

UNIVERSIDAD NACIONAL DE CÓRDOBA

FACULTAD DE MATEMÁTICA ASTRONOMÍA, FÍSICA Y  
COMPUTACIÓN.

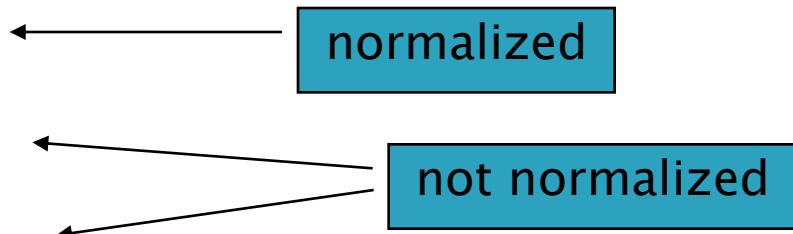
ORGANIZACIÓN DEL COMPUTADOR

TEÓRICOS:  
PABLO A. FERREYRA - NICOLÁS WOLOVIC

PRÁCTICOS:  
DELFINA VELEZ  
AGUSTÍN LAPROVITA  
GONZALO VODANOVICK

# Floating Point

- ▶ Representation for non-integral numbers
  - Including very small and very large numbers
- ▶ Like scientific notation
  - $-2.34 \times 10^{56}$
  - $+0.002 \times 10^{-4}$
  - $+987.02 \times 10^9$
- ▶ In binary
  - $\pm 1.xxxxxxx_2 \times 2^{yyyy}$
- ▶ Types `float` and `double` in C



# Floating Point Standard

- ▶ Defined by IEEE Std 754–1985
- ▶ Developed in response to divergence of representations
  - Portability issues for scientific code
- ▶ Now almost universally adopted
- ▶ Two representations
  - Single precision (32-bit)
  - Double precision (64-bit)

# IEEE Floating-Point Format

single: 8 bits

double: 11 bits

single: 23 bits

double: 52 bits

S	Exponent	Fraction
---	----------	----------

$$x = (-1)^S \times (1 + \text{Fraction}) \times 2^{(\text{Exponent} - \text{Bias})}$$

- ▶ S: sign bit ( $0 \Rightarrow$  non-negative,  $1 \Rightarrow$  negative)
- ▶ Normalize significand:  $1.0 \leq |\text{significand}| < 2.0$ 
  - Always has a leading pre-binary-point 1 bit, so no need to represent it explicitly (hidden bit)
  - Significand is Fraction with the “1.” restored
- ▶ Exponent: excess representation: actual exponent + Bias
  - Ensures exponent is unsigned
  - Single: Bias = 127; Double: Bias = 1023

# Single-Precision Range

- ▶ Exponents 00000000 and 11111111 reserved
- ▶ Smallest value
  - Exponent: 00000001  
⇒ actual exponent =  $1 - 127 = -126$
  - Fraction: 000...00 ⇒ significand = 1.0
  - $\pm 1.0 \times 2^{-126} \approx \pm 1.2 \times 10^{-38}$
- ▶ Largest value
  - exponent: 11111110  
⇒ actual exponent =  $254 - 127 = +127$
  - Fraction: 111...11 ⇒ significand  $\approx 2.0$   
 $\pm 2.0 \times 2^{+127} \approx \pm 3.4 \times 10^{+38}$

# Double-Precision Range

- ▶ Exponents 0000...00 and 1111...11 reserved
- ▶ Smallest value
  - Exponent: 00000000001  
⇒ actual exponent =  $1 - 1023 = -1022$
  - Fraction: 000...00 ⇒ significand = 1.0
  - $\pm 1.0 \times 2^{-1022} \approx \pm 2.2 \times 10^{-308}$
- ▶ Largest value
  - Exponent: 11111111110  
⇒ actual exponent =  $2046 - 1023 = +1023$
  - Fraction: 111...11 ⇒ significand  $\approx 2.0$   
 $\pm 2.0 \times 2^{+1023} \approx \pm 1.8 \times 10^{+308}$

# Floating–Point Precision

## ▶ Relative precision

- all fraction bits are significant
- Single: approx  $2^{-23}$ 
  - Equivalent to  $23 \times \log_{10}2 \approx 23 \times 0.3 \approx 6$  decimal digits of precision
- Double: approx  $2^{-52}$ 
  - Equivalent to  $52 \times \log_{10}2 \approx 52 \times 0.3 \approx 16$  decimal digits of precision

# Floating-Point Example

- ▶ Represent -0.75
  - $-0.75 = (-1)^1 \times 1.1_2 \times 2^{-1}$
  - S = 1
  - Fraction = 1000...00<sub>2</sub>
  - Exponent = -1 + Bias
    - Single: -1 + 127 = 126 = 01111110<sub>2</sub>
    - Double: -1 + 1023 = 1022 = 01111111110<sub>2</sub>
- ▶ Single: 101111101000...00
- ▶ Double: 101111111101000...00

# Floating-Point Example

- ▶ What number is represented by the single-precision float

11000000101000...00

- S = 1
  - Fraction = 01000...00<sub>2</sub>
  - Exponent = 10000001<sub>2</sub> = 129
- ▶  $x = (-1)^1 \times (1 + 01_2) \times 2^{(129 - 127)}$
- $$\begin{aligned} &= (-1) \times 1.25 \times 2^2 \\ &= -5.0 \end{aligned}$$

# Infinities and NaNs

- ▶ Exponent = 111...1, Fraction = 000...0
  - ±Infinity
  - Can be used in subsequent calculations, avoiding need for overflow check
- ▶ Exponent = 111...1, Fraction ≠ 000...0
  - Not-a-Number (NaN)
  - Indicates illegal or undefined result
    - e.g.,  $0.0 / 0.0$
  - Can be used in subsequent calculations

# Floating-Point Addition

- ▶ Consider a 4-digit decimal example
  - $9.999 \times 10^1 + 1.610 \times 10^{-1}$
- ▶ 1. Align decimal points
  - Shift number with smaller exponent
  - $9.999 \times 10^1 + 0.016 \times 10^1$
- ▶ 2. Add significands
  - $9.999 \times 10^1 + 0.016 \times 10^1 = 10.015 \times 10^1$
- ▶ 3. Normalize result & check for over/underflow
  - $1.0015 \times 10^2$
- ▶ 4. Round and renormalize if necessary
  - $1.002 \times 10^2$

# Floating-Point Addition

- ▶ Now consider a 4-digit binary example
  - $1.000_2 \times 2^{-1} + -1.110_2 \times 2^{-2}$  ( $0.5 + -0.4375$ )
- ▶ 1. Align binary points
  - Shift number with smaller exponent
  - $1.000_2 \times 2^{-1} + -0.111_2 \times 2^{-1}$
- ▶ 2. Add significands
  - $1.000_2 \times 2^{-1} + -0.111_2 \times 2^{-1} = 0.001_2 \times 2^{-1}$
- ▶ 3. Normalize result & check for over/underflow
  - $1.000_2 \times 2^{-4}$ , with no over/underflow
- ▶ 4. Round and renormalize if necessary
  - $1.000_2 \times 2^{-4}$  (no change) = 0.0625

# **ALGEBRA DE BOOLE**

## **AXIOMAS Y POSTULADOS BÁSICOS**

# Elementos Básicos

- ▶ Sea  $B$  un conjunto de elementos:
- ▶  $B=\{a, b, c, \dots\}$  y el conjunto  $\{0, 1\}$
- ▶ Sea  $\phi$  un conjunto de operaciones en  $B$  con resultados en  $\{0, 1\}$  :
- ▶  $\phi= \{+, ., '\}$
- ▶ Estos elementos conforman un álgebra de Boole si satisfacen los siguientes axiomas:

# AXIOMAS

► Clausura:

- Si  $a \in B$  y  $b \in B$ , entonces  $a+b \in B$ ,  $a.b \in B$

► Axioma del 0:

- $\exists 0 \in B$  tal que  $a + 0 = a$ , para todo  $a \in B$

► Axioma del 1:

- $\exists 1 \in B$  tal que  $a.1 = a$ , para todo  $a \in B$

► Commutatividad:

- $a+b = b+a$   $a.b = b.a$ , para todo  $a, b$ , en  $B$ .

# Axiomas (continuación)

- ▶ **Distributividad de “+” respecto a “.”**
  - $\forall a, b, c \in B, a + (b \cdot c) = (a + b) \cdot (a + c),$
  
- ▶ **Distributividad de “.” respecto a “+”**
  - $\forall a, b, c \in B, a \cdot (b + c) = (a \cdot b) + (a \cdot c),$

# Axiomas (continuación)

## ► Axioma Inverso:

- Sea  $a \in B$ , entonces existe  $a'$  tal que:  $a + a' = 1$   $a \cdot a' = 0$

## ► Cardinalidad:

- $|B| \geq 2$ ,

# Teoremas, Idempotencia, (Primera parte)

- ▶  $a \cdot a = a$  ?
- ▶  $a \cdot a = a \cdot a + 0$  , Axioma del 0
- ▶  $a \cdot a = a \cdot a + a \cdot a'$  , Axioma del Inverso
- ▶  $a \cdot a = a \cdot (a + a')$  , Distributividad
- ▶  $a \cdot a = a \cdot 1$  , Axioma del inverso
- ▶  $a \cdot a = a$  , Axioma del 1

# Teoremas, Idempotencia, (Segunda parte)

- ▶  $a+a = a$  ?
- ▶  $a + a = a + a \cdot 1$  , Axioma del 1
- ▶  $a+a = a + a \cdot a + a'$  , Axioma del Inverso
- ▶  $a+a = a + (a \cdot a')$  , Distributividad
- ▶  $a + a = a + 0$  , Axioma del inverso
- ▶  $a + a = a$  , Axioma del 0

# Teoremas, propiedad del 0

- ▶  $a \cdot 0 = 0$  ?
- ▶  $a \cdot 0 = a \cdot 0 + 0$  , Axioma del 0
- ▶  $= a \cdot 0 + a \cdot a'$  , Axioma del Inverso
- ▶  $= a \cdot (0 + a')$  , Distributividad
- ▶  $= a \cdot a'$  , Axioma del 0
- ▶  $= 0$  , Axioma del inverso

# Teoremas, propiedad del 1

- ▶  $a+1=1$  ¿?
- ▶  $a + 1 = a + 1 \cdot 1$  , Axioma del 1
- ▶  $= a + 1 \cdot a + a'$  , Axioma del Inverso
- ▶  $= a + (1 \cdot a')$  , Distributividad
- ▶  $= a + a'$  , Axioma del 1
- ▶  $= 1$ , Axioma del inverso

# Teoremas, Unicidad del Inverso

- ▶ Sean  $a$ ,  $a_1$  y  $a_2$  tales que:  $a+a_1 = 1$ ,  $a.a_1=0$ ,  $a+a_2=1$ ,  $a.a_2=0$ , entonces  $a_1=a_2$ . En efecto:

# Teoremas, Unicidad del Inverso

- ▶  $a2 = a2 \cdot 1,$  axioma del 1
- ▶  $a2 = a2 \cdot (a + a1),$  hipótesis
- ▶  $a2 = a2 \cdot a + a2 \cdot a1,$  Distr.
- ▶  $a2 = 0 + a2 \cdot a1,$  hipótesis
- ▶  $a2 = a \cdot a1 + a2 \cdot a1,$  hipótesis
- ▶  $a2 = a1 \cdot (a + a2)$  Distr.
- ▶  $a2 = a1 \cdot 1$  Hipot.
- ▶  $a2 = a1$  Tesis

# Teoremas, absorción, parte 1

- ▶  $a + (a \cdot b) = a,$  ¿
  - ▶  $a + a \cdot b = a \cdot 1 + a \cdot b,$  Ax. del 1
  - ▶  $a + a \cdot b = a \cdot (1 + b),$  Distrib.
  - ▶  $a + a \cdot b = a \cdot 1,$  Prop. Del 1
  - ▶  $a + a \cdot b = a,$  Ax. del 1

# Teoremas, absorción, parte 2

# Leyes de De Morgan

- ▶ Leyes de De Morgan
- ▶
- ▶  $(a + b)' = a' \cdot b'$
- ▶
- ▶  $(a \cdot b)' = a' + b'$
  
- ▶ Ejercicio:
- ▶
- ▶ Demostrar las leyes de De Morgan.

