

## Tema 2: Álgebra de Boole y puertas lógicas

# Álgebra booleana

- En 1938, Shannon propuso aplicar al diseño de circuitos digitales un método matemático para tratar funciones digitales.
- Dicho método había sido propuesto en el siglo XIX por George Boole para analizar proposiciones lógicas de tipo Verdadero/Falso: el álgebra de Boole.

# Representación de funciones lógicas

- Este método se basa en el uso de variables de dos valores y tres operaciones entre ellas.
- Las relaciones se definen mediante tablas que contienen todos los valores posibles de las variables y de la función → Tablas de la verdad.

AND		
X	Y	$Z = X \cdot Y$
0	0	0
0	1	0
1	0	0
1	1	1

OR		
X	Y	$Z = X + Y$
0	0	0
0	1	1
1	0	1
1	1	1

NOT	
X	$Z = \bar{X}$
0	1
1	0

# Postulados del álgebra de Boole

- Partimos de una serie de afirmaciones que no necesitan ser demostradas → Postulados.
- Utilizando estas afirmaciones (que son la definición de los números y las operaciones de este álgebra), podemos crear nuevas proposiciones y relaciones.

$$X = 0 \text{ si } X \neq 1$$

$$X = 1 \text{ si } X \neq 0$$

$$0 + 0 = 0$$

$$1 \cdot 1 = 1$$

$$1 + 1 = 1$$

$$0 \cdot 0 = 0$$

$$0 + 1 = 1$$

$$1 \cdot 0 = 0$$

$$1 + 0 = 1$$

$$0 \cdot 1 = 0$$

$$\bar{X} = 0 \text{ si } X = 1$$

$$\bar{X} = 1 \text{ si } X = 0$$

# Identidades básicas del álgebra de Boole

---

1.	$X + 0 = X$	2.	$X \cdot 1 = X$	
3.	$X + 1 = 1$	4.	$X \cdot 0 = 0$	
5.	$X + X = X$	6.	$X \cdot X = X$	
7.	$X + \overline{X} = 1$	8.	$X \cdot \overline{X} = 0$	
9.	$\overline{\overline{X}} = X$			

---

10.	$X + Y = Y + X$	11.	$XY = YX$	Conmutativa
12.	$X + (Y + Z) = (X + Y) + Z$	13.	$X(YZ) = (XY)Z$	Asociativa
14.	$X(Y + Z) = XY + XZ$	15.	$X + YZ = (X + Y)(X + Z)$	Distributiva
16.	$\overline{X + Y} = \overline{X} \cdot \overline{Y}$	17.	$\overline{X \cdot Y} = \overline{X} + \overline{Y}$	De Morgan

---

$$(X + Y) \cdot \left( X + \overline{Y} \right) = X \quad \text{Combinación} \quad X \cdot Y + X \cdot \overline{Y} = X$$

Comprobando que cumplen los postulados, sabemos que estas relaciones son correctas

# Identidades básicas del álgebra de Boole

- Las identidades se pueden comprobar definiendo la tabla de la verdad de las funciones a ambos lados de la igualdad y comprobando que son iguales
- Aquí tenéis la prueba de la distributiva del producto respecto a la suma

X	Y	Z	$Y+Z$	$X \cdot (Y+Z)$	$X \cdot Y$	$X \cdot Z$	$X \cdot Y + X \cdot Z$
0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	0	0	0	0
1	0	0	0	0	0	0	0
1	0	1	1	1	0	1	1
1	1	0	1	1	1	0	1
1	1	1	1	1	1	1	1

# Identidades básicas del álgebra de Boole

- Aunque no forma parte de las operaciones básicas, existe otra operación que se define por su utilidad: la suma exclusiva (XOR).

$X$	$Y$	$X \oplus Y$
0	0	0
0	1	1
1	0	1
1	1	0

- El resultado es uno si los operandos son distintos.

# Identidades básicas del álgebra de Boole

- La suma exclusiva también tiene la propiedad asociativa.
- Por tanto, se puede definir como una función de tres o más variables que vale 1 si el número de 1s en la entrada es impar → Función impar.

$X$	$Y$	$Z$	$Y \oplus Z$	$X \oplus (Y \oplus Z)$	$X \oplus Y$	$(X \oplus Y) \oplus Z$
0	0	0	0	0	0	0
0	0	1	1	1	0	1
0	1	0	1	1	1	1
0	1	1	0	0	1	0
1	0	0	0	1	1	1
1	0	1	1	0	1	0
1	1	0	1	0	0	0
1	1	1	0	1	0	1



# Expresión canónica de una función

- Mediante el álgebra booleana podemos crear funciones nuevas.
- De la tabla de la verdad de una función lógica, podemos obtener su expresión algebraica → Expresión canónica.
- La expresión canónica tiene todas las variables de la función en todos sus términos.
- Los términos de la expresión canónica pueden ser productos (mintérminos) o sumas (maxtérminos).

# Expresión canónica de una función

X	Y	Z	Término de producto	Símbolo	$m_0$	$m_1$	$m_2$	$m_3$	$m_4$	$m_5$	$m_6$	$m_7$
0	0	0	$\overline{X}\overline{Y}\overline{Z}$	$m_0$	1	0	0	0	0	0	0	0
0	0	1	$\overline{X}\overline{Y}Z$	$m_1$	0	1	0	0	0	0	0	0
0	1	0	$\overline{X}Y\overline{Z}$	$m_2$	0	0	1	0	0	0	0	0
0	1	1	$\overline{X}YZ$	$m_3$	0	0	0	1	0	0	0	0
1	0	0	$X\overline{Y}\overline{Z}$	$m_4$	0	0	0	0	1	0	0	0
1	0	1	$X\overline{Y}Z$	$m_5$	0	0	0	0	0	1	0	0
1	1	0	$XY\overline{Z}$	$m_6$	0	0	0	0	0	0	1	0
1	1	1	$XYZ$	$m_7$	0	0	0	0	0	0	0	1

- Un mintérmino es un producto que vale 1 sólo para una combinación de valores de las variables y se anula para todas las demás.
- En la expresión del mintérmino, si la variable que no anula el producto vale 1, aparece sin negar, y si vale 0 aparece negada.

