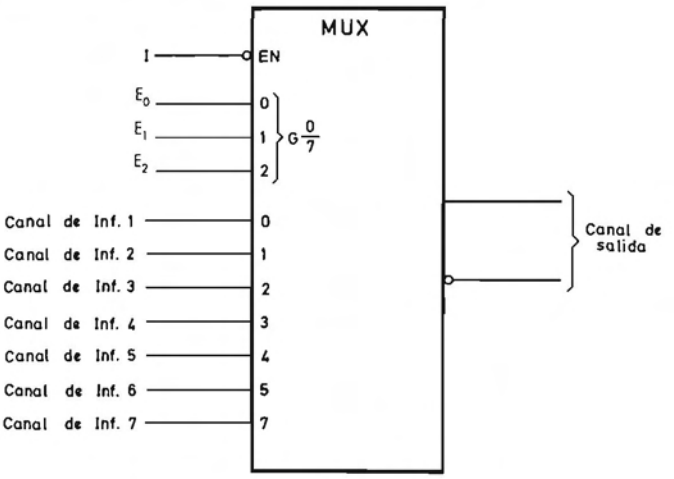


FIGURA 3.93.—**Símbolo lógico de un multiplexor de ocho canales con entrada de desinhibición (*EN*).**

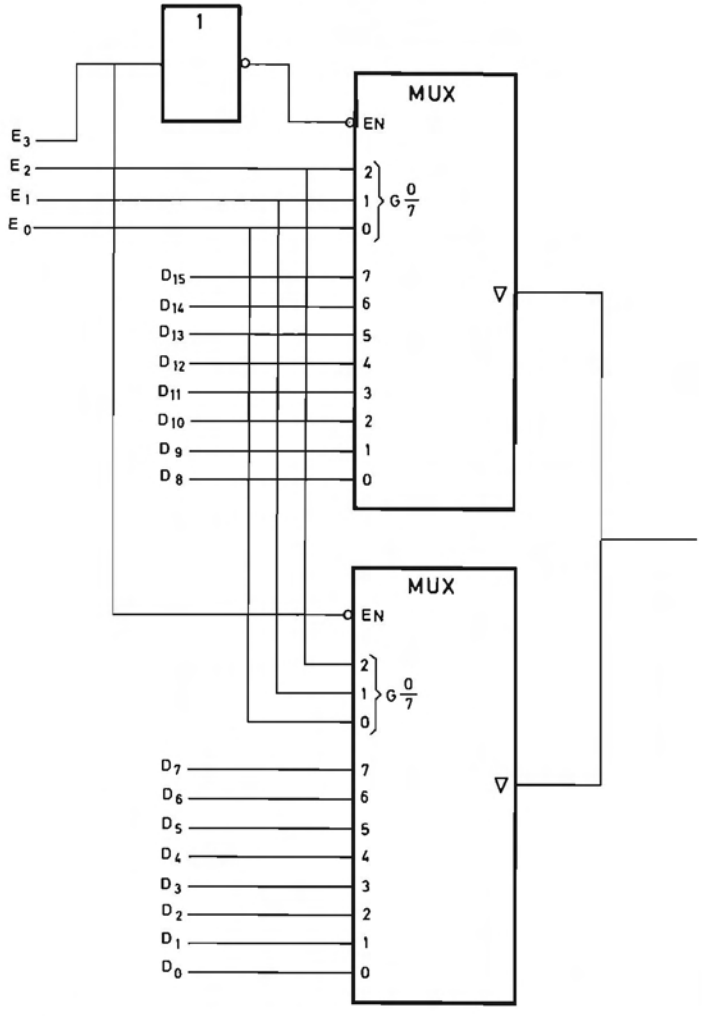
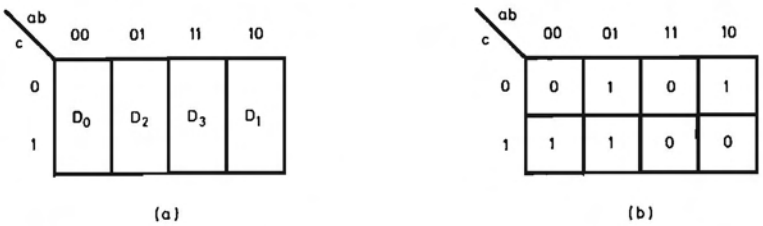


FIGURA 3.94.—**Multiplexor de dieciséis canales realizado con dos multiplexores de ocho canales con salida de tres estados y entrada de desinhibición (*EN*).**

En la figura 3.95*a* se indica para cada combinación de las variables *a* y *b* la entrada *D* cuya información aparece a la salida *Z*. En la figura 3.95*b* se indica el valor que debe tomar *f* para cada combinación de entrada. Si una determinada columna coincide con la variable *c*, ésta ha de ser conectada a la entrada *D* correspondiente. Si, por el contrario, en una columna la función *f* toma los valores inversos de *c*, ha de ser la variable $\bar{c}$ la que se conecte a la entrada correspondiente. Finalmente si una columna es igual a cero o uno, deberá hacerse que la entrada *D* correspondiente sea igual a cero o uno respectivamente. En la figura 3.95*c* se indican los valores que deben tomar las *D* y el esquema lógico de realización de la función *f* con el multiplexor de cuatro canales.



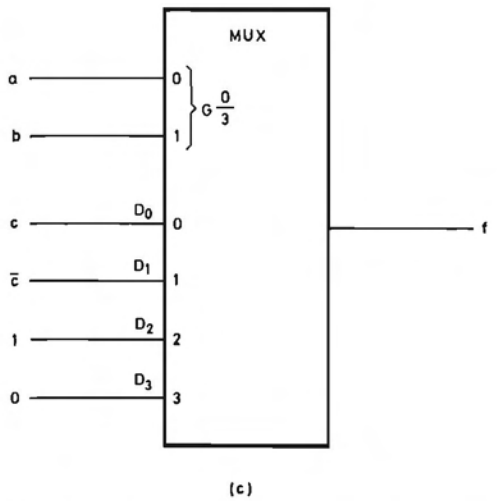


FIGURA 3.95.—**Generación de una función lógica con un multiplexor.**

c	*b*	*a*	*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	1
1	1	1	0

TABLA 3.27

Por ejemplo, cuando *a* y *b* toman respectivamente los valores uno y cero lógicos, la función *f* debe tomar el valor lógico uno si la variable *c* se encuentra en estado cero y el valor lógico cero en caso contrario. Por tanto, a la entrada D_1 del multiplexor se debe conectar la variable $\bar{c}$.

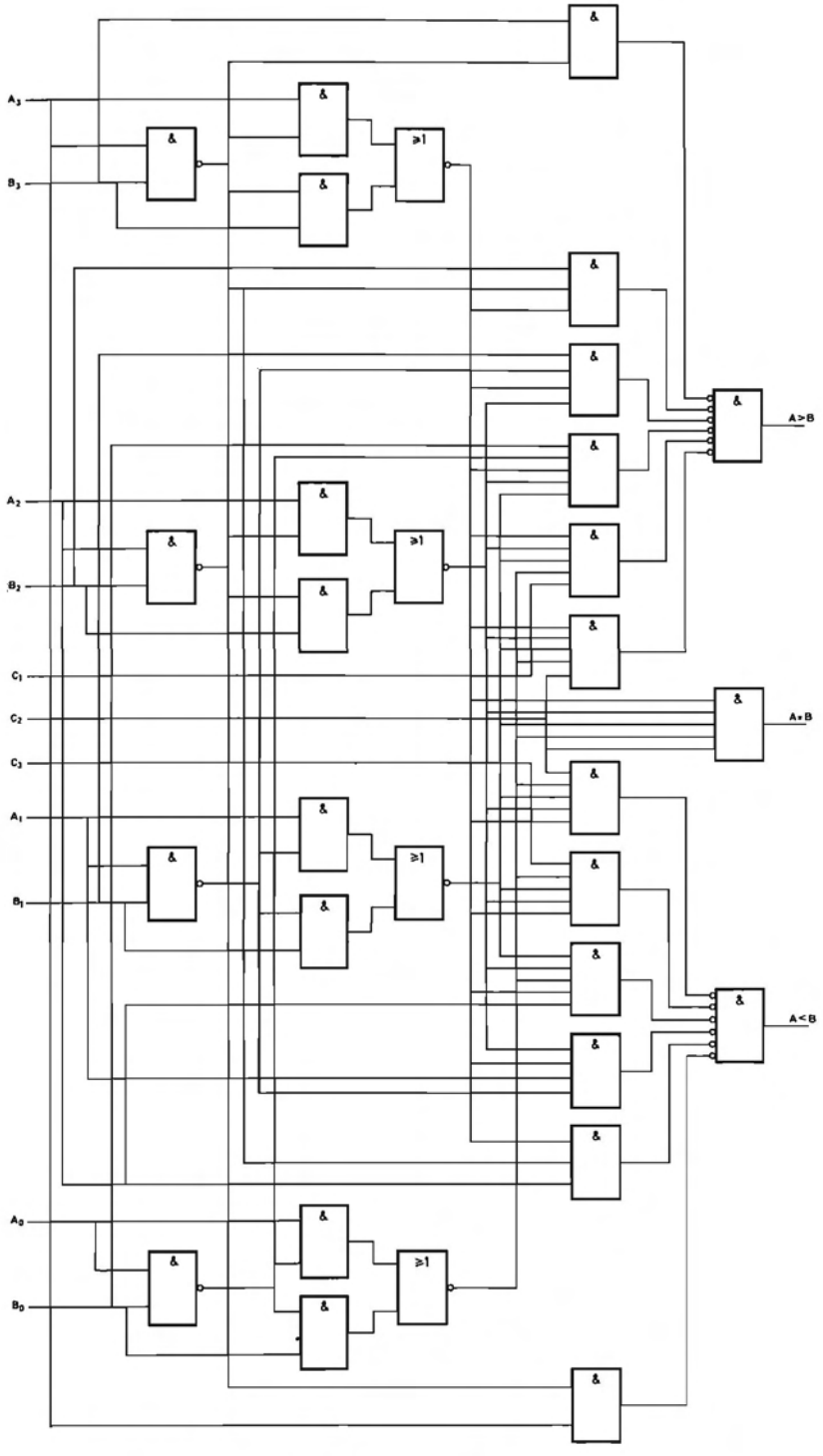


FIGURA 3.96.—**Esquema lógico de un comparador binario de cuatro bits.**

Con un multiplexor de ocho canales se puede generar cualquier función de cuatro variables; con uno de dieciséis cualquier función de cinco variables y mediante combinaciones de todos los tipos de multiplexores descritos se pueden generar todas las funciones lógicas de cualquier número de variables.

3.7.4 Comparadores binarios

Los circuitos comparadores son sistemas combinacionales que detectan si dos combinaciones binarias de n bits en el sistema binario natural son iguales o no y en este último caso cuál de ellas es mayor. Esta función lógica se utiliza con frecuencia en el diseño de sistemas digitales y, por tanto, su disponibilidad como un bloque funcional permite simplificar en gran medida la complejidad de realización de aquéllos.

En escala de integración media (MSI) se han realizado diversos bloques funcionales comparadores.

En la figura 3.96 se representa el esquema de un comparador binario de 4 bits cuya tabla de verdad se indica en la tabla 3.28.

En la figura 3.97 se representan los símbolos no normalizados y normalizados

PQ	$<$	$=$	$>$	$P<Q$	$P=Q$	$P>Q$
$P<Q$	X	X	X	1	0	0
$P>Q$	X	X	X	0	0	0
$P=Q$	1	0	0	1	0	0
$P=Q$	0	1	0	0	1	0
$P=Q$	0	0	1	0	0	1

X Las entradas correspondientes pueden tomar el valor cero o uno lógicos

TABLA 3.28

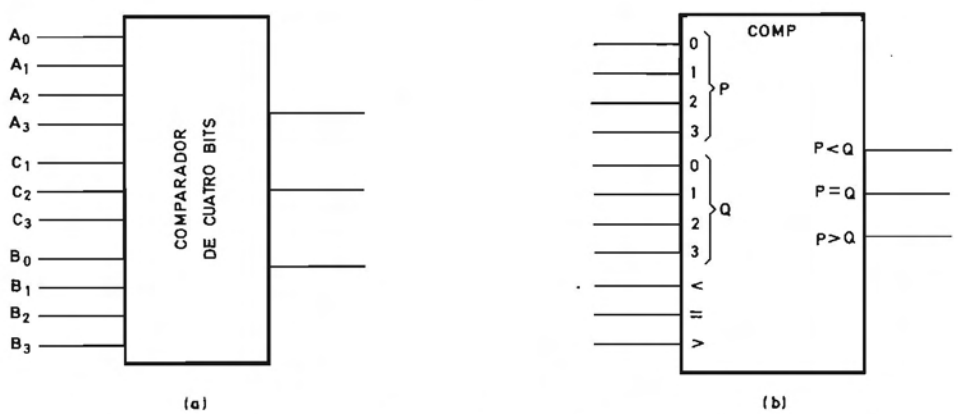


FIGURA 3.97.—**Símbolo lógico del comparador de la figura 3.96: *a*) no normalizado; *b*) normalizado.**

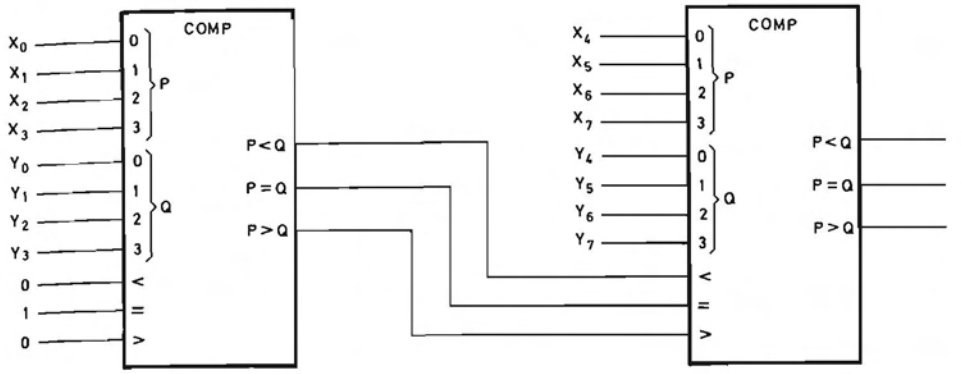


FIGURA 3.98.—**Comparador de ocho bits realizado con el comparador de cuatro cuyo símbolo lógico se representa en la figura 3.97.**

de este comparador que posee tres entradas $<$, $=$ y $>$, que permiten realizar la comparación de combinaciones binarias de un número cualquiera de bits. En la tabla 3.28 se observa que, en el caso de que P y Q sean iguales, el resultado de la comparación depende del valor lógico de $<$, $=$ y $>$ y, por tanto, conectando estas entradas respectivamente a las salidas $P<Q$, $P = Q$ y $P>Q$ de otro comparador y así sucesivamente, se amplía la capacidad de comparación.

En la figura 3.98 se indica el esquema de un comparador de ocho bits realizado con estos circuitos. Primeramente se comparan los cuatro bits menos significativos de cada número y el resultado de esta comparación se conecta a las entradas $<$, $=$ y $>$ de un nuevo comparador a cuyas entradas P_0 a P_3 y Q_0 a Q_3 se conectan los cuatro bits más significativos.

En la figura 3.99 se representa el símbolo lógico normalizado de un comparador de 8 bits que posee una entrada de control $\overline{G}$. La tabla de verdad es la representada en la tabla 3.29. La entrada $\overline{G}$ se comporta como una inhibición/desinhibición que cuando se encuentra en nivel uno fija la salida al nivel cero independientemente de las combinaciones presentes en las entradas P y Q. Por ello en el símbolo normalizado de la figura 3.99 se le asigna el dígito uno tanto a la entrada $\overline{G}$ ($G1$) como a la salida ($1P = Q$). De esta forma se hace superflua la tabla de verdad.

A continuación se analiza un ejemplo de utilización de los circuitos comparadores en combinación con los multiplexores.

Datos P y Q	G	$P = Q$
$P = Q$	0	1
$P > Q$	0	0
$P < Q$	0	0
X	1	0

TABLA 3.29.—**Tabla de verdad del comparador binario de 8 bits de la figura 3.99.**

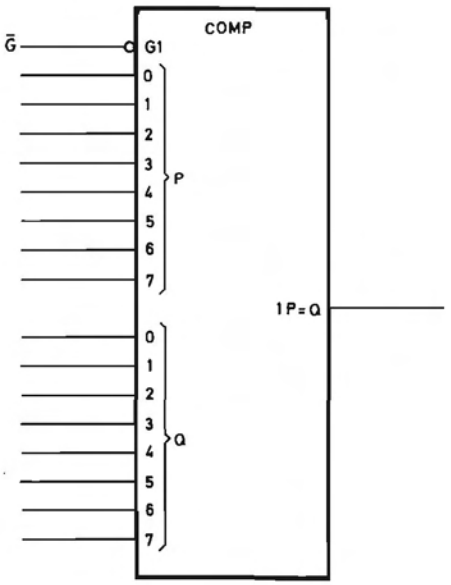


FIGURA 3.99.—**Símbolo lógico normalizado de un comparador de ocho bits con entrada de inhibición /desinhibición (*G*1).**

Ejemplo 3.9: Diseñar un sistema combinacional a cuya entrada se aplican dos combinaciones binarias de cuatro bits y a cuya salida debe aparecer la mayor de ambas y en caso de igualdad, la combinación lógica cero.

En la figura 3.100 se indica la realización práctica de este sistema. Los cuatro bits de ambos números se aplican a las entradas de un comparador de cuatro bits a cuyas salidas se obtiene una indicación de su igualdad o desigualdad y, en este último caso, de cuál es el mayor de ambos. Al mismo tiempo, los bits de igual peso de cada número se llevan a cada uno de los canales de un multiplexor de dos canales. Se necesita, por tanto, un cuádruple multiplexor de dos canales. La decisión del número que aparece a la salida del multiplexor se realiza mediante las entradas de inhibición (*EN*) y de selección (*G*1). La entrada de inhibición *EN* del multiplexor se conecta a la salida $P = Q$ del comparador; cuando ambos números son iguales, esta salida toma el valor uno e inhibe al multiplexor cuyas salidas adoptan todas el estado lógico cero. La salida $P < Q$ del comparador se conecta a la entrada *G*1 de selección del multiplexor. Cuando el número *P* es menor que el *Q*, esta salida adopta el valor lógico uno y a las salidas del multiplexor aparece el número *Q*. Por el contrario, cuando *P* es mayor que *Q* esta salida toma el valor lógico cero y, por tanto, a la salida del multiplexor se presenta el número *P*.

