Algebra de Boole y puertas lógicas

© Luis Entrena, Celia López,
Mario García, Enrique San Millán

Universidad Carlos III de Madrid

Índice

- Postulados y propiedades fundamentales del Álgebra de Boole
- Funciones y expresiones booleanas
- Puertas lógicas. Tecnologías digitales. Implementación de funciones lógicas
- Minimización de funciones lógicas

Álgebra de Boole

- Fundamentos matemáticos de los circuitos digitales
- Denominada Álgebra de Boole en honor de su inventor, George Boole
  - "*An Investigation of the Laws of Thought*" (1854)
- Un álgebra se define por un conjunto de elementos con unas operaciones. En nuestro caso:
  - B = {0, 1}
  - Φ = {+, •}

Postulados del Álgebra de Boole

- Ley de composición interna
  - $\forall\ a, b \in B \Rightarrow a + b \in B,\ a \bullet b \in B$
- Elementos neutros
  - $\forall\ a \in B \Rightarrow \exists$ elementos neutros (0 y 1 respectivamente)

    $a + 0 = a$

    $a \bullet 1 = a$
- Propiedad conmutativa
  - $\forall\ a, b \in B \Rightarrow$ $a + b = b + a$

    $a \bullet b = b \bullet a$
- Propiedad distributiva
  - $\forall\ a, b, c \in B \Rightarrow$ $a + b \bullet c = (a + b) \bullet (a + c)$

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

Postulados del Álgebra de Boole

- Elemento inverso o complementario
  - $\forall\ a \in B \Rightarrow \exists\ \bar{a} \in B$

$$a + \bar{a} = 1$$

$$a \bullet \bar{a} = 0$$

Propiedades fundamentales del Álgebra de Boole

- Dualidad: Toda ley válida tiene una dual, que se obtiene cambiando $0 \leftrightarrow 1$ y $+ \leftrightarrow \bullet$
- Idempotencia
  - $\forall\ a \in B \Rightarrow$ $a + a = a$
    $a \bullet a = a$
  - Demostración:

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

- $\forall\ a \in B \Rightarrow$ $a + 1 = 1$
  $a \bullet 0 = 0$

Propiedades fundamentales del Álgebra de Boole

- De las propiedades anteriores se pueden definir las operaciones básicas

a	b	a+b
0	0	0
0	1	1
1	0	1
1	1	1

a	b	a•b
0	0	0
0	1	0
1	0	0
1	1	1

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

- Tabla de verdad: proporciona el valor de una función para todas las posibles combinaciones de valores de las entradas

Propiedades fundamentales del Álgebra de Boole

- Involución
  - $\forall a \in B \Rightarrow \quad \overline{\overline{a}} = a$
- Absorción
  - $\forall a, b \in B \Rightarrow$ $a + ab = a$
    $a\,(a+b) = a$
  - Demostración:

$$a + ab = a \bullet 1 + ab = a(1 + b) = a \bullet 1 = a$$

- Propiedad asociativa
  - $\forall a, b, c \in B \Rightarrow$ $(a + b) + c = a + (b + c)$
    $(a \bullet b) \bullet c = a \bullet (b \bullet c)$

Propiedades fundamentales del Álgebra de Boole

- Leyes de De Morgan:
  - $\forall\ a, b \in B \Rightarrow$ $\overline{a+b} = \bar{a}\,\bar{b}$

    $\overline{a \bullet b} = \bar{a} + \bar{b}$
  - Demostración:

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

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

    luego (a+b) es el inverso de $\bar{a}\,\bar{b}$

Funciones y expresiones booleanas

- Definiciones:
  - Una variable lógica o booleana es cualquier elemento $x \in B = \{0, 1\}$
  - Un literal es una variable negada o sin negar
  - Función lógica o booleana:

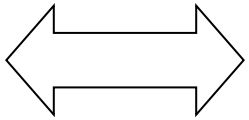
$$f: B^n \rightarrow B$$
$$(x_1, x_2, \ldots, x_n) \rightarrow y$$

Representación de funciones lógicas

- Expresión

$f(a, b) = a + b$

- Tabla de verdad

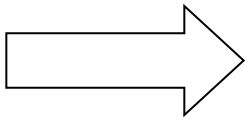


a	b	f(a,b)
0	0	0
0	1	1
1	0	1
1	1	1

Obtención de la tabla de verdad a partir de una expresión

- Basta evaluar la expresión para cada una de las combinaciones de valores de las entradas

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

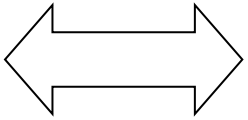


a	b	c	f
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Función mintérmino

- Expresión: un producto en el que aparecen todas las variables, negadas o no
- Tabla de verdad: tiene un 1 en una posición y 0 en todas las demás
- Ejemplo:

$$f(a,b,c) = \bar{a}\,b\,\bar{c} = m_2$$



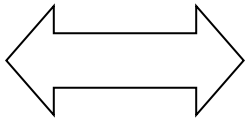
a	b	c	f
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

- Regla para obtener la expresión:
  - 0 → variable negada
  - 1 → variable sin negar

Función maxtérmino

- Expresión: una suma en la que aparecen todas las variables, negadas o no
- Tabla de verdad: tiene un 0 en una posición y 1 en todas las demás
- Ejemplo:

$$f(a,b,c) = (a + \bar{b} + c) = M_2$$



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

- Regla para obtener la expresión:
  - 0 → variable sin negar
  - 1 → variable negada

CUIDADO: al contrario que los mintérminos!

