

UNIVERSIDAD NACIONAL DE CÓRDOBA

FACULTAD DE MATEMÁTICA ASTRONOMÍA, FÍSICA Y  
COMPUTACIÓN.

ORGANIZACIÓN DEL COMPUTADOR

TEÓRICOS:

PABLO A. FERREYRA – NICOLAS WOLOVIC

PRÁCTICOS:

DELFINA VELEZ

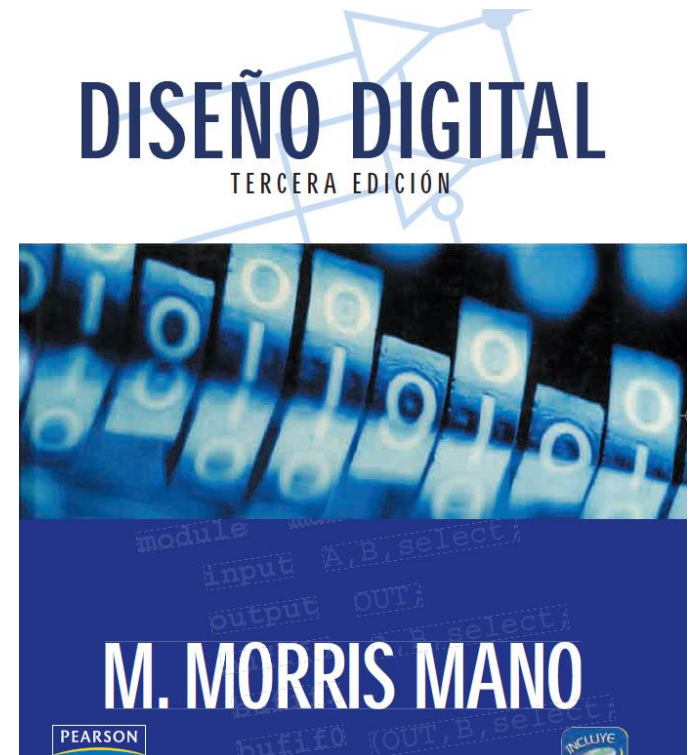
AGUSTÍN LAPROVITA

GONZALO VODANOVICK



# Algebra Booleana y Computetas

- ▶ Para facilitar el seguimiento del tema nos basaremos fielmente en el siguiente libro



# Algebra Booleana y Computetas

- ▶ Entiendo que está en Biblioteca y si no por favor pídanlo a los profesores del práctico.

**DISEÑO DIGITAL**

TERCERA EDICIÓN

**M. Morris Mano**

CALIFORNIA STATE UNIVERSITY, LOS ANGELES

**TRADUCCIÓN**

Roberto Escalona García  
*Ingeniero Químico*  
*Universidad Nacional Autónoma de México*

**REVISIÓN TÉCNICA**

Gonzalo Duchén Sánchez  
*Sección de Estudios de Postgrado e Investigación*  
*Escuela Superior de Ingeniería Mecánica y Eléctrica*  
*Unidad Culhuacán*  
*Instituto Politécnico Nacional*

# Libro de Base para este Tema

- ▶ En particular este tema se desarrolla en el capítulo 2 del libro:

## 2      **ÁLGEBRA BOOLEANA Y COMPUERTAS LÓGICAS**      33

---

2-1	Definiciones básicas	33
2-2	Definición axiomática del álgebra booleana	34
2-3	Teoremas y propiedades básicos del álgebra booleana	37
2-4	Funciones booleanas	40
2-5	Formas canónicas y estándar	44
2-6	Otras operaciones lógicas	51
2-7	Compuertas lógicas digitales	53
2-8	Circuitos integrados	59

# Función O-Exclusivo

## 3-8 FUNCIÓN OR EXCLUSIVO

---

La función OR exclusivo (XOR), denotada por el símbolo  $\oplus$ , es una operación lógica que efectúa la operación booleana siguiente:

$$x \oplus y = xy' + x'y$$

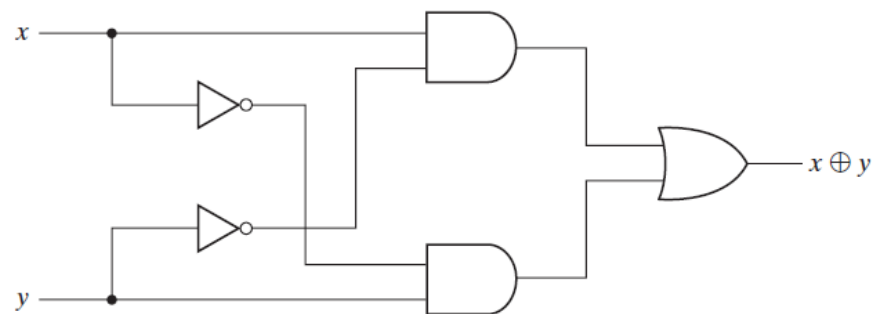
Es igual a 1 si sólo  $x$  es igual a 1 o sólo  $y$  es igual a 1, pero no si ambas son 1. El NOR exclusivo, también llamado equivalencia, realiza la operación booleana siguiente:

$$(x \oplus y)' = xy + x'y'$$

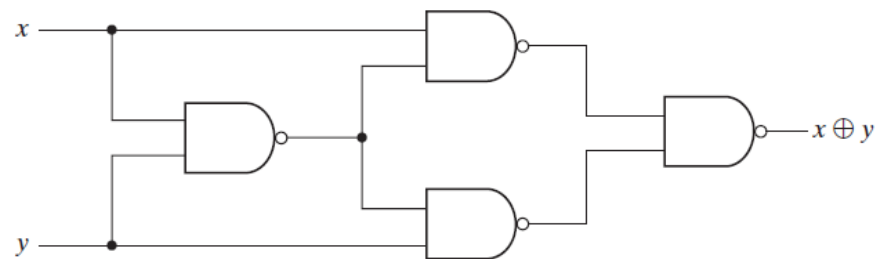
Es igual a 1 si tanto  $x$  como  $y$  son 1 o si ambas son 0. Se puede demostrar que el NOR exclusivo es el complemento del OR exclusivo con la ayuda de una tabla de verdad o por manipulación algebraica:

$$(x \oplus y)' = (xy' + x'y)' = (x' + y)(x + y') = xy + x'y'$$

# Función O-Exclusivo



a) Con compuertas AND-OR-NOT



b) Con compuertas NAND

**FIGURA 3-32**  
Implementaciones del OR exclusivo

# Funciones Par e Impar

## 96 Capítulo 3 Minimización en el nivel de compuertas

		$BC$		$B$	
		00	01	11	10
$A$	0		1		1
	1	1		1	

a) Función impar  
 $F = A \oplus B \oplus C$

		$BC$		$B$	
		00	01	11	10
$A$	0	1		1	
	1		1		1

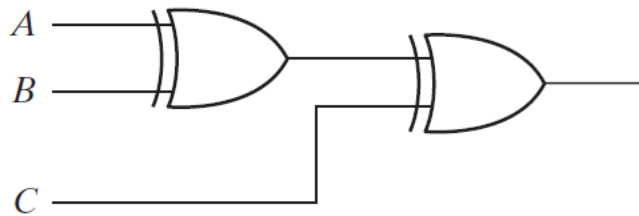
b) Función par  
 $F = (A \oplus B \oplus C)'$

**FIGURA 3-33**

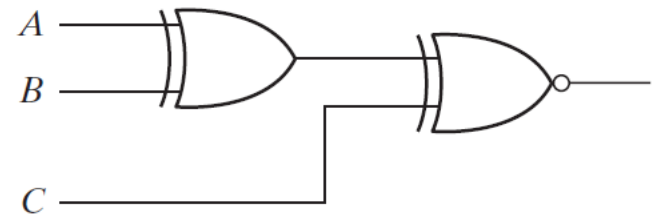
Mapa para una función OR exclusivo de tres variables

# Funciones Par e Impar

$$\begin{aligned} A \oplus B \oplus C &= (AB' + A'B)C' + (AB + A'B')C \\ &= AB'C' + A'BC' + ABC + A'B'C \\ &= \Sigma(1, 2, 4, 7) \end{aligned}$$



a) Función impar de tres entradas



b) Función par de tres entradas

**FIGURA 3-34**

Diagrama lógico de funciones impar y par



# Función Par e Impar

		$C$			
		$CD$			
$AB$	00		1		1
	01	1		1	
	11		1		1
	10	1		1	

$A$  { 00, 01, 11, 10 }  
 $B$  { 00, 01, 11, 10 }  
 $D$  { 00, 01, 11, 10 }

a) Función impar  
 $F = A \oplus B \oplus C \oplus D$

		$C$			
		$CD$			
$AB$	00	1		1	
	01		1		1
	11	1		1	
	10		1		1

$A$  { 00, 01, 11, 10 }  
 $B$  { 00, 01, 11, 10 }  
 $D$  { 00, 01, 11, 10 }

b) Función par  
 $F = (A \oplus B \oplus C \oplus D)'$

**FIGURA 3-35**

Mapa de una función OR exclusivo de cuatro variables

# Función Par e Impar

$$\begin{aligned} A \oplus B \oplus C \oplus D &= (AB' + A'B) \oplus (CD' + C'D) \\ &= (AB' + A'B)(CD + C'D') + (AB + A'B')(CD' + C'D) \\ &= \Sigma(1, 2, 4, 7, 8, 11, 13, 14) \end{aligned}$$

# Generador de Paridad Par

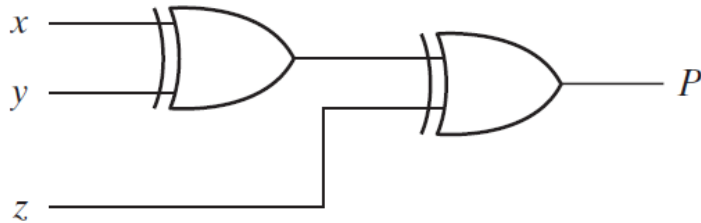
98      Capítulo 3    Minimización en el nivel de compuertas

**Tabla 3-4**

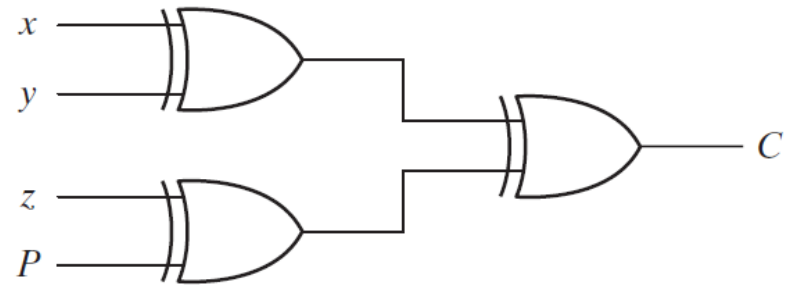
*Tabla de verdad de un generador de paridad par*

Mensaje de tres bits			Bit de paridad
<i>x</i>	<i>y</i>	<i>z</i>	<i>P</i>
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

# Generador de Paridad Par



a) Generador de paridad par de tres bits



b) Verificador de paridad par de cuatro bits

**FIGURA 3-36**

Diagrama lógico de un generador y un verificador de paridad

# Generador de Paridad

**Tabla 3-5**

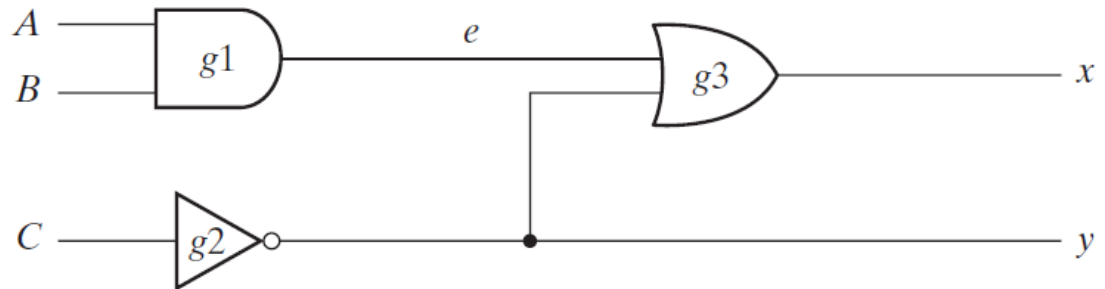
*Tabla de verdad de un verificador de paridad par*

Cuatro bits recibidos				Verificador de errores de paridad
<i>x</i>	<i>y</i>	<i>z</i>	<i>P</i>	<i>C</i>
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

# Lenguajes de Descripción de Hardware

## Ejemplo HDL 3-1

```
//Descripción del circuito simple de la fig. 3-37
module circuito_smpl (A,B,C,x,y);
    input A,B,C;
    output x,y;
    wire e;
    and g1 (e,A,B);
    not g2 (y, C);
    or g3 (x,e,y);
endmodule
```



**FIGURA 3-37**  
Circuito para ilustrar HDL

# Lenguajes de Descripción de Hardware

## 102      Capítulo 3    Minimización en el nivel de compuertas

### Retardos de compuerta

Cuando se usa HDL para hacer simulaciones, tal vez sea necesario especificar la magnitud del retardo que hay entre la entrada y la salida de las compuertas. En Verilog, el retardo se especifica en términos de *unidades de tiempo* y el símbolo #. La asociación de la unidad de tiempo con el tiempo físico se efectúa con la directriz de compilador **``timescale`**. (Las directrices al compilador inician con el símbolo ``` (acento grave).) Tales directrices se especifican antes de declarar módulos. Un ejemplo de directriz de escala de tiempo es:

```
`timescale 1ns/100ps
```

# Lenguajes de Descripción de Hardware

## Ejemplo HDL 3-2

---

```
//Descripción de circuito con retardo
module circuito_con_retardo(A,B,C,x,y);
    input A,B,C;
    output x,y;
    wire e;
    and #(30) g1(e,A,B);
    or #(20) g3(x,e,y);
    not #(10) g2(y,C);
endmodule
```

---



# Lenguajes de Descripción de Hardware

**Tabla 3-6**

*Salida de las compuertas después del retardo*

	<b>Unidades de tiempo</b>	<b>Entrada</b>	<b>Salida</b>
	<b>(ns)</b>	<b>ABC</b>	<b>y e x</b>
Inicial	—	000	1 0 1
Cambio	—	111	1 0 1
	10	111	0 0 1
	20	111	0 0 1
	30	111	0 1 0
	40	111	0 1 0
	50	111	0 1 1