3.7.5 Detectores/generadores de paridad

Este sistema combinacional realiza en esencia la función O-exclusiva de un número de *n* variables y, en consecuencia, su salida adopta el estado lógico uno si

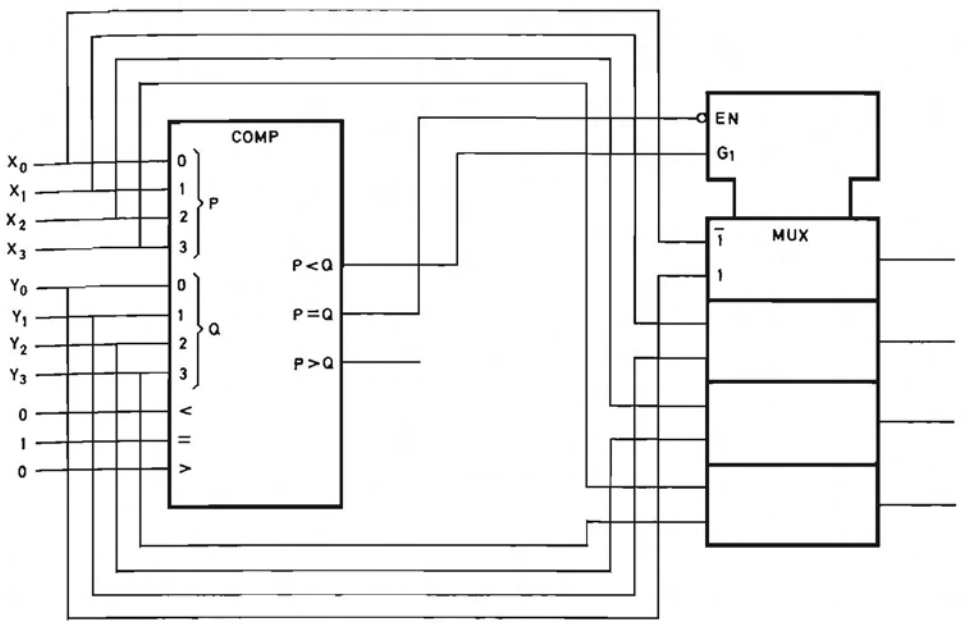


FIGURA 3.100.—**Sistema combinacional que proporciona a su salida el mayor de dos números de cuatro bits, y un cero si son iguales.**

el número de unos aplicados a sus entradas es impar y un cero lógico si dicho número es par. Por tanto, este circuito permite detectar la paridad de una combinación binaria y, si se añade el bit obtenido a la combinación de n bits, se logra una combinación de $n + 1$ bits cuya paridad es constante.

La aplicación más importante de este circuito es la detección de errores en códigos detectores y correctores de error, así como la generación de estos códigos.

En la figura 3.101 se representa el esquema de un detector/generador de paridad de 9 bits realizado en escala de integración media cuya tabla de verdad se indica en la tabla 3.30.

El lector puede comprobar que la ecuación lógica de las salidas *PI* (paridad impar) y *PP* (paridad par) es:

$$PI = A \oplus B \oplus C \oplus D \oplus E \oplus F \oplus G \oplus H \oplus I$$

$$PP = \overline{PI}$$

PI adopta el estado lógico uno si se encuentra en dicho estado un número impar de las variables *A* a *I*.

En la figura 3.102 se representa el símbolo lógico normalizado de este circuito.

Mediante el acoplamiento en serie de dos circuitos de este tipo se consigue un detector/generador de paridad de 17 bits cuyo esquema se representa en la figura 3.103.

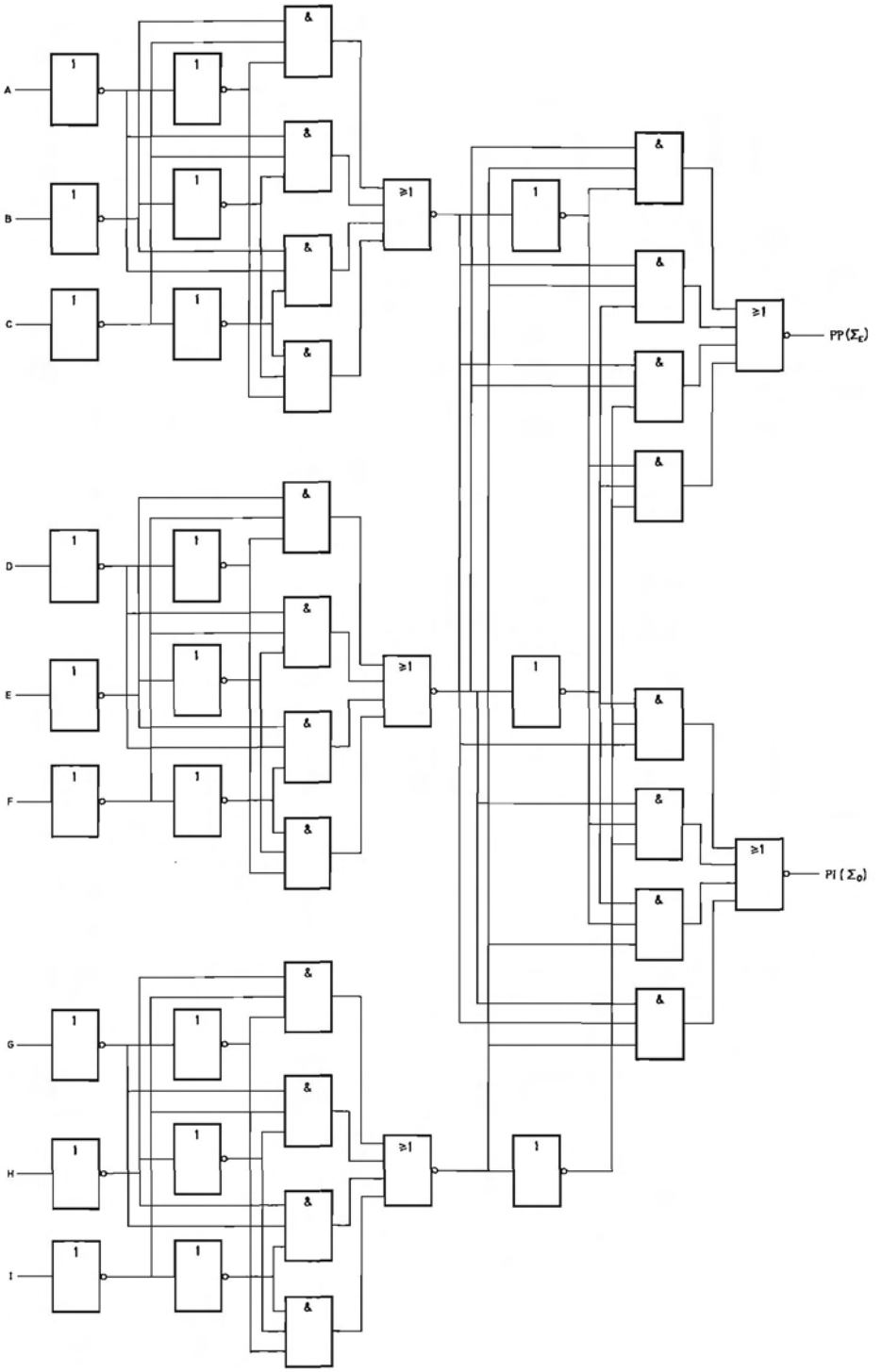


FIGURA 3.101.—**Esquema lógico de un detector/generador de paridad de nueve bits.**

I H G F E D C B A	*PI*	*PP*
Todas las entradas en estado cero	0	1
Una entrada cualquiera en estado uno	1	0
Dos entradas cualesquiera en estado uno	0	1
Tres entradas cualesquiera en estado uno	1	0
Cuatro entradas cualesquiera en estado uno	0	1
Cinco entradas cualesquiera en estado uno	1	0
Seis entradas cualesquiera en estado uno	0	1
Siete entradas cualesquiera en estado uno	1	0
Ocho entradas cualesquiera en estado uno	0	1
Nueve entradas cualesquiera en estado uno	1	0

TABLA 3.30

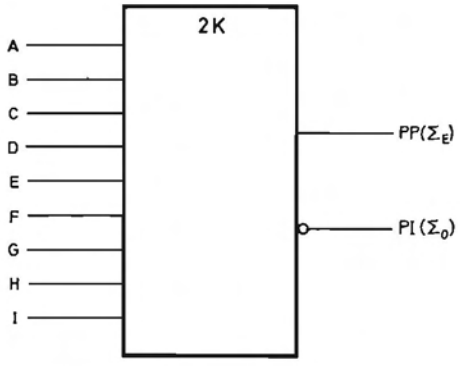


FIGURA 3.102.—**Símbolo lógico del detector/generador de paridad de la figura 3.101.**

3.8 SISTEMAS COMBINACIONALES PROGRAMABLES

Se pueden definir los sistemas combinacionales programables como aquellos cuya tabla de verdad puede ser cambiada sin necesidad de modificar el cableado entre los elementos que los constituyen.

El progreso de las técnicas de integración ha permitido la realización física de sistemas combinacionales programables de elevado número de variables de entrada y salida en escala de integración media (MSI) y gran escala de integración (LSI).

Los sistemas combinacionales programables se pueden clasificar en dos grandes tipos según se indica en la tabla 3.31:

a) Sistemas combinacionales programables no universales que realizan funciones específicas de aplicación general.

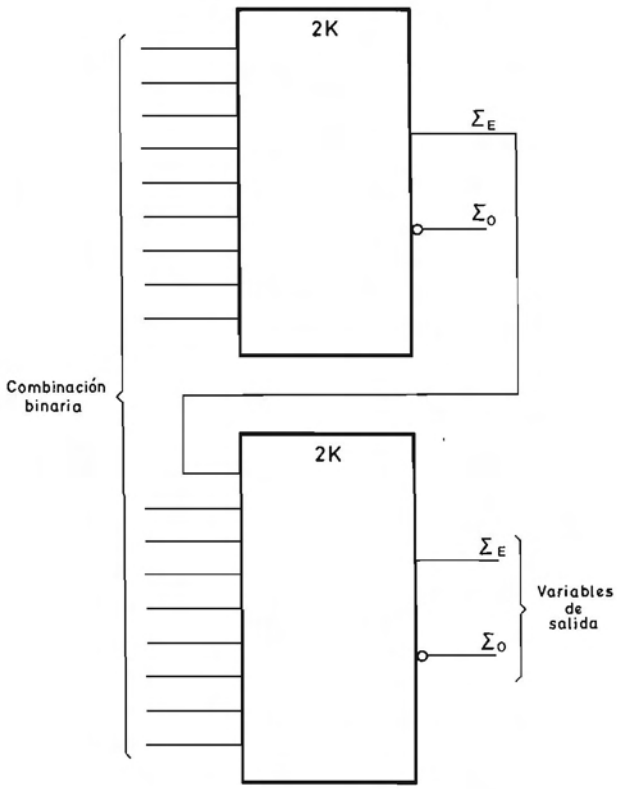


FIGURA 3.103.—**Detector/generador de paridad de 17 bits realizado con el detector/generador de paridad cuyo símbolo lógico se representa en la figura 3.102.**

b) Sistemas combinacionales programables universales con los que se puede realizar cualquier función lógica.

La programación consiste en la supresión de determinadas conexiones realizadas mediante un diodo o un transistor. Inicialmente se utilizaron diodos y la programación consistía en hacer pasar a través de ellos una corriente superior a la de funcionamiento normal que los fundía. El desarrollo de los transistores MOS ha hecho que se realicen sistemas combinacionales programables cuyas conexiones están constituidas por transistores MOS de puerta flotante que se hacen conducir o no mediante la aplicación de una tensión superior a la de funcionamiento normal que inyecta portadores de carga a la puerta durante el proceso de programación. En el apartado 7.3.2.2 se describen las memorias pasivas que utilizan estas técnicas. Al lector que desee estudiarlas en mayor profundidad se le remite a la bibliografía [INTE 89] [PHIL 89] [CYPR 89].

En sucesivos apartados se estudian los diferentes tipos de sistemas combinacionales programables.

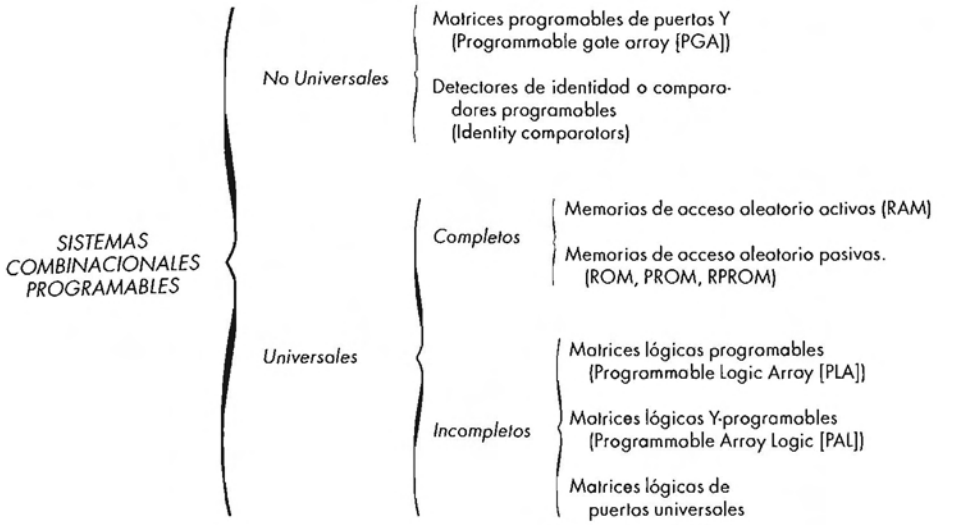


TABLA 3.31

3.8.1 Sistemas combinacionales programables no universales

Existen algunas funciones lógicas que, aunque no son universales, se utilizan en numerosas aplicaciones y presentan variantes cuya programación resulta interesante desde un punto de vista práctico.

A continuación se estudian dos sistemas combinacionales programables que realizan funciones de ese tipo.

3.8.1.1 Matrices programables de puertas Y o decodificadores programables [Programmable gate arrays (PGA)]. Su esquema se representa en la figura 3.104*a* en la que se observa que están formadas por un cierto número n' de puertas Y conectadas a un número n de variables de entrada y sus inversas. El número n' de puertas es menor que 2^n y mediante la supresión de las conexiones adecuadas se logra que la salida de cada una de ellas constituya un producto canónico entre los 2^n posibles. De lo expuesto se deduce que este circuito constituye. un decodificador programable. El esquema de la figura 3.104*a* se puede representar de forma simplificada tal como se indica en la figura 3.104*b* en la que todas las conexiones de cada puerta Y se indican en una sola línea. En la figura 3.105 se representa este circuito mediante símbolos normalizados.

3.8.1.2 Detectores de identidad (Identity comparators). Su esquema básico se representa en la figura 3.106*a* y están constituidos por un comparador, una de cuyas combinaciones de entrada es programable, es decir, sus bits se pueden colocar en cero o en uno mediante la supresión de la conexión adecuada. En efecto, cada uno de los bits de la combinación Q de la figura 3.106*a* está a cero si el

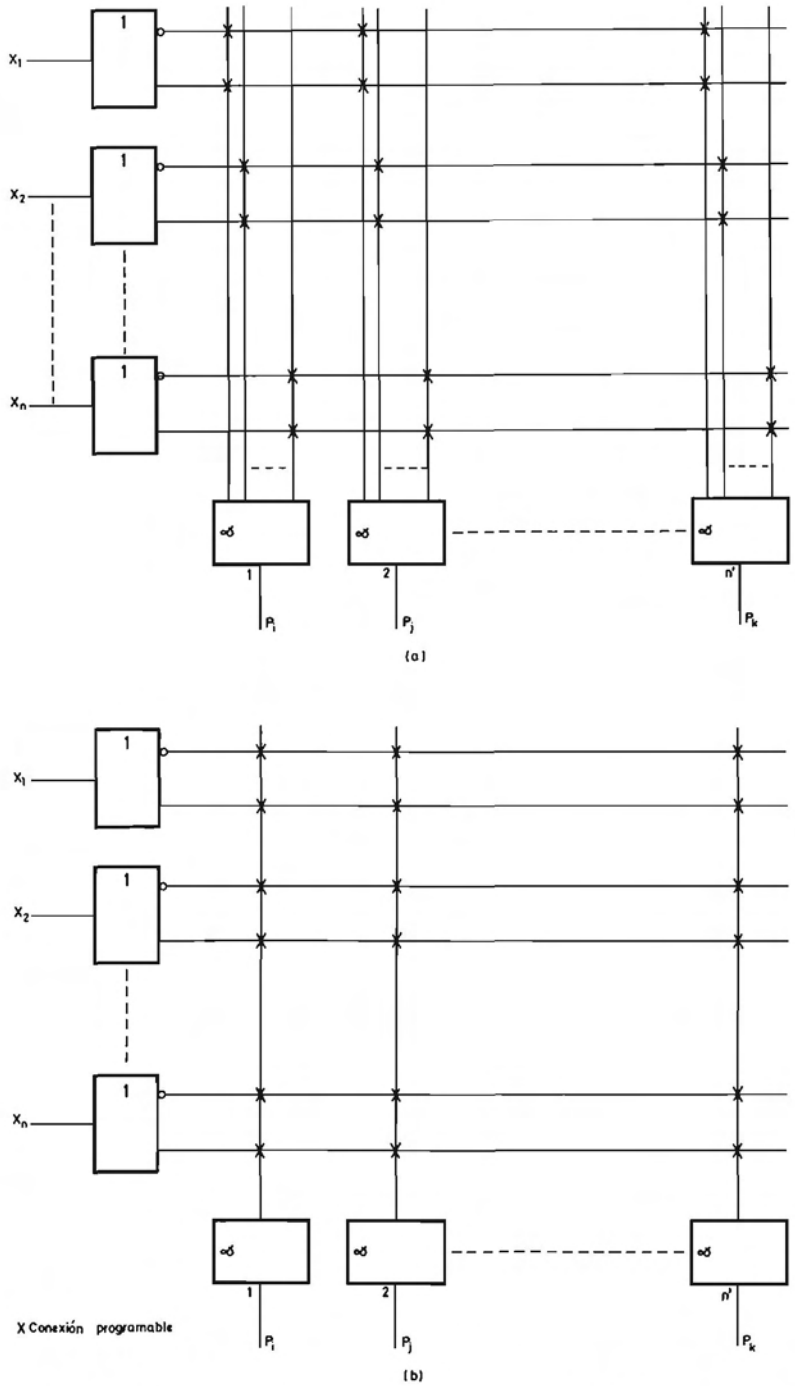


FIGURA 3.104.—**Esquema lógico de una matriz programable de puertas Y [Programmable gate array (PGA)]: *a*) no simplificado; *b*) simplificado.**

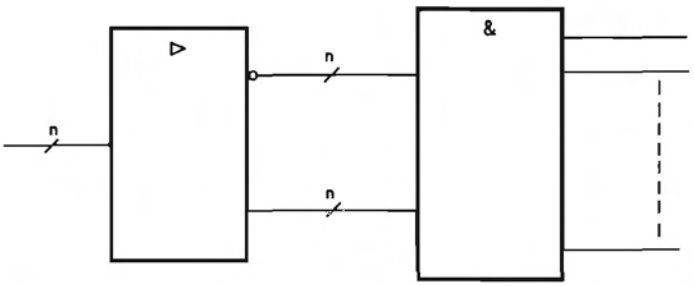


FIGURA 3.105.—**Esquema de una matriz programable de puertas Y realizado con símbolos lógicos normalizados.**

elemento (diodo, transistor, etc.) representado mediante el símbolo X conduce y está a uno si por el contrario está cortado. La salida $P = Q$ se pone a nivel uno cuando la combinación presente en la entrada P coincide con la programada en la entrada Q. Este circuito, por lo tanto, identifica la combinación presente en Q y tiene como principal aplicación la selección de unidades de entrada o salida modulares de un procesador digital [MAND 92].