Teorema de Expansión de Shannon

- Toda función booleana se puede descomponer de las siguientes formas

$$f(x_1, x_2, \ldots, x_n) = \bar{x}_i \, f(x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n) + x_i \, f(x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n)$$

$$f(x_1, x_2, \ldots, x_n) = [\bar{x}_i + f(x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n)][x_i + f(x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n)]$$

- Demostración

$$x_i = 0 \Rightarrow f(x_1, x_2, \ldots, x_n) = 1 \bullet f(x_1, \ldots, 0, \ldots, x_n) + 0 \bullet f(x_1, \ldots, 1, \ldots, x_n) = \\ = f(x_1, \ldots, 0, \ldots, x_n)$$

$$x_i = 1 \Rightarrow f(x_1, x_2, \ldots, x_n) = 0 \bullet f(x_1, \ldots, 0, \ldots, x_n) + 1 \bullet f(x_1, \ldots, 1, \ldots, x_n) = \\ = f(x_1, \ldots, 1, \ldots, x_n)$$

  - La otra forma se demuestra por dualidad

Corolario del Teorema de Expansión de Shannon

- Aplicando recursivamente el Teorema:

$$
\begin{aligned}
f(a,b,c) &= \bar{a}f(0,b,c) + af(1,b,c) = \\
&= \bar{a}(\bar{b}f(0,0,c) + bf(0,1,c)) + a(\bar{b}f(1,0,c) + bf(0,1,c)) = \\
&= \bar{a}\bar{b}f(0,0,c) + \bar{a}bf(0,1,c)) + a\bar{b}f(1,0,c) + abf(0,1,c) = \\
&= \bar{a}\bar{b}\bar{c}f(0,0,0) + \bar{a}\bar{b}cf(0,0,1) + \bar{a}b\bar{c}f(0,1,0) + \bar{a}bcf(0,1,1) + \\
&+ a\bar{b}\bar{c}f(1,0,0) + a\bar{b}cf(1,0,1) + ab\bar{c}f(1,1,0) + abcf(1,1,1) = \\
&= \sum_{3} m_i k_i
\end{aligned}
$$

- Una función es igual a la suma de todos los mintérminos (m_i) afectados por un coeficiente (k_i) igual al valor que toma la función al sustituir cada variable por un 0 o un 1 según que en el mintérmino aparezca la variable negada o sin negar, respectivamente

Primera forma canónica

- Una función se puede expresar como la suma de los mintérminos para los que la función vale 1

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

$$f(a,b,c) = \sum_{3}(0,2,5) = \sum_{3} m(0,2,5) = \\ = \bar{a}\bar{b}\bar{c} + \bar{a}b\bar{c} + a\bar{b}c$$

Segunda forma canónica

- Una función se puede expresar como el producto de los maxtérminos para los que la función vale 0

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

$$f(a,b,c) = \prod_{3}(1,3,4,6,7) = \prod_{3} M(1,3,4,6,7) =$$
$$= (a+b+\bar{c})(a+\bar{b}+\bar{c})(\bar{a}+b+c)$$
$$(\bar{a}+\bar{b}+c)(\bar{a}+\bar{b}+\bar{c})$$

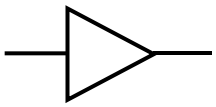
CUIDADO: al contrario que los mintérminos!

Puertas lógicas

- Las puertas lógicas son circuitos electrónicos que realizan las funciones básicas del Álgebra de Boole
- Para cada puerta utilizaremos un símbolo

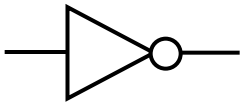
- Identidad
  $z = a$

a	a
0	0
1	1



- Puerta NOT o inversor
  $z = \overline{a}$

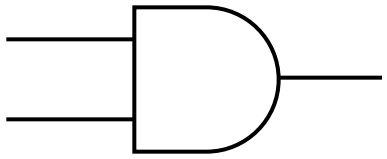
a	$\overline{a}$
0	1
1	0



Puertas AND y OR

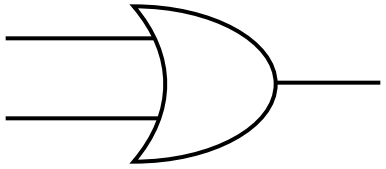
- Puerta AND
  $z = a \bullet b$

a	b	a•b
0	0	0
0	1	0
1	0	0
1	1	1



- Puerta OR
  $z = a + b$

a	b	a+b
0	0	0
0	1	1
1	0	1
1	1	1

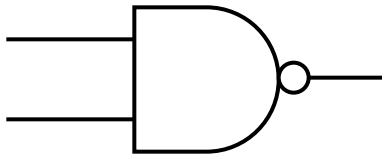


Puertas NAND y NOR

- Puerta NAND

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

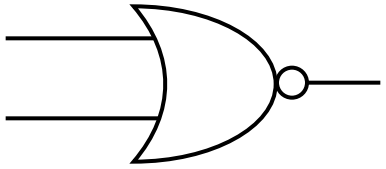
a	b	$\overline{a \bullet b}$
0	0	1
0	1	1
1	0	1
1	1	0