# Expresión canónica de una función

X	Y	Z	Término de sumas	Símbolo	M <sub>0</sub>	M <sub>1</sub>	M <sub>2</sub>	M <sub>3</sub>	M <sub>4</sub>	M <sub>5</sub>	M <sub>6</sub>	M <sub>7</sub>
0	0	0	$X+Y+Z$	M <sub>0</sub>	0	1	1	1	1	1	1	1
0	0	1	$X+Y+\overline{Z}$	M <sub>1</sub>	1	0	1	1	1	1	1	1
0	1	0	$X+\overline{Y}+Z$	M <sub>2</sub>	1	1	0	1	1	1	1	1
0	1	1	$X+\overline{Y}+\overline{Z}$	M <sub>3</sub>	1	1	1	0	1	1	1	1
1	0	0	$\overline{X}+Y+Z$	M <sub>4</sub>	1	1	1	1	0	1	1	1
1	0	1	$\overline{X}+Y+\overline{Z}$	M <sub>5</sub>	1	1	1	1	1	0	1	1
1	1	0	$\overline{X}+\overline{Y}+Z$	M <sub>6</sub>	1	1	1	1	1	1	0	1
1	1	1	$\overline{X}+\overline{Y}+\overline{Z}$	M <sub>7</sub>	1	1	1	1	1	1	1	0

- Un maxtérmino es una suma que vale 0 sólo para una combinación de valores de las variables.
- En la expresión del maxtérmino, si la variable que anula la suma vale 0, aparece sin negar, y si vale 1 aparece negada.
- Cada maxtérmino es la negación de su mintérmino correspondiente  $\rightarrow M_i = m'_i$ .

# Expresión canónica de una función

- La expresión canónica es la suma de todos los minterminos correspondientes a las combinaciones de valores de variables que hacen 1 la función.
- También es expresión canónica el producto de los maxtérminos correspondientes a las combinaciones de valores de variables que hacen 0 la función.
- De este modo, cada función tiene asociadas dos expresiones canónicas → Suma de minterminos/Producto de maxtérminos.
- De las expresiones canónicas se puede obtener cualquier función, cuya tabla de la verdad conozcamos (pero pueden ser muy largas...).

# Expresión canónica de una función

X	Y	Z	F	$\bar{F}$
0	0	0	1	0
0	0	1	0	1
0	1	0	1	0
0	1	1	0	1
1	0	0	0	1
1	0	1	1	0
1	1	0	0	1
1	1	1	1	0

$$F = \bar{X}\bar{Y}\bar{Z} + \bar{X}Y\bar{Z} + X\bar{Y}Z + XYZ$$

$$= m_0 + m_2 + m_5 + m_7$$

$$\bar{F} = \bar{X}\bar{Y}Z + \bar{X}YZ + X\bar{Y}\bar{Z} + XY\bar{Z}$$

$$= m_1 + m_3 + m_4 + m_6$$

$$F = \overline{m_1 + m_3 + m_4 + m_6} = \bar{m}_1 \cdot \bar{m}_3 \cdot \bar{m}_4 \cdot \bar{m}_6$$

$$M_i = \bar{m}_i$$

$$F = M_1 \cdot M_3 \cdot M_4 \cdot M_6$$

A través de De Morgan se demuestra que las dos expresiones canónicas son idénticas.

# Simplificación de funciones lógicas

- Las expresiones canónicas nos permiten expresar algebraicamente cualquier función lógica.
- Estas expresiones suelen ser largas.
- Mediante el álgebra booleana, es posible transformar una expresión en otra equivalente de menos términos, que sea más sencilla de implementar → Simplificación.
- El método de los mapas de Karnaugh nos permite sistematizar la simplificación de funciones.

# Simplificación mediante mapas de Karnaugh

- El mapa de Karnaugh es una representación de la tabla de la verdad como tabla de doble entrada.
- Cada punto de intersección de fila y columna (casilla del mapa) es un valor de la función.
- Cada casilla se corresponde con un mintérmino, se escribe un uno en los mintérminos de la función.

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

		$y$	
		0	1
$x$	0	$x'y'$	$x'y$
	1	$xy'$	$xy$

# Simplificación mediante mapas de Karnaugh

- Según el número de variables de la función, los mapas de Karnaugh tienen 4 casillas (dos variables), 8 casillas (tres variables), 16 casillas (cuatro variables),...
- Los minterminos correspondientes a casillas adyacentes en el mapa de Karnaugh se pueden agrupar en un sólo producto (combinación).
- En la expresión de ese término (implicante) sólo aparecen las variables de valores comunes a las dos casillas.