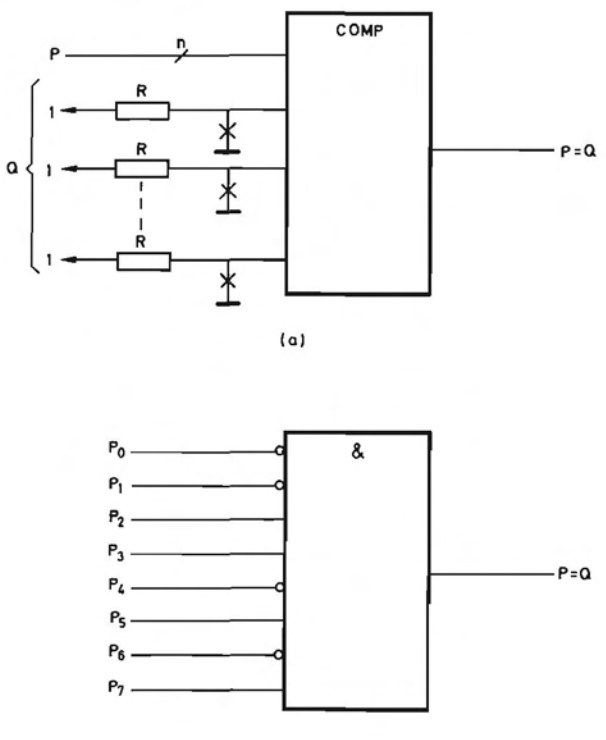


FIGURA 3.106.—**Detector de identidad (Identity comparator): *a*) esquema básico; *b*) símbolo lógico.**

En la figura 3.106*b* se representa el símbolo lógico de este bloque funcional una vez programado. La entrada *Pi* cuya homóloga *Qi* ha sido programada en uno no lleva el símbolo de inversión de entrada y, por el contrario, sí lo lleva aquella que ha sido programada en cero.

El circuito de la figura 3.106*b* detecta, por lo tanto, la combinación 10101100.

La forma de programación depende de la tecnología y la realización física utilizada por el fabricante de circuitos integrados.

El lector debe por ello estudiar los manuales de datos [TEXA 85] para diseñar sistemas con estos componentes.

3.8.2 Sistemas combinacionales universales programables

Estos sistemas se pueden a su vez clasificar en completos o incompletos, según sea o no posible programar el valor de las variables de salida para cada combinación de las variables de entrada de forma independiente. A continuación se estudia cada uno de estos tipos.

3.8.2.1 Sistemas combinacionales universales programables completos. Memorias de acceso aleatorio. Se definen como sistemas combinacionales completos aquellos en los que es posible programar de forma independiente el valor de las variables de salida correspondiente a cada una de las combinaciones de las variables de entrada. Las memorias de acceso aleatorio [en inglés «Random Access Memories» (RAM)] en sus diferentes versiones constituyen dispositivos lógicos programables combinacionales universales completos. Debido a que las memorias de acceso aleatorio forman parte de la práctica totalidad de los procesadores digitales secuenciales, a su estudio se dedica el capítulo 7 de este libro. No obstante, a continuación se realiza una breve introducción a este tipo memorias y para un estudio más profundo se remite al lector al citado capítulo.

Una memoria de acceso aleatorio (RAM) está constituida por un cierto número N de células capaces de almacenar una información binaria (0 o 1) agrupadas en posiciones de m células de manera que el número total N' de posiciones cumple la ecuación $N = N' \cdot m$. La memoria posee en el caso más general m terminales de entrada cuya información puede ser introducida en las m células de cualquier posición en una operación de escritura y m terminales de salida en los que puede aparecer la información de las m células de cualquier posición en una operación de lectura. Ambos grupos de terminales se pueden confundir en uno solo, utilizado indistintamente para introducir información en la memoria o leer la que contiene ésta. En la figura 3.107 se representan los símbolos lógicos no normalizado (*a*) y normalizado (*b*) de una memoria de acceso aleatorio (RAM).

Para poder seleccionar cuál de las N' posiciones se lee o escribe, la memoria posee n terminales de dirección tales que $2^n = N'$. Cada una de las 2^n combinaciones posibles de las n variables de dirección selecciona una de las N' posiciones de la memoria.

El lector puede comprender fácilmente que una memoria de acceso aleatorio constituye un sistema combinacional programable. En efecto, una memoria de ac-

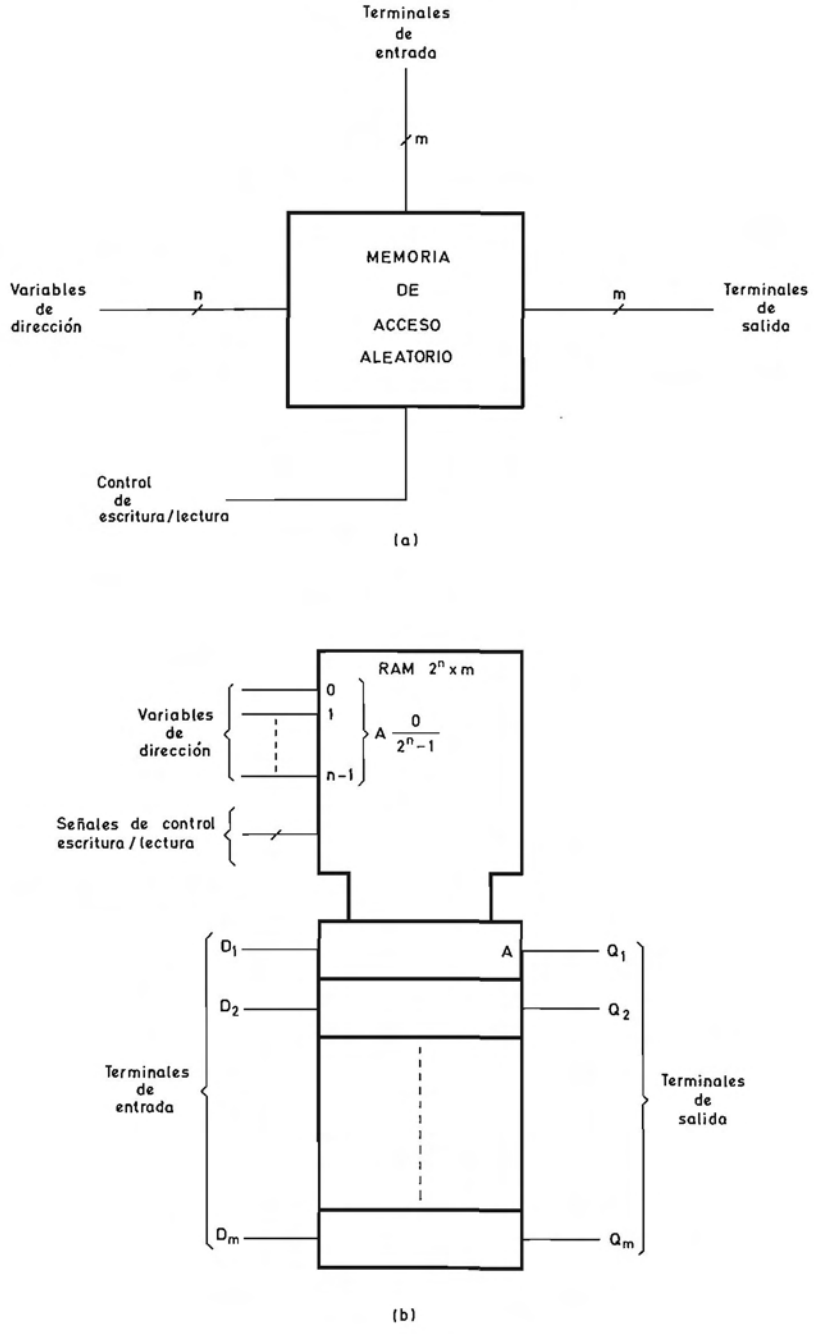
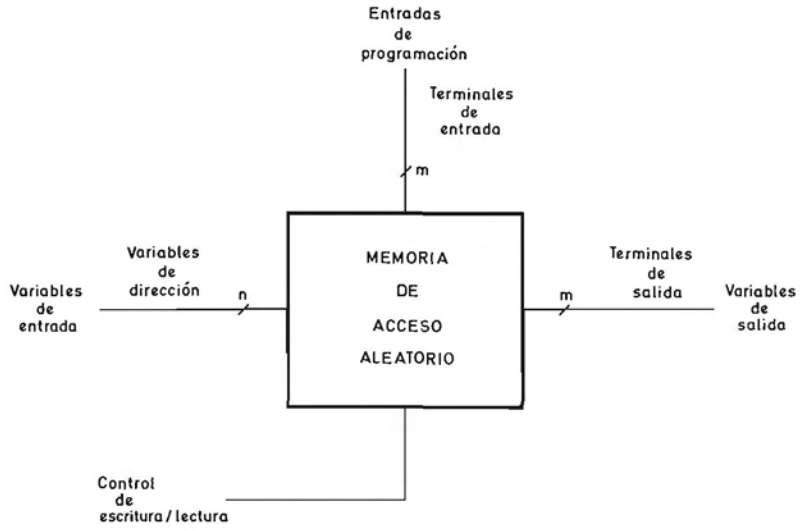
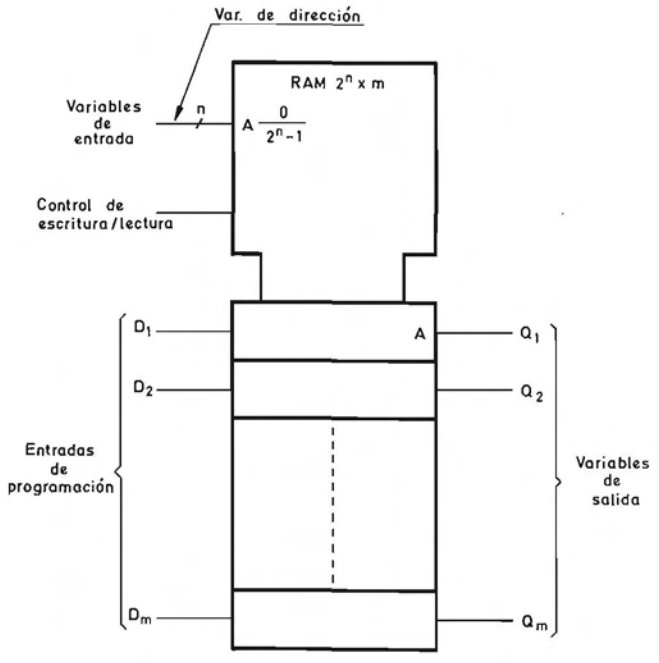


FIGURA 3.107.—**Memoria de acceso aleatorio: *a*) símbolo lógico no normalizado; *b*) símbolo lógico normalizado.**



(a)



(b)

FIGURA 3.108.—**Memoria de acceso aleatorio utilizada como circuito combinacional programable completo.**

ceso aleatorio se comporta como un sistema combinacional durante la operación de lectura, porque para cada combinación binaria aplicada a sus *n* entradas de dirección aparece una información de salida igual a la contenida en la posición seleccionada, que es independiente de la secuencia de combinaciones de las variables de entrada. Las entradas de dirección de la memoria constituyen las variables de entrada del sistema combinacional y las salidas de información constituyen las variables de salida, tal como se indica en la figura 3.108. Pero además este sistema combinacional es programable, porque mediante operaciones de escritura se puede modificar la información contenida en cada posición introduciendo en ella la información colocada en los terminales de entrada, que constituyen las entradas de programación del circuito (fig. 3.108). La señal de «control de la escritura/lectura» permite seleccionar mediante su nivel lógico la operación de escritura o lectura.

Por lo tanto, una memoria de acceso aleatorio se comporta durante la operación de lectura como un sistema combinacional, tal como se indica de forma gráfica en la figura 3.109 en la cual se prescinde de la señal de «control de escritura/lectura» que se supone conectada rígidamente al nivel lógico correspondiente a la operación de lectura. Los terminales de entrada se utilizan exclusivamente para realizar la programación mediante operaciones de escritura en las diferentes posiciones de la memoria.

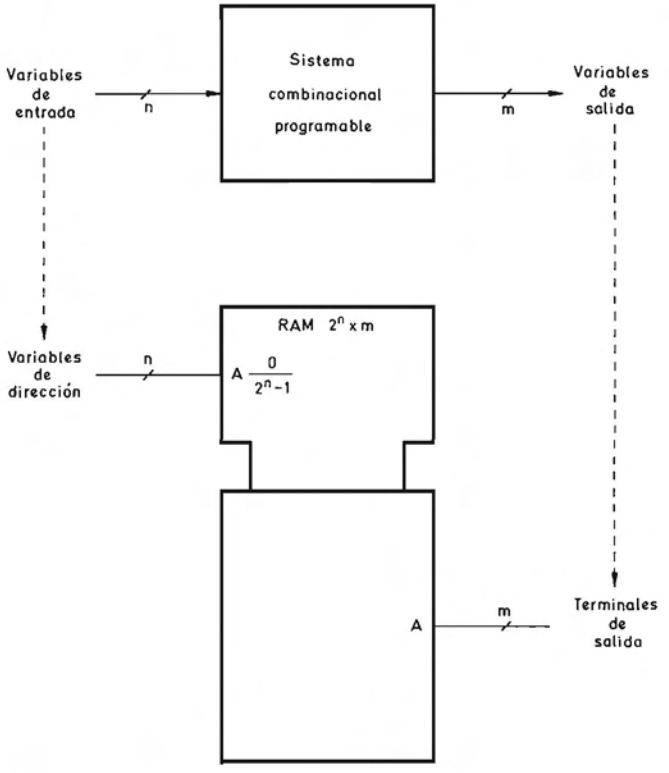


FIGURA 3.109.—**Memoria de acceso aleatorio utilizada como circuito combinacional.**

Mediante un ejemplo se aclara todo lo que se acaba de exponer.

Ejemplo 3.10: Realizar mediante una memoria de acceso aleatorio el sistema combinacional cuya tabla se representa en la tabla 3.32.

Es necesario deducir la organización de la memoria, es decir el número de posiciones y el número de bits de cada posición y el tipo de memoria de acceso aleatorio.

El número de posiciones ha de ser igual al de combinaciones de las variables de entrada, es decir, $2^4 = 16$. Cada posición ha de tener dos bits correspondientes a las dos variables de salida f_1 y f_2. Por ello, la memoria ha de poseer cuatro entradas de dirección y dos variables de salida y su símbolo lógico se representa en la figura 3.110. Las variables de entrada a, b, c y d se conectan respectivamente a las entradas de dirección A_0, A_1, A_2 y A_3 y los terminales de salida S_1 y S_2 coinciden con las variables de salida f_1 y f_2 respectivamente.

En cada una de las posiciones de la memoria se ha de colocar la información de f_1 y f_2 correspondiente a cada combinación de las variables de entrada indicada en la tabla 3.32.

En el capítulo 7 se estudian los diferentes tipos de memorias de acceso aleatorio (RAM) y se demuestra que las más adecuadas para la realización de sistemas combinacionales son las pasivas en sus diferentes versiones debido a su característica de no perder la información al suprimir la tensión de alimentación.

En este ejemplo se ha elegido una memoria pasiva programable (PROM) (figura 3.110).

d	c	b	a	f_1	f_2
0	0	0	0	1	1
0	0	0	1	0	0
0	0	1	0	1	1
0	0	1	1	1	1
0	1	0	0	1	0
0	1	0	1	1	0
0	1	1	0	1	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	0	1	0	0
1	0	1	0	0	1
1	0	1	1	0	0
1	1	0	0	0	0
1	1	0	1	0	0
1	1	1	0	0	1
1	1	1	1	0	0

TABLA 3.32

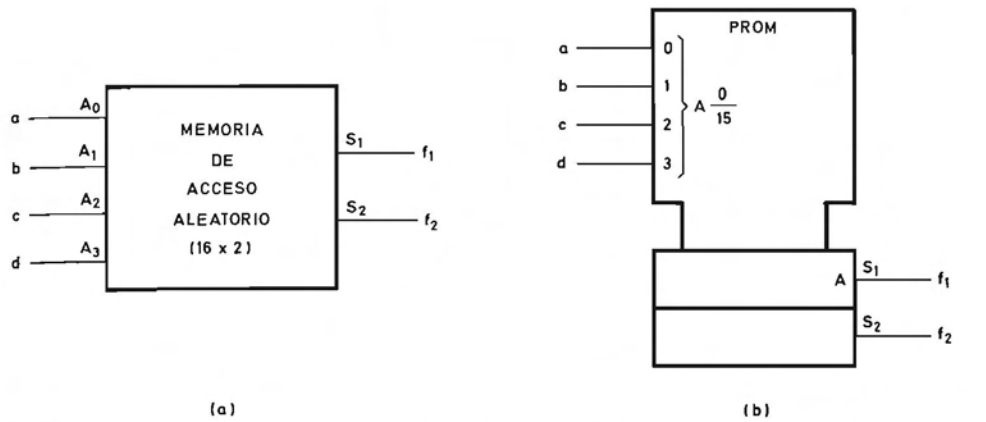


FIGURA 3.110.—**Realización con una memoria de acceso aleatorio del circuito combinacional cuya tabla de verdad se representa en la tabla 3.32: *a*) símbolo no normalizado; *b*) símbolo normalizado.**

El símbolo lógico normalizado asignado a las memorias pasivas es idéntico al de las memorias de acceso aleatorio representado en la figura 3.107. En la figura 3.111*a* se representa el símbolo correspondiente a una memoria totalmente pasiva (ROM) de 2^n posiciones de m bits con salida de tres estados controlada mediante

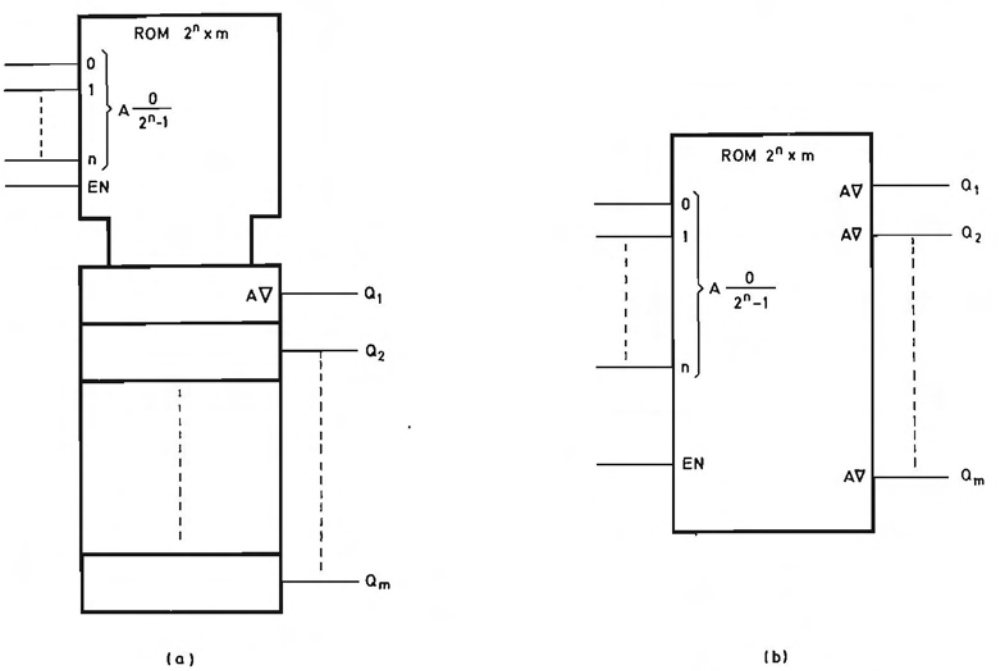


FIGURA 3.111.—**Símbolos lógicos normalizados de una memoria de acceso aleatorio totalmente pasiva.**

la variable de entrada *EN*. En la figura 3.111*b* se representa un símbolo también normalizado alternativo.