- Puerta NOR

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

a	b	$\overline{a + b}$
0	0	1
0	1	0
1	0	0
1	1	0

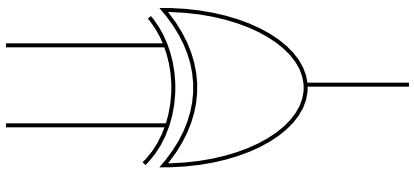


Puertas XOR y XNOR

- Puerta XOR (OR-Exclusiva)

$$z = a \oplus b = \bar{a}b + a\bar{b} = (\bar{a} + \bar{b})(a + b)$$

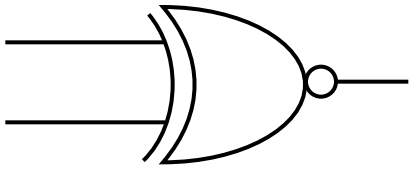
a	b	a⊕b
0	0	0
0	1	1
1	0	1
1	1	0



- Puerta XNOR (NOR-Exclusiva)

$$z = \overline{a \oplus b} = ab + \bar{a}\bar{b} = (\bar{a} + b)(a + \bar{b})$$

a	b	$\overline{a \oplus b}$
0	0	1
0	1	0
1	0	0
1	1	1

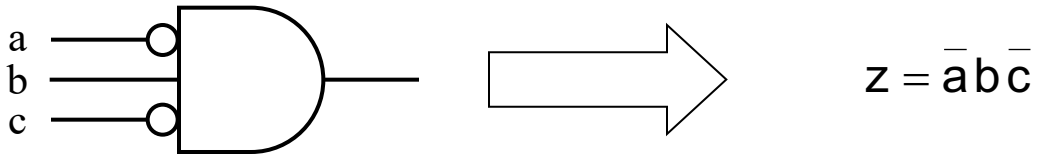


Generalización a n entradas

	Valor de la salida	
Puerta	0	1
AND	Alguna entrada = 0	Todas las entradas = 1
OR	Todas las entradas = 0	Alguna entrada = 1
NAND	Todas las entradas = 1	Alguna entrada = 0
NOR	Alguna entrada = 1	Todas las entradas = 0
XOR	Hay un nº par de entradas = 1	Hay un nº impar de entradas = 1
XNOR	Hay un nº impar de entradas = 1	Hay un nº par de entradas = 1

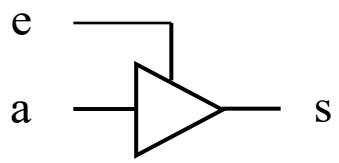
Otros símbolos

- Un círculo en una entrada o una salida indica negación



Buffer triestado

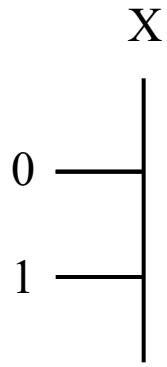
- Un tipo especial de puerta lógica que puede poner su salida en alta impedancia



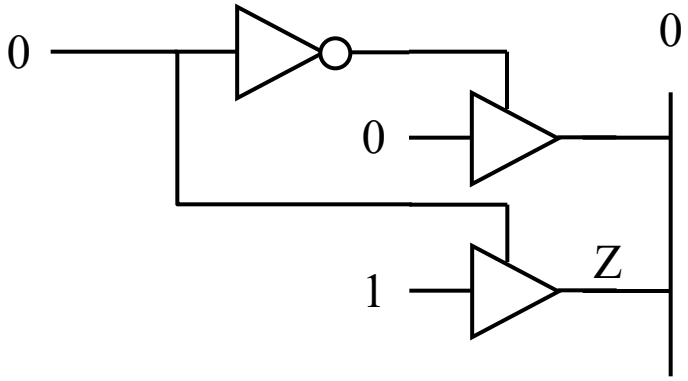
e	a	s
0	0	Z
0	1	Z
1	0	0
1	1	1

Buffer triestado

- Los buffers triestado son útiles para permitir múltiples conexiones a un mismo punto evitando cortocircuitos



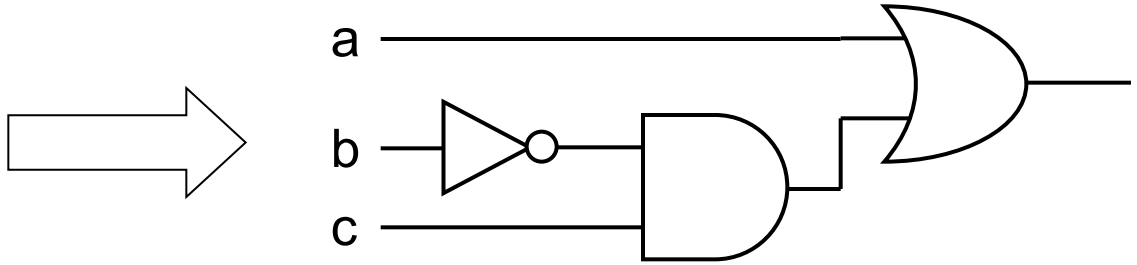
Cortocircuito!