# Leyes de De Morgan

- ▶ Leyes de De Morgan
- ▶
- ▶  $(a + b)' = a' \cdot b'$
- ▶
- ▶  $(a \cdot b)' = a' + b'$
  
- ▶ Ejercicio:
- ▶
- ▶ Demostrar las leyes de De Morgan.

# Operaciones elementales

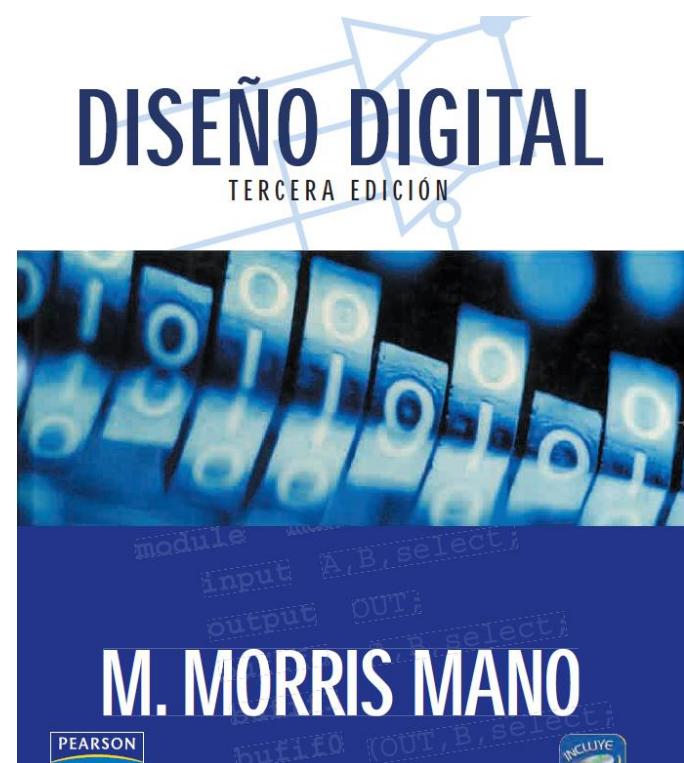
$x$	$y$	$x \cdot y$
0	0	0
0	1	0
1	0	0
1	1	1

$x$	$y$	$x + y$
0	0	0
0	1	1
1	0	1
1	1	1

$x$	$x'$
0	1
1	0

# Algebra Booleana y Compuestas

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



# Algebra Booleana y Compuestas

- ▶ 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

# Algebra Booleana y Compuertas

## 2-4 FUNCIONES BOOLEANAS

$$F_1 = x + y'z$$

$$F_2 = x'y'z + x'yz + xy'$$

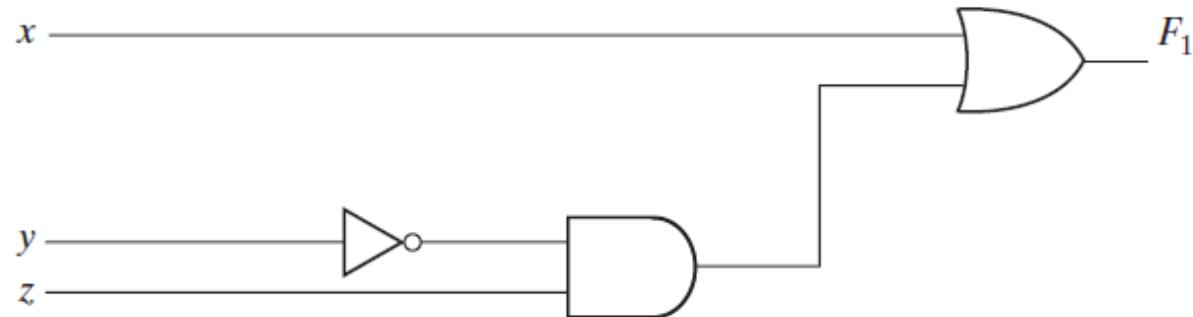
**Tabla 2-2**

*Tablas de verdad para  $F_1$  y  $F_2$*

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

# Algebra Booleana y Compuertas

## 2-4 FUNCIONES BOOLEANAS

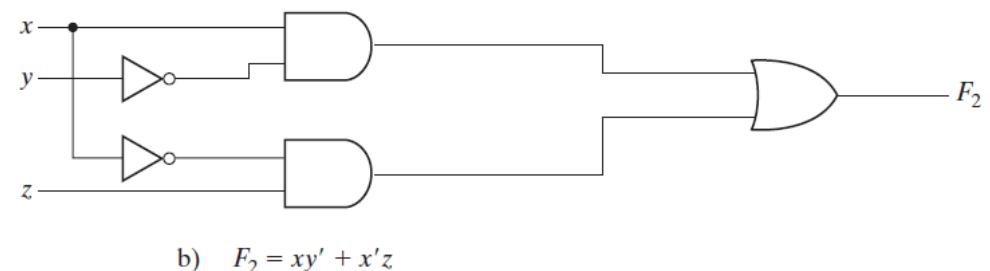
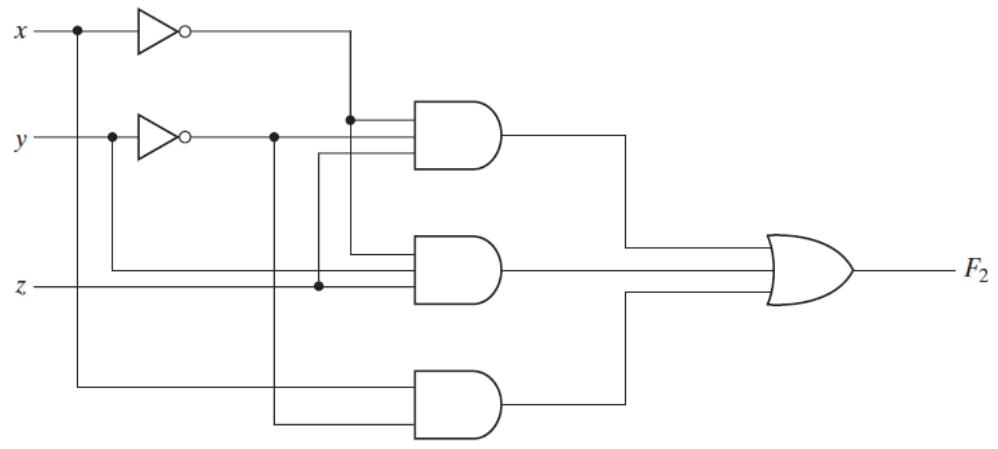


**FIGURA 2-1**

Implementación de  $F_1 = x + y'z$  con compuertas

# Algebra Booleana y Compuertas

## 2-4 FUNCIONES BOOLEANAS



**FIGURA 2-2**  
Implementación de la función booleana  $F_2$  con compuertas

# Algebra Booleana y Compuertas

## 2-5 FORMAS CANÓNICAS Y ESTÁNDAR

### Minitérminos y maxitérminos

**Tabla 2-3**

*Minitérminos y maxitérminos para tres variables binarias*

x	y	z	Minitérminos		Maxitérminos	
			Términos	Designación	Términos	Designación
0	0	0	$x'y'z'$	$m_0$	$x + y + z$	$M_0$
0	0	1	$x'y'z$	$m_1$	$x + y + z'$	$M_1$
0	1	0	$x'yz'$	$m_2$	$x + y' + z$	$M_2$
0	1	1	$x'yz$	$m_3$	$x + y' + z'$	$M_3$
1	0	0	$xy'z'$	$m_4$	$x' + y + z$	$M_4$
1	0	1	$xy'z$	$m_5$	$x' + y + z'$	$M_5$
1	1	0	$xyz'$	$m_6$	$x' + y' + z$	$M_6$
1	1	1	$xyz$	$m_7$	$x' + y' + z'$	$M_7$

# Algebra Booleana y Compuertas

## 2-5 FORMAS CANÓNICAS Y ESTÁNDAR

### Minitérminos y maxitérminos

**Tabla 2-4**  
*Funciones de tres variables*

<b>x</b>	<b>y</b>	<b>z</b>	<b>Función <math>f_1</math></b>	<b>Función <math>f_2</math></b>
0	0	0	0	0
0	0	1	1	0
0	1	0	0	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

# Algebra Booleana y Compuertas

## 2-5 FORMAS CANÓNICAS Y ESTÁNDAR

### Minitérminos y maxitérminos

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

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

# Algebra Booleana y Compuertas

## 2-5 FORMAS CANÓNICAS Y ESTÁNDAR

### Minitérminos y maxitérminos

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

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

# Algebra Booleana y Compuertas

## 2-5 FORMAS CANÓNICAS Y ESTÁNDAR

### Minitérminos y maxitérminos

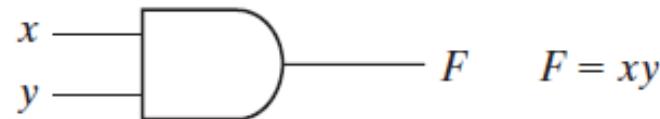
$$\begin{aligned}f_2 &= (x + y + z)(x + y + z')(x + y' + z)(x' + y + z) \\&= M_0M_1M_2M_4\end{aligned}$$

# Algebra Booleana y Compuertas

## 2-7 COMPUERTAS LÓGICAS DIGITALES

Nombre	Símbolo gráfico	Función algebraica	Tabla de verdad
--------	-----------------	--------------------	-----------------

AND



$$F = xy$$

$x$	$y$	$F$
0	0	0
0	1	0
1	0	0
1	1	1

OR



$$F = x + y$$

$x$	$y$	$F$
0	0	0
0	1	1
1	0	1
1	1	1

# Algebra Booleana y Compuertas

## 2-7 COMPUERTAS LÓGICAS DIGITALES

Nombre	Símbolo gráfico	Función algebraica	Tabla de verdad
--------	-----------------	--------------------	-----------------