Las memorias de acceso aleatorio pasivas pueden tener una organización 2D, 3D o 2½ D (ver apartado 7.2.3.1.2 del capítulo 7). La organización 2D se representa en la figura 3.112. El decodificador selecciona mediante cada una de sus variables de salida las distintas posiciones. El decodificador se puede realizar mediante una matriz de puertas Y adecuadamente conectadas a las variables de entrada de tal manera que cada puerta Y decodifica un producto canónico. Se tiene así una matriz de $n\ 2^n$ conexiones fijas (figura 3.113).

Las salidas de las puertas Y constituyen las columnas de otra matriz cuyas filas se conectan a un número m de puertas O. Las puertas O se conectan a las salidas de todas las puertas Y (figura 3.113). La complejidad de la figura 3.113 lleva a la representación simplificada de la figura 3.114 de la que se deduce que una memoria pasiva (ROM, PROM o RPROM) está constituida por una matriz fija de puertas Y y una matriz programable de puertas O.

En la figura 3.115 se representa la realización de las funciones f_1 y f_2 de la tabla 3.32 mediante una memoria pasiva. En la matriz de puertas O se eliminan las conexiones correspondientes a los productos canónicos que no forman parte de cada una de las funciones.

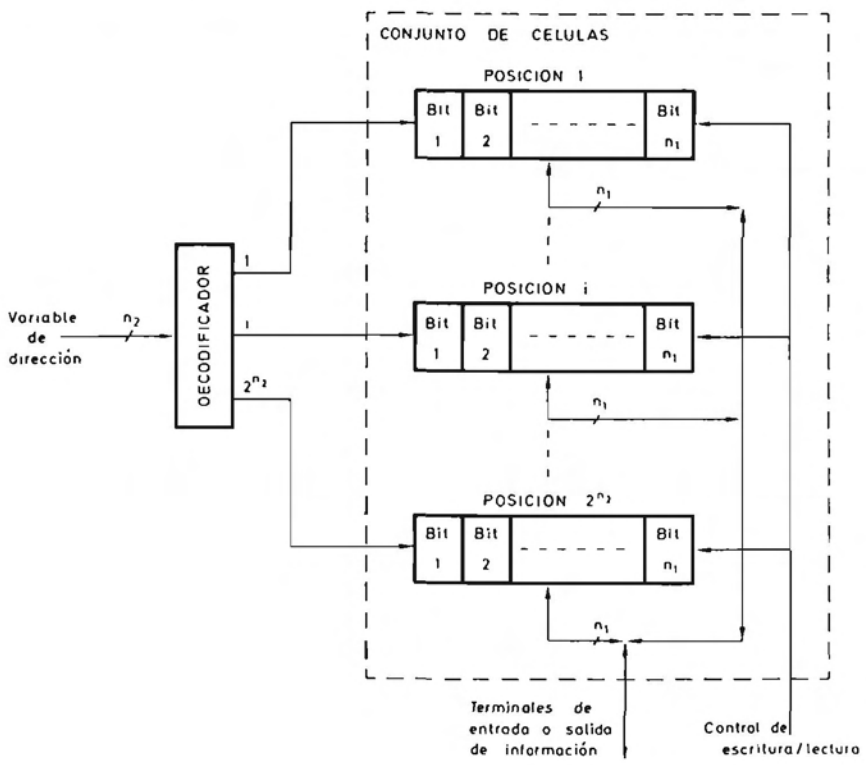


FIGURA 3.112.—**Organización 2D de una memoria de acceso aleatorio.**

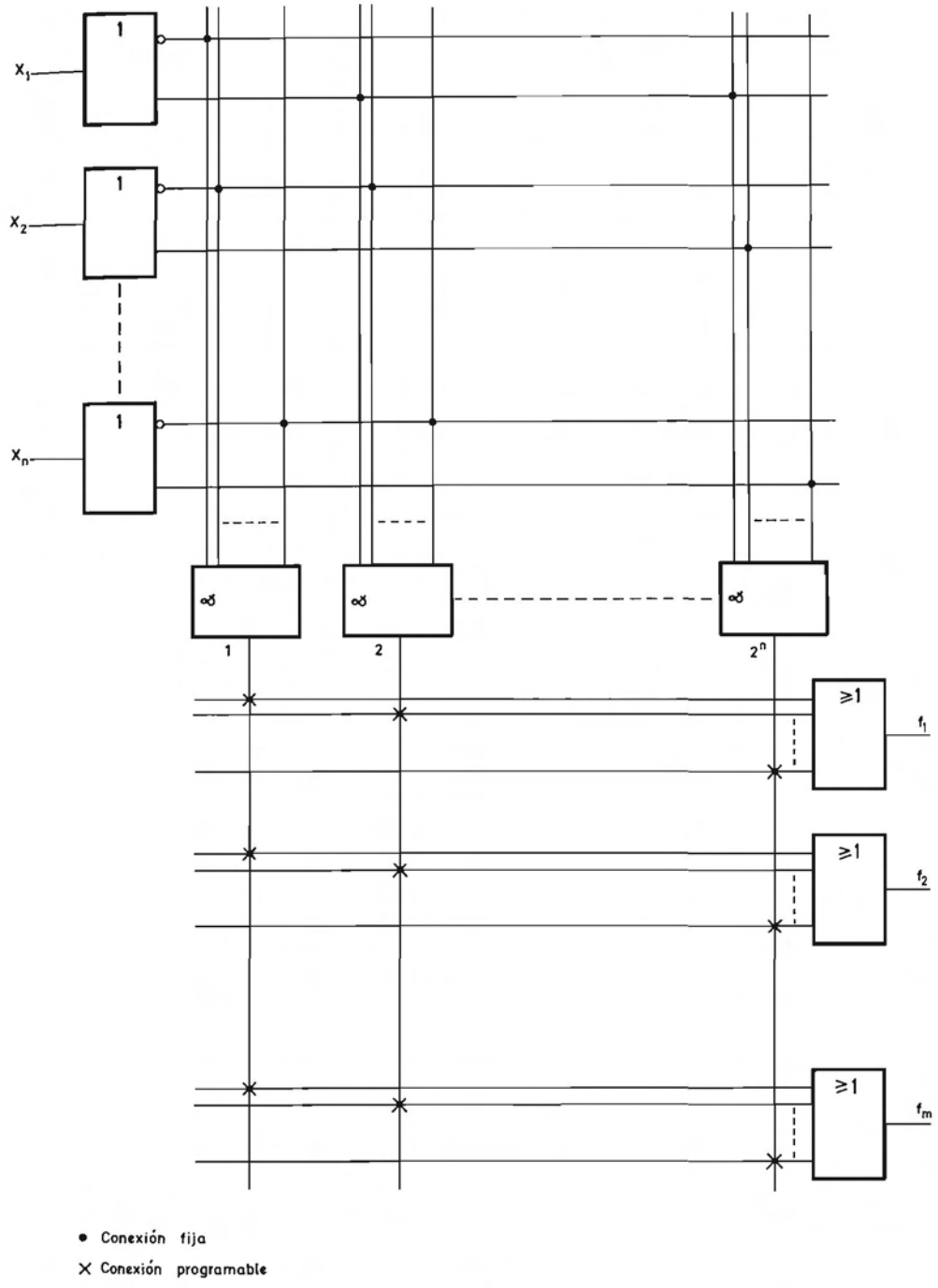


FIGURA 3.113.—**Esquema de una memoria pasiva (ROM, PROM o RPROM) realizada mediante una matriz de puertas Y y otra de puertas O.**

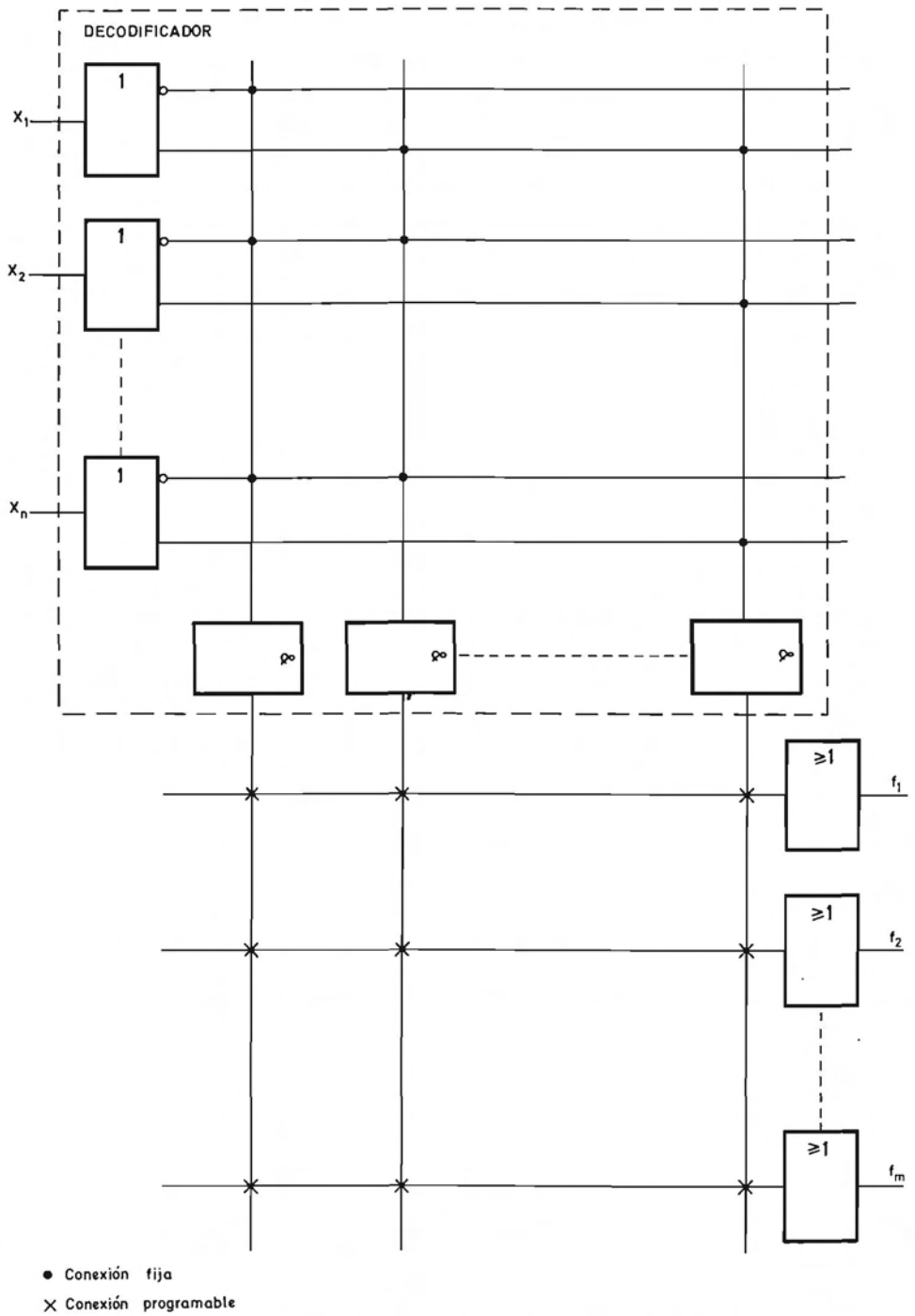


FIGURA 3.114.—**Esquema simplificado de la memoria pasiva de la figura 3.113.**

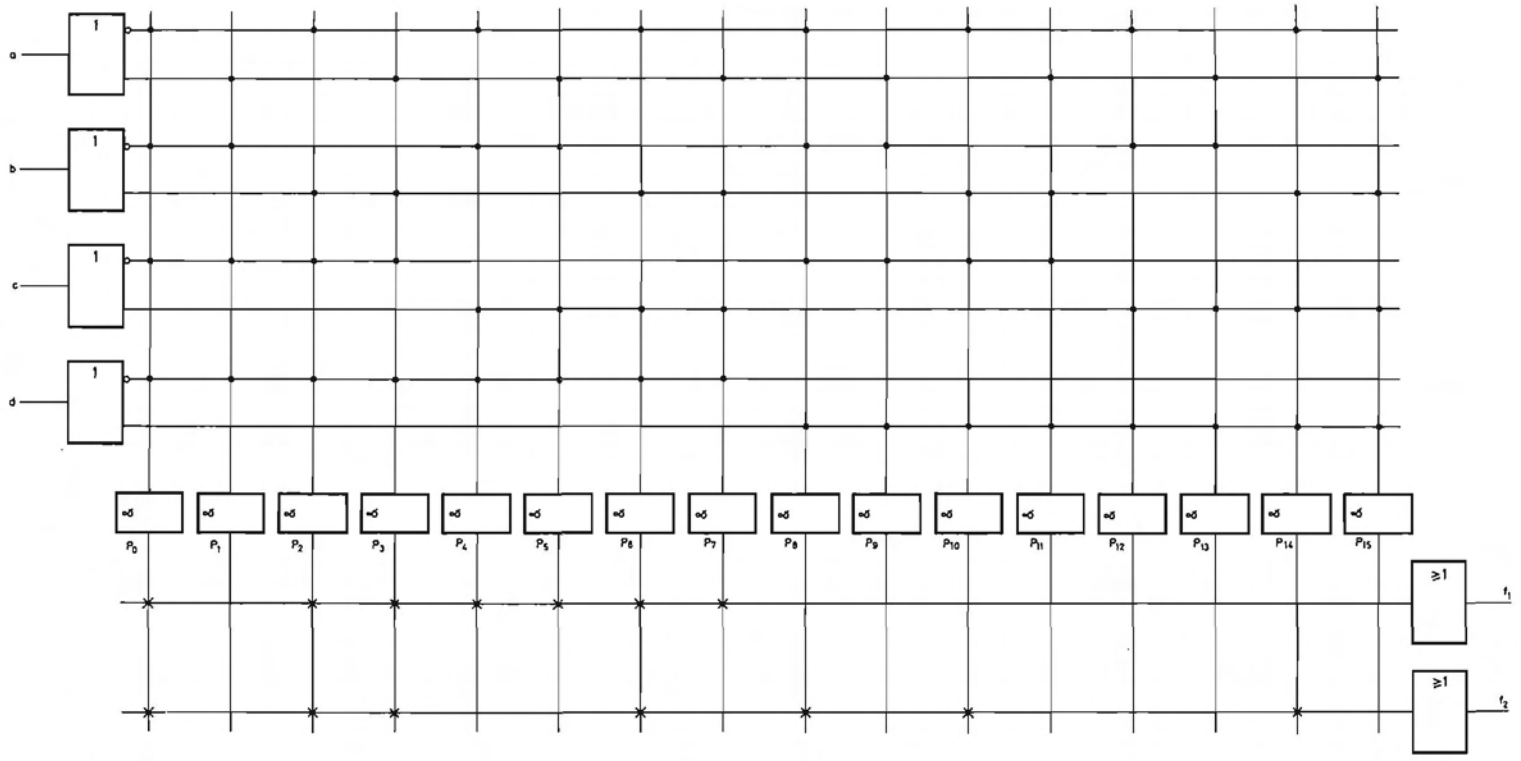


FIGURA 3.115.—**Esquema de la programación de una memoria pasiva de cuatro variables de entrada para realizar las funciones f_1 y f_2 de la tabla 3.32.**

Las memorias pasivas son útiles fundamentalmente para la construcción de tablas, generadores de caracteres, convertidores de código y, en general, cualquier aplicación que requiera programar el valor de las variables de salida para todas y cada una de las combinaciones de entrada. El ejemplo más significativo es la memoria que contiene el programa de un microprocesador en aplicaciones de control. La existencia de muchas aplicaciones en las que no es necesario programar el estado de las salidas para todas las combinaciones de entrada, en especial cuando el número de entradas es elevado, hace preferible en estos casos la utilización de sistemas combinacionales programables incompletos que se estudian en el apartado siguiente.

En el capítulo 7 se estudian con mayor profundidad las memorias pasivas y, en particular, las formas de realizarlas físicamente.

3.8.2.2 Sistemas combinacionales universales programables incompletos (SCUPI). En los sistemas combinacionales programables universales completos que se estudian en el apartado anterior es posible programar de forma independiente el valor de las variables de salida correspondiente a cada una de las combinaciones posibles de las variables de entrada. Pero en la práctica se suelen presentar las siguientes situaciones:

a) Las funciones lógicas sólo toman el valor uno para un cierto número de las combinaciones de las variables de entrada inferior al total 2^n.
b) La expresión canónica de suma de productos se puede simplificar por los métodos numérico o tabular de Karnaugh.
c) La función no está definida para algunas combinaciones de las variables de entrada.

Ninguno de estos tres casos se simplifica cuando la función se realiza mediante

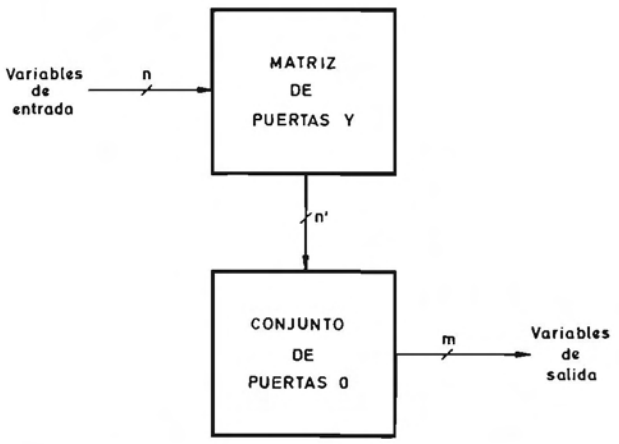


FIGURA 3.116.—**Diagrama de bloques de un sistema combinacional universal programable incompleto realizado mediante una matriz de puertas Y y un conjunto de puertas O.**

un sistema combinacional universal programable completo y de ahí el interés de la utilización de los sistemas combinacionales universales programables incompletos (SCUPI) que están formados por dos o una matriz de puertas lógicas.

Estos sistemas se pueden realizar de dos formas:

a) Mediante una matriz de puertas Y conectada a un conjunto de puertas O (figura 3.116). Según la forma en que se realiza la conexión de la matriz de puertas Y con el conjunto de puertas O, se tienen dos tipos diferentes de SCUPI:

—Las matrices lógicas programables [en inglés «Programmable Logic Array» (PLA)]

—Las matrices lógicas Y-programables [en inglés «Programmable Array Logic» (PAL)]

b) Mediante una matriz de puertas universales.

En sucesivos apartados se estudian los diferentes tipos de SCUPI.