Realización de una función lógica con puertas lógicas

- A partir de la expresión de la función, sustituimos las operaciones lógicas por puertas lógicas
- Ejemplo:

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



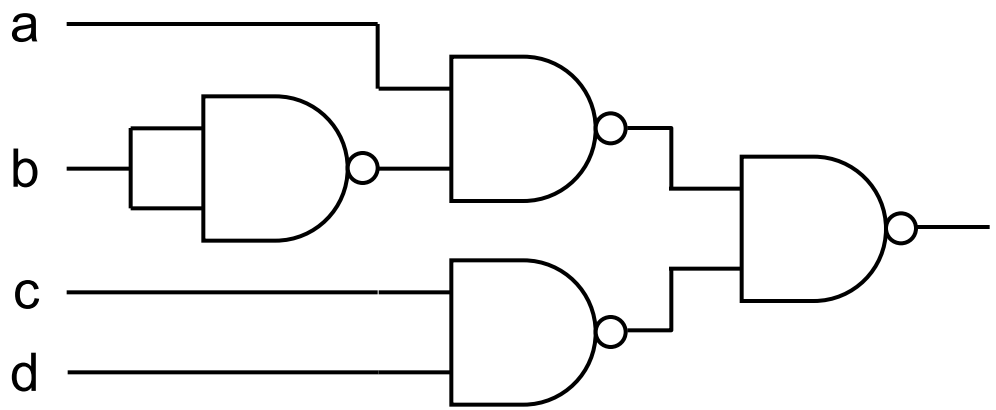
Conjuntos completos

- Un conjunto de funciones es funcionalmente completo si cualquier función lógica puede realizarse con las funciones del conjunto solamente
  - {AND} ***no*** es un conjunto completo
  - {AND, NOT} es un conjunto completo
  - {OR, NOT} es un conjunto completo
  - {NAND} es un conjunto completo
  - {NOR} es un conjunto completo
- Los conjuntos {NAND} y {NOR} tienen la ventaja de que permiten realizar cualquier función lógica con un sólo tipo de puerta lógica

Realización de circuitos con puertas NAND

- Aplicación directa de las leyes de De Morgan
- Ejemplo: $f(a,b,c) = a\bar{b} + cd =$

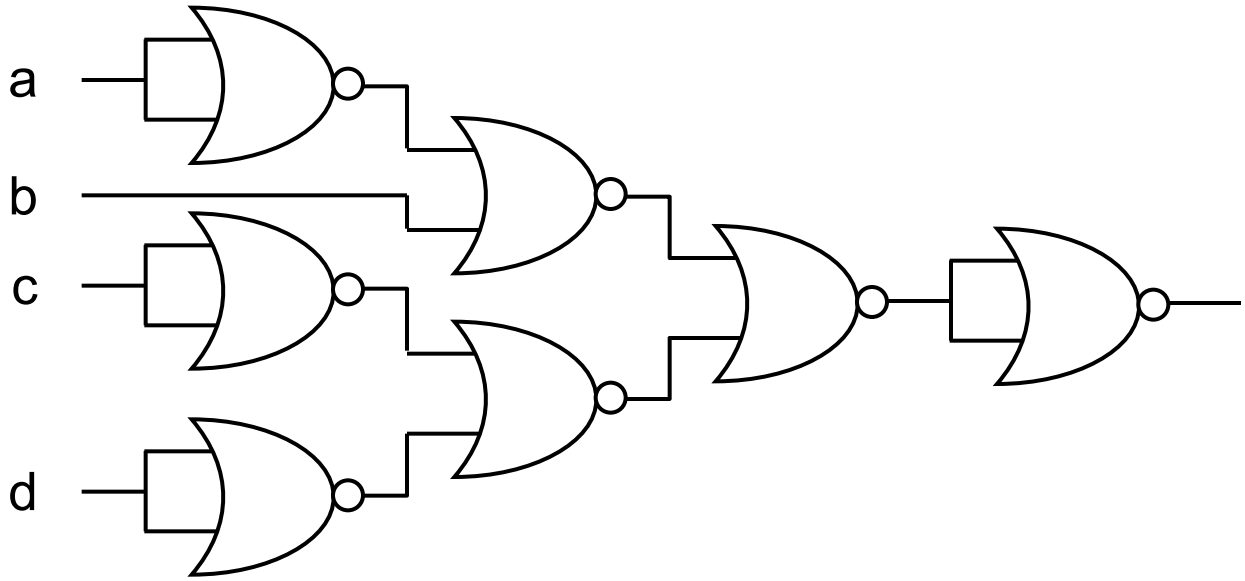
$$= \overline{\overline{a\bar{b} + cd}} = \overline{\overline{a\bar{b}} \bullet \overline{cd}}$$



Realización de circuitos con puertas NOR

- Aplicación directa de las leyes de De Morgan
- Ejemplo: $f(a,b,c) = a\bar{b} + cd =$

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



Minimización de funciones lógicas

- Una función lógica tiene múltiples expresiones equivalentes
  - La forma más sencilla dará lugar a una implementación mejor
- Criterios de optimización:
  - En tamaño o área:
    - Menor número de puertas lógicas
    - Puertas lógicas con el menor número de entradas
  - En velocidad o retardo:
    - Menor número de puertas lógicas desde una entrada hasta la salida
- Nos centraremos en la optimización en área