Inversor



$$F = x'$$

$x$	$F$
0	1
1	0

Búfer



$$F = x$$

$x$	$F$
0	0
1	1

# Algebra Booleana y Compuertas

## 2-7 COMPUERTAS LÓGICAS DIGITALES

Nombre	Símbolo gráfico	Función algebraica	Tabla de verdad
--------	-----------------	--------------------	-----------------

NAND



$$F = (xy)'$$

$x$	$y$	$F$
0	0	1
0	1	1
1	0	1
1	1	0

NOR



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

$x$	$y$	$F$
0	0	1
0	1	0
1	0	0
1	1	0

# Algebra Booleana y Compuertas

## 2-7 COMPUERTAS LÓGICAS DIGITALES

Nombre	Símbolo gráfico	Función algebraica	Tabla de verdad
--------	-----------------	--------------------	-----------------

OR exclusivo  
(XOR)



$$\begin{aligned}F &= xy' + x'y \\&= x \oplus y\end{aligned}$$

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

NOR exclusivo  
o  
equivalencia



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

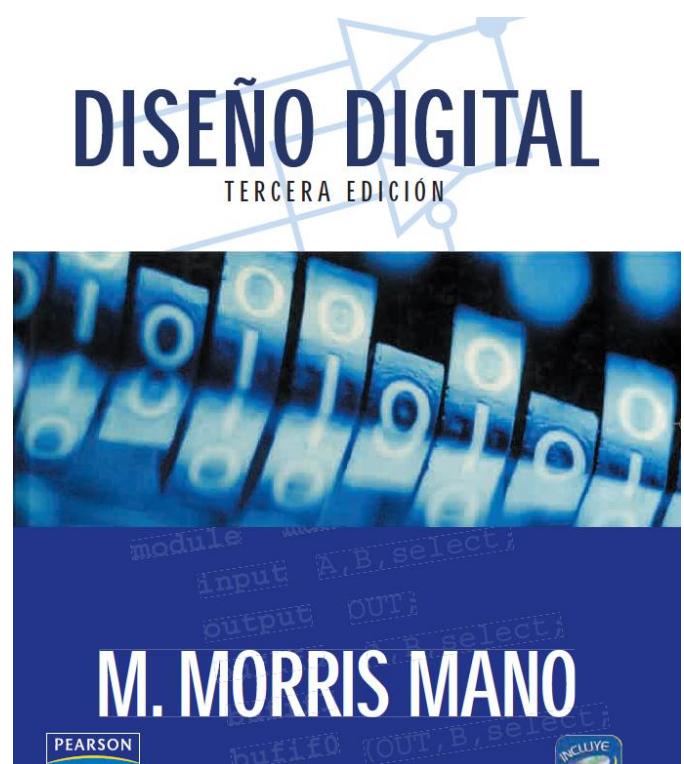
x	y	F
0	0	1
0	1	0
1	0	0
1	1	1

# **MINIMIZACIÓN DE FUNCIONES LÓGICAS**

## **MÉTODO GRÁFICOS DE KARNAUGH**

# Libro de Base para este Tema

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



# Libro de Base para este Tema

- ▶ 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 3 del libro:

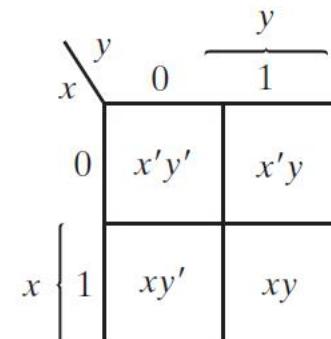


PREFACIO	ix
<b>1 SISTEMAS BINARIOS</b>	<b>1</b>
1-1 Sistemas digitales	1
1-2 Números binarios	3
1-3 Conversiones de base numérica	5
1-4 Números octales y hexadecimales	7
1-5 Complementos	9
1-6 Números binarios con signo	13
1-7 Códigos binarios	16
1-8 Almacenamiento binario y registros	24
1-9 Lógica binaria	27
<b>2 ÁLGEBRA BOOLEANA Y COMPUERTAS LÓGICAS</b>	<b>33</b>
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
<b>3 MINIMIZACIÓN EN EL NIVEL DE COMPUERTAS</b>	<b>64</b>
3-1 El método del mapa	64
3-2 Mapa de cuatro variables	70

# Mapa de 2 Variables

$m_0$	$m_1$
$m_2$	$m_3$

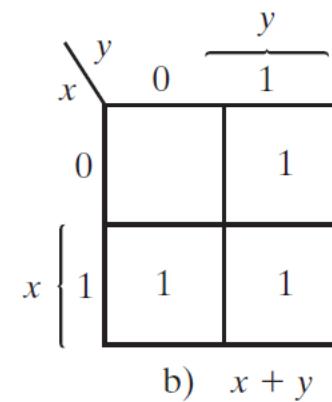
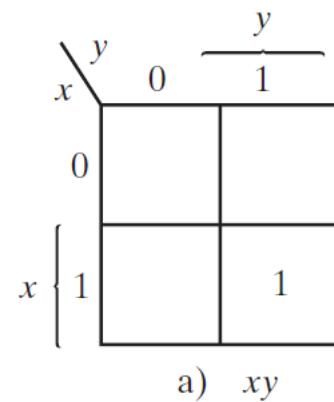
a)



b)

**FIGURA 3-1**  
Mapa de dos variables

# Mapa de 2 Variables



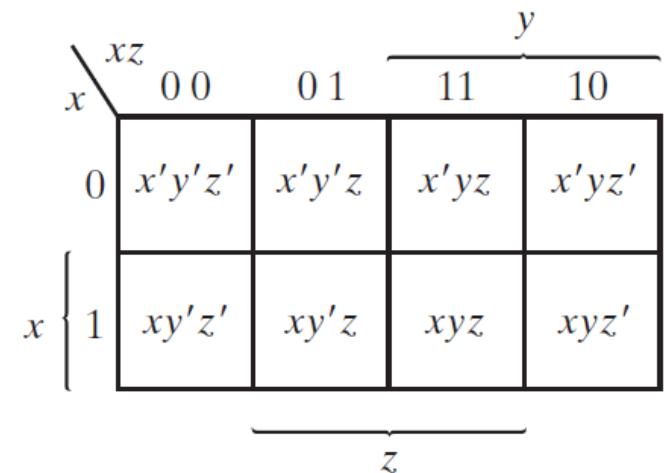
**FIGURA 3-2**  
Representación de funciones en el mapa

# Mapa de 3 Variables

66 Capítulo 3 Minimización en el nivel de compuertas

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

a)



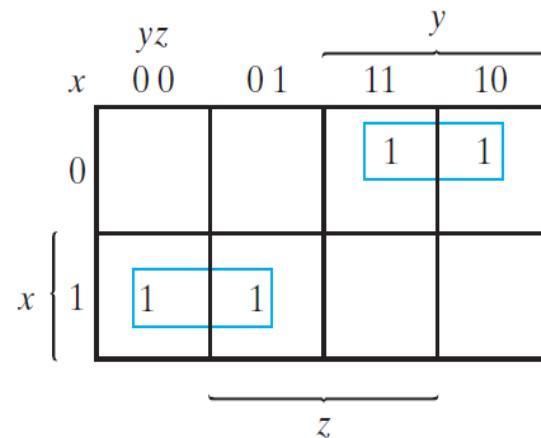
b)