3.8.2.2.1 Matrices lógicas programables [Programmable logic array (PLA)]. Las matrices lógicas programables están constituidas por una matriz programable de n' ($n' < 2^n$, siendo n el número de variables de entrada) puertas Y y una matriz programable de m puertas O (OR). Las puertas Y poseen n entradas que se unen a cada variable de entrada a través de una conexión que puede ser eliminada (figura 3.117).

El número n' es menor que 2^n precisamente porque en la práctica nunca es necesario realizar todos los productos canónicos posibles.

Para programar la matriz lógica programable es necesario realizar las siguientes acciones:

— Suprimir las conexiones adecuadas de la matriz de puertas Y para que la salida de cada una de ellas represente un determinado producto lógico.
— Suprimir las conexiones adecuadas de la matriz de puertas O para que cada variable de salida sea la suma lógica de las salidas de las puertas Y convenientes.

El esquema de la figura 3.117 se puede representar de forma simplificada tal como se indica en la figura 3.118.

Para simplificar el esquema de un sistema digital que posea una matriz lógica programable como las representadas en las figuras 3.117 y 3.118, se puede utilizar el diagrama de bloques de la figura 3.119 en el que se indican por separado la matriz de n' puertas Y y la matriz de m puertas O.

En este diagrama se utilizan los nuevos símbolos normalizados colocando debajo de los indicativos de las puertas Y (&) y O (≥ 1) la cantidad de ellas que es respectivamente n' y m, y el número de entradas de cada una que es respectivamente $2n$ y n'.

Un diagrama de bloques aún más sencillo que constituye un símbolo lógico, es el de la figura 3.120 en el que se utiliza un único símbolo lógico al que se asignan las siglas PLA junto con el número n' de puertas Y que forman parte de la matriz.

A continuación se expone mediante un ejemplo la forma de realizar una función con una matriz lógica programable.

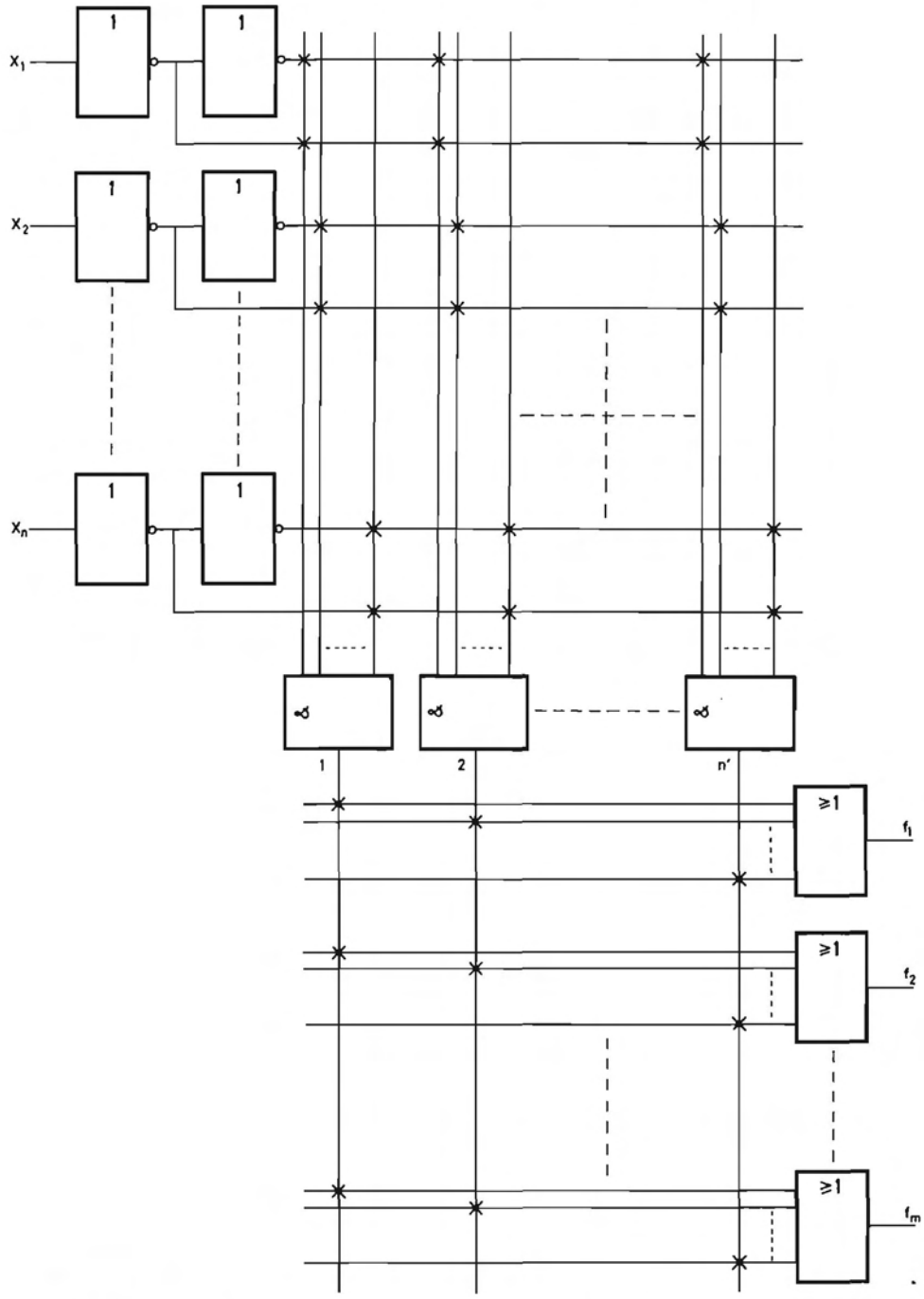


FIGURA 3.117.—**Matriz lógica programable (PLA) de *n* variables de entrada, *n'* puertas Y y *m* variables de salida.**

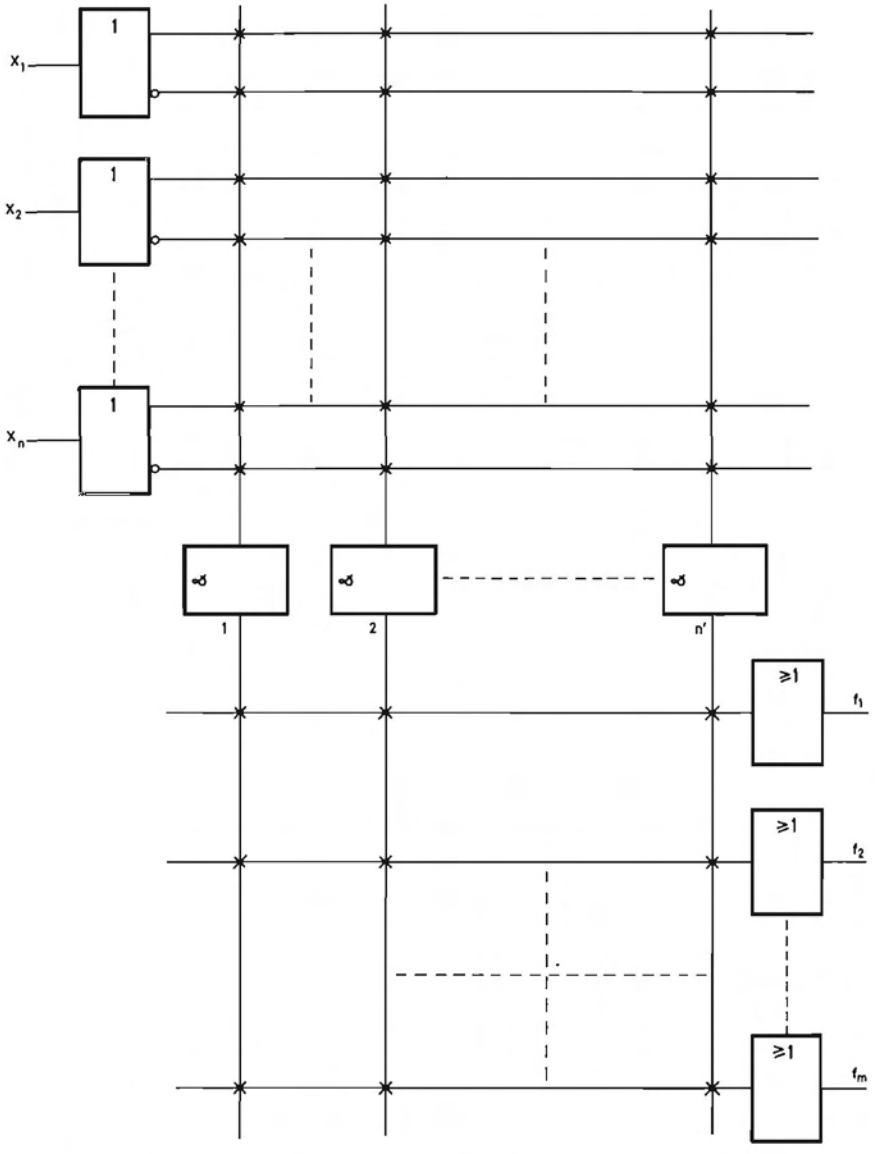


FIGURA 3.118.—**Esquema simplificado de la matriz lógica programable de la figura 3.117.**

Ejemplo 3.11: Realizar mediante una matriz lógica programable las funciones f_1 y f_2 cuya tabla de verdad se representa en la tabla 3.32.

El número de variables de entrada es 4 y el de variables de salida 2 y, por lo tanto, la matriz lógica utilizada ha de tener esta capacidad mínima.

En primer lugar, a partir de la tabla 3.32 se deducen las expresiones canónicas de sumas de productos de las dos funciones:

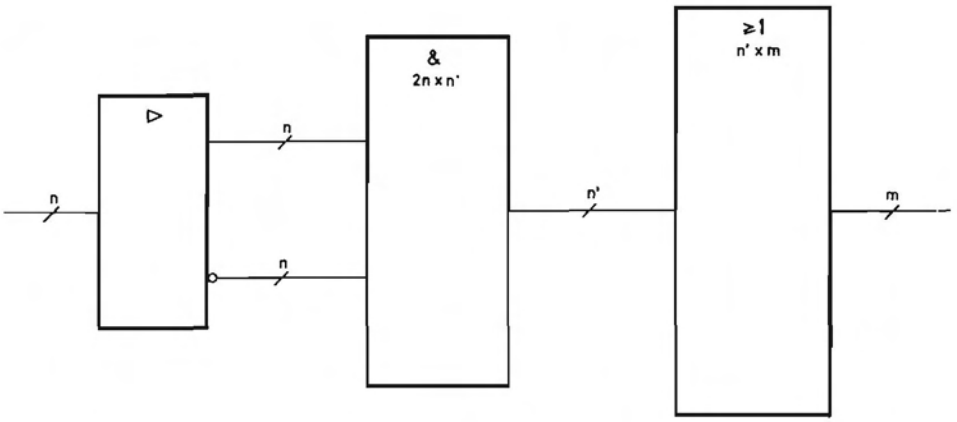


FIGURA 3.119.—**Diagrama de bloques de una matriz lógica programable.**

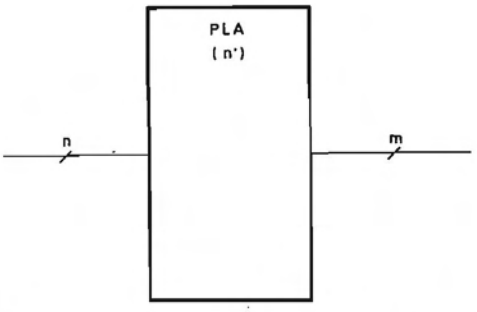


FIGURA 3.120.—**Símbolo lógico de una matriz lógica programable.**

$$f_1 = \sum_4 (0, 2, 3, 4, 5, 6, 7)$$
$$f_2 = \sum_4 (0, 2, 3, 6, 8, 10, 14)$$

El número de productos canónicos diferentes de ambas funciones es 10.

Es posible realizar directamente ambas expresiones con una matriz lógica programable.

La matriz ha de tener un número mínimo de productos igual al de productos canónicos diferentes de ambas funciones, que es 10. En la figura 3.121 se representa su esquema sin programar.

Para realizar la programación se deduce la expresión algebraica de cada producto canónico:

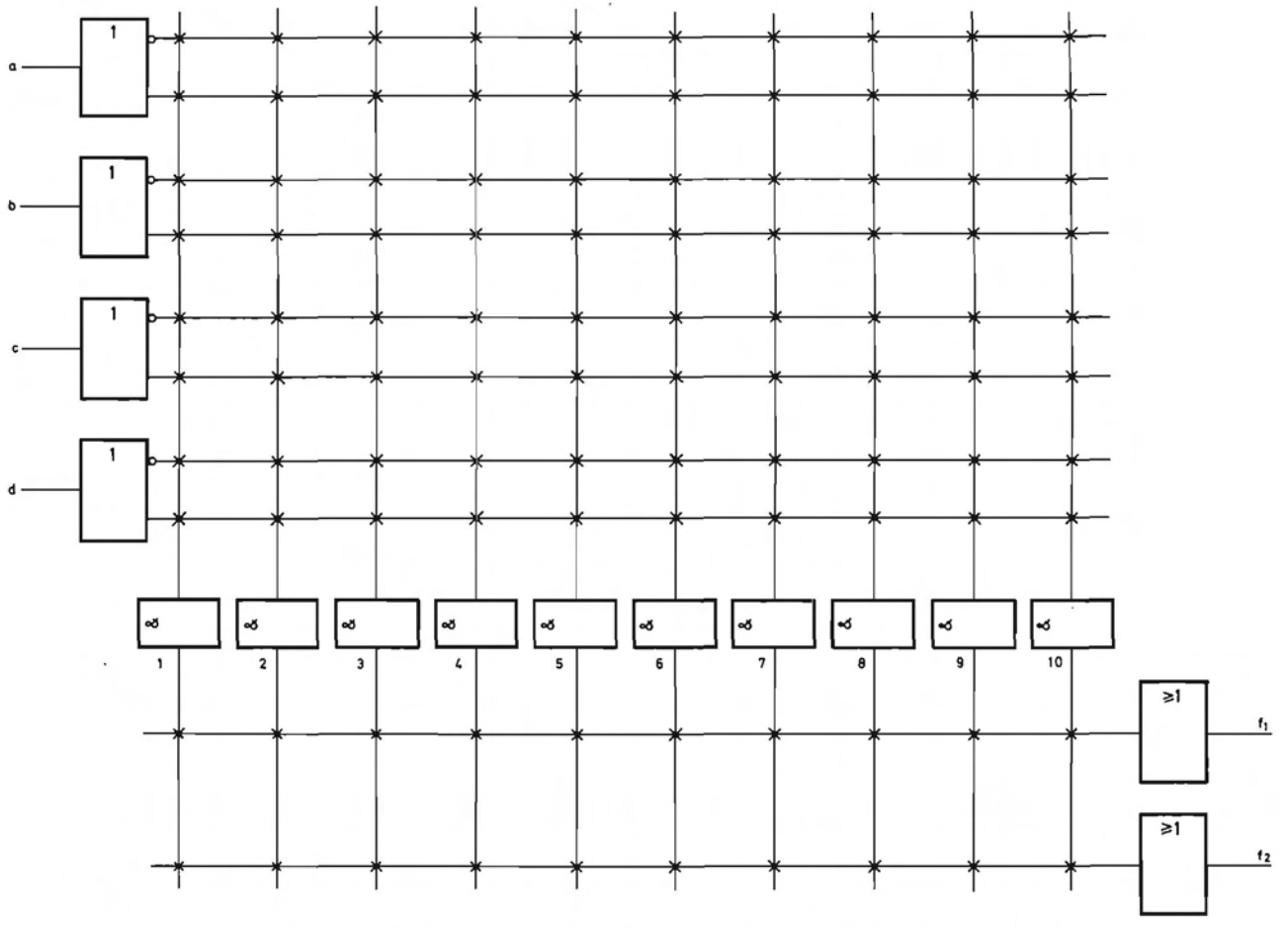


FIGURA 3.121.—**Matriz lógica programable de 4 variables de entrada, 10 productos lógicos y dos variables de salida, sin programar.**

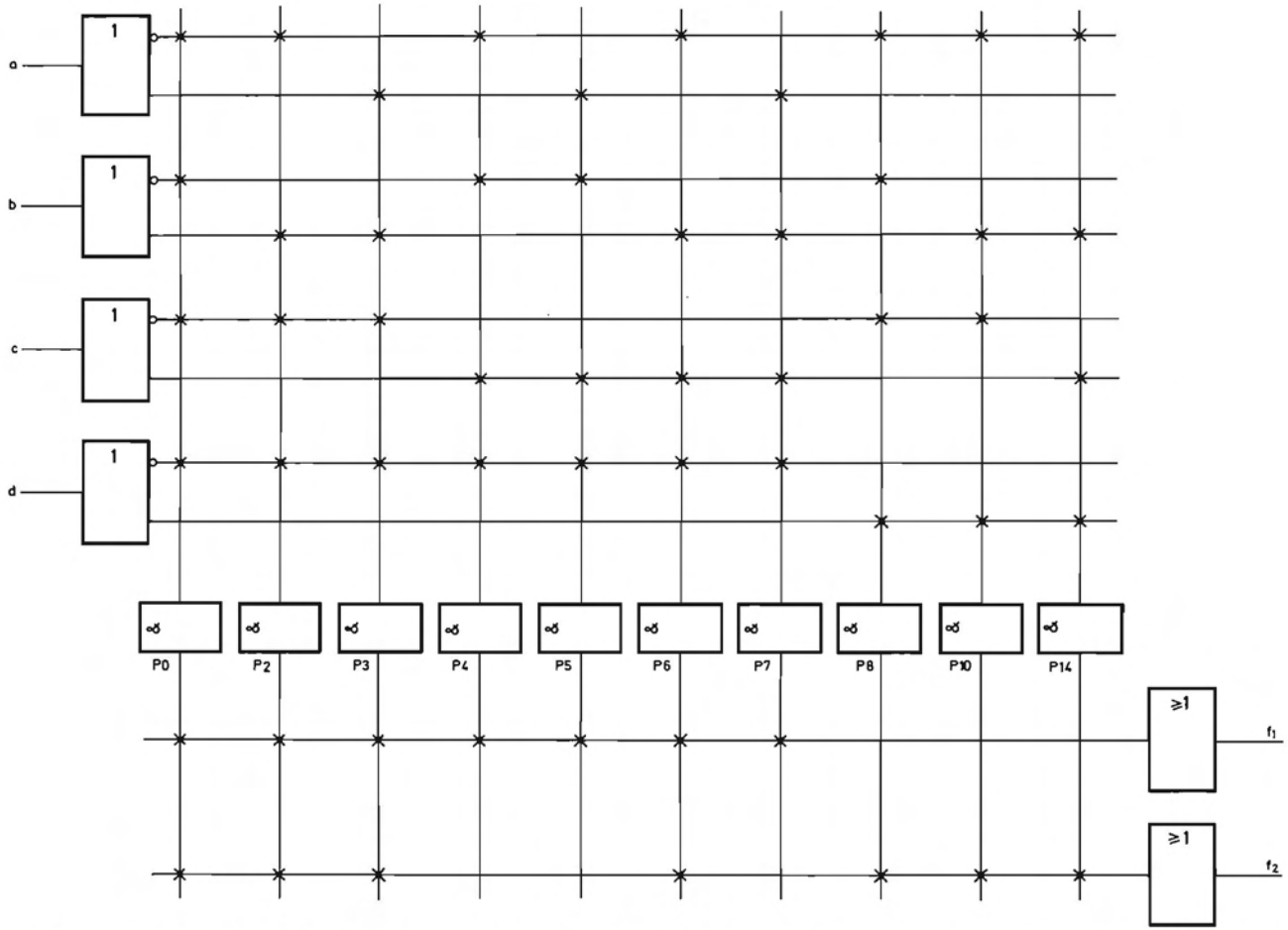


FIGURA 3.122.—**Matriz lógica de la figura 3.121 programada para realizar las funciones f_1 y f_2 cuyas tablas de verdad se representan en la tabla 3.32.**

$$
\begin{aligned}
P_0 &= \bar{a}\,\bar{b}\,\bar{c}\,\bar{d} \\
P_2 &= \bar{a}\,b\,\bar{c}\,\bar{d} \\
P_3 &= a\,b\,\bar{c}\,\bar{d} \\
P_4 &= \bar{a}\,\bar{b}\,c\,\bar{d} \\
P_5 &= a\,\bar{b}\,c\,\bar{d} \\
P_6 &= \bar{a}\,b\,c\,\bar{d} \\
P_7 &= a\,b\,c\,\bar{d} \\
P_8 &= \bar{a}\,\bar{b}\,\bar{c}\,d \\
P_{10} &= \bar{a}\,b\,\bar{c}\,d \\
P_{14} &= \bar{a}\,b\,c\,d
\end{aligned}
$$

En la figura 3.122 se representa la matriz lógica programable una vez programada para realizar las funciones f_1 y f_2. Cada puerta Y realiza uno de los productos lógicos que forman parte de una o ambas funciones f_1 y f_2. Por ejemplo, en la puerta Y que realiza el producto P_3 se suprimen las conexiones de sus entradas a las variables $\bar{a}$, $\bar{b}$, c y d y se mantienen las conexiones a las variables a, b, $\bar{c}$ y $\bar{d}$. Las entradas de las puertas O correspondientes a los productos que no forman parte de la función se suprimen también. Por ejemplo, en la puerta O que realiza f_1 se suprimen las uniones de las entradas correspondientes a P_8, P_{10} y P_{14}.