Minimización de funciones lógicas

- Métodos de optimización
  - Manual: aplicación directa de las leyes del Álgebra de Boole
    - Muy difícil, no sistemático
  - En dos niveles: el objetivo es obtener una expresión óptima en forma de suma de productos o productos de sumas
    - Existen soluciones sistemáticas y óptimas
    - Aplicable manualmente (para pocas variables) o con ayuda de un computador
  - Multinivel
    - Mejor solución, aunque mucho más difícil
    - Sólo posible con ayuda de un computador

Métodos de los mapas de Karnaugh

- Método de optimización en dos niveles
- Se puede realizar manualmente hasta 6 variables
- Se basa en la Propiedad de adyacencia
  - $\forall E, x \in B \Rightarrow Ex + E\bar{x} = E(x + \bar{x}) = E$

    $(E + x)(E + \bar{x}) = E + (x \bullet \bar{x}) = E$ (dual)

  - Dos términos son adyacentes si son idénticos excepto por un literal, que aparece negado en un término y no negado en el otro
  - Los dos términos se simplifican en uno sólo con eliminación del literal que los diferencia

Aplicación de la propiedad de adyacencia

- Ejemplo:

$$f(a,b,c) = \sum_{3}(0,1,2,3,7) = \bar{a}\bar{b}\bar{c} + \bar{a}\bar{b}c + \bar{a}b\bar{c} + \bar{a}bc + abc =$$

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

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

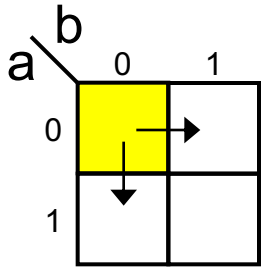
- La observación de las adyacencias puede ser difícil en la práctica

Mapas de Karnaugh

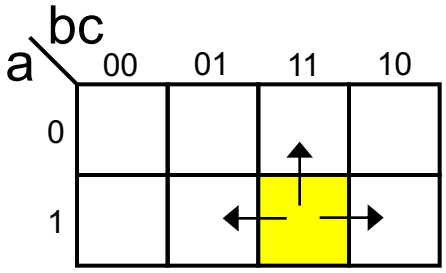
- Mapa que presenta la tabla de verdad de una función de manera que los términos adyacentes son contiguos:
  - Una casilla para cada combinación o término
  - Las casillas se numeran en código Gray
  - En un mapa de n variables, cada casilla tiene n casillas adyacentes que se corresponden con las combinaciones que resultan de invertir el valor de cada una de las n variables