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

		$y$			
		$yz$			
		00	01	11	10
$x$	0	$x'y'z'$	$x'y'z$	$x'yz$	$x'yz'$
	1	$xy'z'$	$xy'z$	$xyz$	$xyz'$
		$z$			



# Simplificación mediante mapas de Karnaugh

- $m_1, m_3, m_5$  y  $m_7$  se agrupan en un implicante único:  $C$ .
- $m_3$  y  $m_2$  se agrupan en  $A' \cdot B$ .
- *La suma de estos dos implicantes contiene a todos los minterminos de la función, luego es equivalente a su expresión canónica  $\Rightarrow F = C + A' \cdot B$ .*

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

$C$

$$F = m_1 + m_2 + m_3 + m_5 + m_7$$

$$F = C + A' \cdot B$$

# Simplificación mediante mapas de Karnaugh

- Llamaremos **suma mínima** de una función, a la suma de menor número de términos y con menor número de variables en cada término que incluya a todos los mintérminos de la función.
- Los implicantes que contengan el mayor número (siempre potencias de 2) de casillas, sin que ninguna no sea mintérmino de la función (casilla vacía), se llaman **implicantes primos**.

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

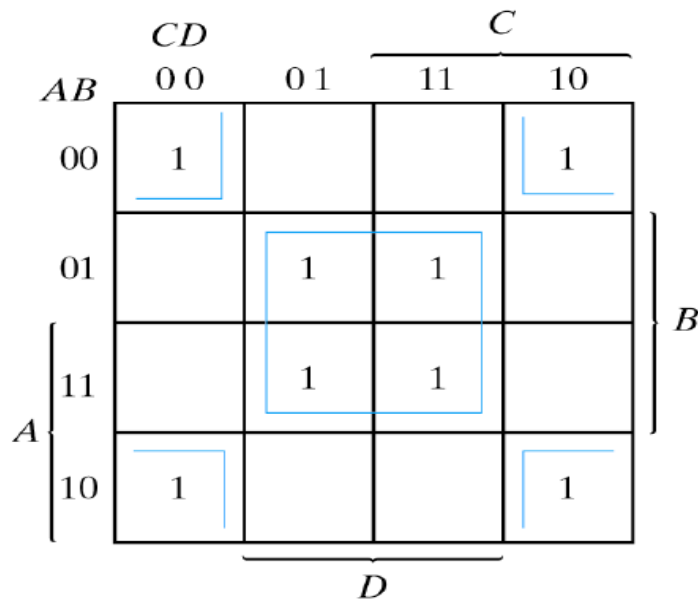
		$yz$		$y$	
		0 0	0 1	1 1	1 0
$w$	$wx$	$w'x'y'z'$	$w'x'y'z$	$w'x'yz$	$w'x'yz'$
	00	$w'xy'z'$	$w'xy'z$	$w'xyz$	$w'xyz'$
	01	$wxy'z'$	$wxy'z$	$wxyz$	$wxyz'$
	11	$wx'y'z'$	$wx'y'z$	$wx'yz$	$wx'yz'$
$w$	10	$wx'y'z'$	$wx'y'z$	$wx'yz$	$wx'yz'$
	10	$wx'y'z'$	$wx'y'z$	$wx'yz$	$wx'yz'$

$z$

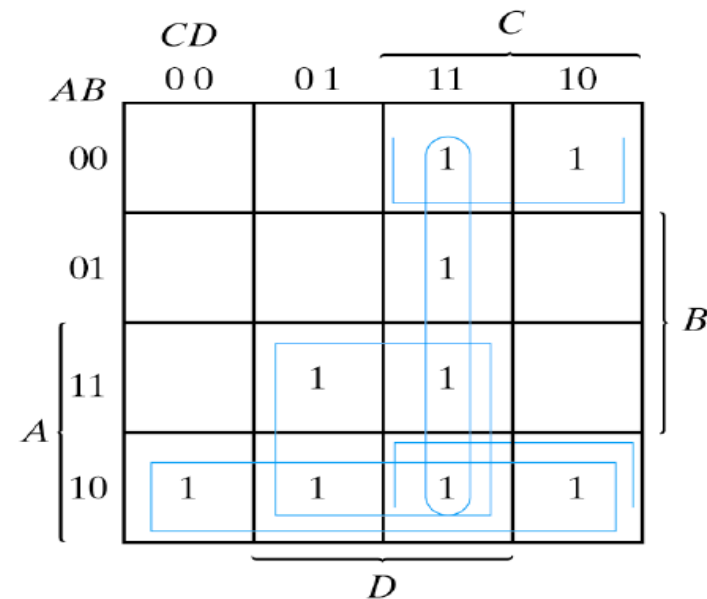
$x$

# Simplificación mediante mapas de Karnaugh

- Las celdas que sólo pertenecen a un implicante primo se llaman **celdas distinguidas**.
- Los implicante primos que contienen una celda distinguida se llaman **implicantes primos esenciales**.
- La **suma mínima** de una función es una **suma de implicantes primos** de esa función (puede que no todos), y siempre contiene a **todos los implicantes primos esenciales**.



*Implicantes primos esenciales  
 $BD$  y  $B'D'$*



*Implicantes primos esenciales  
 $CD$ ,  $B'C$ ,  $AD$  y  $AB'$*

# Simplificación mediante mapas de Karnaugh: usando los ceros

- Las celdas vacías (ceros de la función) corresponden a los maxtérminos de la función.
- Se pueden agrupar (combinación) celdas de ceros adyacentes en un sólo implicante (suma) que contiene a los maxtérminos correspondientes.
- En su expresión sólo aparecen las variables de valor común en todas las casillas.
- El producto de los implicantes de ceros que contiene a todos los maxtérminos de la función, es equivalente a su expresión canónica.
- Llamaremos **producto mínimo** de una función al producto de menor número de términos y con menor número de variables en cada término que incluya a todos los maxtérminos de la función.