# Lenguajes de Descripción de Hardware

## Ejemplo HDL 3-3

---

//Estímulo para el circuito simple

**module** stimcrt;

**reg** A,B,C;

**wire** x,y;

circuito\_con\_retardo ccr(A,B,C,x,y);

**initial**

**begin**

A = 1'b0; B = 1'b0; C = 1'b0;

#100

A = 1'b1; B = 1'b1; C = 1'b1;

#100 \$finish;

**end**

**endmodule**

//Descripción de circuito con retardo

**module** circuito\_con\_retardo (A,B,C,x,y);

**input** A,B,C;

**output** x,y;

**wire** e;

**and** #(30) g1(e,A,B);

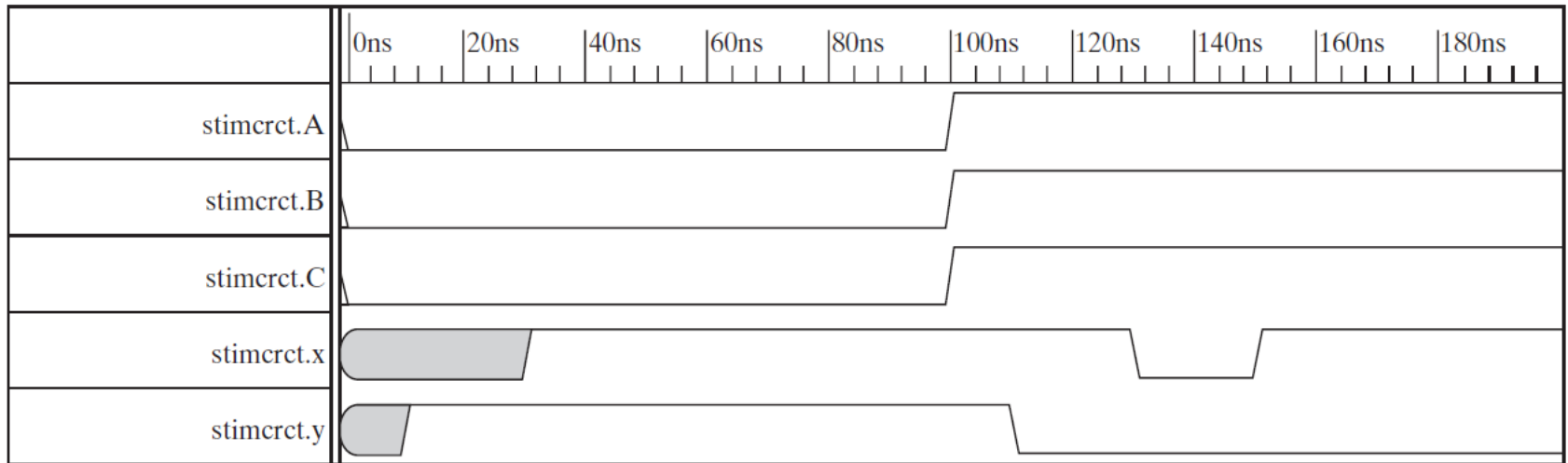
**or** #(20) g3(x,e,y);

**not** #(10) g2(y,C);

**endmodule**

---

# Lenguajes de Descripción de Hardware



**FIGURA 3-38**

Salida de la simulación del ejemplo HDL 3-3

# Lenguajes de Descripción de Hardware

## Ejemplo HDL 3-4

---

```
//Circuito especificado con expresiones booleanas
module circuit_bln (x,y,A,B,C,D);
    input A,B,C,D;
    output x,y;
    assign x = A | (B & C) | (~B & D);
    assign y = (~B & C) | (B & ~C & ~D);
endmodule
```

---

# Lenguajes de Descripción de Hardware

## Ejemplo HDL 3-5

---

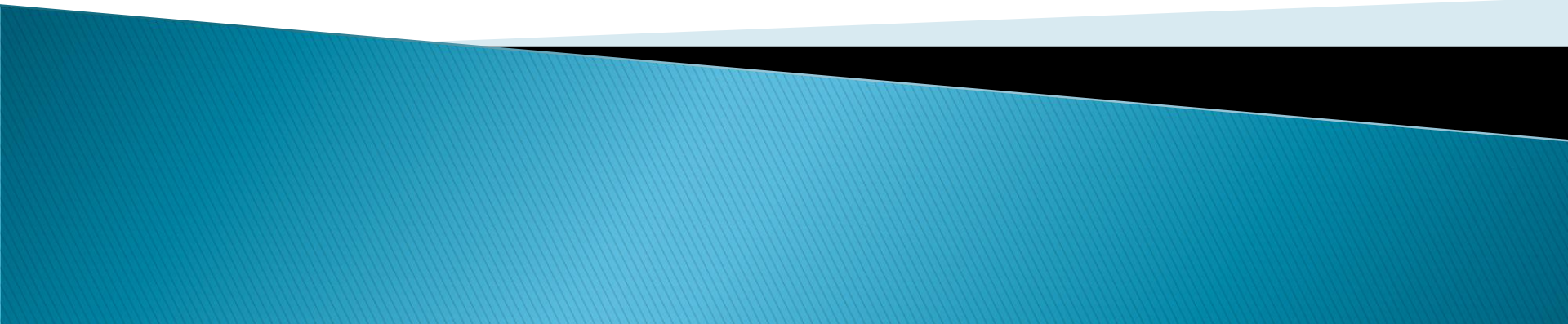
```
//Primitiva definida por el usuario (UDP)
primitive crctp (x,A,B,C);
    output x;
    input A,B,C;
//Tabla de verdad para  $x(A,B,C) = \Sigma(0,2,4,6,7)$ 
    table
//      A   B   C   :   x   (Esto es sólo un comentario)
      0   0   0   :   1;
      0   0   1   :   0;
      0   1   0   :   1;
      0   1   1   :   0;
      1   0   0   :   1;
      1   0   1   :   0;
      1   1   0   :   1;
      1   1   1   :   1;
    endtable
endprimitive

//Crear una copia de la primitiva
module declare_crctp;
    reg x,y,z;
    wire w;
    crctp (w,x,y,z);
endmodule
```

---

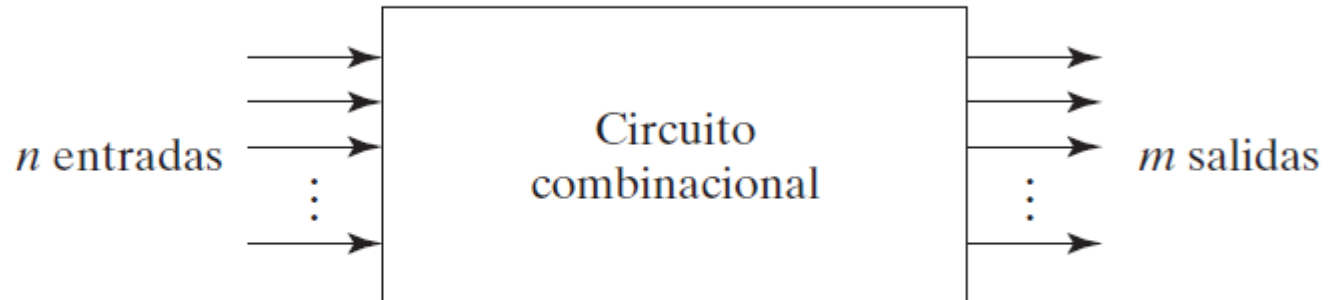
# LÓGICA COMBINACIONAL

## ALGUNOS MÓDULOS BÁSICOS



# Circuitos Combinacionales

## ► Esquema General

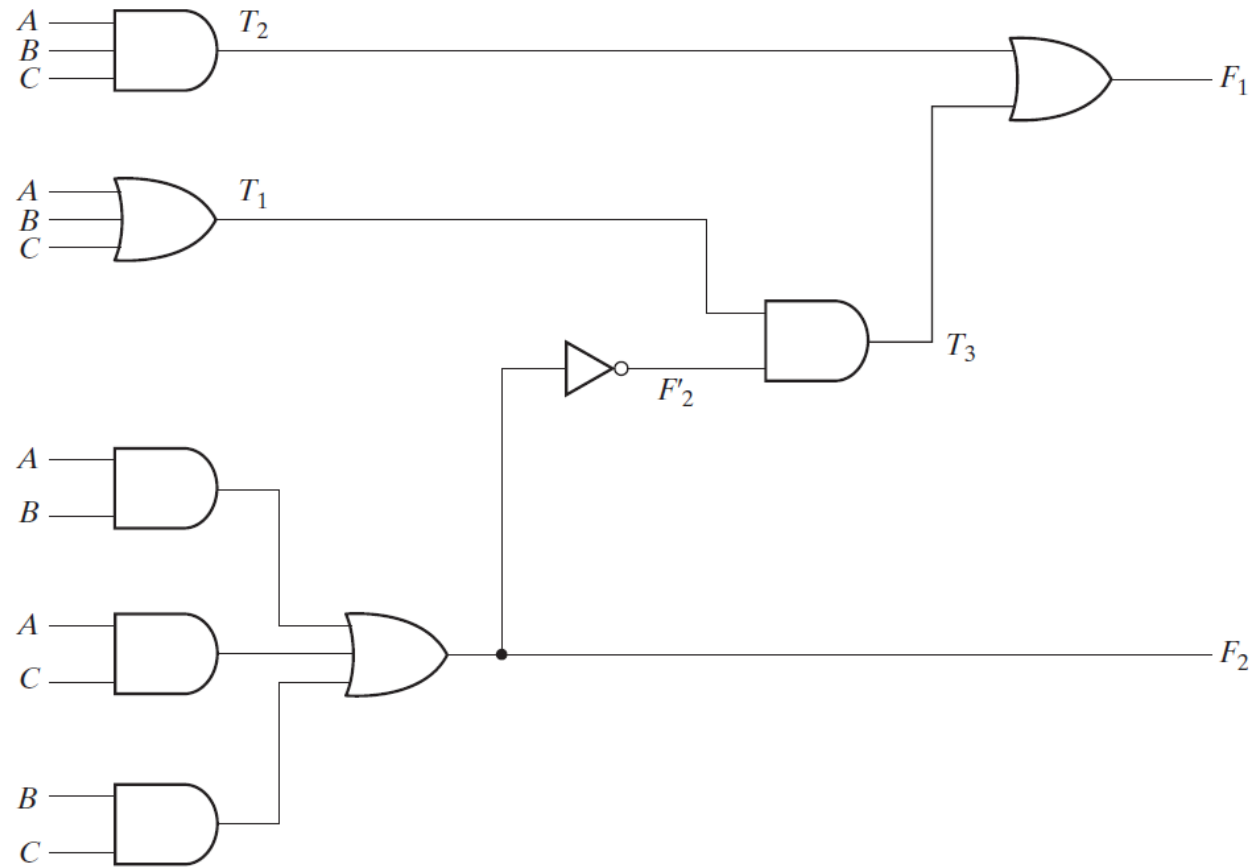


**FIGURA 4-1**

Diagrama de bloques de un circuito combinacional

# Circuitos Combinacionales

## ► Análisis



**FIGURA 4-2**

Diagrama lógico para el ejemplo de análisis



# Circuitos Combinacionales

## ► Ejemplo de Diseño

**Tabla 4-2**

*Tabla de verdad para el ejemplo de conversión de código*

Entrada BCD				Salida código exceso-3			
A	B	C	D	w	x	y	z
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0