**FIGURA 3-3**  
Mapa de tres variables

# Mapa de 3 Variables

## Sección 3-1 El método del mapa

67



**FIGURA 3-4**

Mapa para el ejemplo 3-1;  $F(x, y, z) = \Sigma(2, 3, 4, 5) = x'y + xy'$

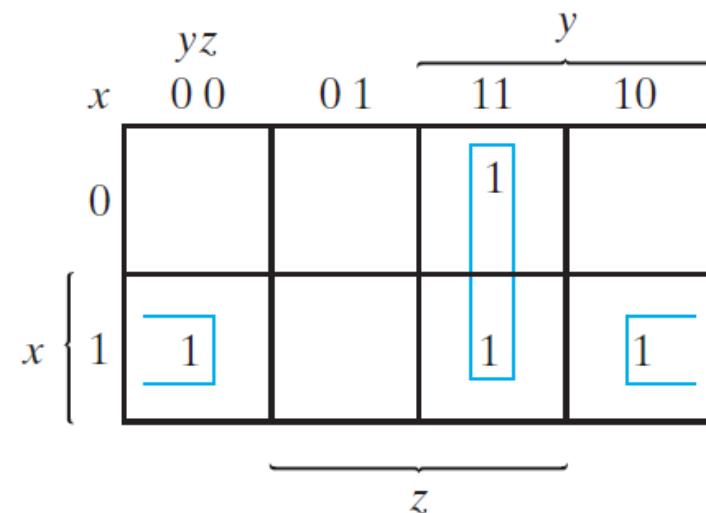
## EJEMPLO 3-1

Simplifique la función booleana

$$F(x, y, z) = \Sigma(2, 3, 4, 5)$$

# Mapa de 3 Variables

68 Capítulo 3 Minimización en el nivel de compuertas



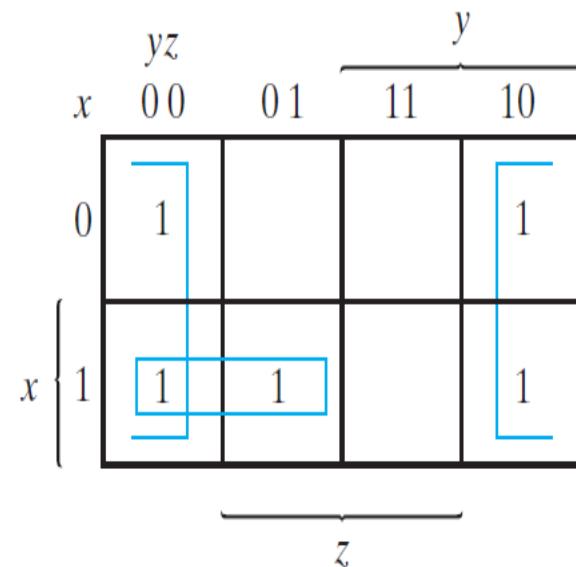
**FIGURA 3-5**

Mapa para el ejemplo 3-2;  $F(x, y, z) = \Sigma(3, 4, 6, 7) = yz + xz'$

# Mapa de 3 Variables

## Sección 3-1 El método del mapa

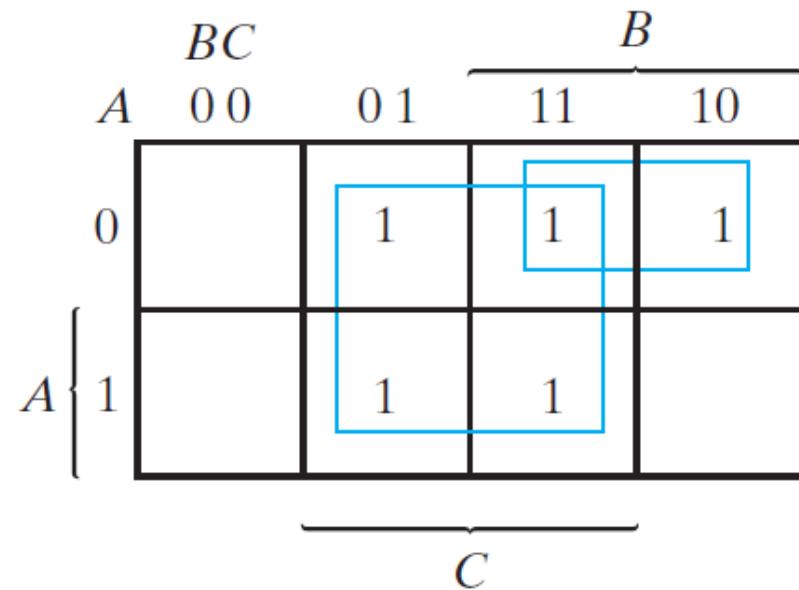
69



**FIGURA 3-6**

Mapa para el ejemplo 3-3;  $F(x, y, z) = \Sigma(0, 2, 4, 5, 6) = z' + xy'$

# Mapa de 3 Variables



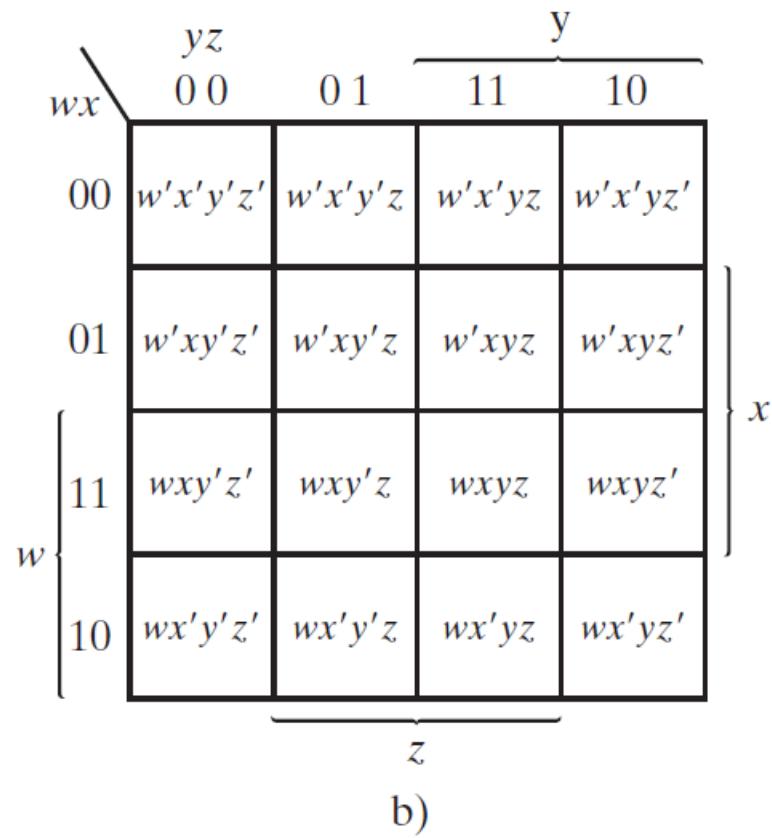
**FIGURA 3-7**

Mapa para el ejemplo 3-4;  $A'C + A'B + AB'C + BC = C + A'B$

# Mapa de 4 Variables

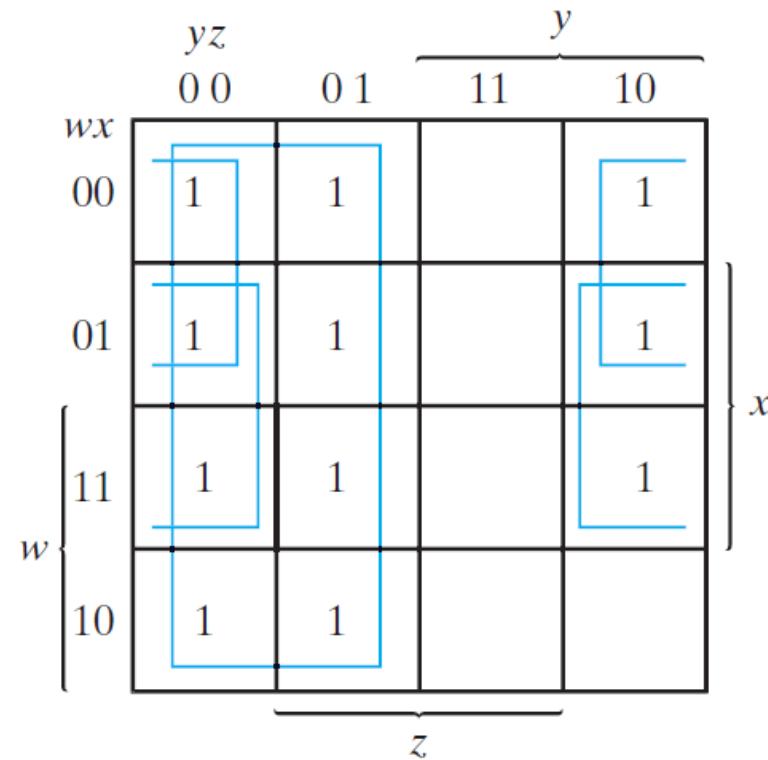
$m_0$	$m_1$	$m_3$	$m_2$
$m_4$	$m_5$	$m_7$	$m_6$
$m_{12}$	$m_{13}$	$m_{15}$	$m_{14}$
$m_8$	$m_9$	$m_{11}$	$m_{10}$

a)