# Simplificación mediante mapas de Karnaugh: usando los ceros

- $M_4, M_6, M_{12}$  y  $M_{14}$  se agrupan en un sólo implicante primo:  $B'+D$ .
- Los otros dos son  $A'+B'$  y  $C'+D'$ .
- Puesto que todos los implicantes son esenciales, el producto mínimo es:

$$F=(B'+D) \cdot (A'+B') \cdot (C'+D')$$

Mapa para la función  $F(A, B, C, D)=\Sigma (0, 1, 2, 5, 8, 9, 10)$

		$CD$		$C$	
		00	01	11	10
$AB$	00	1	1	0	1
	01	0	1	0	0
	11	0	0	0	0
	10	1	1	0	1

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

# Simplificación mediante mapas de Karnaugh

- La simplificación se consigue sustituyendo la expresión canónica por la expresión mínima (puede ser la suma o el producto, la más sencilla de las dos).
- Para número de variables de la función mayor que cuatro, no se suelen usar los mapas de Karnaugh, sino métodos iterativos por computador.

# Simplificación de funciones de especificación incompleta

- Las funciones de especificación incompleta son aquéllas para las que no están asignados valores para todas las combinaciones de variables de entrada.
- En muchos casos, no todas las combinaciones de variables son posibles, por lo que no se define valor de la función para ellas, se indica con una X.
- El término asignado para estas combinaciones de variables se llama término “no importa”, y se nombra con  $d$  y el subíndice correspondiente a la combinación de variables para la que no está definida la función.
- En estos casos, se puede escoger para la función el valor que suponga mayor simplificación: 0 ó 1.

# Simplificación de funciones de especificación incompleta

$$F = \sum m(1, 3, 7, 11, 15) + \sum d(0, 2, 5)$$

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

z

a)  $F = y \cdot z + w' \cdot x'$

- Si tomamos 0, 2 y 5 como ceros de la función, el implicante primo sólo contiene 1 y 3:  $w' \cdot x' \cdot z$ .
- Los términos 0 y 2 son tomados como 1 para formar el implicante primo:  $w' \cdot x'$ .
- Dejando 5 como 0, la expresión a) es la suma mínima.



# Simplificación de funciones de especificación incompleta

$$F = \sum m(1, 3, 7, 11, 15) + \sum d(0, 2, 5)$$

		$y$			
		$yz$			
		00	01	11	10
$w$	$x$	00	1	1	$X$
	01	0	$X$	1	0
	11	0	0	1	0
	10	0	0	1	0

- Si tomamos 0 y 2 como ceros y 5 como 1, el nuevo implicante primo es:  $w' \cdot z$ .
- De este modo, la suma mínima es b).
- Esta función es diferente a a), pero responde a la misma especificación incompleta de F.

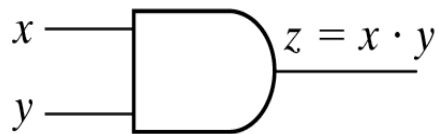
b)  $F = y \cdot z + w' \cdot z$

# Puertas lógicas

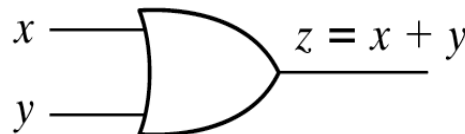
- El álgebra booleana nos permite diseñar funciones lógicas.
- Para aprovechar estas funciones en el diseño digital, necesitamos circuitos digitales que implementen las operaciones booleanas.
- Estos circuitos se llaman puertas lógicas.

# Puertas lógicas

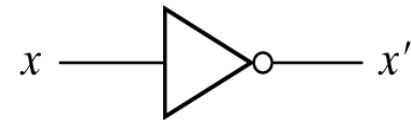
- Las puertas lógicas básicas se corresponden con las tres operaciones del álgebra booleana.



Función lógica Y (AND)



Función lógica O (OR)



Función lógica NO (NOT)

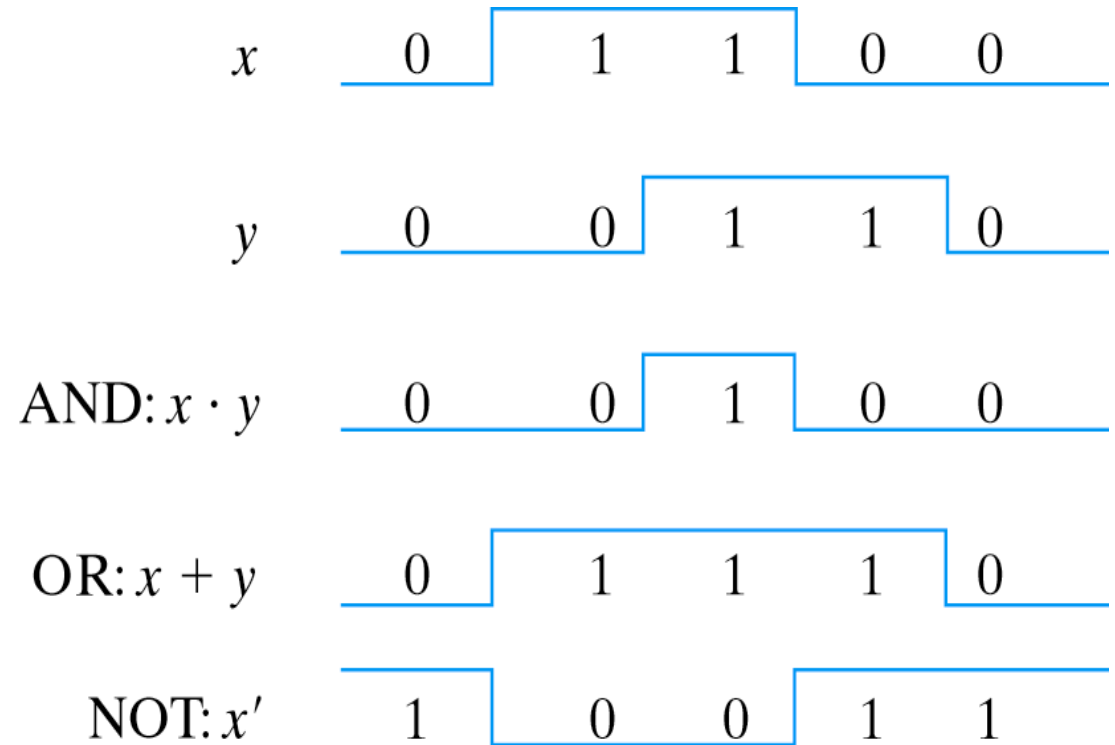
AND		
X	Y	Z = X · Y
0	0	0
0	1	0
1	0	0
1	1	1