# Circuitos Combinacionales

## ► Ejemplo de Diseño

Sección 4-3 Procedimiento de diseño

117

		$CD$		$C$	
		00	01	11	10
$AB$	00	1			1
	01	1			1
	11	$X$	$X$	$X$	$X$
	10	1		$X$	$X$
		$D$			

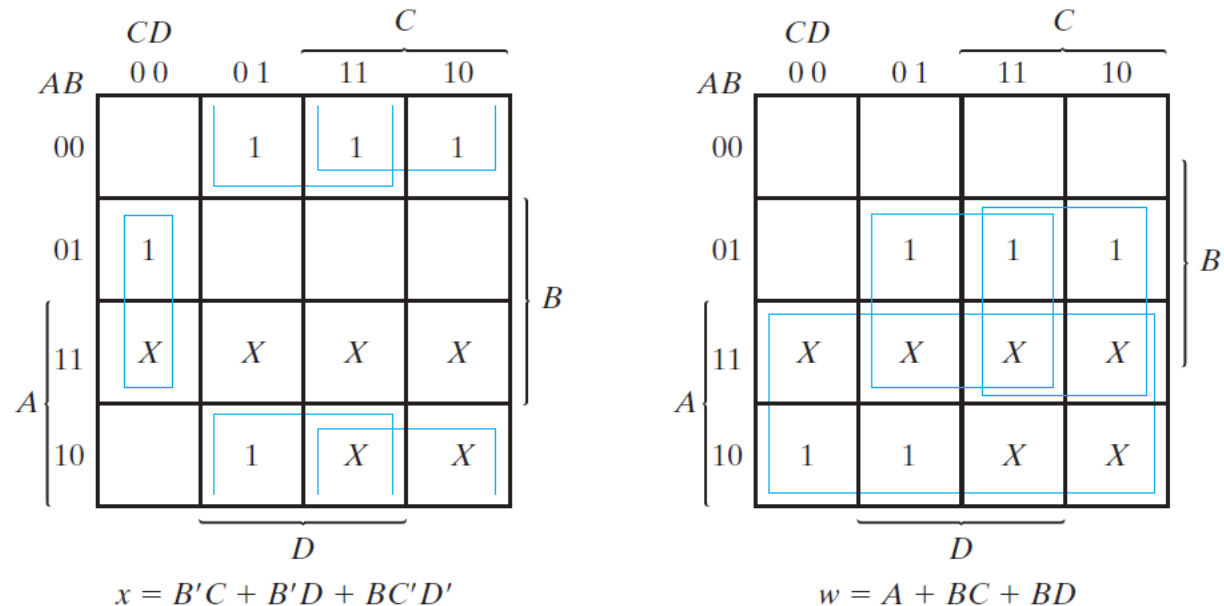
$z = D'$

		$CD$		$C$	
		00	01	11	10
$AB$	00	1		1	
	01	1		1	
	11	$X$	$X$	$X$	$X$
	10	1		$X$	$X$
		$D$			

$y = CD + C'D'$

# Circuitos Combinacionales

## ► Ejemplo de Diseño

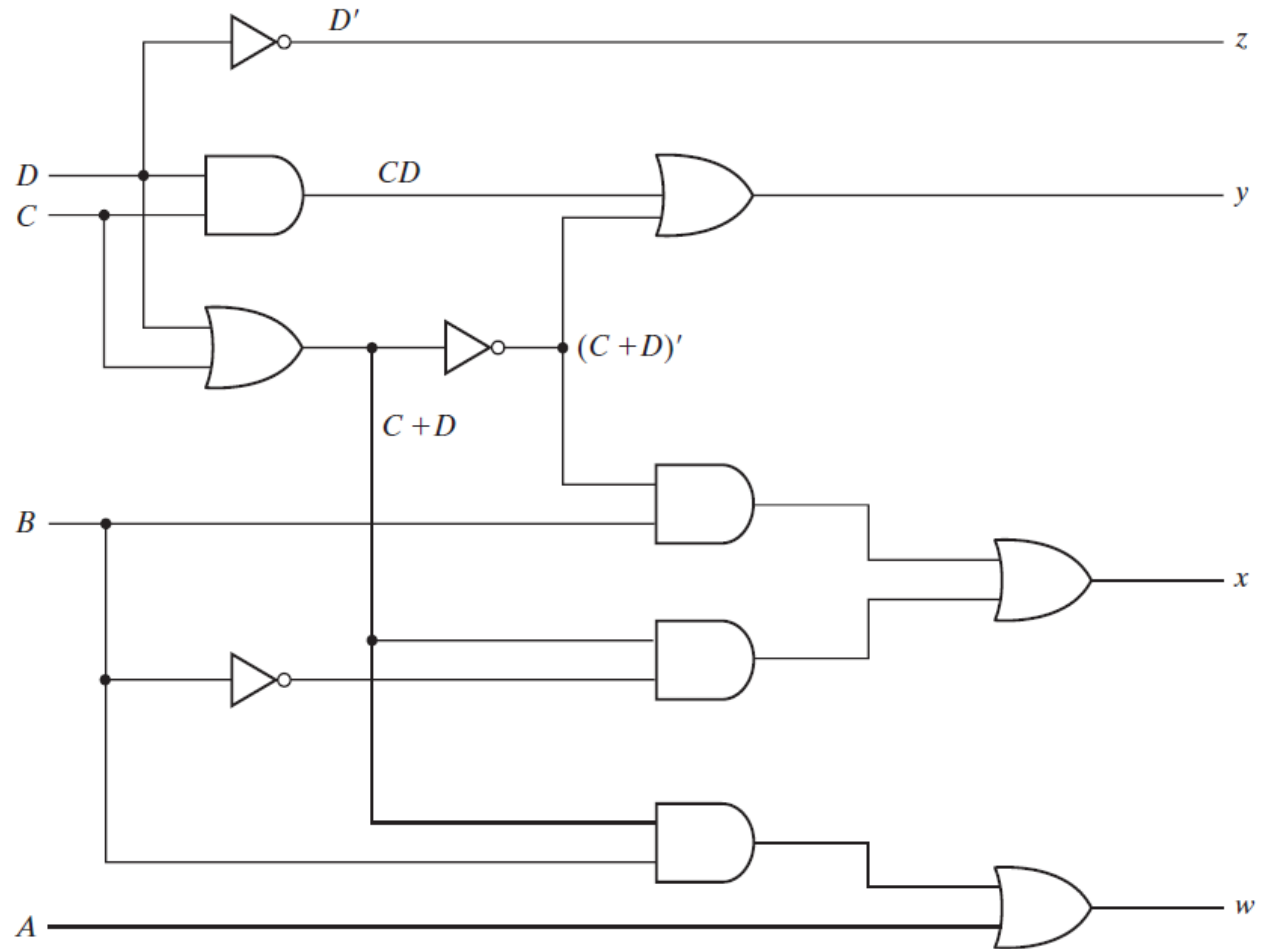


**FIGURA 4-3**

Mapas para el convertidor de código BCD a exceso-3

# Circuitos Combinacionales

► Ej:



**FIGURA 4-4**

Diagrama lógico para el convertidor de código BCD a exceso-3

# Ejemplo de un Sumador Restador

- ▶ Semisumador

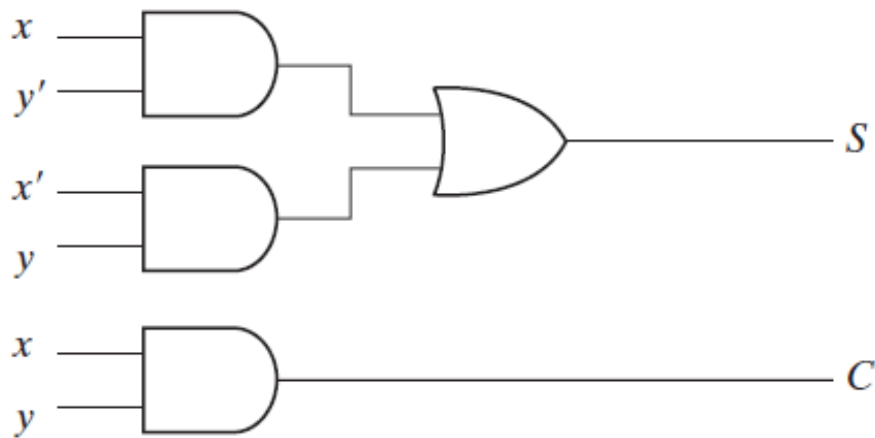
**Tabla 4-3**  
*Semisumador*

$x$	$y$	$C$	$S$
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

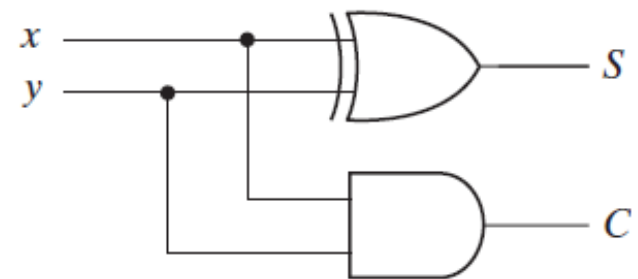
# Ejemplo de un Sumador Restador

## ► Semisumador

### Capítulo 4 Lógica combinacional



$$\begin{aligned} \text{a) } S &= xy' + x'y \\ C &= xy \end{aligned}$$



$$\begin{aligned} \text{b) } S &= x \oplus y \\ C &= xy \end{aligned}$$

**FIGURA 4-5**

Implementación de semisumador

# Ejemplo de un Sumador Restador

- ▶ Sumador Completo

**Tabla 4-4**  
*Sumador completo*

$x$	$y$	$z$	$C$	$S$
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

# Ejemplo de un Sumador Restador

## ► Sumador Completo

Sección 4-4 Sumador-restador binario

121

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

$$S = x'y'z + x'yz' + xy'z' + xyz$$

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

$$\begin{aligned} C &= xy + xz + yz \\ &= xy + xy'z + x'yz \end{aligned}$$

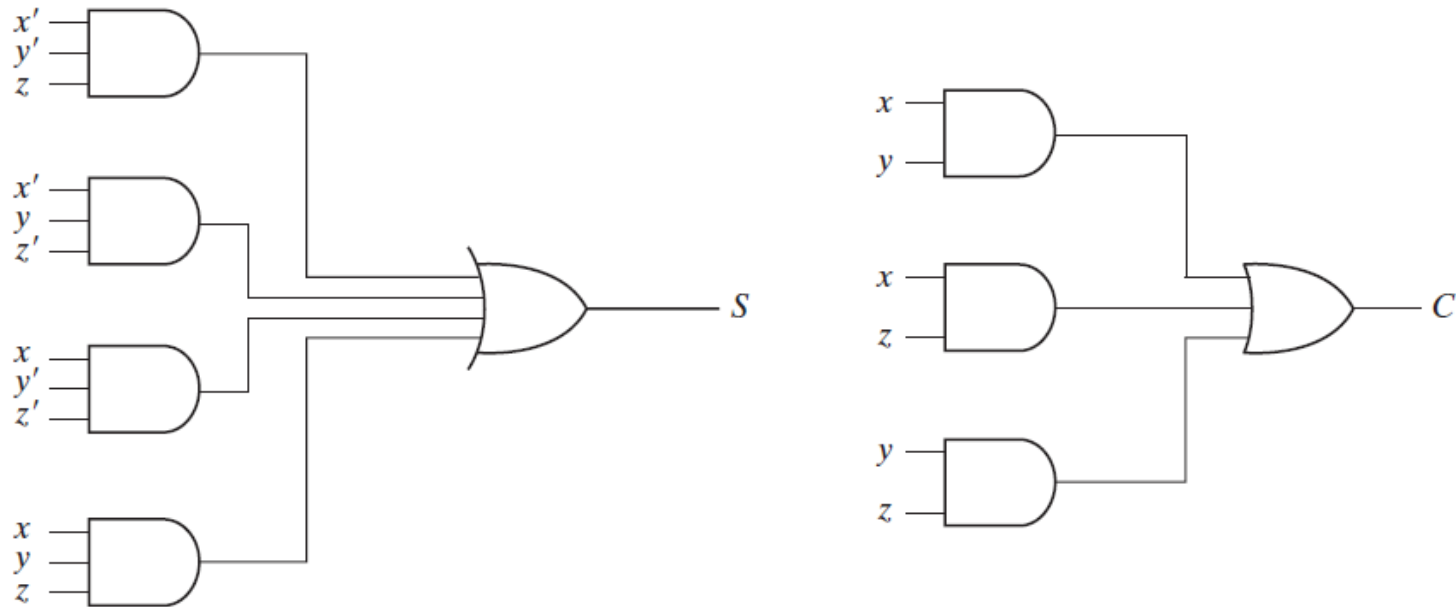
**FIGURA 4-6**

Mapas para el sumador completo



# Ejemplo de un Sumador Restador

## ► Sumador Completo



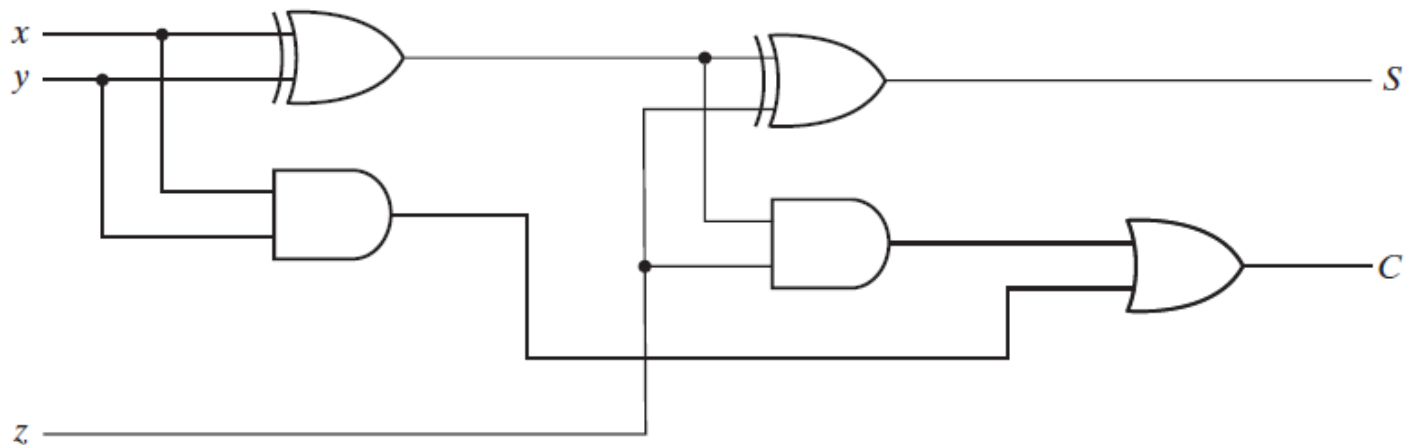
**FIGURA 4-7**

Implementación de un sumador completo como suma de productos

# Ejemplo de un Sumador Restador

## ► Sumador Completo

122 Capítulo 4 Lógica combinacional

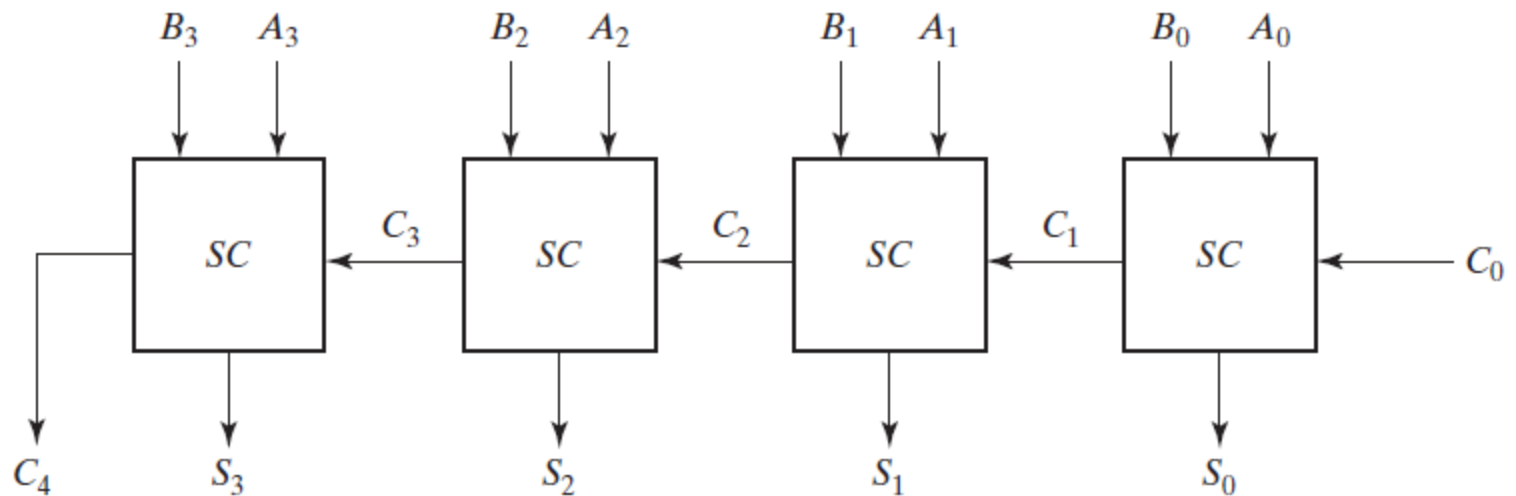


**FIGURA 4-8**

Implementación de un sumador completo con dos semisumadores y una compuerta OR

# Ejemplo de Sumador Restador

- Sumador Completo

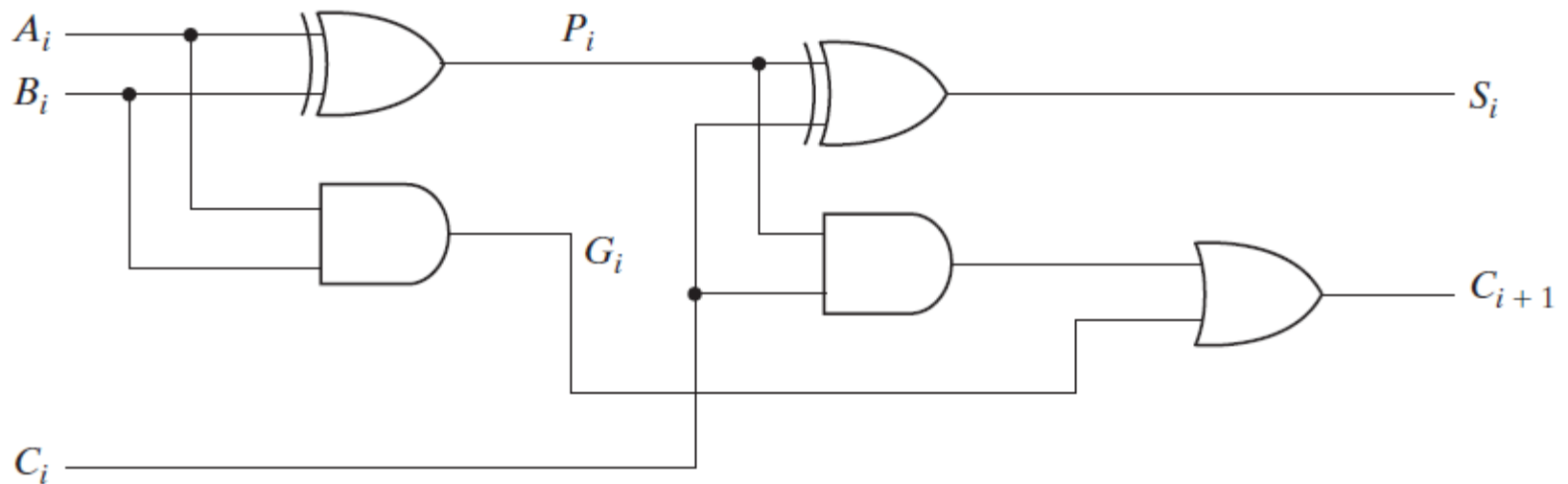


**FIGURA 4-9**

Sumador de cuatro bits

# Ejemplo de Sumador Restador

## ► Sumador Completo



**FIGURA 4-10**

Sumador completo en el que se indican P y G

# Ejemplo de Sumador Restador

## ► Acarreo Anticipado

Considere el circuito del sumador completo que se aprecia en la figura 4-10. Si definimos dos nuevas variables binarias

$$P_i = A_i \oplus B_i$$

$$G_i = A_i B_i$$

la suma y el acarreo se expresarán así:

$$S_i = P_i \oplus C_i$$

$$C_{i+1} = G_i + P_i C_i$$

# Ejemplo de Sumador Restador

## ► Acarreo Anticipado

$C_0$  = acarreo de entrada

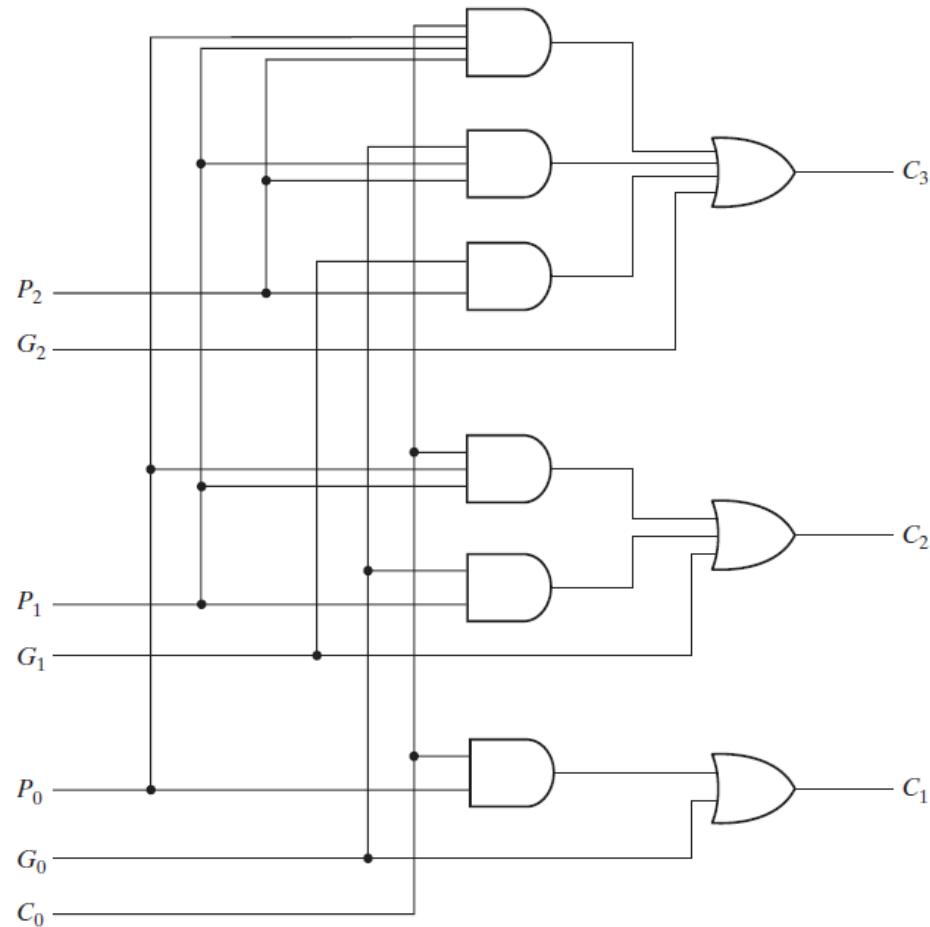
$$C_1 = G_0 + P_0C_0$$

$$C_2 = G_1 + P_1C_1 = G_1 + P_1(G_0 + P_0C_0) = G_1 + P_1G_0 + P_1P_0C_0$$

$$C_3 = G_2 + P_2C_2 = G_2 + P_2G_1 + P_2P_1G_0 + P_2P_1P_0C_0$$

# Ejemplo de Sumador Restador

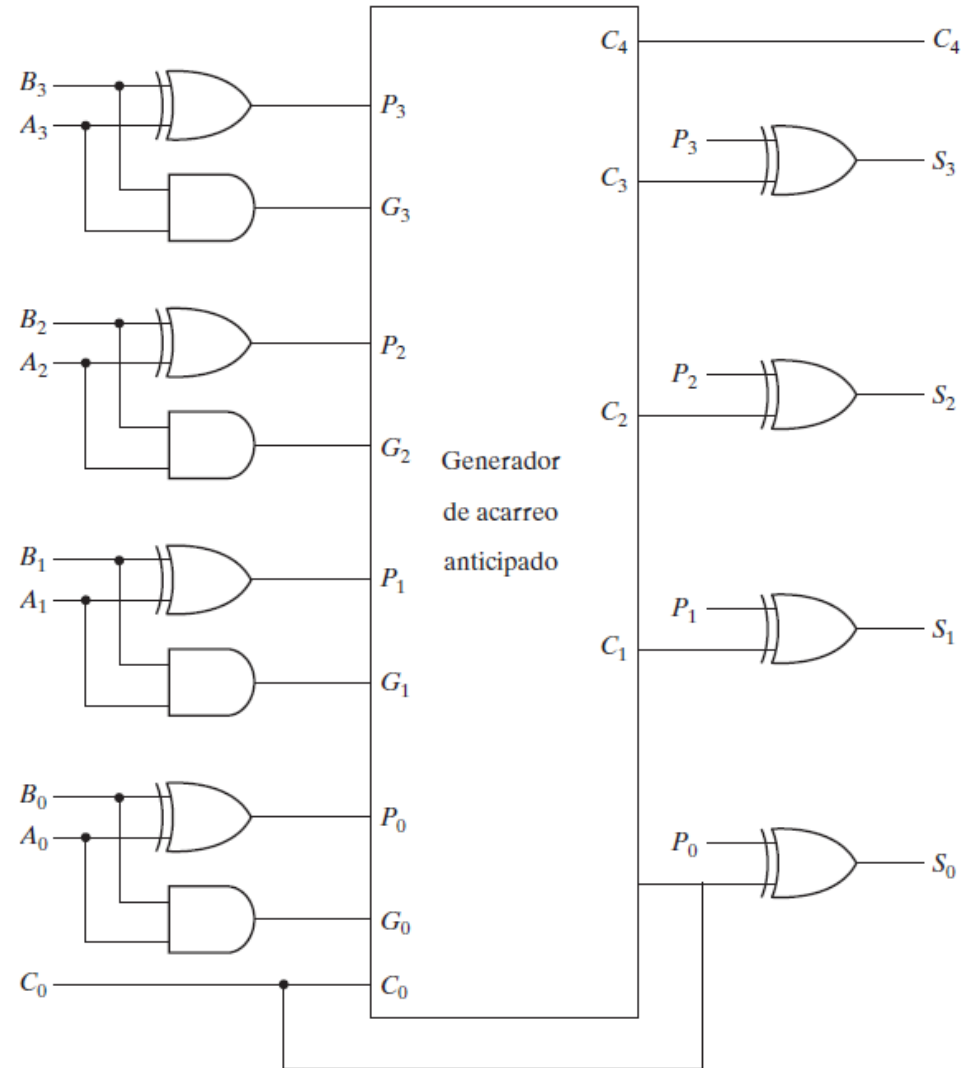
Sección 4-4 Sumador-restador binario 125