**FIGURA 3-8**  
Mapa de cuatro variables

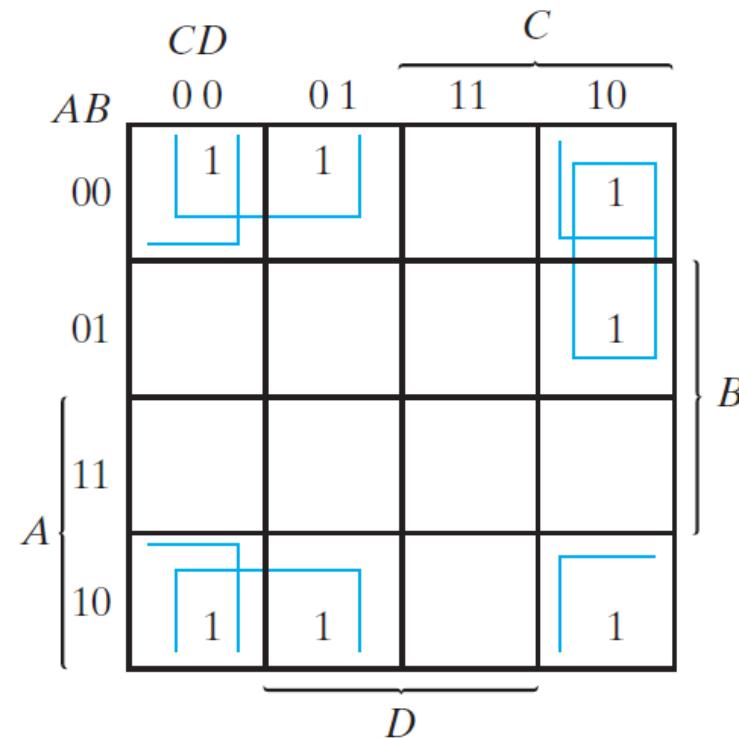
# Mapa de 4 Variables



**FIGURA 3-9**

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

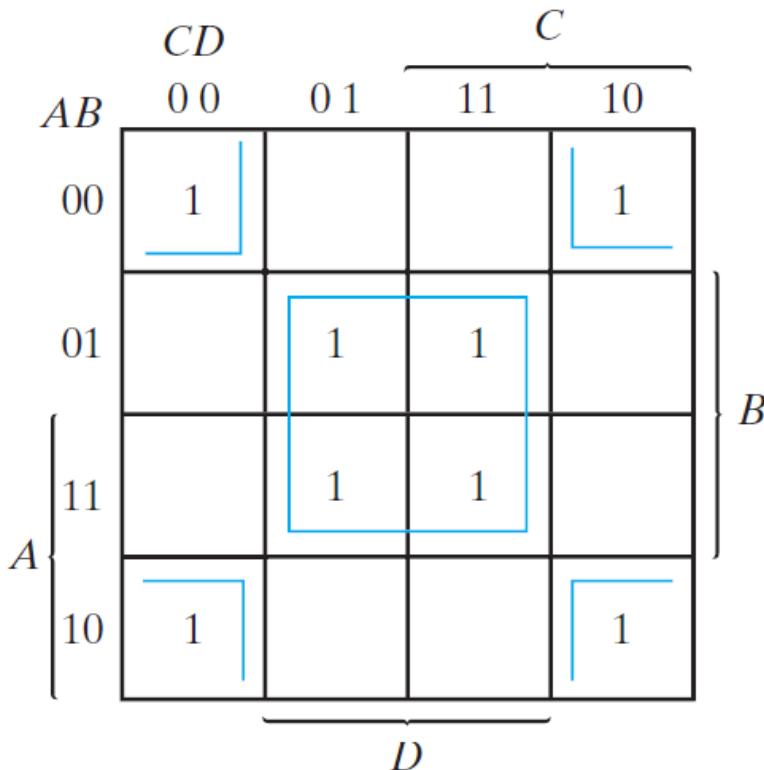
# Mapa de 4 Variables



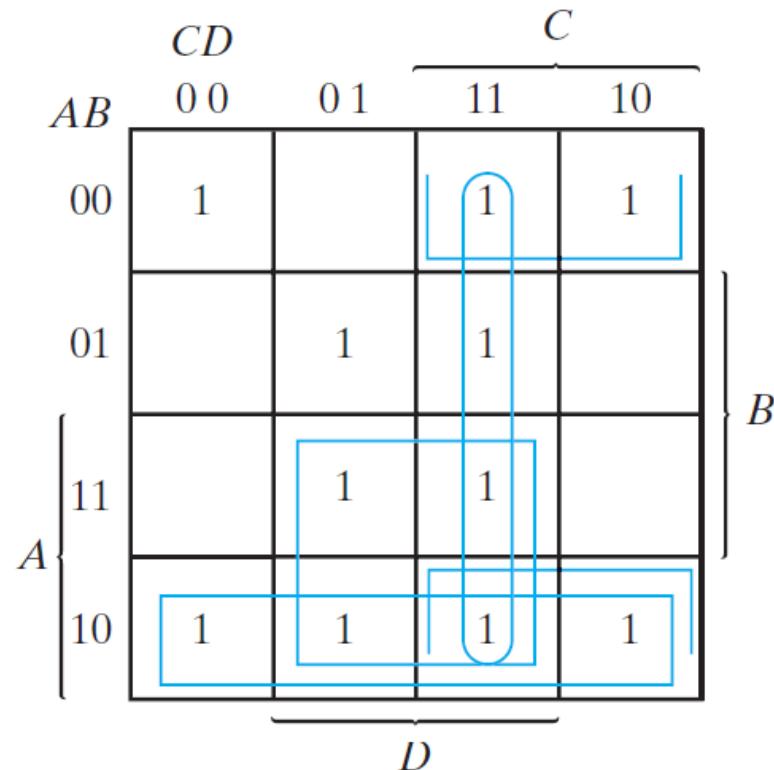
**FIGURA 3-10**

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

# Mapa de 4 Variables



a) Implicantes primos esenciales  
BD y  $B'D'$



b) Implicantes primos CD,  $B'C$ ,  
AD y  $AB'$

**FIGURA 3-11**

Simplificación empleando implicantes primos

# Mapa de 5 Variables

## Sección 3-3 Mapa de cinco variables

75

				$A = 0$	
DE		D			
BC	00	01	11	10	
00	0	1	3	2	
01	4	5	7	6	
B	11	12	13	15	14
10	8	9	11	10	
				$E$	

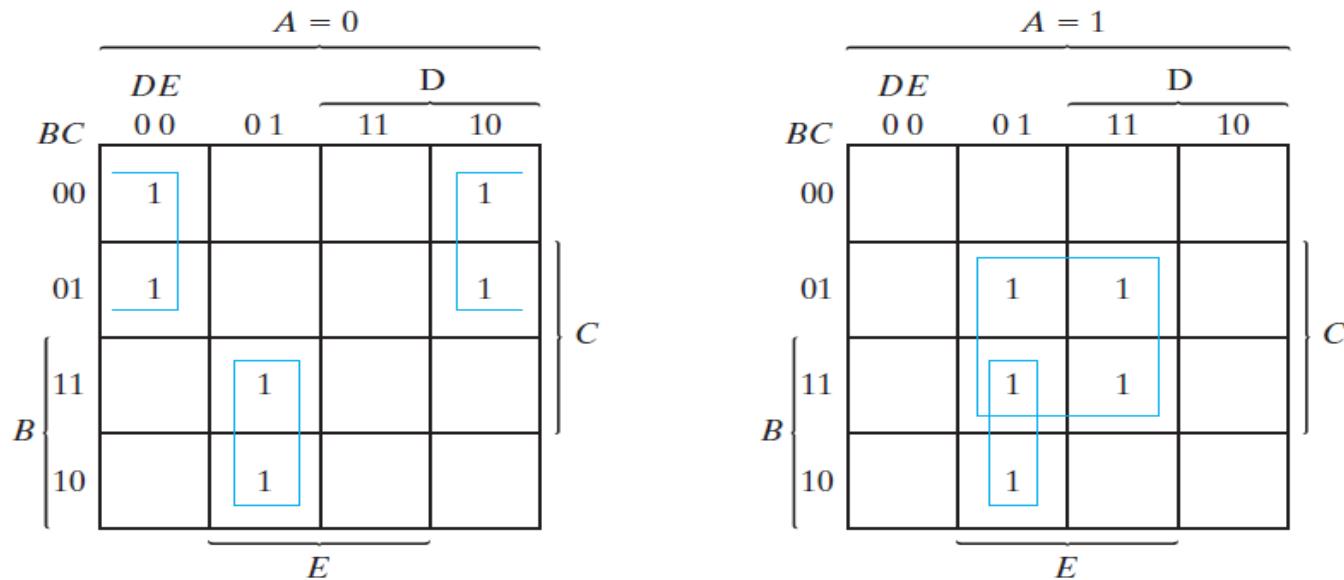
				$A = 1$	
DE		D			
BC	00	01	11	10	
00	16	17	19	18	
01	20	21	23	22	
B	11	28	29	31	30
10	24	25	27	26	
				$E$	

**FIGURA 3-12**  
Mapa de cinco variables

# Mapa de 5 Variables

76

## Capítulo 3 Minimización en el nivel de compuertas



**FIGURA 3-13**

Mapa para el ejemplo 3-7;  $F = A'B'E' + BD'E + ACE$

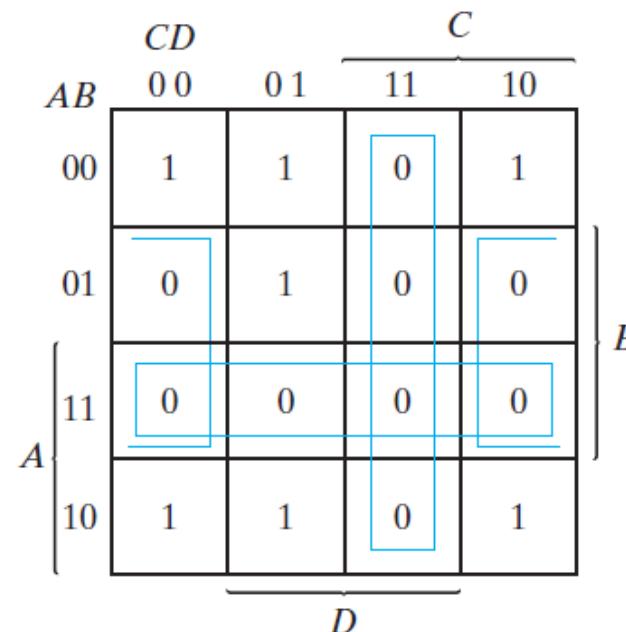
### EJEMPLO 3-7

Simplifique la función booleana

$$F(A, B, C, D, E) = (0, 2, 4, 6, 9, 13, 21, 23, 25, 29, 31)$$

# Suma de Productos y Producto de Sumas

- ▶ Es una aplicación directa de De Morgan



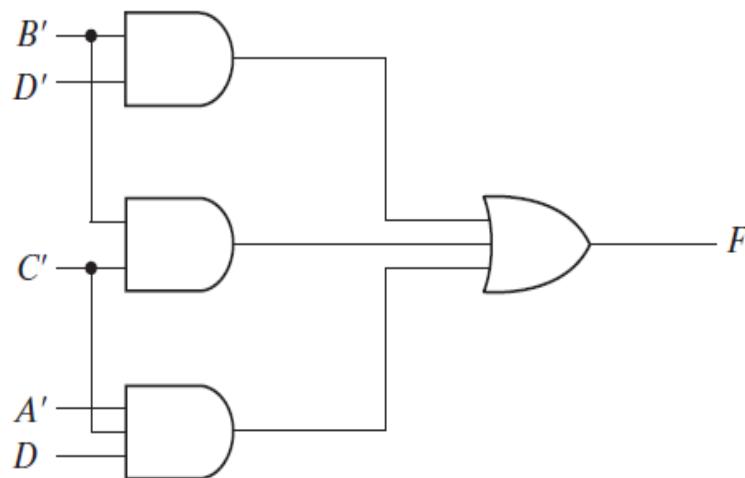
**FIGURA 3-14**

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

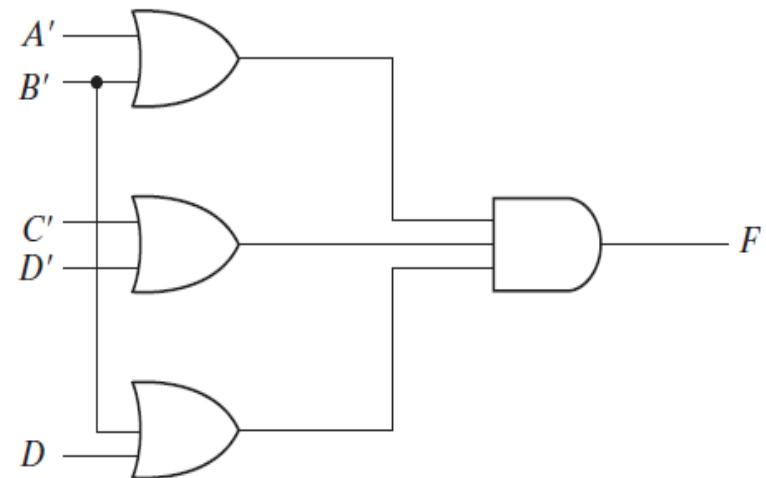
# Suma de Productos y Producto de Sumas

78

Capítulo 3 Minimización en el nivel de compuertas



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



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

**FIGURA 3-15**

Implementación con compuertas de la función del ejemplo 3-8

# Condiciones Sin Cuidado o de Indiferencia (x)

## Sección 3-5 Condiciones de indiferencia

81

		yz		y	
		00	01	11	10
wx		X	1	1	X
w	00	0	X	1	0
	01	0	0	1	0
	11	0	0	1	0
	10	0	0	1	0

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

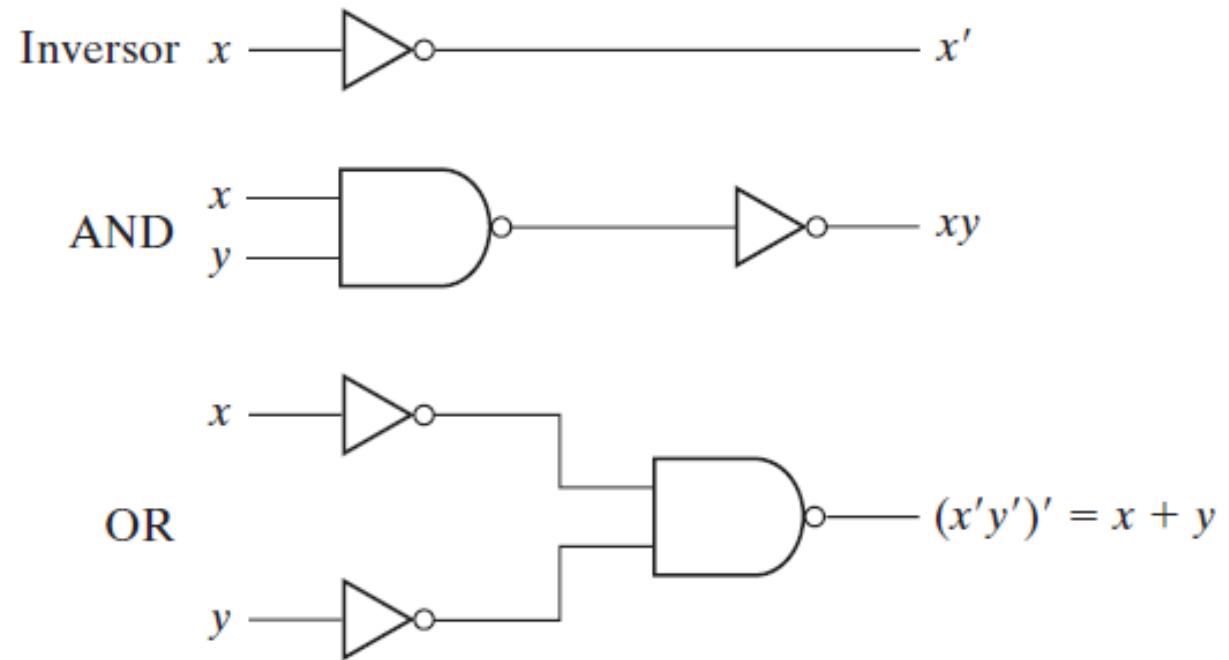
		yz		y	
		00	01	11	10
wx		X	1	1	X
w	00	0	X	1	0
	01	0	0	X	1
	11	0	0	1	0
	10	0	0	1	0