OR		
X	Y	Z = X + Y
0	0	0
0	1	1
1	0	1
1	1	1

NOT	
X	Z = $\bar{X}$
0	1
1	0

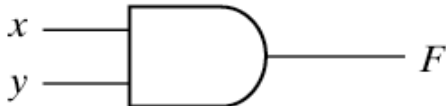

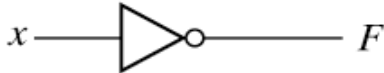
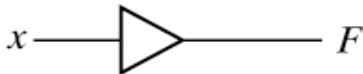
# Puertas lógicas

- Cada variable booleana es sustituida por una señal de tension.
- Al 0 le corresponde la tensión baja (L), y al 1 la tensión alta (H).




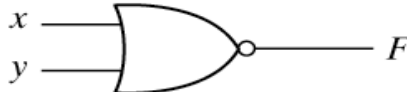


Señales de entrada y de salida de circuitos digitales representadas frente al tiempo (cronogramas).

# Puertas lógicas

Nombre	Símbolo	Expresión algebraica	Tabla de la verdad															
AND		$F = xy$	<table> <tr> <th><math>x</math></th> <th><math>y</math></th> <th><math>F</math></th> </tr> <tr><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>1</td></tr> </table>	$x$	$y$	$F$	0	0	0	0	1	0	1	0	0	1	1	1
$x$	$y$	$F$																
0	0	0																
0	1	0																
1	0	0																
1	1	1																
OR		$F = x + y$	<table> <tr> <th><math>x</math></th> <th><math>y</math></th> <th><math>F</math></th> </tr> <tr><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>1</td></tr> </table>	$x$	$y$	$F$	0	0	0	0	1	1	1	0	1	1	1	1
$x$	$y$	$F$																
0	0	0																
0	1	1																
1	0	1																
1	1	1																
Inversor		$F = x'$	<table> <tr> <th><math>x</math></th> <th><math>F</math></th> </tr> <tr><td>0</td><td>1</td></tr> <tr><td>1</td><td>0</td></tr> </table>	$x$	$F$	0	1	1	0									
$x$	$F$																	
0	1																	
1	0																	
Buffer		$F = x$	<table> <tr> <th><math>x</math></th> <th><math>F</math></th> </tr> <tr><td>0</td><td>0</td></tr> <tr><td>1</td><td>1</td></tr> </table>	$x$	$F$	0	0	1	1									
$x$	$F$																	
0	0																	
1	1																	

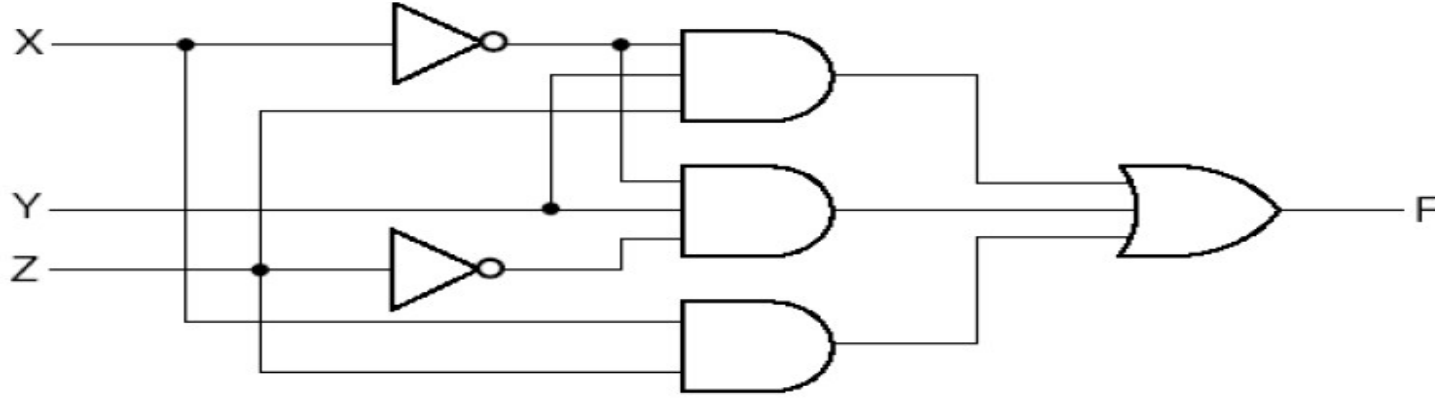
Aunque el buffer no tiene función aparente, se usa para mantener los valores digitales de tensión e intensidad a través de sucesivas puertas lógicas.