Pero la simplificación de las expresiones canónicas de suma de productos de f_1 y f_2 permite reducir las dimensiones de la matriz lógica programable necesaria. Dicha simplificación se puede realizar, considerándolas en conjunto como una multifunción, por el método tabular de Karnaugh descrito en el apartado 3.4.

En la figura 3.123 se representan las tablas de Karnaugh de suma de productos a partir de las cuales se deducen las expresiones:

$$
\begin{aligned}
f_1 &= \bar{a}\,\bar{d} + c\,\bar{d} + b\,\bar{c}\,\bar{d} \\
f_2 &= \bar{a}\,\bar{c} + \bar{a}\,b + b\,\bar{c}\,\bar{d}
\end{aligned}
$$

Partiendo de estas expresiones se deduce que la matriz lógica programable necesaria ha de tener solamente cinco puertas Y correspondientes a los cinco productos diferentes existentes en las expresiones de f_1 y f_2. El lector puede obtener fácilmente la matriz programada representada en la figura 3.124. En las puertas Y que realizan un producto lógico en el cual falta alguna variable, se suprimen las uniones correspondientes a dicha variable y su inversa.

Las matrices lógicas programables son los SCUPI más flexibles, porque en ellos es posible programar la conexión de cada producto a todas y cada una de las puertas O de salida, pero en contrapartida presentan la característica de que es necesario programar dos matrices.

3.8.2.2.2 Matrices lógicas Y-programables (PAL). El diagrama de bloques de las matrices lógicas Y-programables también coincide con el de la figura 3.115 pero se diferencian de las matrices lógicas programables en que las entradas de las puertas O están conectadas rígidamente a un determinado número de puertas Y. En general, si la PAL posee n' productos y m salidas, cada puerta O se conecta a n'/m productos diferentes. En la figura 3.125 se representa como ejemplo una

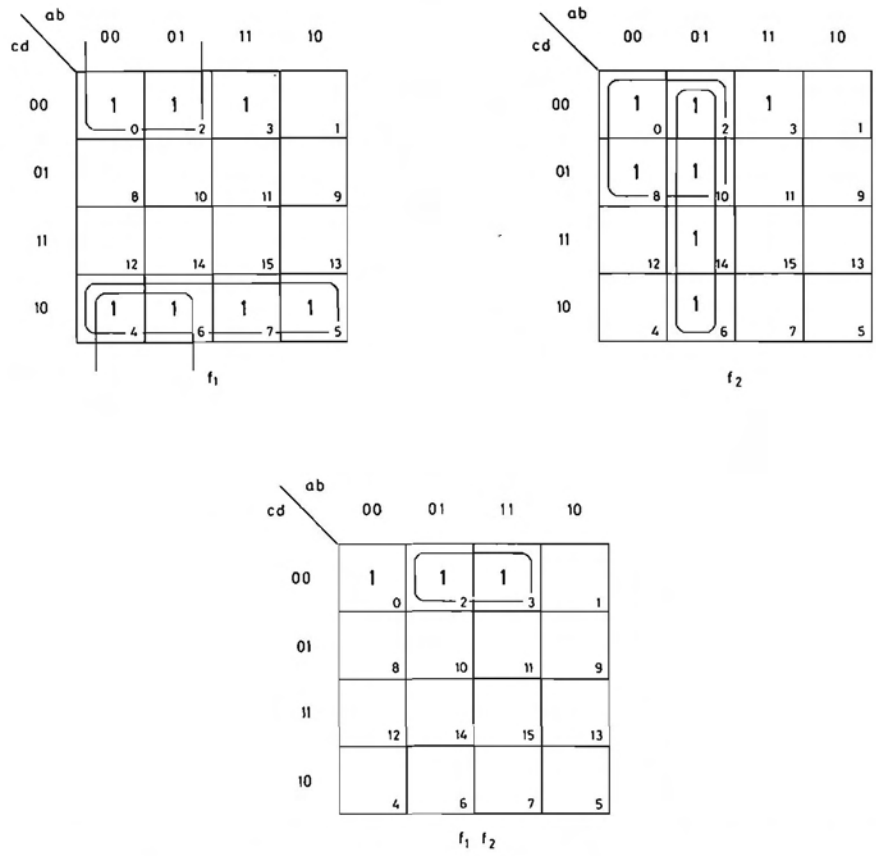


FIGURA 3.123.—**Tablas de Karnaugh de las funciones f_1 y f_2 cuyas tablas de verdad se representan en la tabla 3.32.**

PAL de doce productos lógicos y tres puertas O, conectadas a cuatro productos cada uno.

El nombre de PAL fue registrado por el primer fabricante que se decidió a integrarlas en un único circuito integrado (la empresa ya desaparecida «Monolithic Memories»). Su elección fue debida a que al estar registrado el nombre de PLA, al citado fabricante no se le ocurrió otra cosa que permutar la L por la A. Por ello el nombre de PAL es un ejemplo, cada vez más frecuente en microelectrónica, de la importancia de la imagen de marca en las denominaciones de los circuitos.

Las PAL son menos flexibles que las PLA, y necesitan más puertas Y que éstas porque si un producto lógico ha de formar parte de dos salidas, ha de ser programado dos veces. En compensación, las PAL poseen un menor tiempo de propagación, menor disipación y, lo que es más importante, ocupan una menor superficie de silicio, y su programación es evidentemente más sencilla.

En la figura 3.126 se representa el diagrama de bloques de una matriz lógica Y-

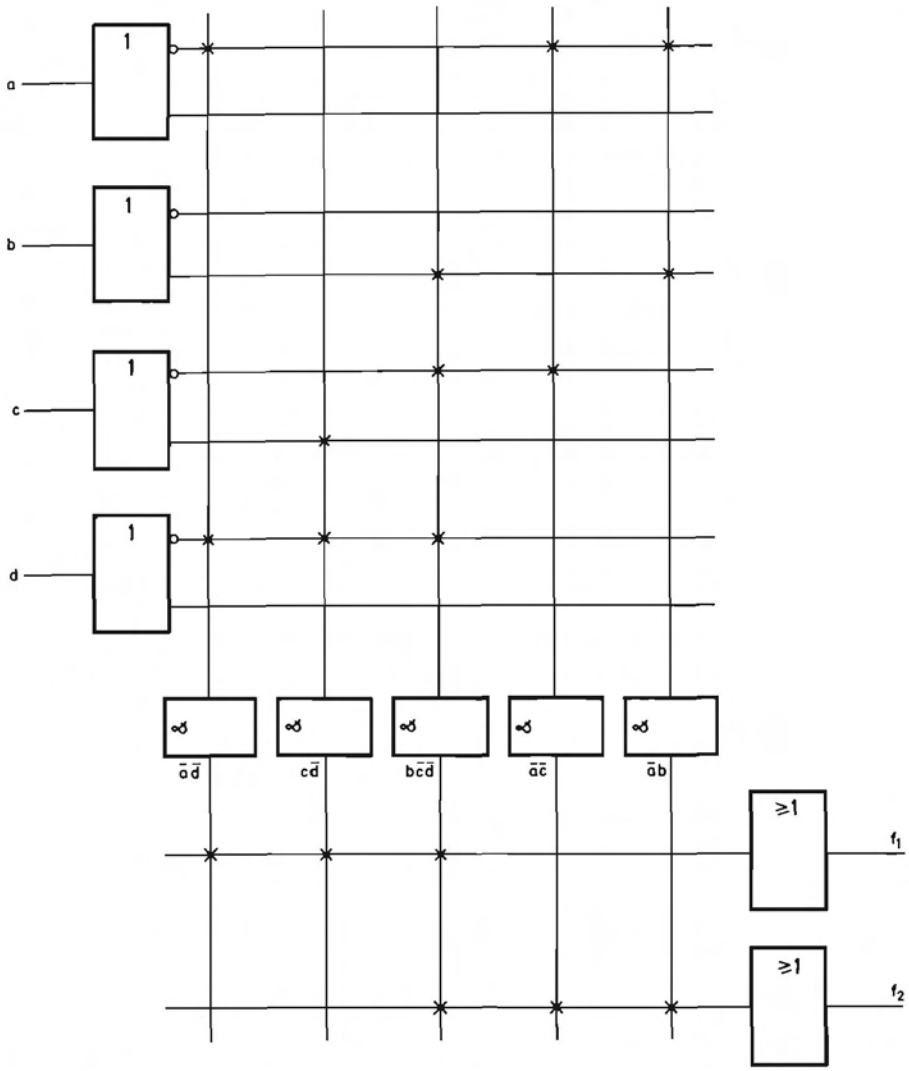


FIGURA 3.124.—**Matriz lógica programable programada para realizar las expresiones mínimas de suma de productos de f_1 y f_2.**

programable (PAL) de *n'* productos y *m* puertas O con *n'/m* entradas cada una y en la figura 3.127 el diagrama de bloques simplificado que constituye un símbolo lógico.

A continuación se expone mediante un ejemplo la forma de realizar una función lógica con una matriz lógica Y-programable.

Ejemplo 3.12: Realizar mediante una matriz lógica Y-programable las mismas funciones realizadas en el ejemplo 3.11 con una matriz lógica programable (PLA). Dichas funciones tienen las expresiones canónicas:

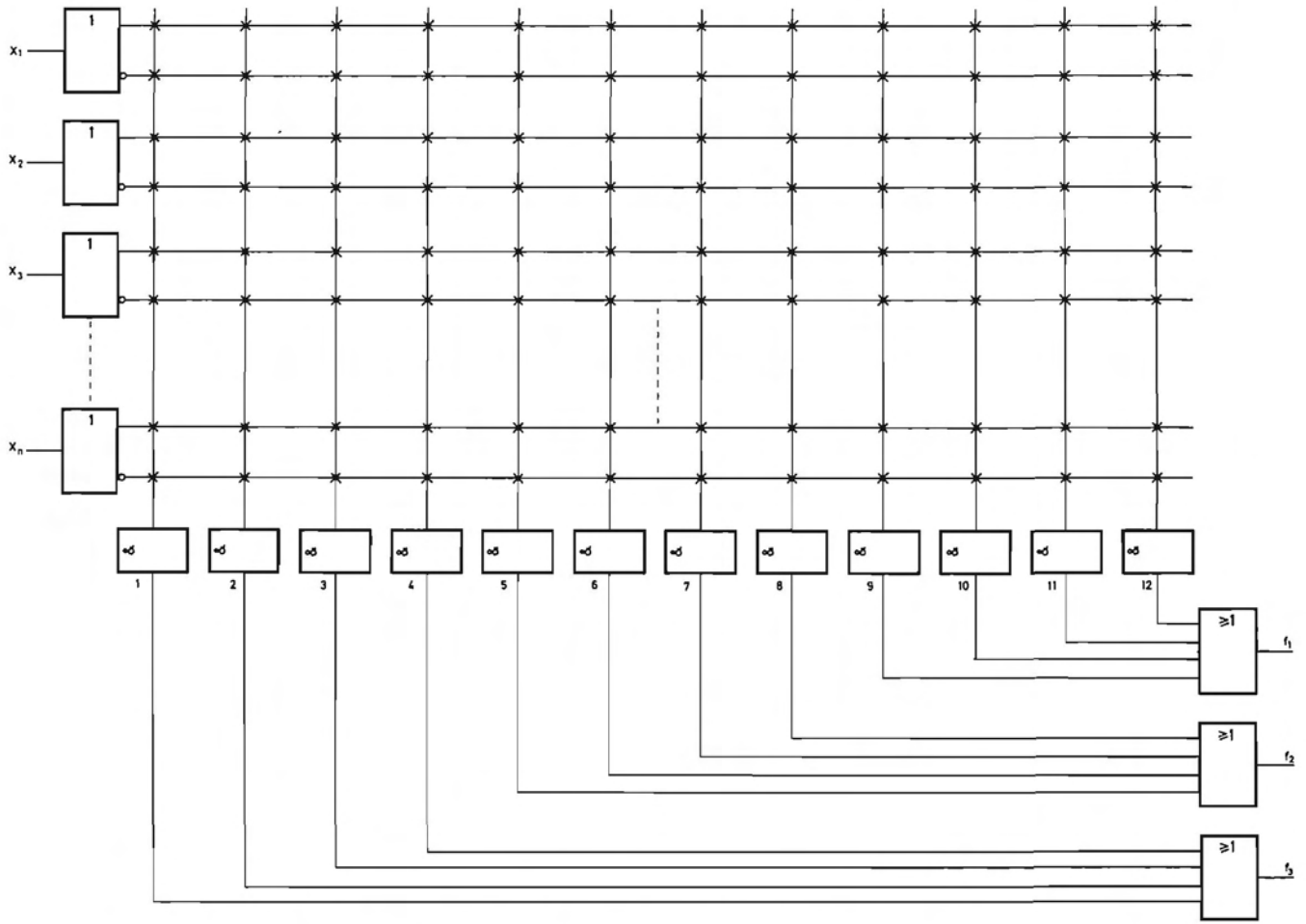


FIGURA 3.125.—**Matriz lógica Y-programable (PAL) de *n* variables de entrada, 12 productos y 3 variables de salida.**

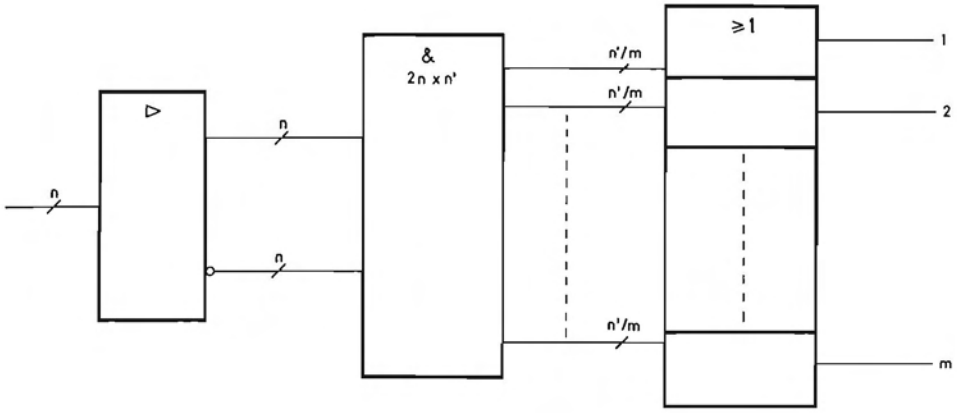


FIGURA 3.126.—**Diagrama de bloques de una matriz lógica Y-programable (PAL).**

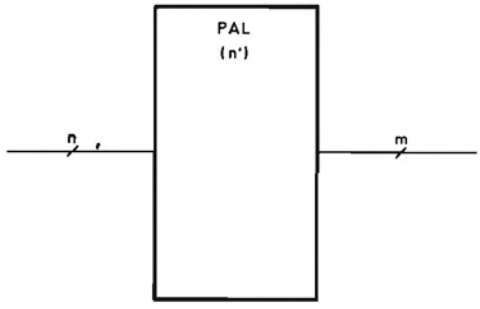


FIGURA 3.127.—**Símbolo lógico de una matriz lógica Y-programable (PAL).**

$$f_1 = \sum_4 (0,2,3,4,5,6,7)$$
$$f_2 = \sum_4 (0,2,3,6,8,10,14)$$

Dado que cada expresión tiene 7 productos, para realizarlas directamente con una PAL, ésta ha de tener como mínimo 14 puertas Y y 2 puertas O, de acuerdo con el esquema representado sin programar en la figura 3.128.

En la figura 3.129 se representa la PAL una vez programada para realizar las funciones f_1 y f_2. Se observa que existen dos puertas Y que realizan el producto P0 y otro tanto sucede con P2, P3 y P6.

La simplificación de las expresiones canónicas de suma de productos permite también reducir la complejidad de la PAL mínima necesaria.