b)  $F = yz + w'z$

FIGURA 3-17

Ejemplo con condiciones de indiferencia

# Implementación con NAND y NOR

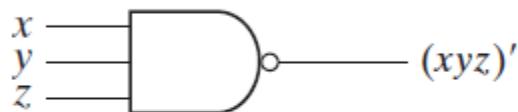


**FIGURA 3-18**  
Operaciones lógicas con compuertas NAND

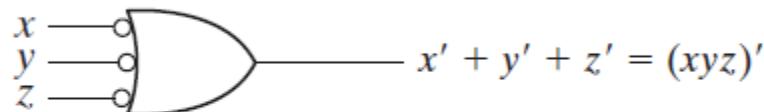
# Implementación con NAND y NOR

## Sección 3-6 Implementación con NAND y NOR

83



a) AND-invertir

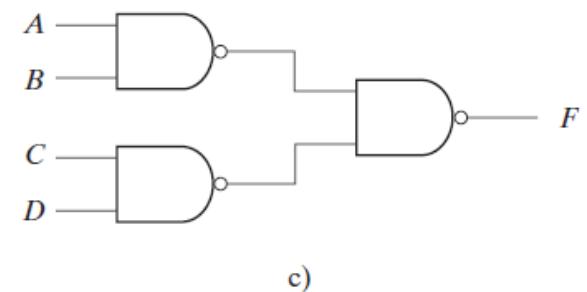
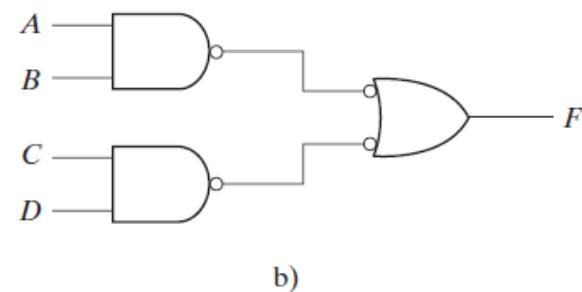
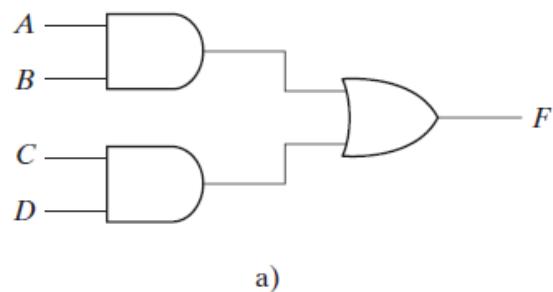


b) Invertir-OR

**FIGURA 3-19**

Dos símbolos gráficos para la compuerta NAND

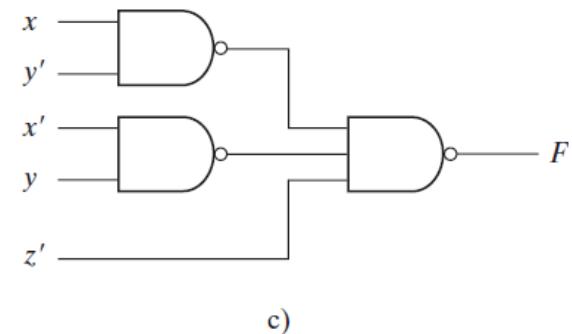
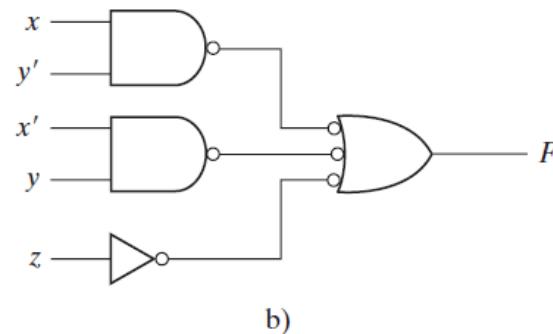
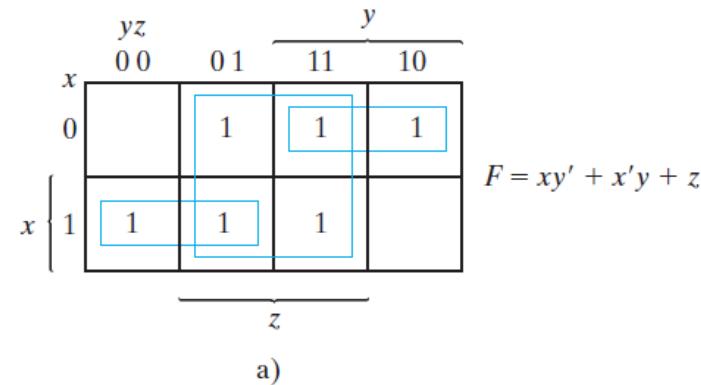
# Implementación con NAND y NOR



**FIGURA 3-20**  
Tres formas de implementar  $F = AB + CD$

# Implementación con Nand y Nor

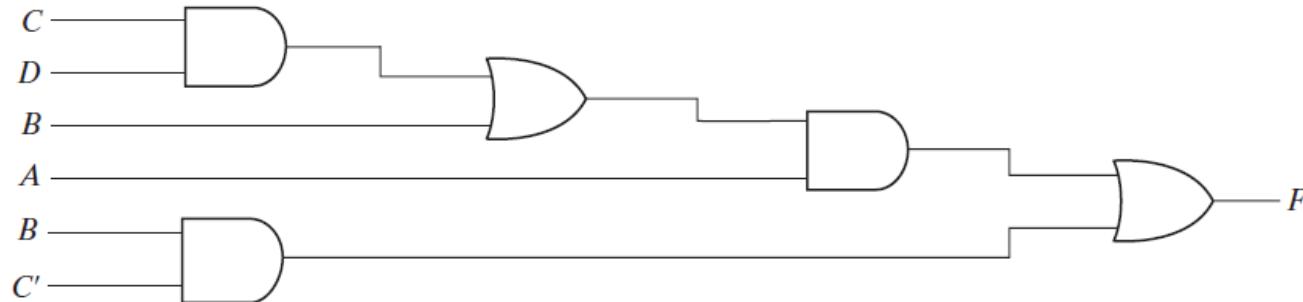
- ▶ También es una aplicación directa de De Morgan



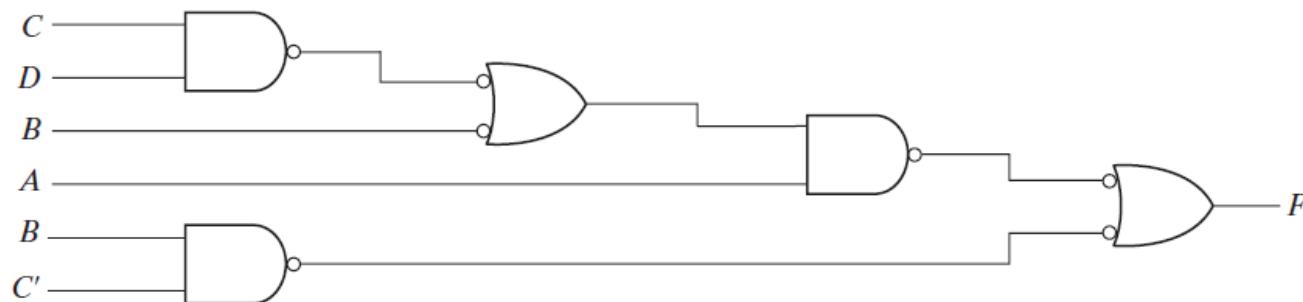
**FIGURA 3-21**  
Solución del ejemplo 3-10

# Implementación con Nand y Nor

86 Capítulo 3 Minimización en el nivel de compuertas



a) Compuertas AND-OR



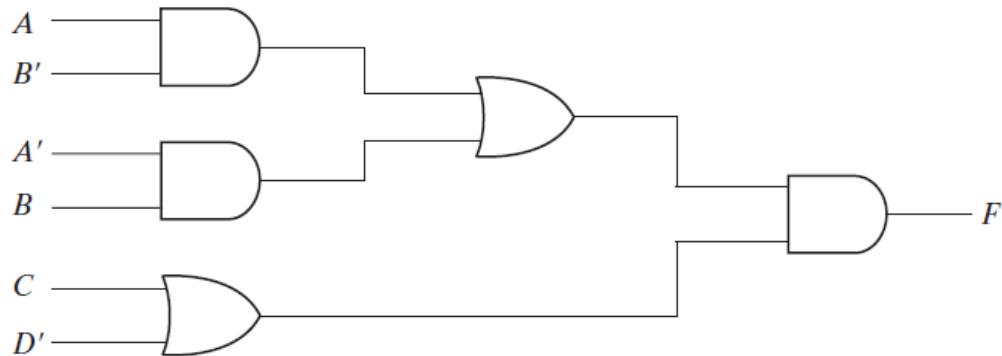
b) Compuertas NAND

**FIGURA 3-22**

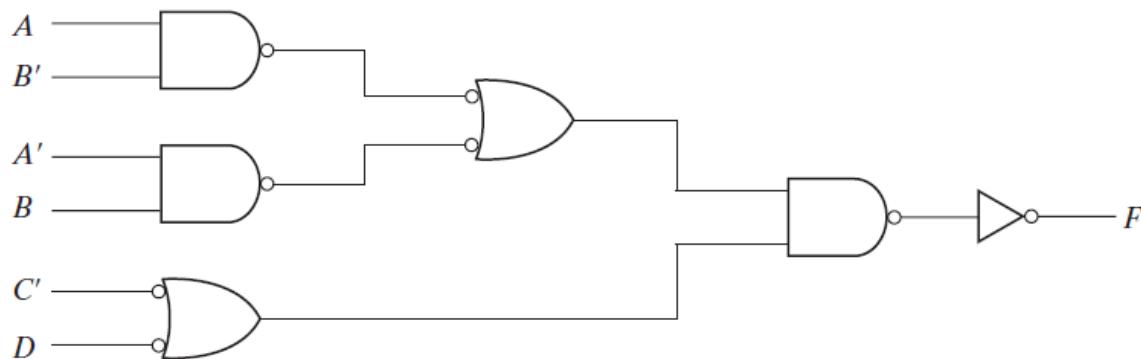
Implementación de  $F = A(CD + B) + BC'$