# Puertas lógicas

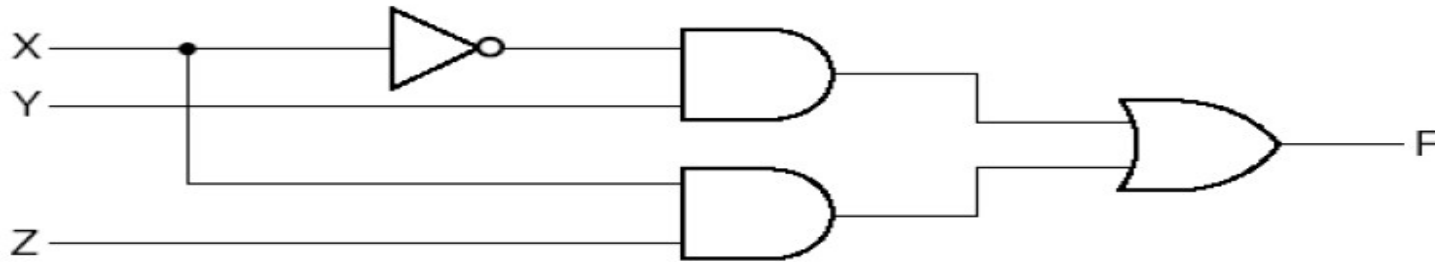
Nombre	Símbolo	Expresión algebraica	Tabla de la verdad															
NAND		$F = (xy)'$	<table> <tr> <th><math>x</math></th> <th><math>y</math></th> <th><math>F</math></th> </tr> <tr><td>0</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>0</td></tr> </table>	$x$	$y$	$F$	0	0	1	0	1	1	1	0	1	1	1	0
$x$	$y$	$F$																
0	0	1																
0	1	1																
1	0	1																
1	1	0																
NOR		$F = (x + y)'$	<table> <tr> <th><math>x</math></th> <th><math>y</math></th> <th><math>F</math></th> </tr> <tr><td>0</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>0</td></tr> </table>	$x$	$y$	$F$	0	0	1	0	1	0	1	0	0	1	1	0
$x$	$y$	$F$																
0	0	1																
0	1	0																
1	0	0																
1	1	0																
OR exclusivo (XOR)		$F = xy' + x'y$ $= x \oplus y$	<table> <tr> <th><math>x</math></th> <th><math>y</math></th> <th><math>F</math></th> </tr> <tr><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>0</td></tr> </table>	$x$	$y$	$F$	0	0	0	0	1	1	1	0	1	1	1	0
$x$	$y$	$F$																
0	0	0																
0	1	1																
1	0	1																
1	1	0																
NOR exclusivo (equivalencia)		$F = xy + x'y'$ $= (x \oplus y)'$	<table> <tr> <th><math>x</math></th> <th><math>y</math></th> <th><math>F</math></th> </tr> <tr><td>0</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>1</td></tr> </table>	$x$	$y$	$F$	0	0	1	0	1	0	1	0	0	1	1	1
$x$	$y$	$F$																
0	0	1																
0	1	0																
1	0	0																
1	1	1																

Por su utilidad, o por facilidad de fabricación, se han diseñado otras puertas lógicas que no son operadores básicos del álgebra booleana.

# Síntesis de circuitos lógicos



(a)  $F = \bar{X}YZ + \bar{X}Y\bar{Z} + XZ$

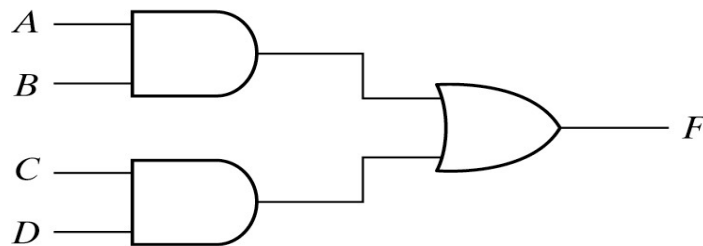


(b)  $F = \bar{X}Y + XZ$

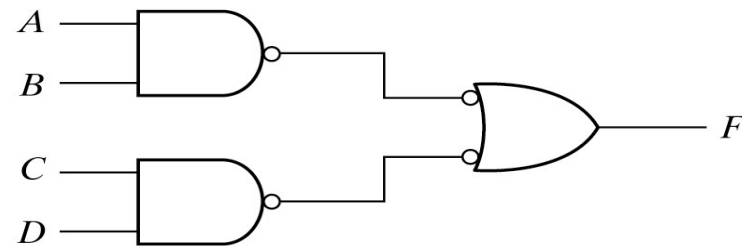
- Cualquier expresión del álgebra de Boole se puede transformar en circuito mediante las puertas lógicas, sustituyendo cada operador de Boole por la puerta lógica correspondiente.
- Cuanto más sencilla sea la expresión, menor será el número de puertas y más sencillo, rápido y eficaz el circuito.

# Circuitos de dos niveles

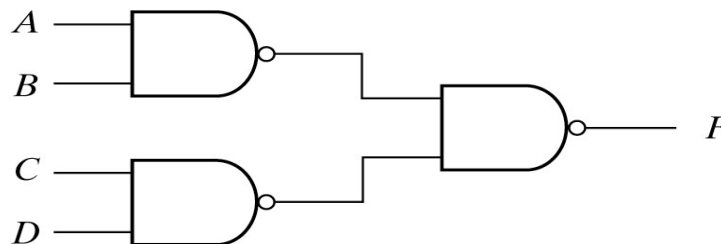
- Si la expresión algebraica es de suma de productos o de producto de sumas, sólo hay dos puertas entre entrada y salida del circuito.
- En ese caso se puede implementar todo el circuito sólo con puertas NAND:



(a)



(b)