**FIGURA 4-11**

Diagrama lógico del generador de acarreo anticipado

# Ejemplo de Sumador Restador

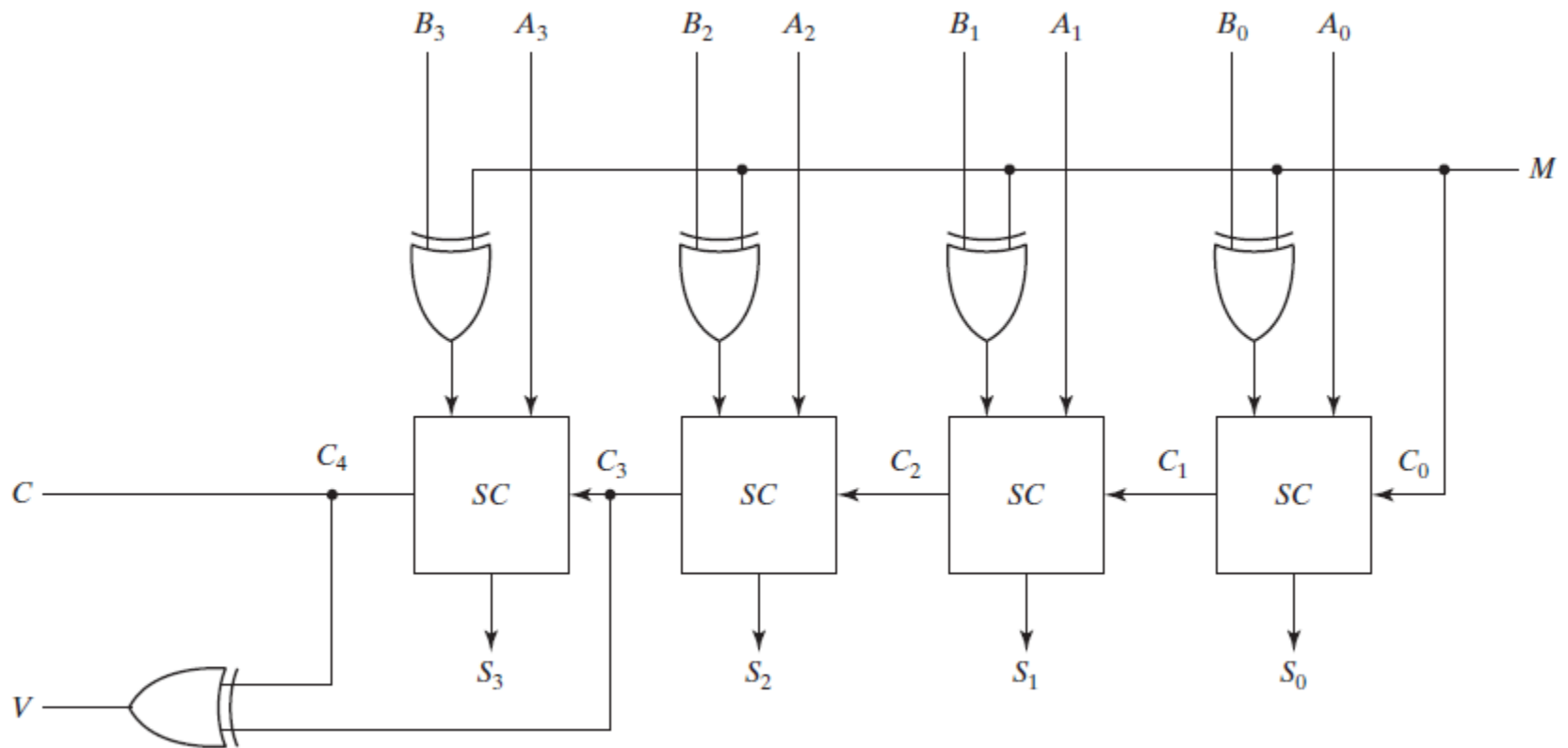


**FIGURA 4-12**

Sumador de cuatro bits con acarreo anticipado



# Ejemplo de Sumador Restador

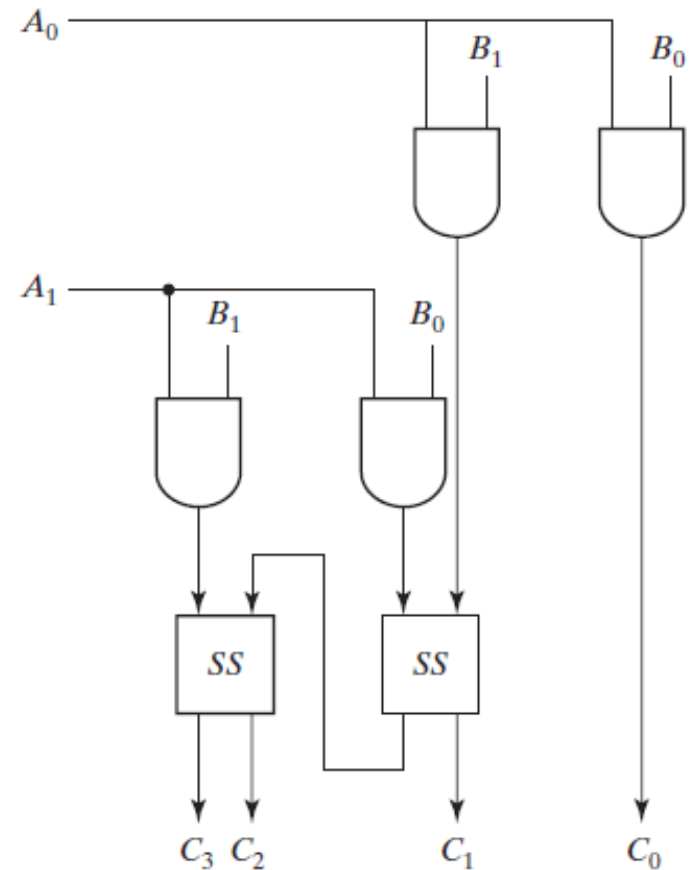


**FIGURA 4-13**

Sumador-restador de cuatro bits

# Ejemplo de Multiplicador Binario

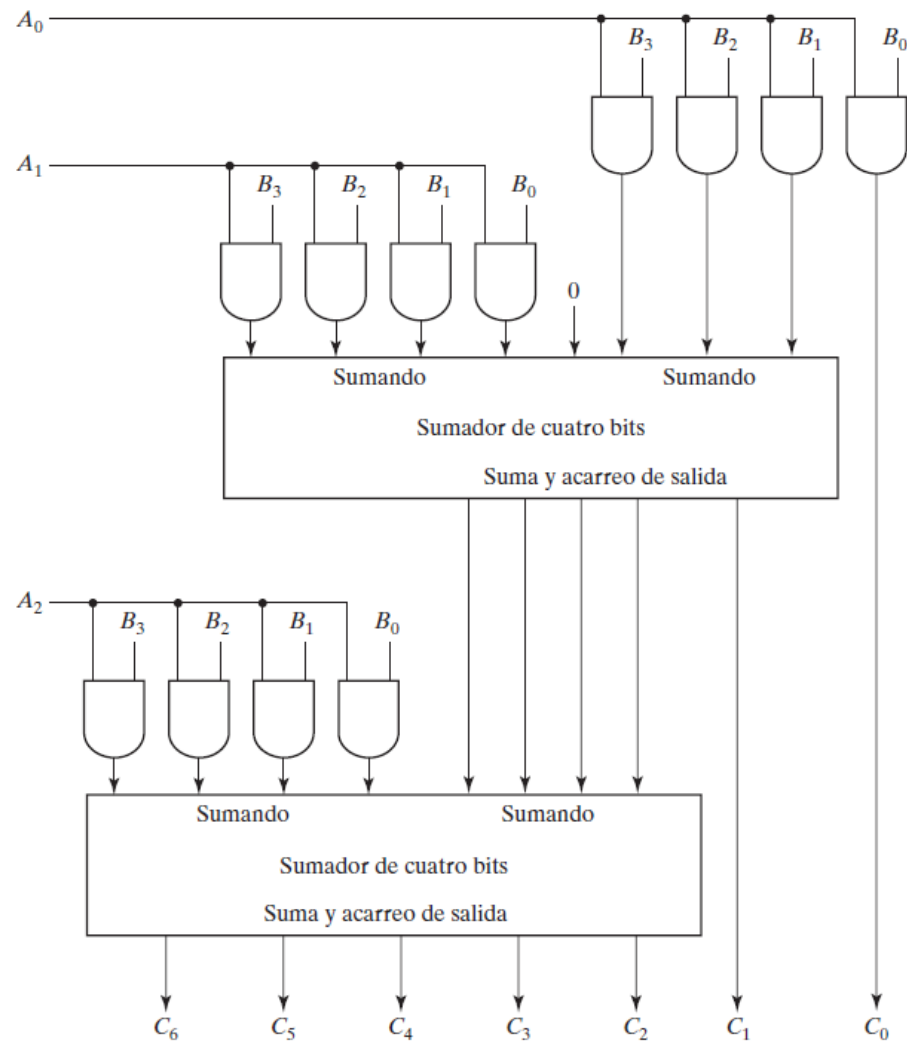
		$B_1$	$B_0$
	$A_1$	$A_0$	
	<hr/>		
	$A_0B_1$	$A_0B_0$	
$A_1B_1$	$A_1B_0$		
<hr/>			
$C_3$	$C_2$	$C_1$	$C_0$



**FIGURA 4-15**

Multiplicador binario de dos bits por dos bits

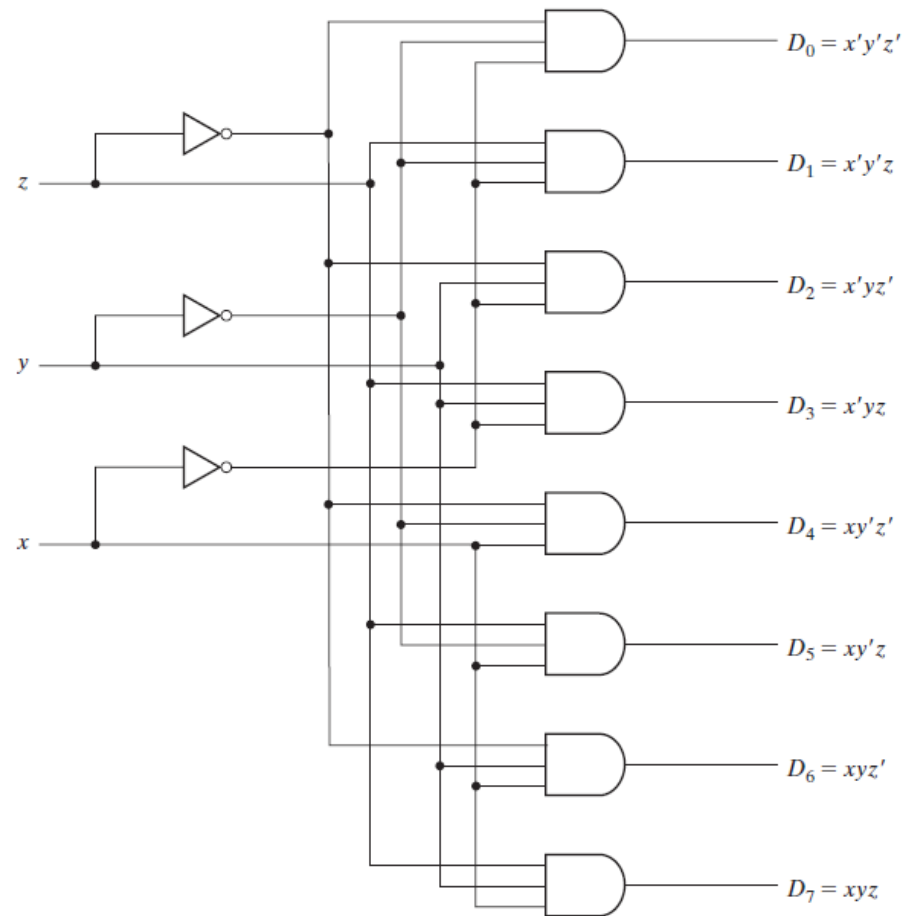
# Ejemplo de Multiplicador Binario



**FIGURA 4-16**  
Multiplicador binario de 4 bits por 3 bits

# DECODIFICADORES

## Sección 4-8 Decodificadores



**FIGURA 4-18**  
Decodificador de 3 a 8 líneas

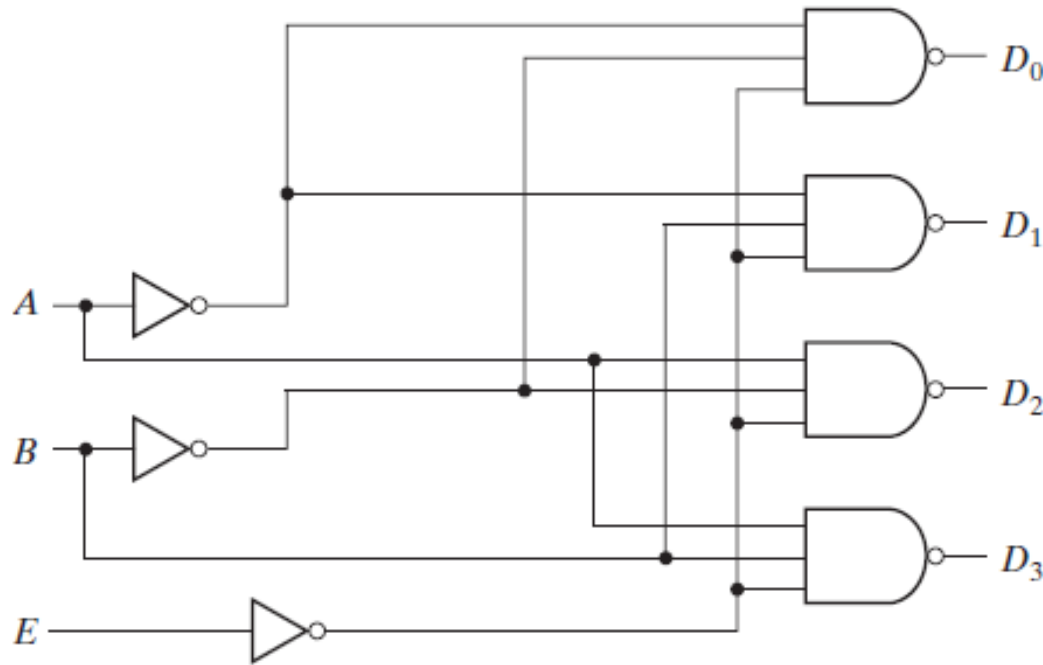
# DECODIFICADORES

**Tabla 4-6**

*Tabla de verdad de un decodificador de 3 a 8 líneas*

Entradas			Salidas							
<i>x</i>	<i>y</i>	<i>z</i>	<i>D</i> <sub>0</sub>	<i>D</i> <sub>1</sub>	<i>D</i> <sub>2</sub>	<i>D</i> <sub>3</sub>	<i>D</i> <sub>4</sub>	<i>D</i> <sub>5</sub>	<i>D</i> <sub>6</sub>	<i>D</i> <sub>7</sub>
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

# DECODIFICADORES



a) Diagrama lógico

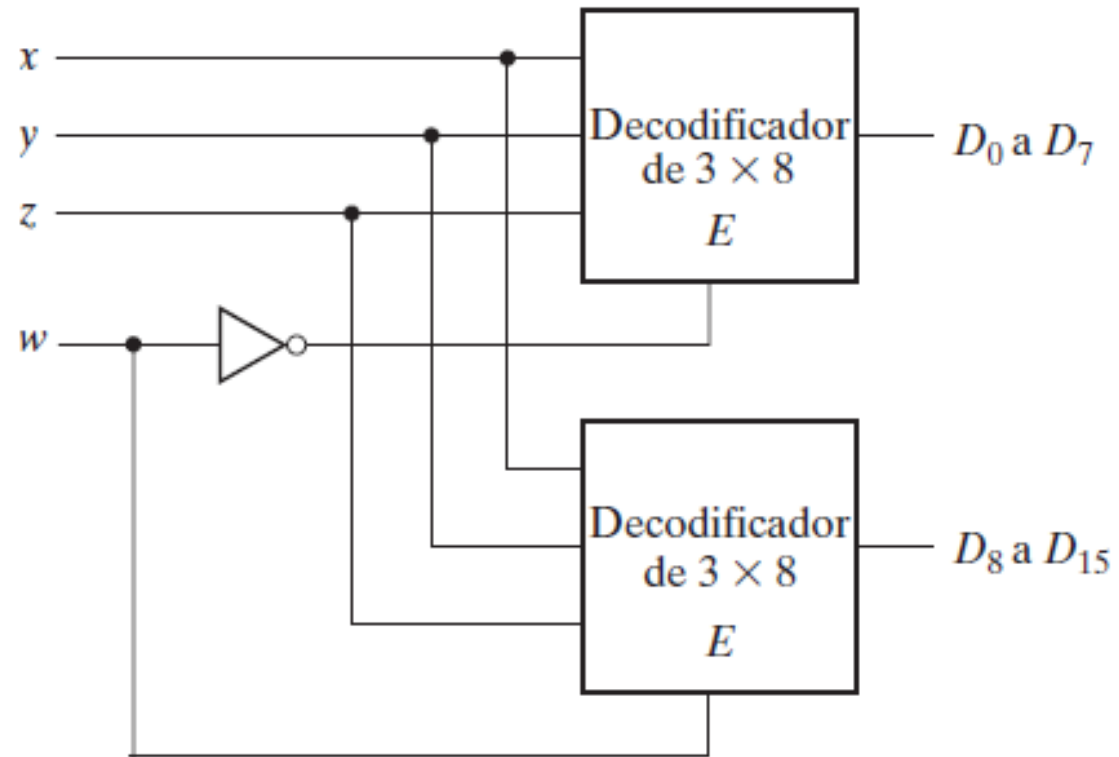
$E$	$A$	$B$	$D_0$	$D_1$	$D_2$	$D_3$
1	$X$	$X$	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