Las expresiones mínimas de productos de sumas de f_1 y f_2, obtenidas a partir de las tablas de Karnaugh de la figura 3.123, son:

$$f_1 = \bar{a}\bar{d} + c\bar{d} + b\bar{c}\bar{d}$$
$$f_2 = \bar{a}\bar{c} + \bar{a}b + b\bar{c}\bar{d}$$

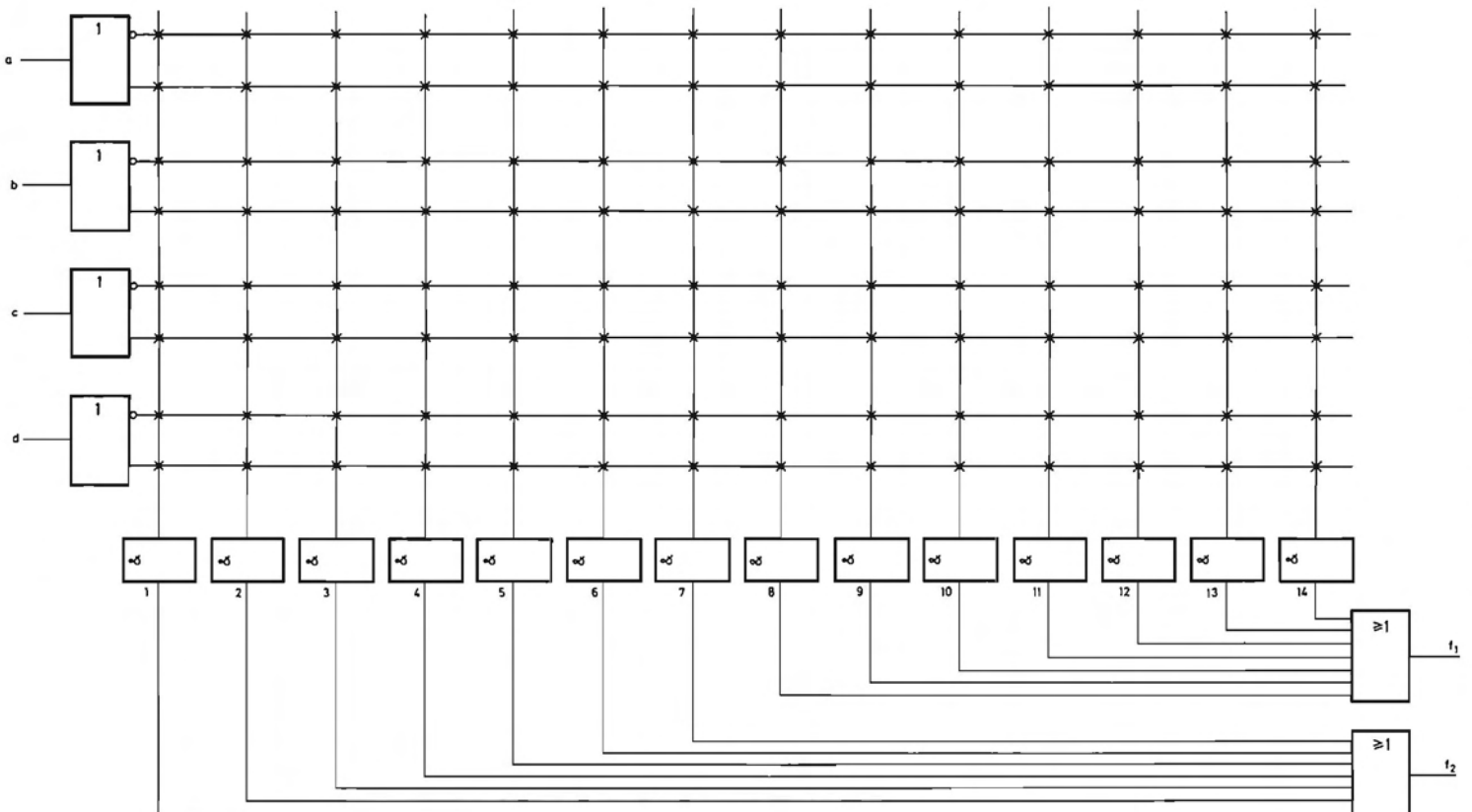


FIGURA 3.128.—**Matriz lógica Y-programable mínima necesaria para realizar las funciones f_1 y f_2 (tabla 3.32) sin programar.**

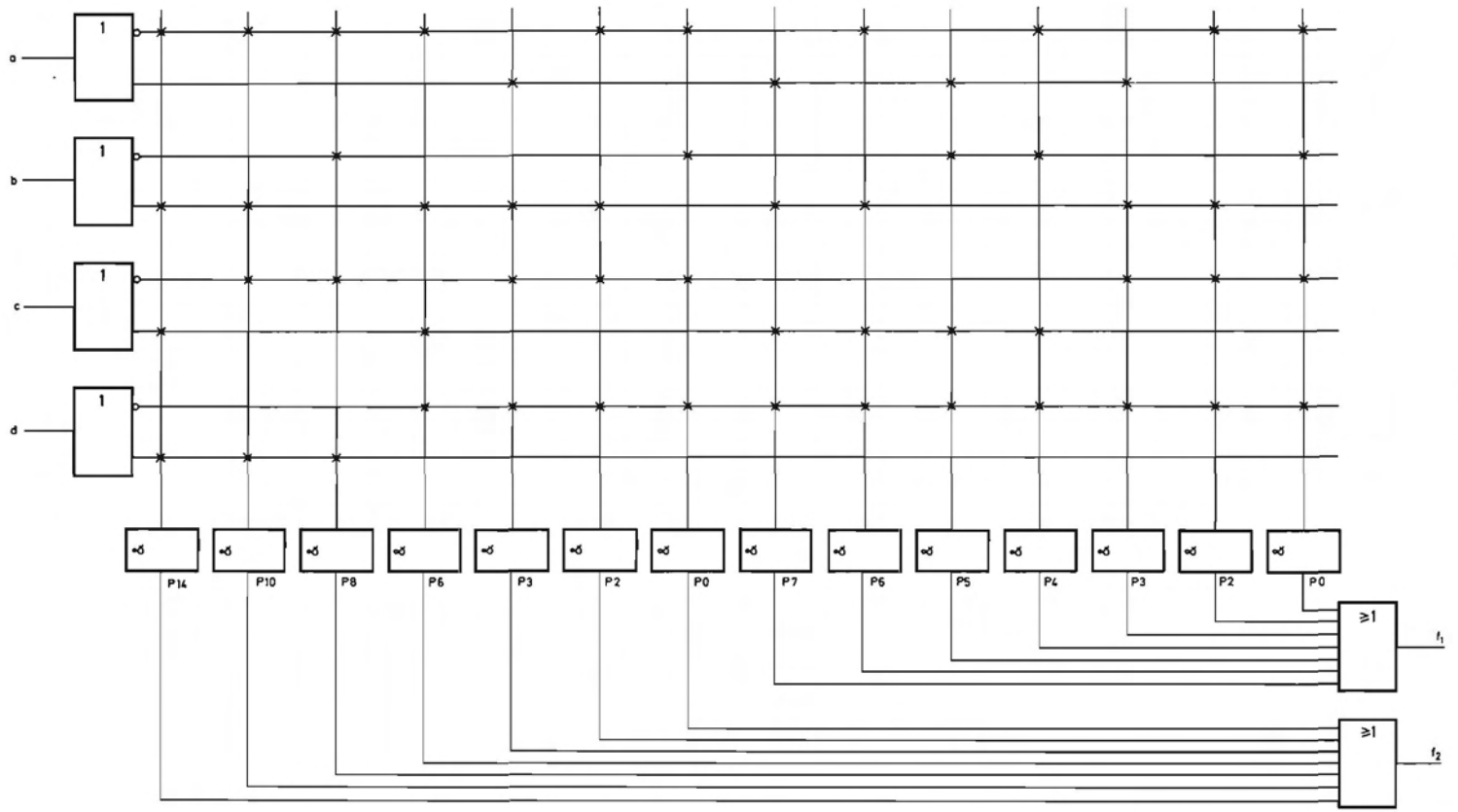


FIGURA 3.129.—**Matriz lógica Y-programable de la figura 3.128 programada para realizar las funciones f_1 y f_2.**

La matriz lógica Y-programable mínima necesaria ha de tener seis puertas Y. En la figura 3.130 se representa su esquema con la programación adecuada cuya comprobación se recomienda al lector.

Las matrices lógicas Y-programables (PAL) adquieren su verdadero interés combinándolas con registros y con un conjunto de recursos lógicos para obtener los denominados dispositivos lógicos programables (DLP) [en inglés «Programmable Logic Devices» (PLD)]. Por ello las PAL que poseen solamente una matriz de puertas Y y un conjunto de puertas O no se suelen realizar, en general, en la actualidad en circuitos independientes.

Los DLP permiten realizar tanto circuitos combinacionales como secuenciales

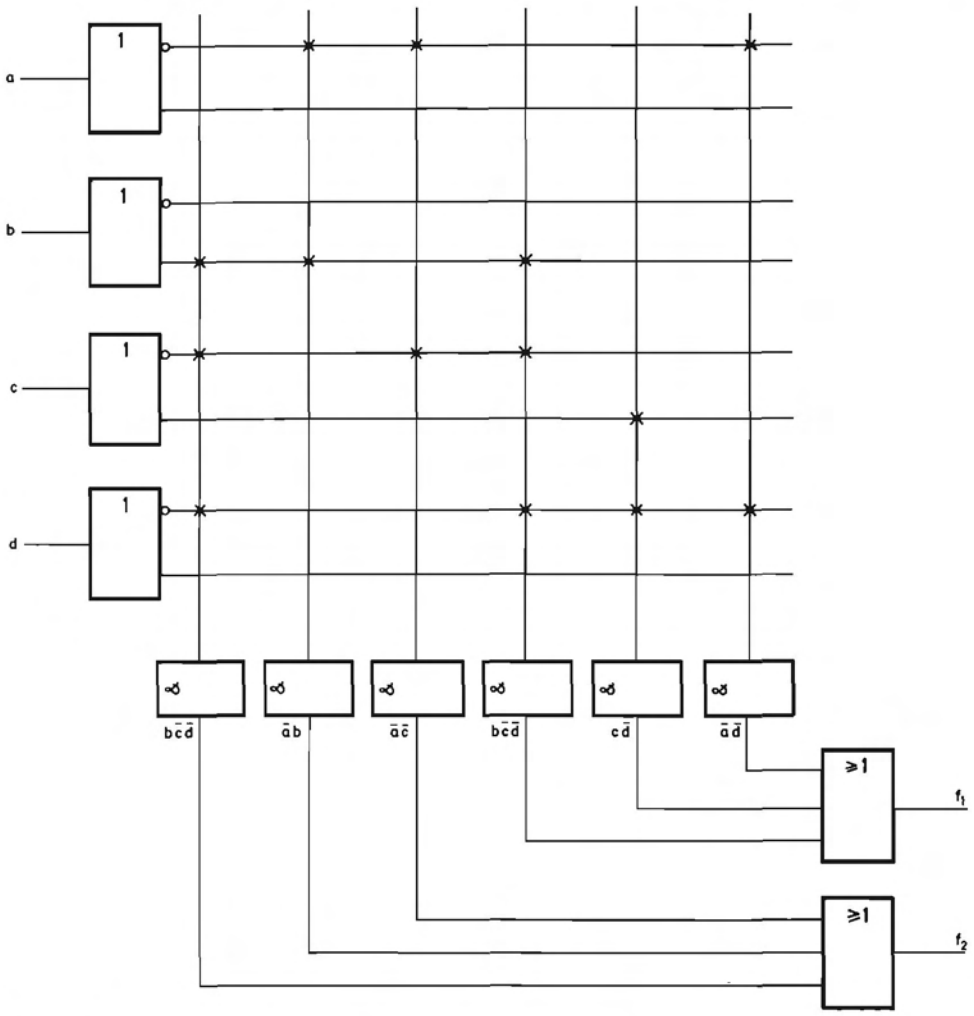


FIGURA 3.130.—**Matriz lógica Y-programable mínima programada para realizar las expresiones simplificadas de suma de productos de f_1 y f_2.**

y por ello su estudio se realiza en el capítulo 6. Los DLP son además circuitos integrados digitales monolíticos configurables y por ello se estudian también en el capítulo 5, en el apartado dedicado a los circuitos integrados digitales monolíticos normalizados. No obstante, un estudio en profundidad de los DLP se sale fuera de los límites de este libro. Al lector interesado se le remite a los libros «Controladores lógicos y autómatas programables» [MAND 92] y «Sistemas digitales configurables y sus aplicaciones. Tomo I: Dispositivos lógicos programables» desarrollados por varios profesores del Departamento de Tecnología Electrónica que, a su vez, son miembros del Instituto de Electrónica Aplicada Pedro Barrié de la Maza de la Universidad de Vigo.

3.8.2.2.3 Ampliación de la capacidad de las matrices lógicas programables (PLA) y las matrices lógicas Y-programables (PAL). Los bloques funcionales poseen un cierto número n de variables de entrada, n' de puertas Y y m de puertas O y variables de salida. En la práctica puede ser necesario realizar una PLA o una PAL de un número n_1 de variables de entrada superior a n, o un número n'_1 de productos superior a n' o un número de variables de salida m_1 superior a m.

Aunque la elevación de la capacidad de integración hace que los métodos de ampliación tengan cada vez menor importancia en la utilización de circuitos comerciales, es interesante conocerlos porque son útiles en las técnicas de diseño microelectrónico.

A continuación se analiza la ampliación de cada uno de los factores.

a) Elevación del número de productos.

Si el número n' de productos lógicos de un bloque funcional no es suficiente, se puede hacer igual a un múltiplo de n', conectando las variables de entrada a varios módulos y uniendo entre sí las salidas correspondientes a las funciones que utilicen productos de varios módulos (fig. 3.131). Para que esta unión sea posible es necesario que las puertas O del bloque funcional tengan una de las siguientes características:

a) Poseer como carga del transistor de salida una resistencia, tal como se describe en el apartado 5.4.4.3.2.
b) Poseer tres estados para lo cual el bloque funcional ha de tener una entrada de desinhibición/inhibición (*EN*) que cuando adopte un cierto estado lógico (por ejemplo el cero lógico), provoque el tercer estado de la salida y en caso contrario aparezca en ella el estado lógico producido por el circuito.

En la figura 3.132 se realiza el acoplamiento de dos matrices de n' productos para obtener una matriz de $2n'$ productos. Las entradas *EN* se conectan entre sí a través de un inversor. La variable X_{n+1} conectada a ellas ha de aparecer en forma directa en los productos de la matriz superior y en forma inversa en los de la matriz inferior. La variable que se conecte a la entrada *EN* debe ser, en general, aquella que haga que los números de productos de ambas matrices sean lo más próximos posibles. Un ejemplo aclarará lo que se acaba de exponer.

Ejemplo 3.13: Sea la función $f = \sum_3 (0, 1, 2, 4, 6)$ (tabla 3.33) que se desea

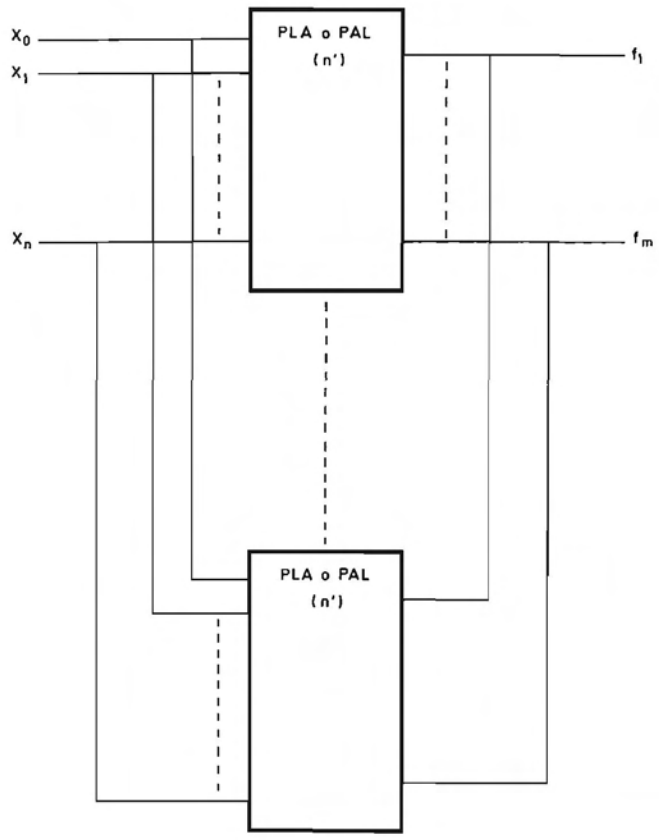


FIGURA 3.131.—**Elevación del número de productos de una PLA o una PAL.**

realizar en forma canónica mediante uña matriz lógica programable de dos variables de entrada y tres productos canónicos, que posee una salida de tres estados y una entrada de desinhibición (*EN*) que se supone que produce la salida de tres estados cuando se encuentra en nivel cero.

Las expresiones algebraicas de los productos canónicos de f son:

$$P_0 = \overline{a}\,\overline{b}\,\overline{c}$$
$$P_1 = a\,\overline{b}\,\overline{c}$$
$$P_2 = \overline{a}\,b\,\overline{c}$$
$$P_4 = \overline{a}\,\overline{b}\,c$$
$$P_6 = \overline{a}\,b\,c$$

Observando estos productos se comprueba que la variable b está en forma directa en dos de ellos y en forma inversa en tres, al igual que la variable c. Cualquiera de ellas puede ser utilizada para gobernar la entrada de inhibición. Por el contrario, la variable a no puede ser utilizada porque se encuentra en forma inver-

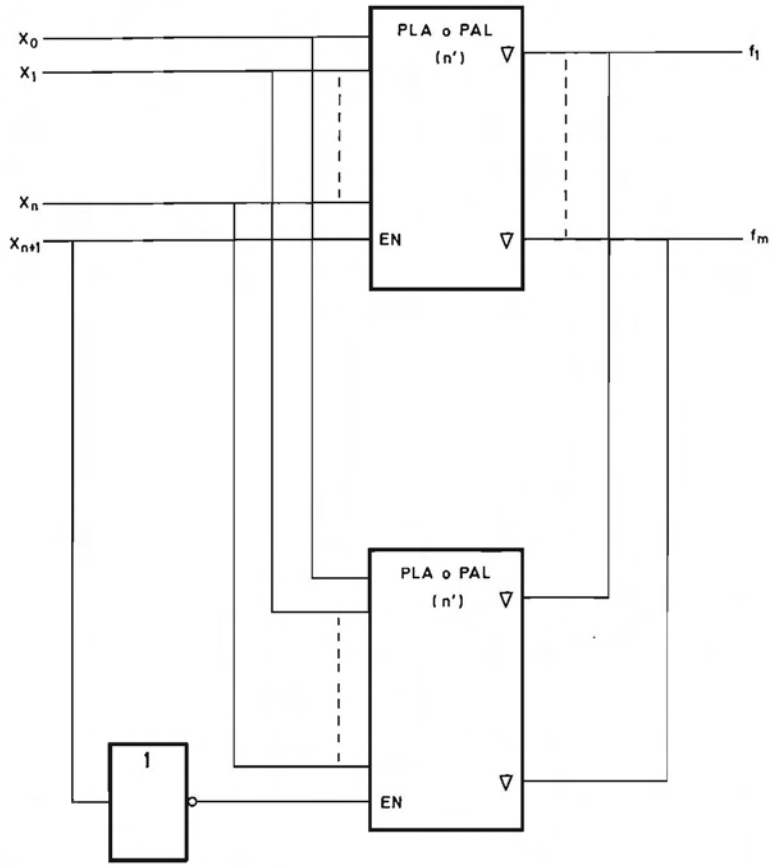


FIGURA 3.132.—**Elevación del número de productos de una PLA o una PAL que posee salida de tres estados.**

c	b	a	f_1
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

TABLA 3.33

sa en cuatro productos canónicos y se necesitaría una matriz lógica programable de esta capacidad.

En la figura 3.133 se indica el diagrama de bloques de la solución en la que se controla la entrada de inhibición con la variable *b*. La entrada de desinhibición *EN* de la matriz lógica programable *N*1 se une a la variable *b* y en ella se realizan los productos canónicos P_2 y P_6 y la entrada de desinhibición de la matriz lógica programada *N*2 se conecta a la variable *b* y en ella se realizan los productos canónicos P_0, P_1 y P_4.

b) Elevación del número de variables de salida.