# Implementación con Nand y Nor

## Sección 3-6 Implementación con NAND y NOR 87



a) Compuertas AND-OR

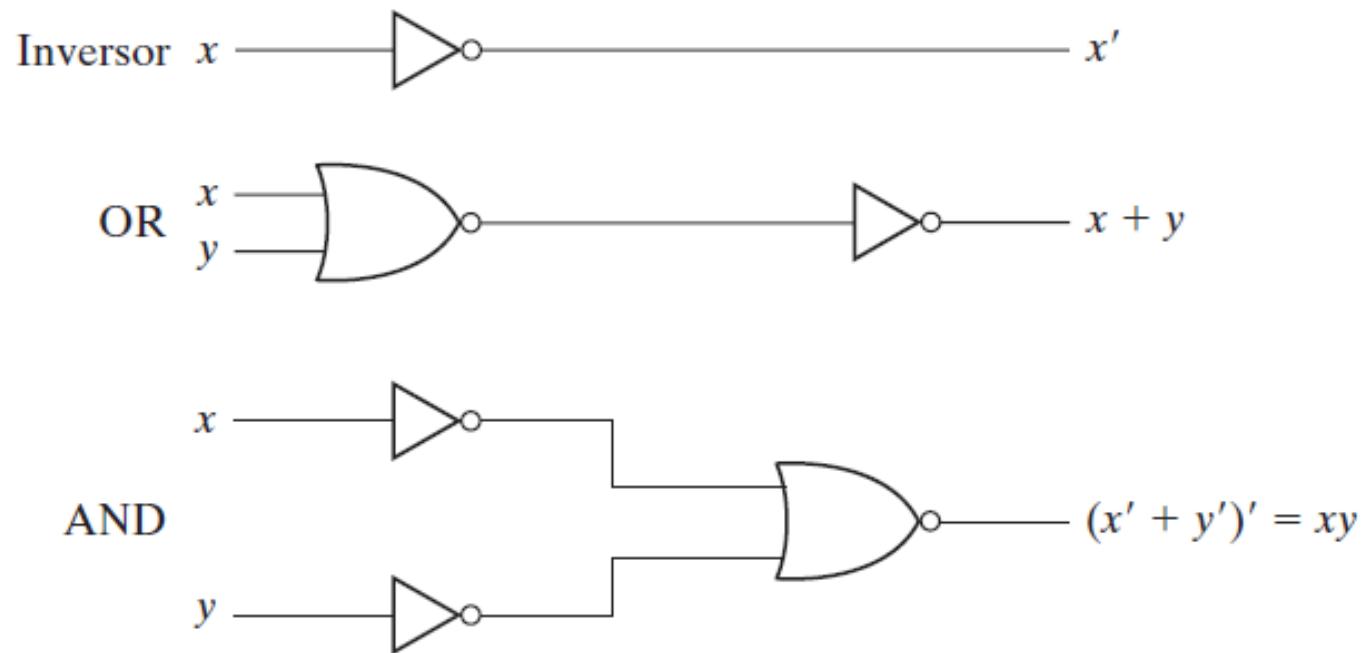


b) Compuertas NAND

FIGURA 3-23

Implementación de  $F = (AB' + A'B)(C + D')$

# Implementación con Nand y Nor



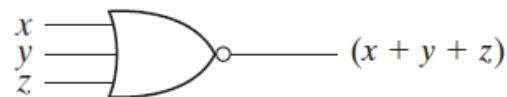
**FIGURA 3-24**

Operaciones lógicas con compuertas NOR

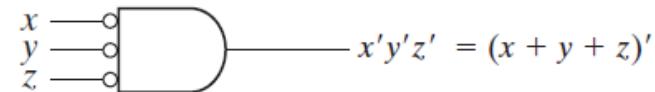
# Implementación con Nand y Nor

88

Capítulo 3 Minimización en el nivel de compuertas



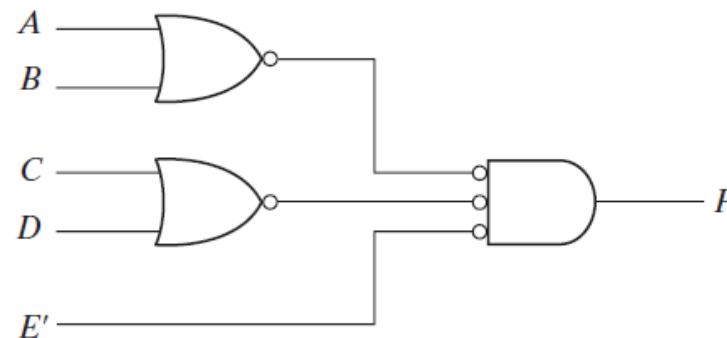
a) OR-invertir



b) invertir-AND

**FIGURA 3-25**

Dos símbolos gráficos para la compuerta NOR



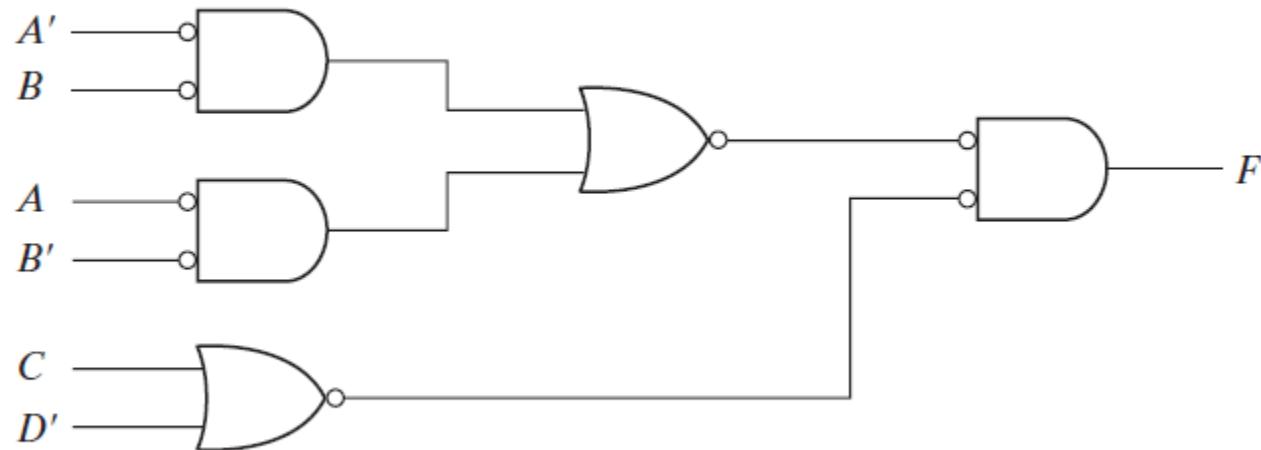
**FIGURA 3-26**

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

# Otras Implementaciones

## Sección 3-7 Otras implementaciones de dos niveles

89



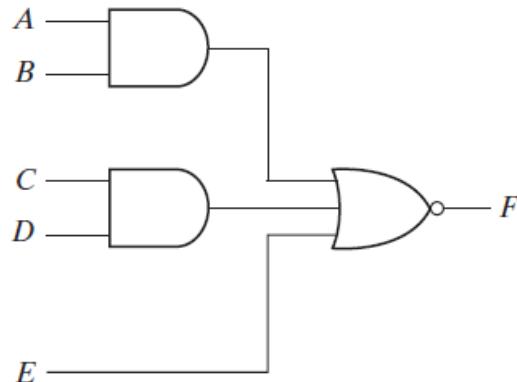
**FIGURA 3-27**

Implementación de  $F = (AB' + A'B)(C + D')$  con compuertas NOR

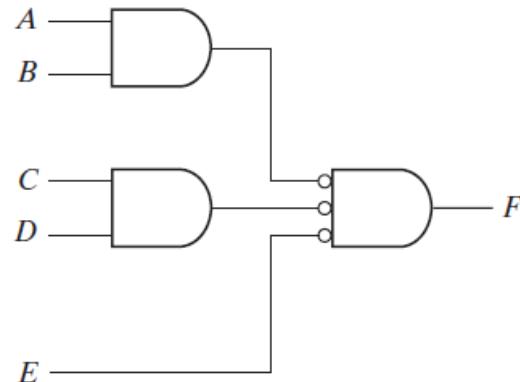
# Otras Implementaciones

## Sección 3-7 Otras implementaciones de dos niveles

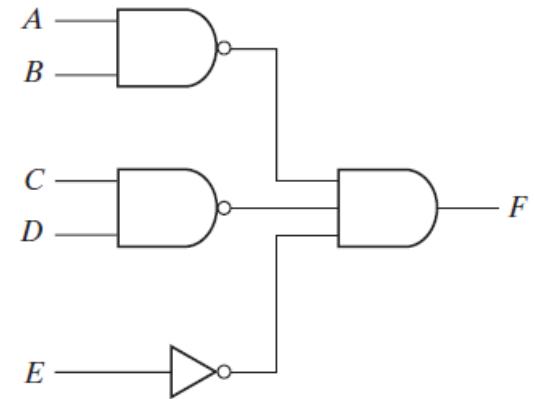
91



a) AND-NOR



b) AND-NOR



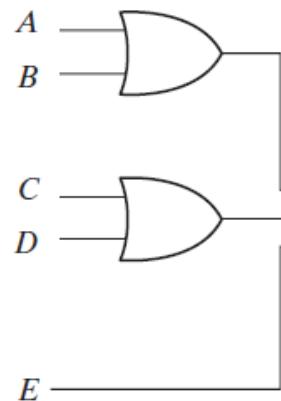
c) NAND-AND

**FIGURA 3-29**

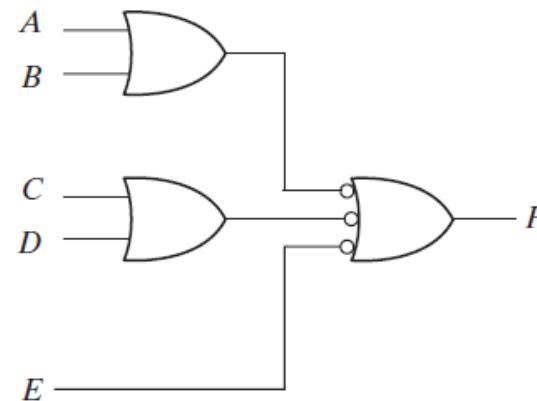
Circuitos AND-OR-INVERT;  $F = (AB + CD + E)'$

# Otras Implementaciones

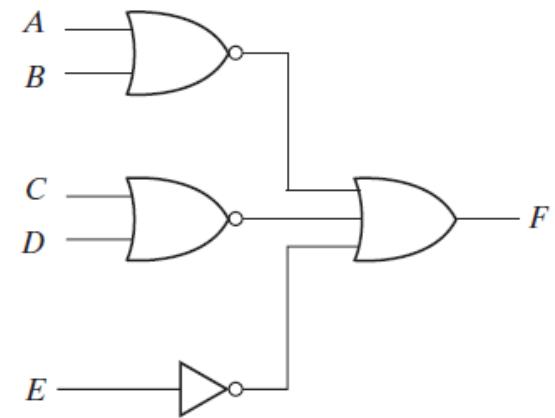
$$F = [(A + B)(C + D)E]'$$



a) OR-NAND



b) OR-NAND



c) NOR-OR

**FIGURA 3-30**

Circuitos OR-AND-INVERT;  $F = [(A + B)(C + D)E]'$

# 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:

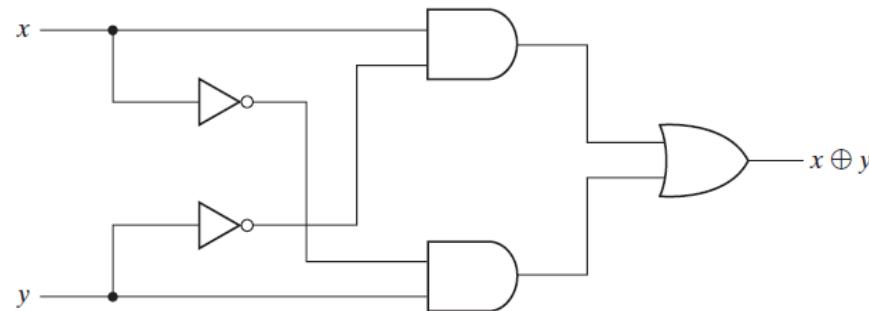
$$(x \oplus y)' = xy + x'y'$$

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:

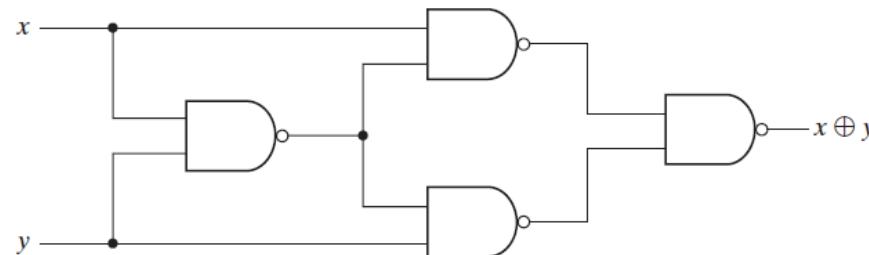
$$(x \oplus y)' = (xy' + x'y)' = (x' + y)(x + y') = xy + x'y'$$

# Función O-Exclusivo

Sección 3-8 Función OR exclusivo 95



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
A	1	1			1
	1			1	

$C$

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

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

$C$

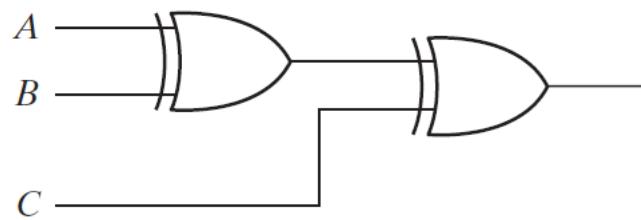
b) Función par  
 $F = (A \oplus B \oplus C)'$

**FIGURA 3-33**

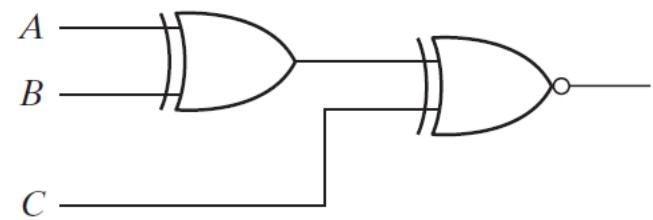
Mapa para una función OR exclusivo de tres variables

# 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

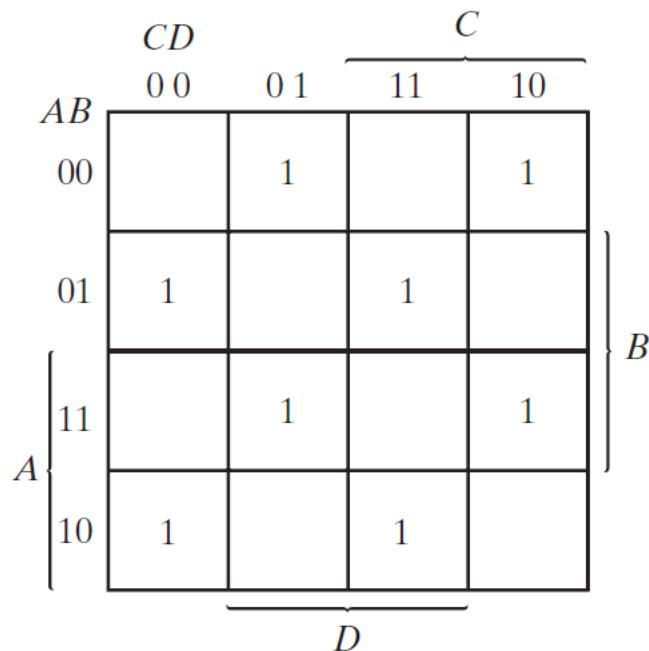


b) Función par de tres entradas

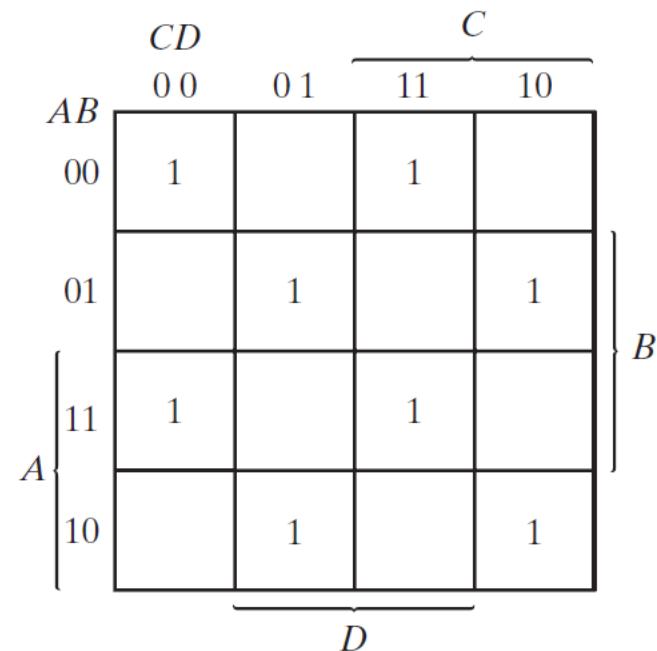
**FIGURA 3-34**

Diagrama lógico de funciones impar y par

# Función Par e Impar



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



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

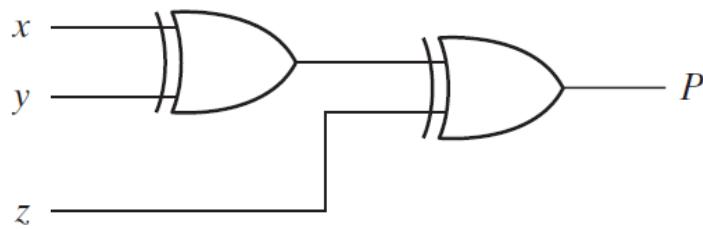
Capítulo 3 Minimización en el nivel de compuertas

**Tabla 3-4**

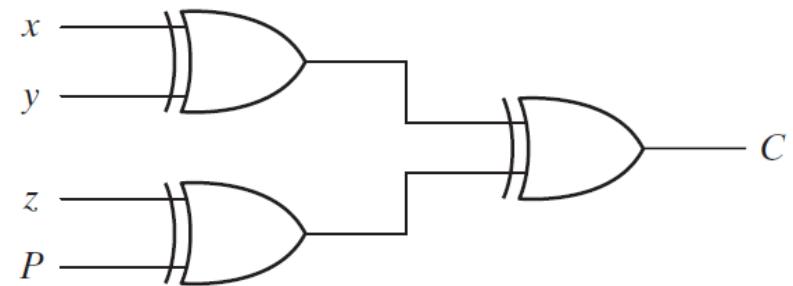
*Tabla de verdad de un generador de paridad par*

Mensaje de tres bits			Bit de paridad
<b>x</b>	<b>y</b>	<b>z</b>	<b>P</b>
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**

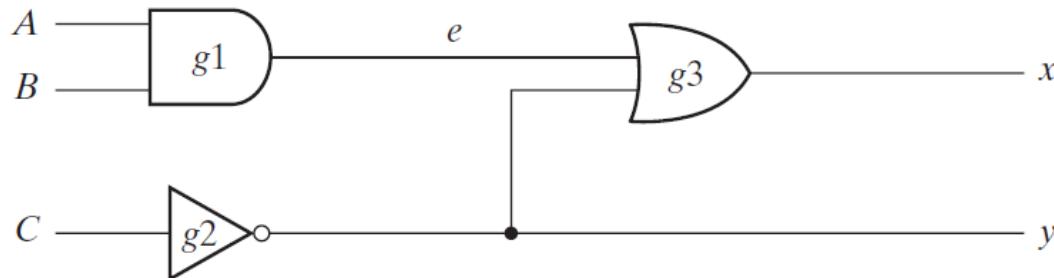
*Tabla de verdad de un verificador de paridad par*

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*

Unidades de tiempo	(ns)	Entrada			Salida			
		A	B	C	y	e	x	
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

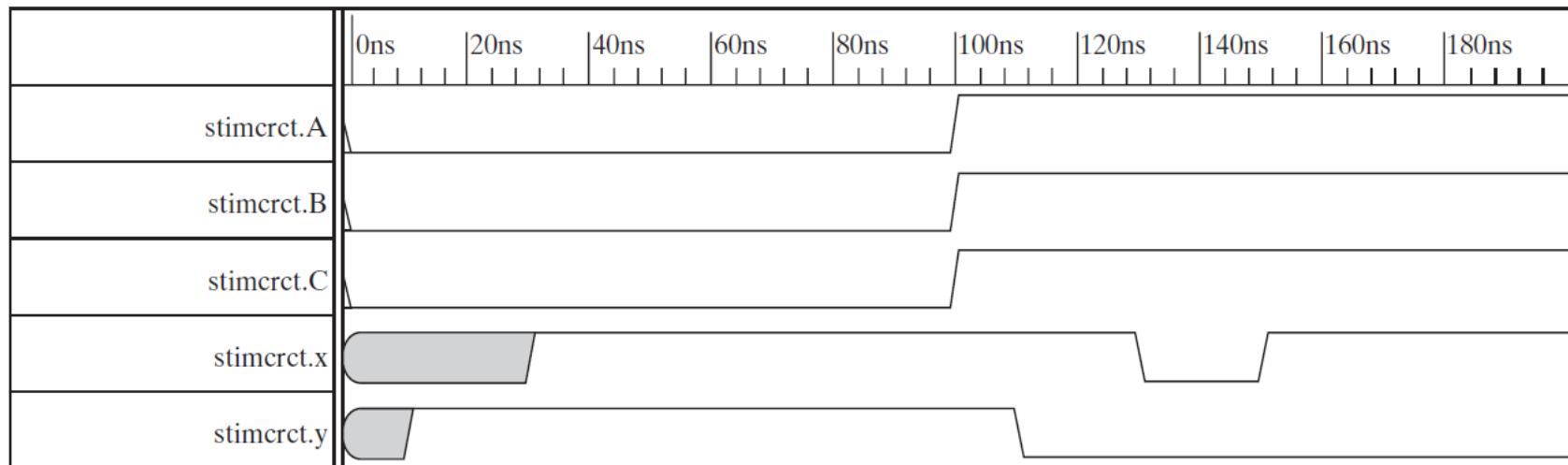
## Ejemplo HDL 3-3

---

```
//Estímulo para el circuito simple
module stimcrct;
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 Hard

## 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) = Σ(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
```

---