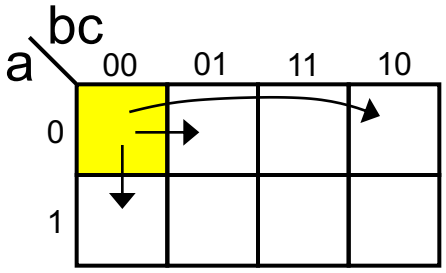
Mapas de Karnaugh: adyacencias

- Dos variables



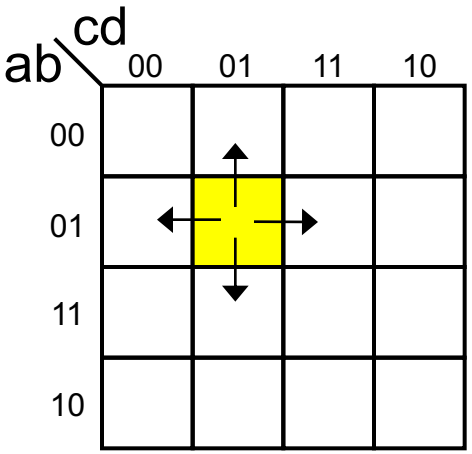
- Tres variables

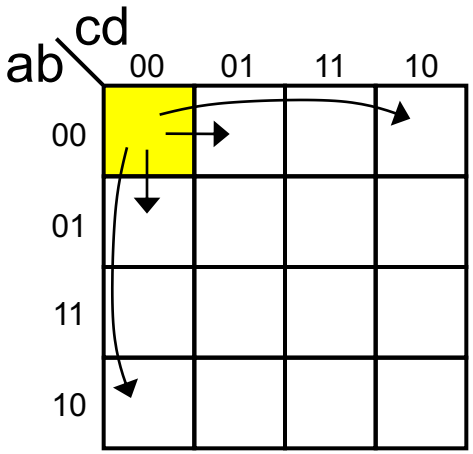




Mapas de Karnaugh: adyacencias

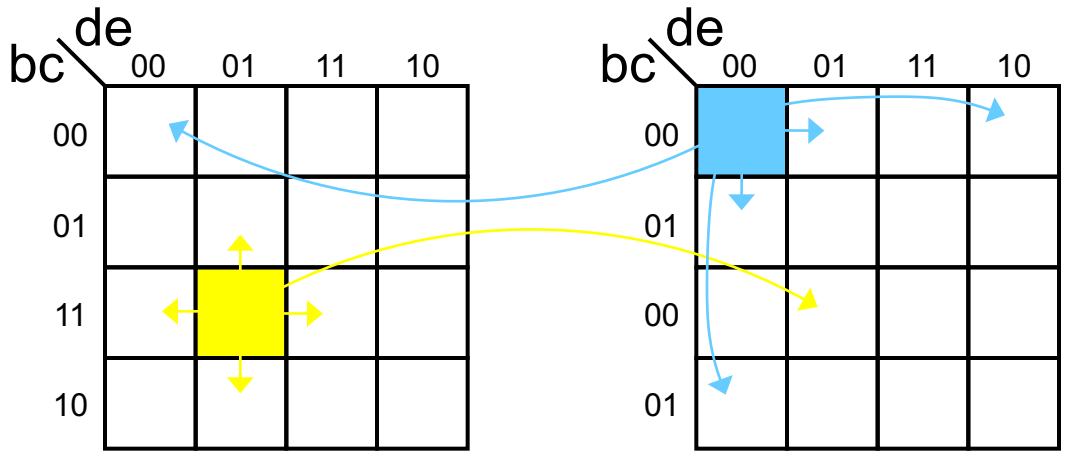
- Cuatro variables





Mapas de Karnaugh: adyacencias

- Cinco variables



Mapas de Karnaugh: numeración de las casillas

- Dos variables

a \ b	0	1
0	0	1
1	2	3

- Tres variables

a \ bc	00	01	11	10
0	0	1	3	2
1	4	5	7	6

- Cuatro variables

ab \ cd	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	12	13	15	14
10	8	9	11	10

Mapas de Karnaugh: numeración de las casillas

- Cinco variables

bc \ de	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	12	13	15	14
10	8	9	11	10

a = 0

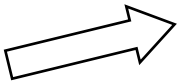
bc \ de	00	01	11	10
00	16	17	19	18
01	20	21	23	22
00	28	29	31	30
01	24	25	27	26

a = 1

Representación de una función en el Mapa de Karnaugh

- Se marcan las casillas que corresponden a los mintérminos o los maxtérminos de la función
- Ejemplo:

$$f(a,b,c) = \sum_3 (0,1,2,3,7) =$$
$$= \prod_3 (4,5,6)$$

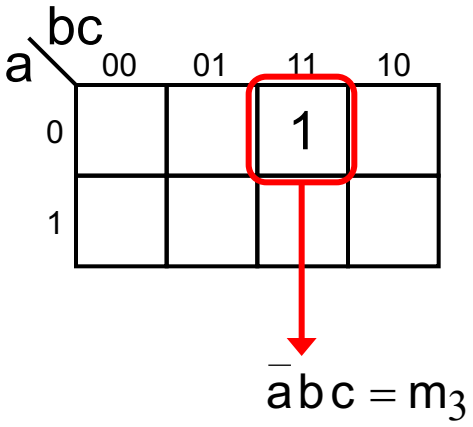


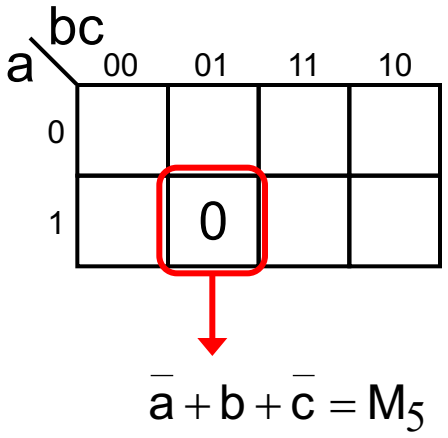
a \ bc	00	01	11	10
0	1	1	1	1
1			1	

a \ bc	00	01	11	10
0				
1	0	0		0

Obtención de una expresión a partir del Mapa de Karnaugh

- Se siguen las reglas para mintérminos y maxtérminos
  - Regla para mintérminos
    - 0 → variable negada
    - 1 → variable sin negar
  - Regla para maxtérminos
    - 0 → variable sin negar
    - 1 → variable negada