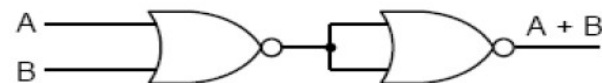
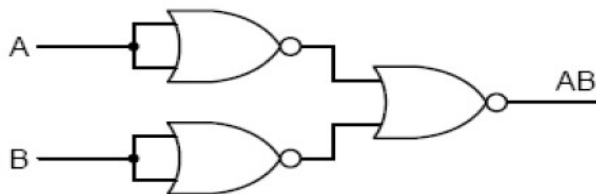
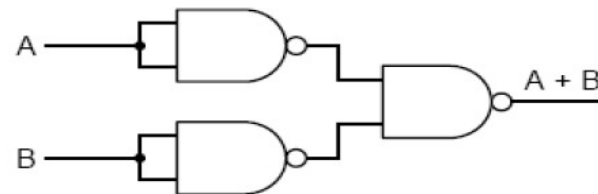
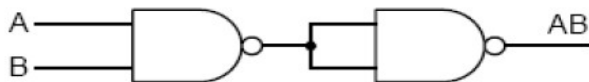
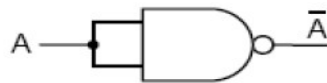


(c)



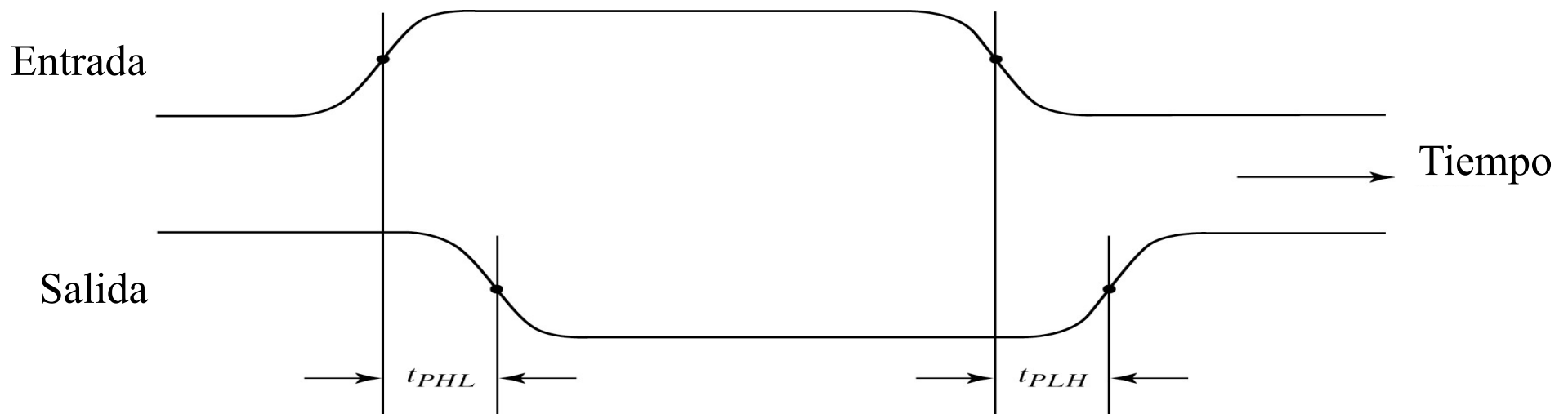
# Circuitos de dos niveles

- Esto se puede aplicar a un producto de sumas para obtener un circuito sólo de puertas NOR.
- NAND y NOR son capaces de implementar por separado todas las operaciones del álgebra de Boole.



# Tiempo de retardo

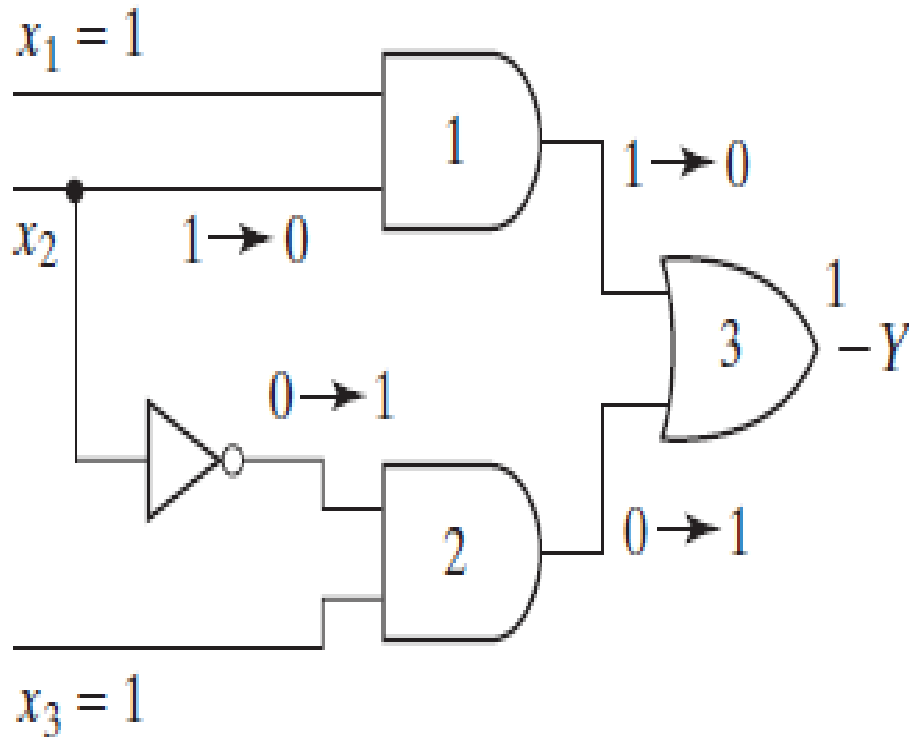
- En los circuitos reales, los cambios en las variables tardan un tiempo en reflejarse en la función.
- Al tiempo que pasa desde que cambia una entrada hasta que cambia la salida del circuito, le llamamos tiempo de retardo  $t_p$ .