b) Tabla de verdad

**FIGURA 4-19**

Decodificador de 2 a 4 líneas con entrada habilitadora

# DECODIFICADORES

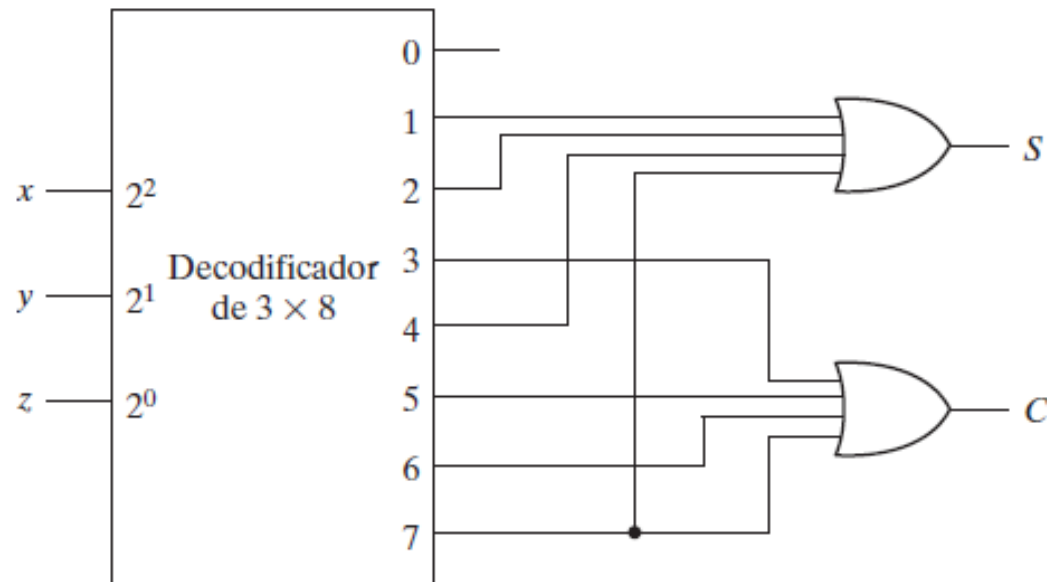


**FIGURA 4-20**

Decodificador  $4 \times 16$  construido con dos decodificadores  $3 \times 8$

# Implementación de lógica con decodificadores

## ► Suma de Productos



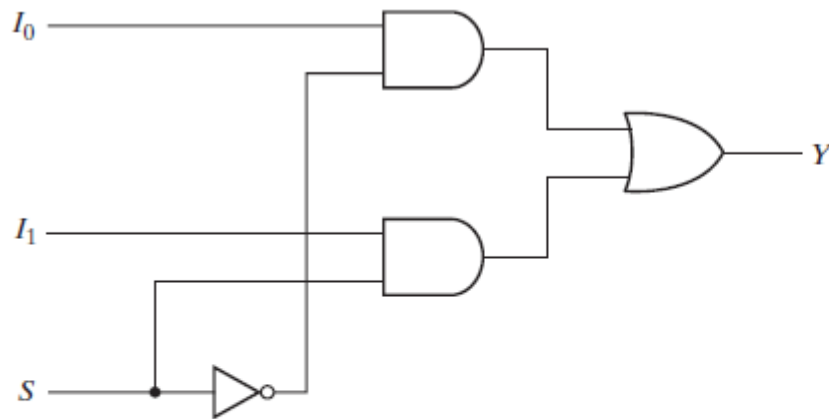
**FIGURA 4-21**

Implementación de un sumador completo con un decodificador

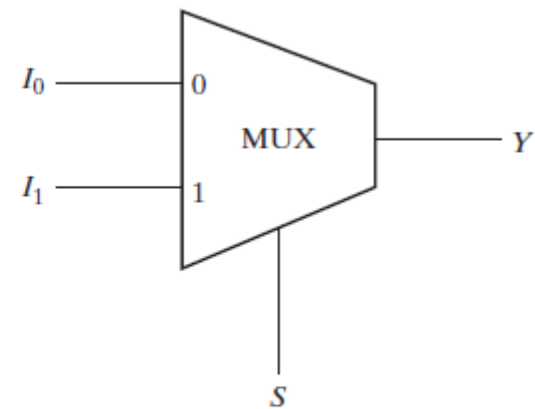


# MULTIPLEXORES

## ► De 2 a 1



a) Diagrama lógico

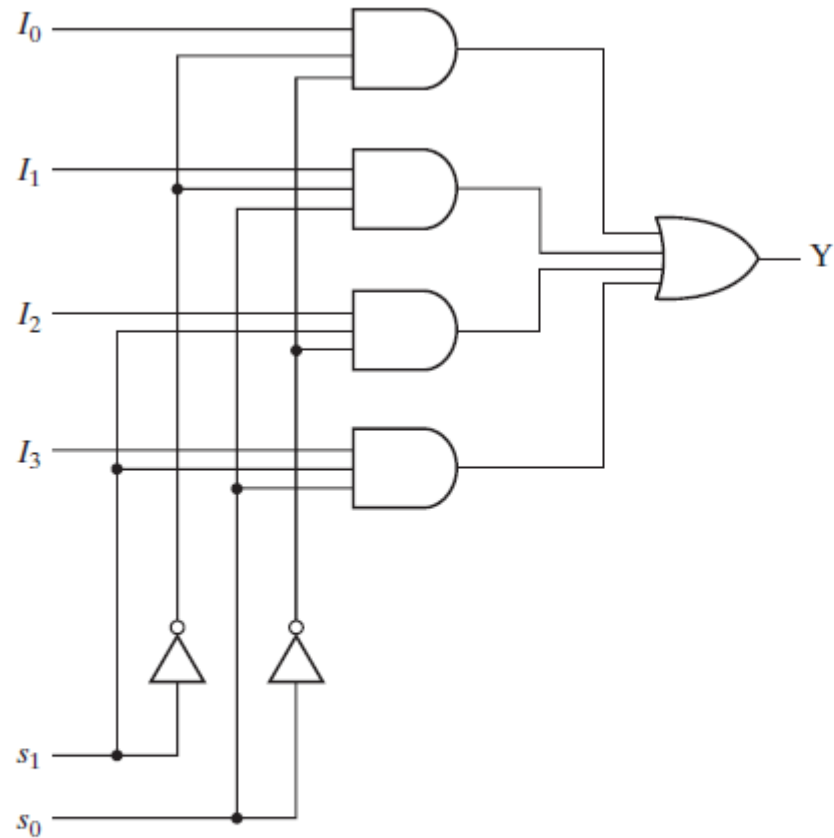


b) Diagrama de bloque

**FIGURA 4-24**  
Multiplexor de 2 líneas a 1

# MULTIPLEXORES

## ► De 4 a 1



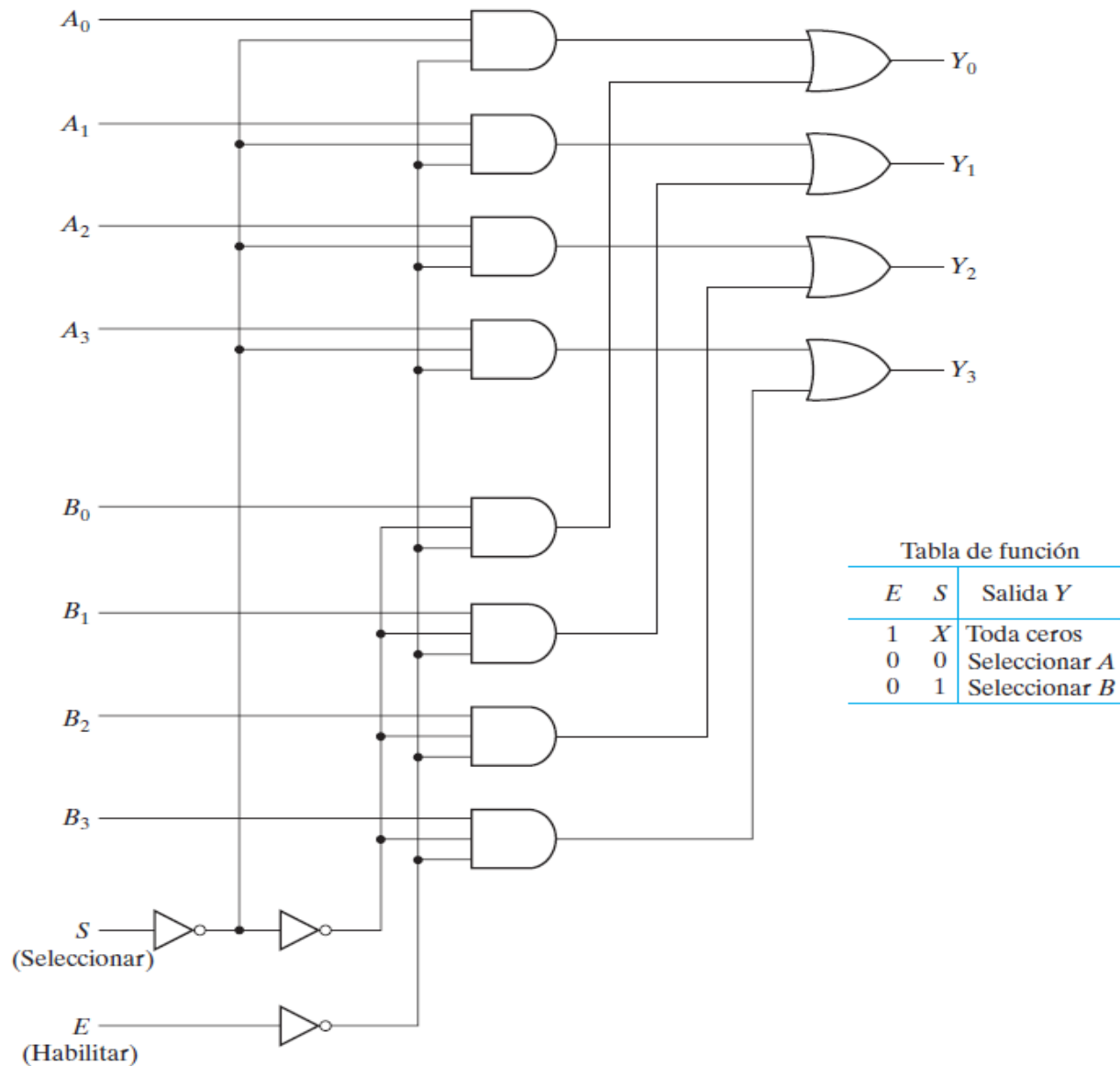
a) Diagrama lógico

$s_1$	$s_0$	$Y$
0	0	$I_0$
0	1	$I_1$
1	0	$I_2$
1	1	$I_3$

b) Tabla de función

**FIGURA 4-25**  
Multiplexor de 4 líneas a 1

# MULTIPLEXORES

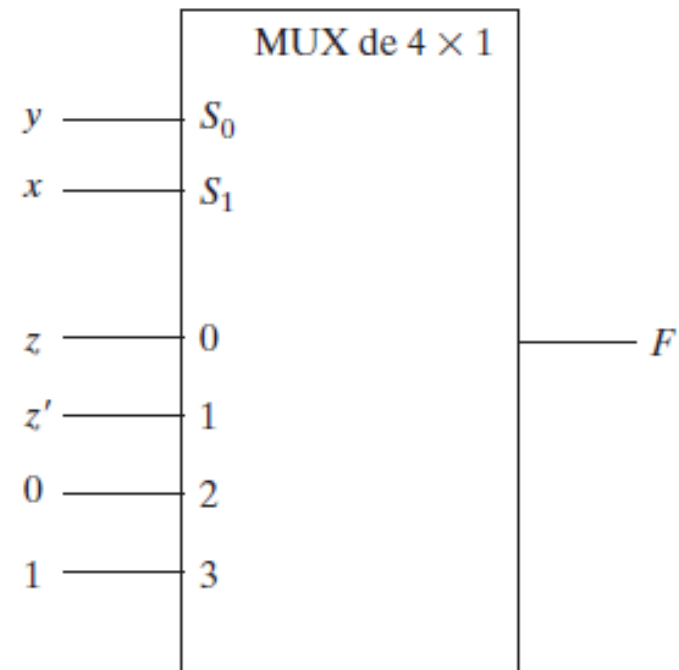


**FIGURA 4-26**  
Multiplexor cuádruple de 2 líneas a 1

# IMPLEMENTACIÓN DE LÓGICA CON MULTIPLEXORES

$x$	$y$	$z$	$F$	
0	0	0	0	$F = z$
0	0	1	1	
0	1	0	1	$F = z'$
0	1	1	0	
1	0	0	0	$F = 0$
1	0	1	0	
1	1	0	1	$F = 1$
1	1	1	1	