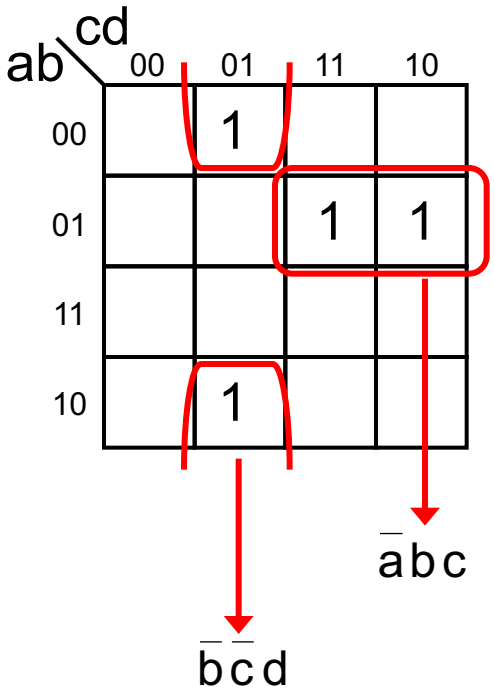


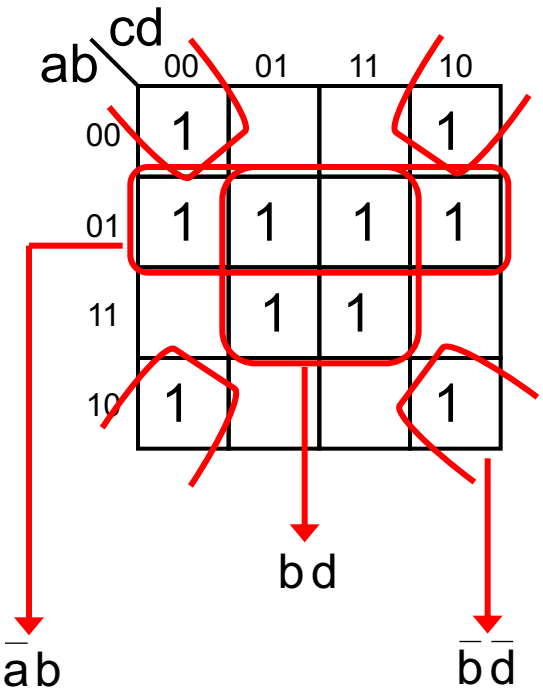


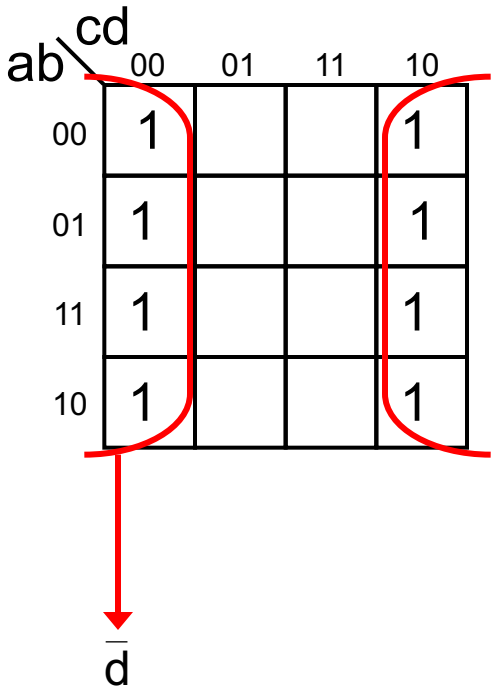
Simplificación mediante Mapas de Karnaugh

- Dos opciones
  - Por mintérminos (unos): se obtiene una suma de productos
  - Por maxtérminos (ceros): se obtiene un producto de sumas
- Buscar grupos de casillas adyacentes
  - Un grupo de 2 casillas adyacentes elimina 1 variable
  - Un grupo de 4 casillas adyacentes elimina 2 variables
  - Un grupo de 8 casillas adyacentes elimina 3 variables
  - Un grupo de 16 casillas adyacentes elimina 4 variables
  - ….
- Objetivo: cubrir todos los mintérminos (maxtérminos) con los grupos más grandes posibles y con el menor número de grupos
  - Se pueden repetir términos, si es necesario (propiedad de absorción)