Si el número *m* de variables de salida de un bloque funcional no es suficiente, se eleva el número de salidas acoplando las variables de entrada a los módulos ne-

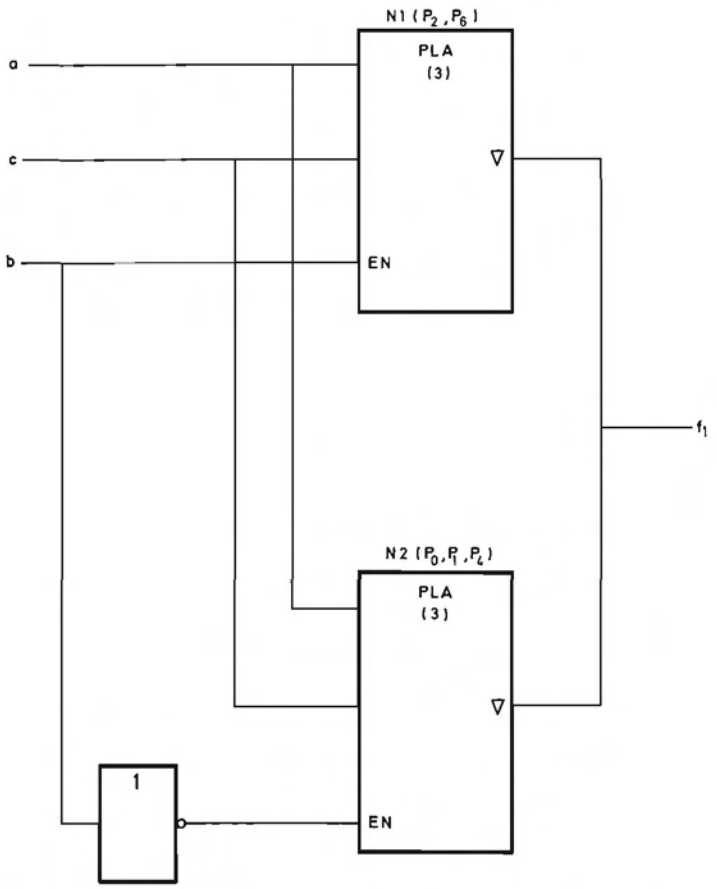


FIGURA 3.133.—**Realización de la función f_1 (tabla 3.33) mediante dos matrices lógicas programables de dos variables de entrada y salida de tres estados con entrada de desinhibición (*EN*).**

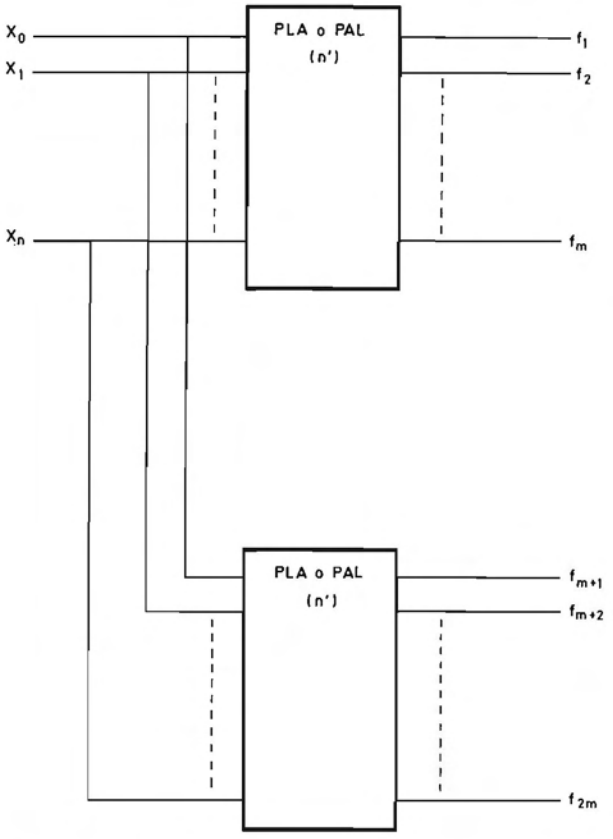


FIGURA 3.134.—**Elevación del número de variables de salida de una PLA o PAL.**

cesarios. En la figura 3.134 se supone que el número de variables de salida está comprendido entre m y $2m$.

c) Elevación del número de variables de entrada.

Si el número n de variables de entrada de un módulo funcional es insuficiente, se puede elevar mediante la entrada de inhibición. En el apartado *a*) se vio la forma de elevar el número de productos mediante dicha entrada y se observó que el número de variables de entrada posibles se eleva en una unidad. La utilización de un decodificador en combinación con las matrices lógicas programables (PLA o PAL) tal como se indica en la figura 3.135 permite elevar el número de variables de entrada. El lector no debe tener ningún problema para comprender su funcionamiento.

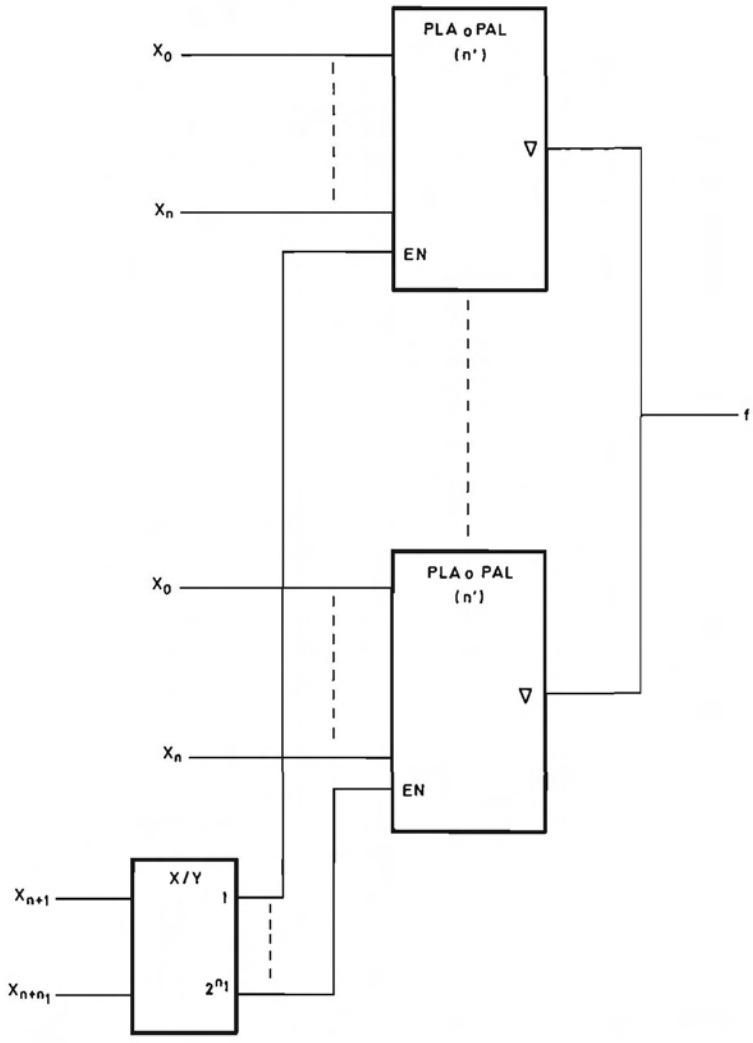


FIGURA 3.135.—**Elevación del número de variables de entrada de una PLA o PAL.**

3.8.2.2.4 Matrices lógicas de puertas universales. Las puertas NO-Y (NAND) y NO-O (NOR) son puertas universales porque mediante una combinación de cualquiera de ambas se puede realizar cualquier función lógica, tal como se indica en el apartado 3.5.1. Por ello, combinándolas con una matriz de conexiones programables se obtiene un sistema combinacional universal programable incompleto (SCUPI).

En la figura 3.136 se representa el esquema básico de un SCUPI formado por un conjunto de puertas NO-Y y una matriz de conexiones programables a cuyas filas

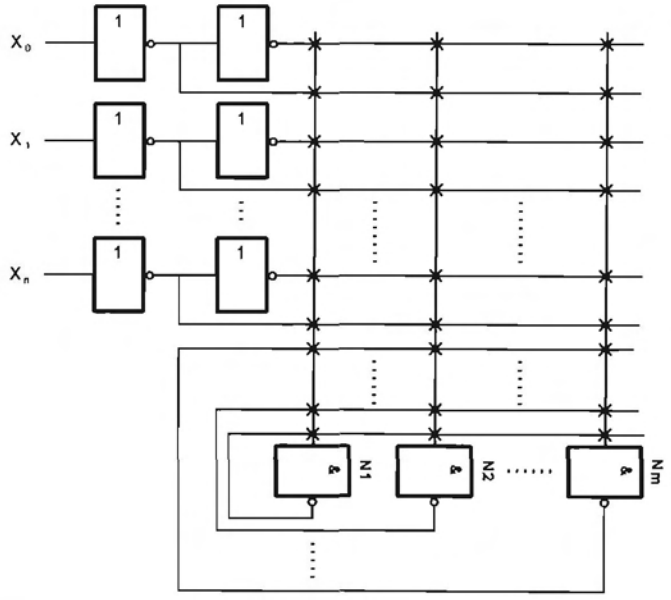


FIGURA 3.136.—**Matriz lógica programable de puertas NO-Y (NAND) realimentadas.**

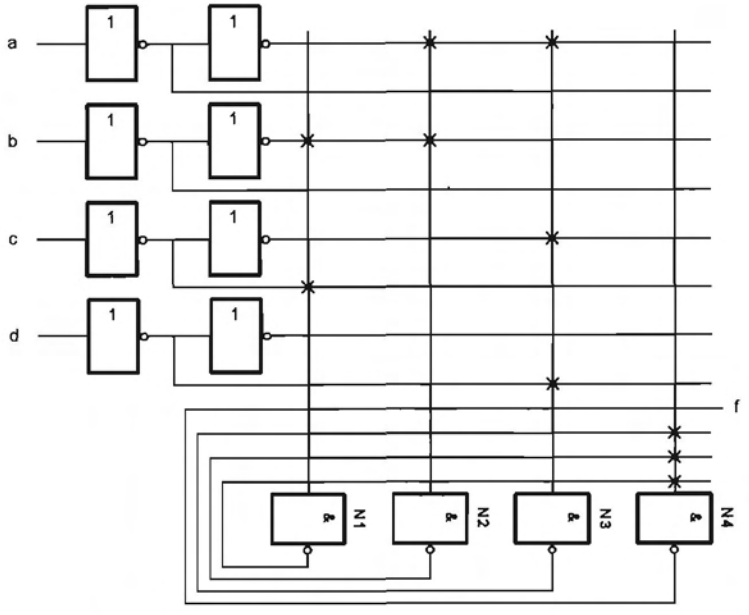


FIGURA 3.137.—**Generación de la función $f = b\bar{c} + ab + ac\bar{d}$ con una matriz lógica programable de puertas NO-Y (NAND) realimentadas.**

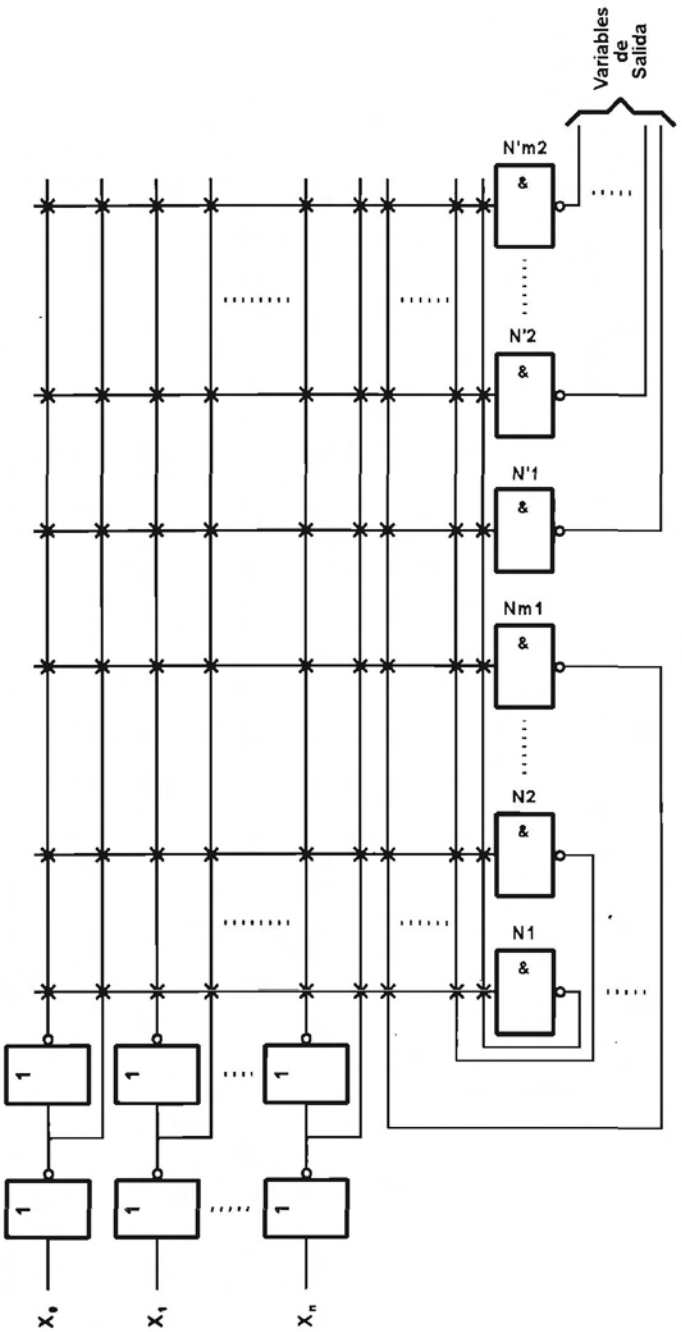


FIGURA 3.138.—Esquema de una matriz lógica con puertas NO-Y (NAND) realimentadas que posee una matriz de puertas de salida.

se conectan las variables de entrada (directas o invertidas) y las salidas de las puertas NO-Y.

Mediante la supresión de las conexiones adecuadas se puede obtener cualquier función lógica. En la figura 3.137 se representa una matriz de puertas NO-Y realimentadas que realiza la función $f = b\bar{c}+ab+ac\bar{d}$, que transformada adecuadamente se convierte en:

$$f = \overline{\overline{b\bar{c}}\;\overline{ab}\;\overline{ac\bar{d}}}$$

Para ello la matriz ha de tener como mínimo cuatro puertas NO-Y, una para realizar cada uno de los tres productos y la cuarta para generar f a partir de las otras.

En la figura 3.137 las puertas N1 a N3 generan los tres productos invertidos y N4 genera la función f.

En el ejemplo de la figura 3.137 se observa que la salida de N4 no se conecta a la entrada de ninguna puerta una vez realizada la programación. Por ello, para no complicar innecesariamente la matriz de conexión, el esquema real de una matriz lógica de puertas NO-Y es el representado en la figura 3.138 en la que hay dos conjuntos de puertas NO-Y:

— Un conjunto de m_1 puertas realimentadas.
— Un conjunto de m_2 puertas no realimentadas.

Las matrices lógicas programables realizadas con puertas NO-Y y NO-O, al igual que las PLA y las PAL se combinan con registros y otros recursos lógicos para obtener dispositivos lógicos programables. Para profundizar en su estudio se remite al lector a la misma bibliografía indicada anteriormente.

BIBLIOGRAFIA

[INTE 89] Programmable logic handbook. Intel Corporation. 1989.
[MAND 92] E. Mandado, J. Marcos, S.A. Pérez. Controladores lógicos y autómatas programables. Capítulo 1: Modularidad de entradas y salidas. 2ª edición. Editorial Marcombo. 1992.
[PHIL 89] Semi-custom Programmable Logic Devices. Philips. 1989.
[TEXA 85] The TTL data book. Volume 2. Fuse-programmable identity comparators. SN74ALS526. Texas Instruments. 1984.
[CYPR 89] CMOS, BICMOS Data book. Cypress Semiconductor 1989.

PROBLEMAS

1. En un registro de cuatro bits cuyas salidas están disponibles al exterior se almacena información en el código BCD Aiken.
   a) Realizar la tabla de verdad de un circuito que detecte que el número contenido en el registro es superior a 7 o inferior a 3.
   b) Minimizar la expresión algebraica de este circuito por el método de Karnaugh o numérico.
   c) Realizar la expresión mínima con puertas NO-Y y NO-O.
   d) Realizar este circuito con un multiplexor de ocho canales.
   e) Realizar este circuito con una memoria pasiva.
   f) Realizar este circuito con una matriz lógica programable (PLA) y una matriz lógica Y-programable (PAL).

2. Generar la función:

$$f = \prod_{4} (1, 3, 7, 8, 10) \prod_{8} (0, 5, 6, 14, 15)$$

con un decodificador y las puertas NO-O necesarias.

3. *a*) Obtener la expresión algebraica mínima de una función lógica de cuatro variables que toma el valor lógico uno cuando el número de variables que están en estado uno es superior al de las que se encuentran en estado cero. Nunca pueden estar más de tres variables en estado uno.
b) Realizar la expresión obtenida con puertas NO-O y NO-Y.
c) Obtener la expresión mínima de esta función con la función O-exclusiva.

4. *a*) Realizar la tabla de verdad de un convertidor del código BCD natural al BCD exceso tres.
b) Minimizar las expresiones algebraicas por el método tabular de Karnaugh y el método numérico.
c) Realizar este convertidor con puertas NO-Y y NO-O.
d) Realizar este convertidor con circuitos multiplexores.
e) Realizar este convertidor con una matriz lógica Y-programable (PAL).

5. Realizar un codificador con prioridad de 16 variables de entrada mediante el codificador de la figura 3.72 cuya tabla de verdad se representa en la tabla 3.23. Utilícense las puertas NO-Y que sean necesarias.

6. Realizar un decodificador del código ASCII de 6 bits (tabla 1.8) utilizando al máximo los circuitos decodificadores de escala de integración media.

7. En un registro de 6 bits cuyas salidas están disponibles en paralelo se almacena información en el código ASCII. Diseñar funciones lógicas que adopten el estado lógico uno, cuando la información contenida en dicho registro corresponda:

a) A un carácter numérico (0 al 9).
b) A un carácter alfabético.
c) A un carácter especial.

Realizar la síntesis de estas funciones en primer lugar con puertas NO-Y o NO-O y después con circuitos multiplexores.

8. a) Realizar por el método numérico la síntesis de la multifunción:

$$f_1 (a, b, c, d) = \prod_{4} (1, 2, 3, 4, 5, 12, 13, 14)$$

$$f_2 (a, b, c, d) = \prod_{4} (1, 2, 3, 7, 8, 9, 12, 13, 14)$$

$$f_3 (a, b, c, d) = \prod_{4} (0, 3, 8, 9)$$

b) Realizar el circuito con puertas NO-Y y NO-O.
c) Realizar el circuito utilizando el montaje «Y por conexión».
d) Realizar el circuito con una matriz lógica programable (PLA)

9. Diseñar un convertidor del código decimal (uno entre diez) al BCD natural con puertas NO-Y. Se supone que las entradas decimales son activas con un cero lógico.