a) Tabla de verdad



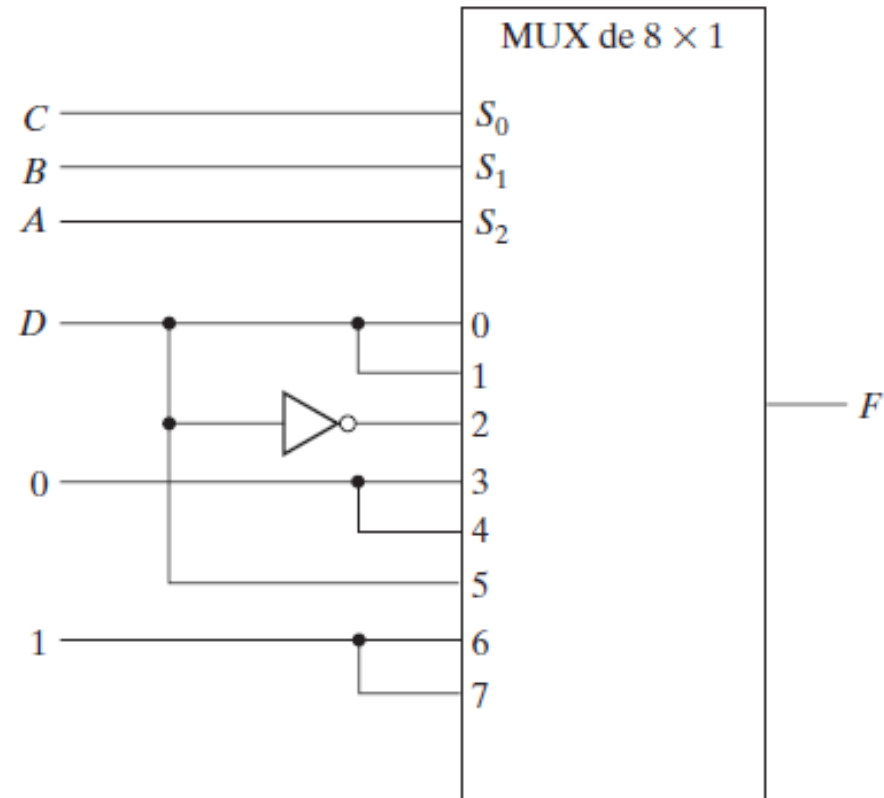
b) Implementación con multiplexor

**FIGURA 4-27**

Implementación de una función booleana con un multiplexor

# IMPLEMENTACIÓN DE LÓGICA CON MULTIPLEXORES

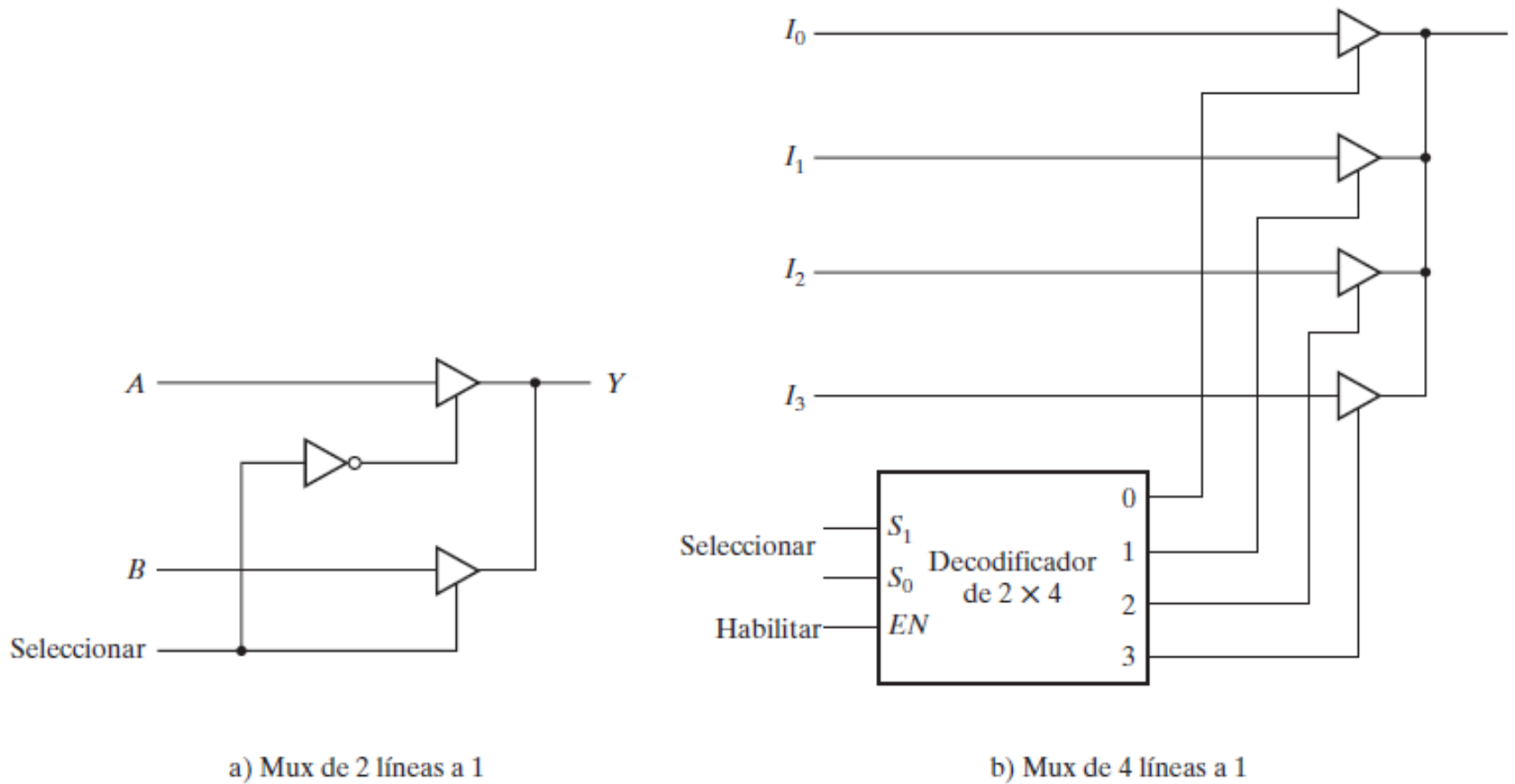
<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>	<i>F</i>	
0	0	0	0	0	$F = D$
0	0	0	1	1	
0	0	1	0	0	$F = D$
0	0	1	1	1	
0	1	0	0	1	$F = D'$
0	1	0	1	0	
0	1	1	0	0	$F = 0$
0	1	1	1	0	
1	0	0	0	0	$F = 0$
1	0	0	1	0	
1	0	1	0	0	$F = D$
1	0	1	1	1	
1	1	0	0	1	$F = 1$
1	1	0	1	1	
1	1	1	0	1	$F = 1$
1	1	1	1	1	



**FIGURA 4-28**

Implementación de una función de cuatro entradas con un multiplexor

# IMPLEMENTACIÓN DE LÓGICA CON BUFFERS DE ALTA CON ALTA IMPEDANCIA

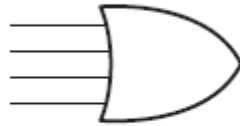


**FIGURA 4-30**

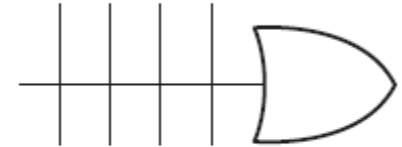
Multiplexores con compuertas de tres estados