# Tiempo de retardo: Riesgos

- Cuando la salida de un circuito es la entrada de otro, el tiempo de retardo se acumula.
- Debido a esto, algunos circuitos pueden producir valores erróneos de la función durante pequeños intervalos de tiempo.
- Estos valores transitorios se llaman riesgos y se producen cuando los trayectos desde las variables hasta la función tienen diferentes tiempos de retardo al recorrer diferente número de puertas lógicas.

# Tiempo de retardo: Riesgos

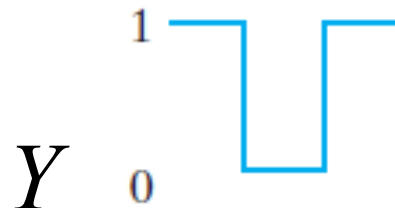


- Cuando  $x_2$  cambia de 1 a 0, las dos entradas de la puerta OR tienen diferente retardo.
- Por ello hay un intervalo en el que ambas son 0  $\Rightarrow Y$  vale 0.

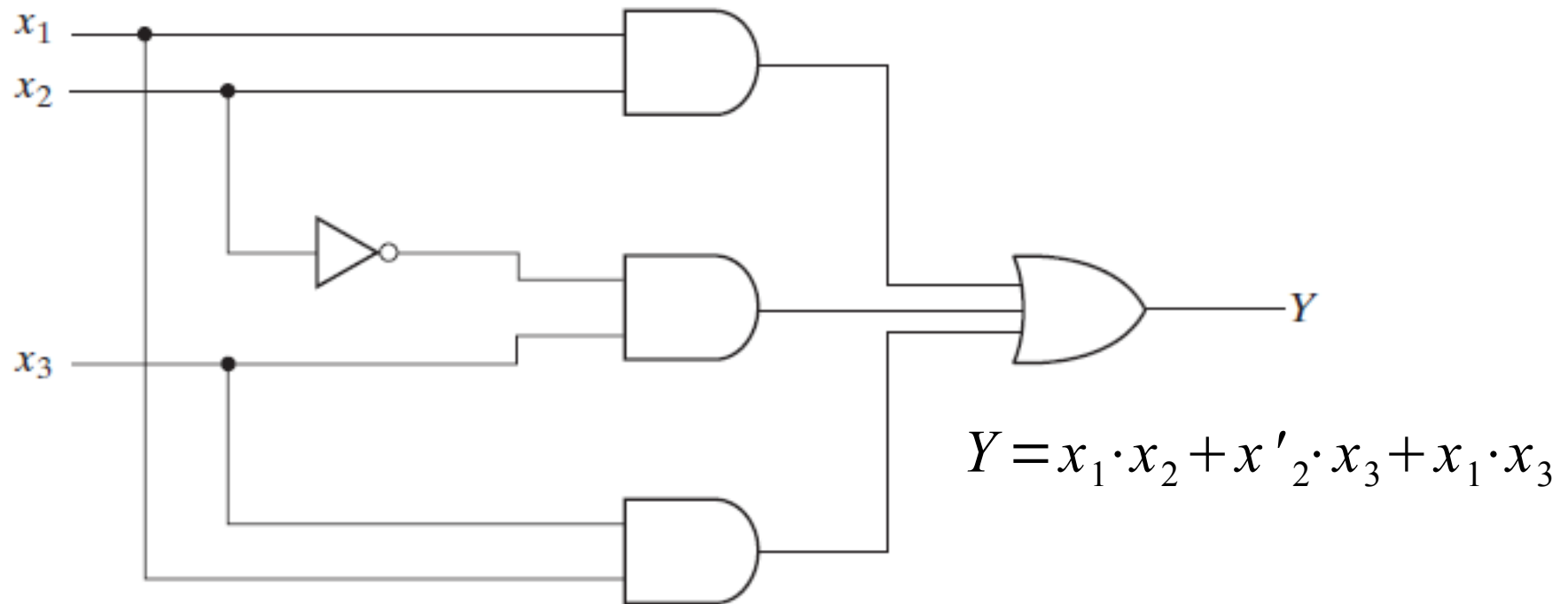
$$Y = x_1 \cdot x_2 + x'_2 \cdot x_3$$

$$x_1 x_2 x_3 = 111 \Rightarrow Y = 1$$

$$x_1 x_2 x_3 = 101 \Rightarrow Y = 1$$



# Tiempo de retardo: Riesgos



- Este tipo de defectos en el diseño sólo se resuelven añadiendo términos a la suma.
- El término nuevo vale 1 mientras se produce el cambio de valor  $\rightarrow Y$  sigue valiendo 1.

# Lenguaje de descripción de hardware: VHDL

- Existe otro método para representar los circuitos digitales: el lenguaje de descripción de hardware.
- Se trata de un texto que describe una o varias funciones lógicas.
- El fichero de texto se usa para la programación o fabricación de un circuito integrado que cumple la función descrita.
- El lenguaje de descripción de hardware más utilizado actualmente es el VHDL.

# Lenguaje de descripción de hardware: VHDL

- El texto VHDL se estructura en dos partes: entidad y arquitectura.
- La entidad sólo especifica cuáles son las variables, su tipo y si son de entrada o salida.
- La arquitectura define las funciones a realizar entre las variables definidas en la entidad.

```
ENTITY example1 IS  
  PORT ( x1,x2,x3      : IN  BIT;  
         f              : OUT BIT );  
END example1 ;
```

```
ARCHITECTURE LogicFunc OF example1 IS  
  BEGIN  
    f <= (x1 AND x2) OR (NOT x2 AND x3);  
  END LogicFunc ;
```