# Memoria y Lógicas Programables

## 256 Capítulo 7 Memoria y lógica programable



a) Símbolo convencional

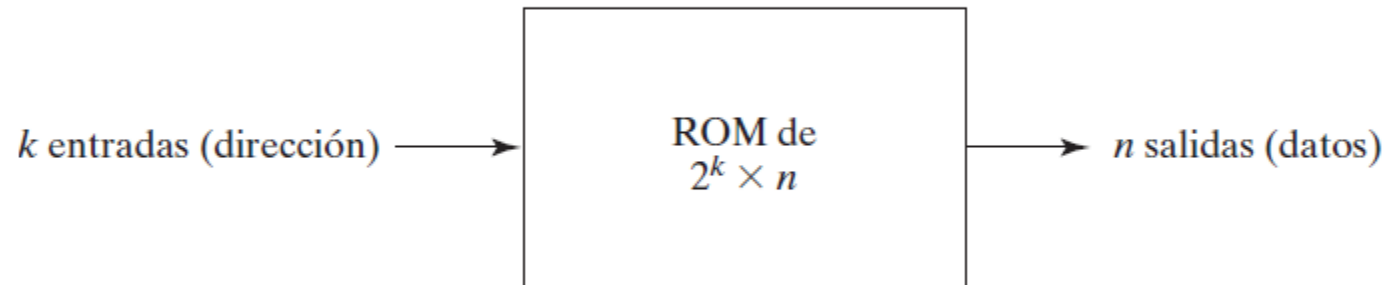


b) Símbolo de arreglo lógico

**FIGURA 7-1**

Diagramas convencional y de arreglo lógico para la compuerta OR

# Read Only Memories ROM

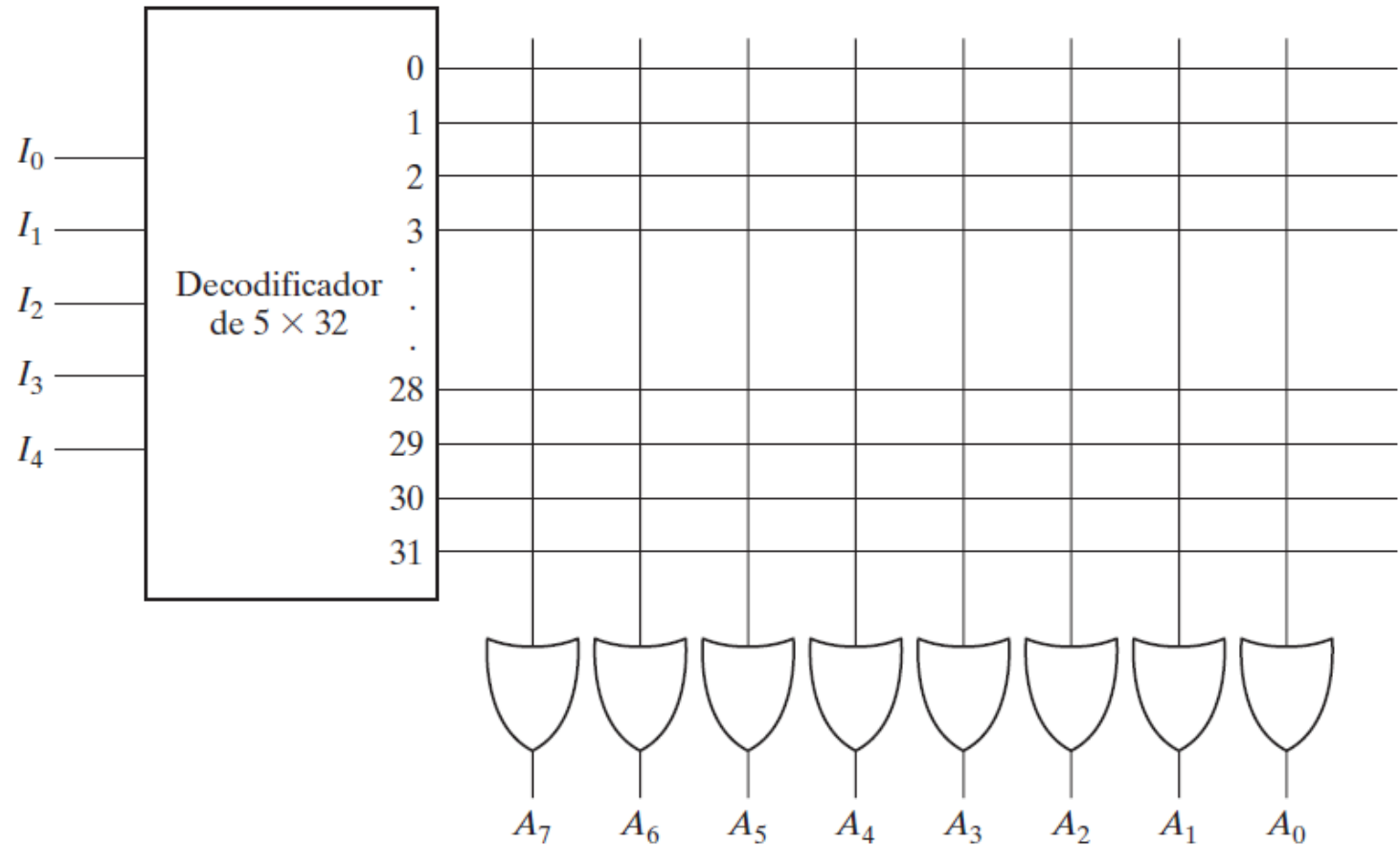


**FIGURA 7-9**

Diagrama de bloques de ROM



# Read Only Memories ROM



**FIGURA 7-10**

Lógica interna de una ROM de  $32 \times 8$

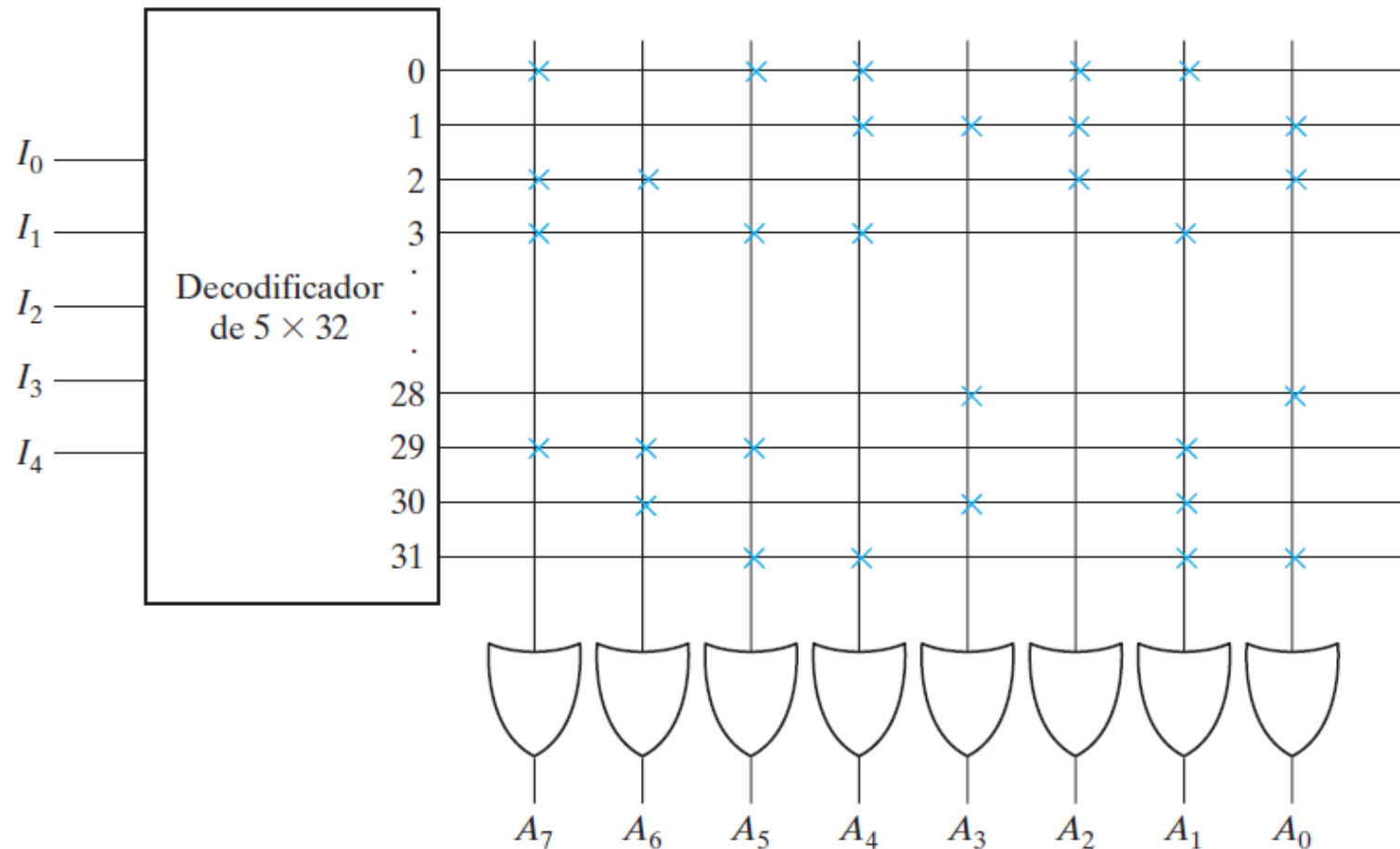
# Implementando Combinacionales con ROM

## 272 Capítulo 7 Memoria y lógica programable

**Tabla 7-3**  
*Tabla de verdad de ROM (parcial)*

Entradas					Salidas							
I4	I3	I2	I1	I0	A7	A6	A5	A4	A3	A2	A1	A0
0	0	0	0	0	1	0	1	1	0	1	1	0
0	0	0	0	1	0	0	0	1	1	1	0	1
0	0	0	1	0	1	1	0	0	0	1	0	1
0	0	0	1	1	1	0	1	1	0	0	1	0
		⋮					⋮					
1	1	1	0	0	0	0	0	0	1	0	0	1
1	1	1	0	1	1	1	1	0	0	0	1	0
1	1	1	1	0	0	1	0	0	1	0	1	0
1	1	1	1	1	0	0	1	1	0	0	1	1

# Implementando Combinacionales con ROM

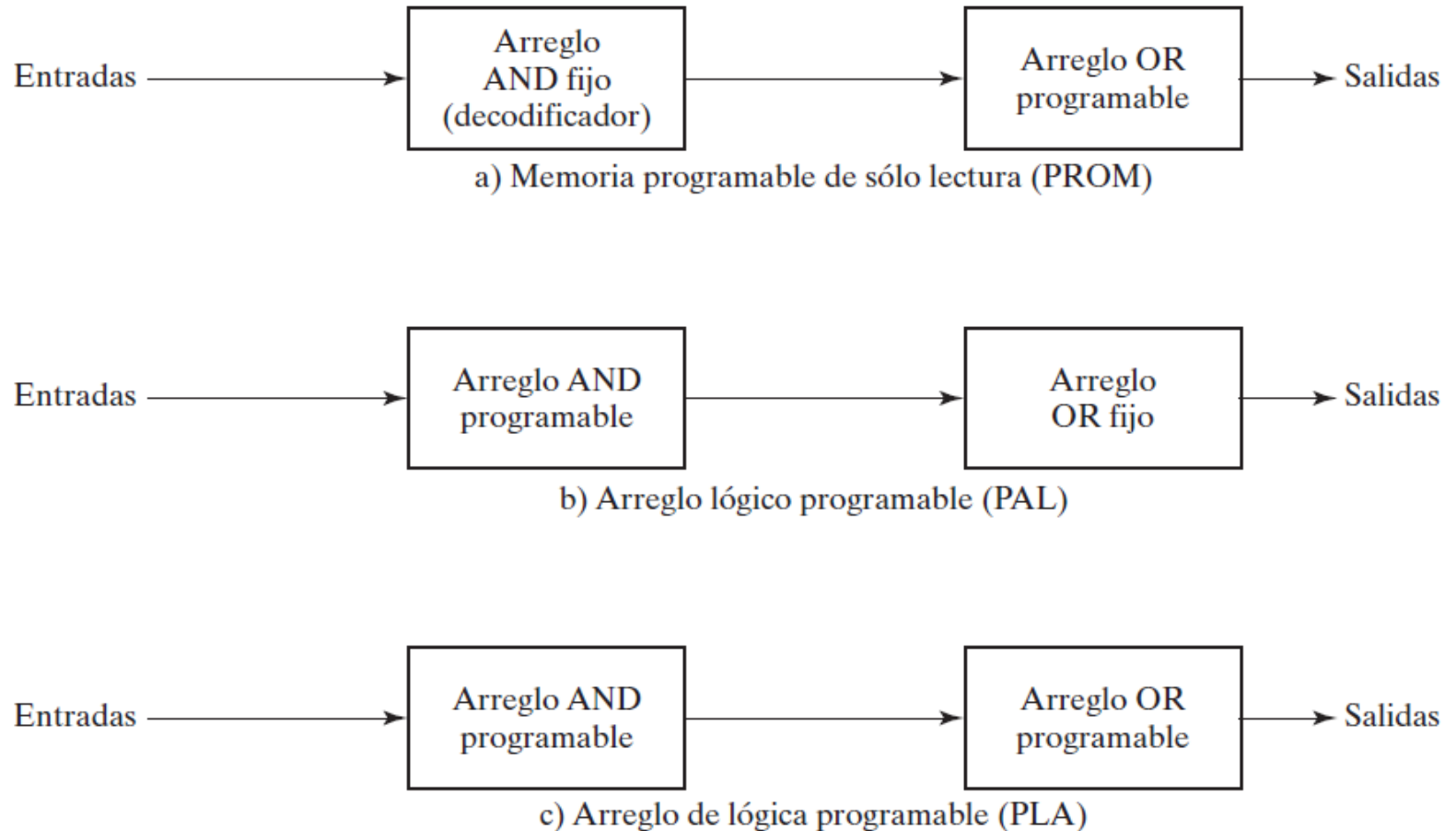


**FIGURA 7-11**

Programación de la ROM según la tabla 7-3

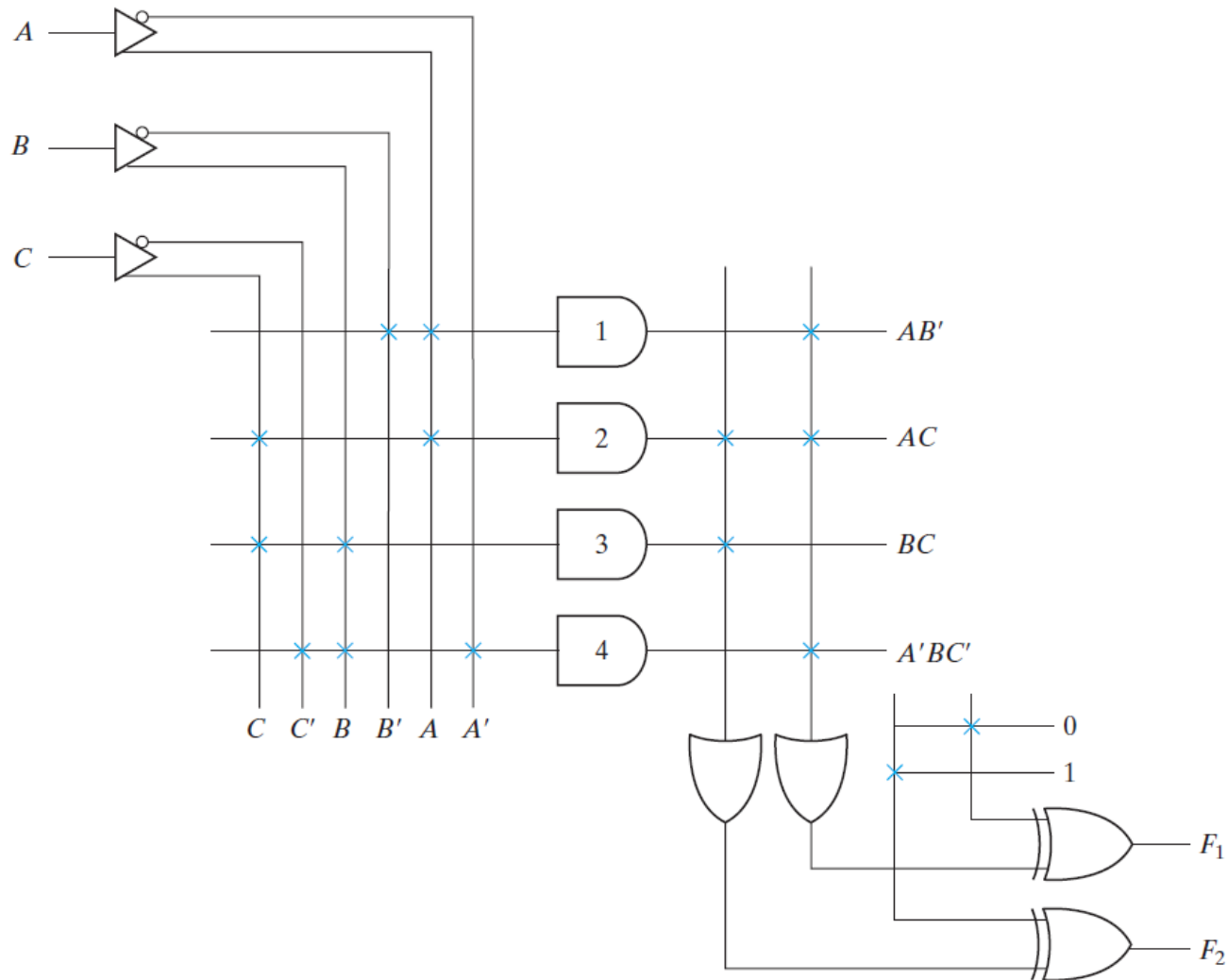
# Programmable Logic Devices (PLDs)

## 276 Capítulo 7 Memoria y lógica programable



**FIGURA 7-13**  
Configuración básica de tres PLD

# Programmable Logic Devices (PLDs)



**FIGURA 7-14**

PLA con tres entradas, cuatro términos producto y dos salidas

# Programmable Logic Devices (PLDs)

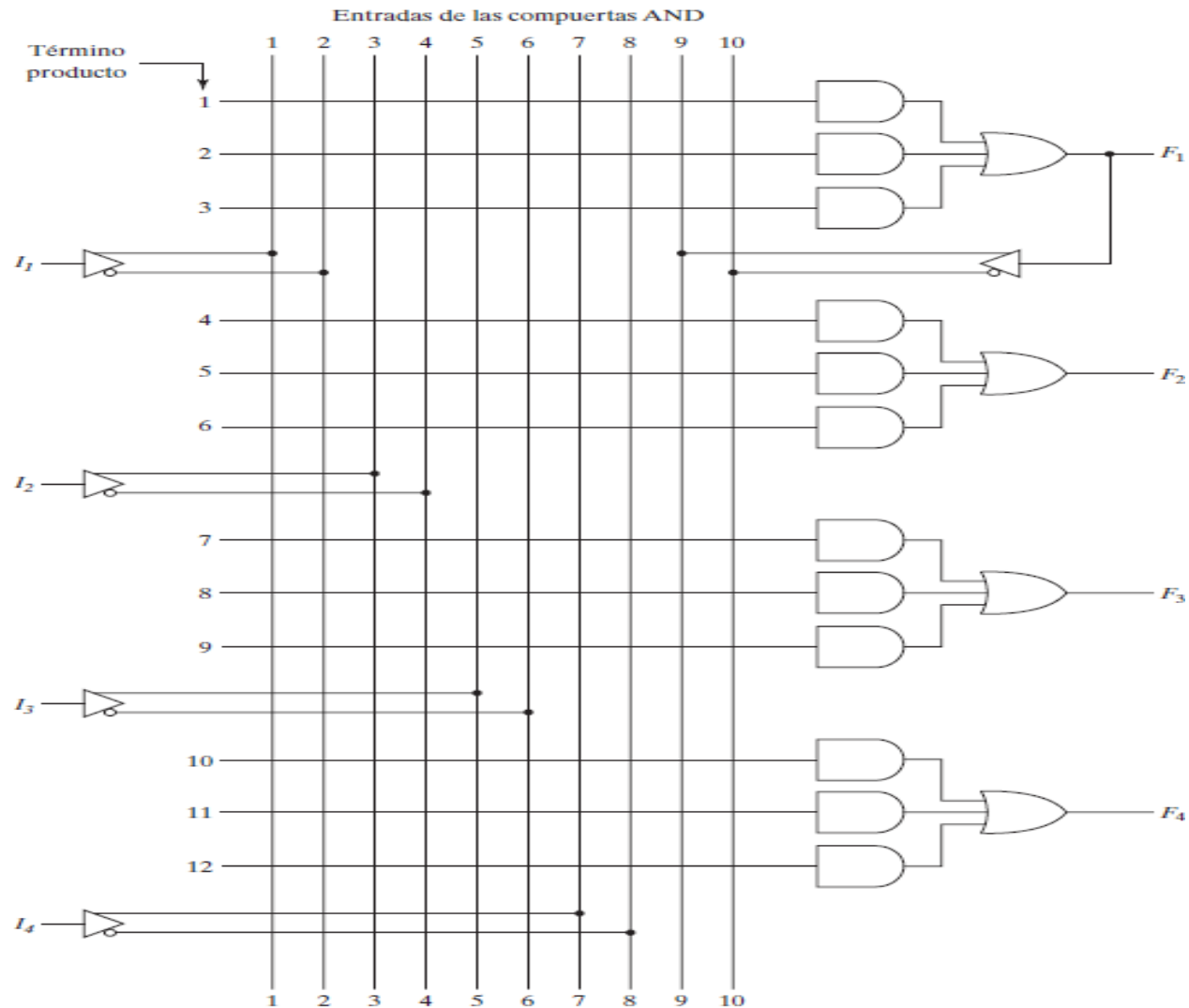


FIGURA 7-16

PAL con cuatro entradas, cuatro salidas y una estructura AND-OR de anchura tres