Simplificación: formación de grupos



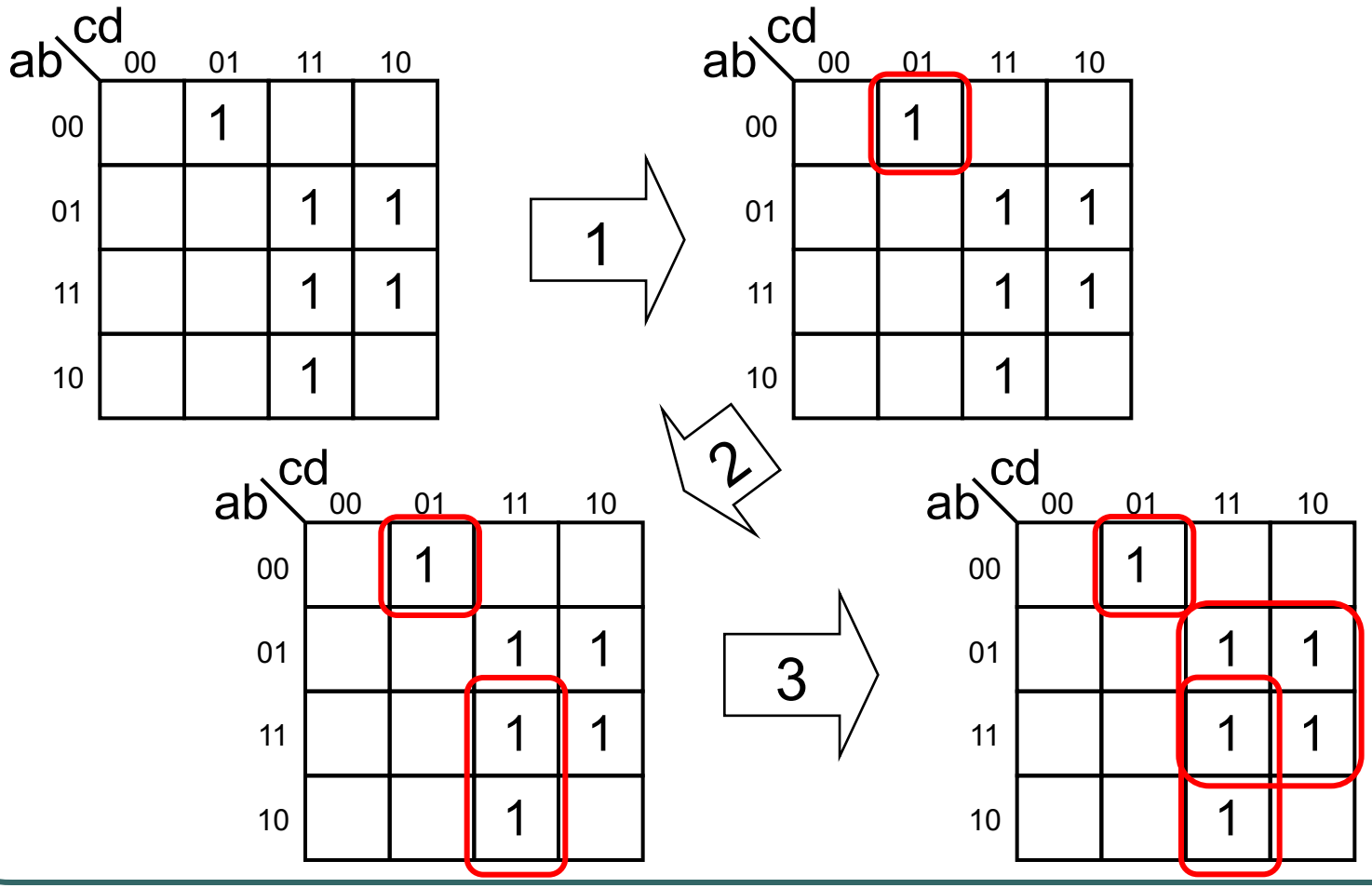




Simplificación mediante Mapas de Karnaugh: Algoritmo

- Algoritmo sistemático
  1. Cubrir las casillas que no pueden formar grupos de 2
  2. Cubrir las casillas que pueden formar grupos de 2, pero no de 4
  3. Cubrir las casillas que pueden formar grupos de 4, pero no de 8
  4. Cubrir las casillas que pueden formar grupos de 8, pero no de 16
  5. …
- Si en algún paso hay más de una opción:
  - Comenzar siempre cubriendo las casillas que tienen menos opciones

Simplificación mediante Mapas de Karnaugh: Ejemplo



Funciones incompletas

- Una función incompletamente especificada (o simplemente incompleta) es aquella que no está especificada para alguna combinación de valores de sus entradas
- Las funciones incompletas se dan en la práctica:
  - Cuando las entradas provienen de otro circuito que no puede producir determinadas combinaciones por construcción
  - Cuando existen casos en que el valor de la función no tiene sentido o es indiferente
- Notación:
  - Un valor indiferente se representa con 'X' ó '-'
  - El conjunto de términos indiferentes ("don't cares") se denota con la letra Δ

Funciones incompletas

- Ejemplo: Función que determina si un número BCD es impar
  - Los números del 10 al 15 no tienen sentido en BCD

$$f(b3,b2,b1,b0) = \sum_4(1,3,5,7,9) + \Delta_4(10,11,12,13,14,15) =$$

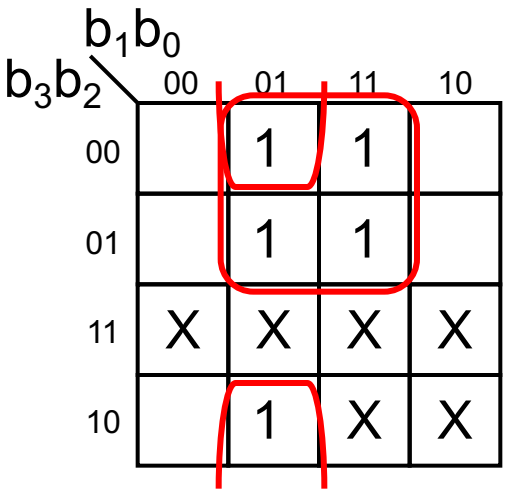
$$= \prod_4(0,2,4,6,8) + \Delta_4(10,11,12,13,14,15)$$

Combinaciones indiferentes

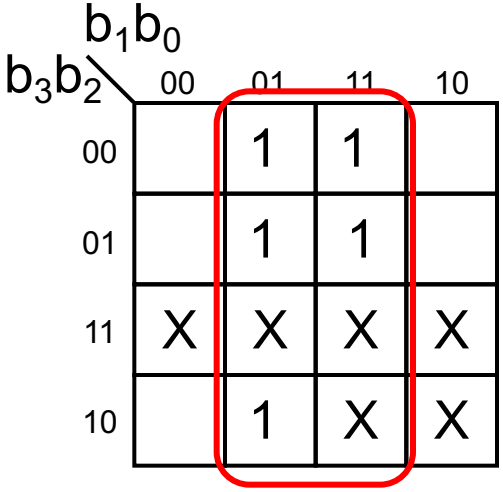
b3	b2	b1	b0	f
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	X
1	0	1	1	X
1	1	0	0	X
1	1	0	1	X
1	1	1	0	X
1	1	1	1	X

Minimización de funciones incompletas

- Los términos indiferentes son “comodines”: se pueden cubrir o no, según convenga para formar grupos más grandes



$$f(b_3,b_2,b_1,b_0) = \overline{b_3}\,b_0 + \overline{b_2}\,\overline{b_1}\,b_0$$



☑ *Correcto*

$$f(b_3,b_2,b_1,b_0) = b_0$$

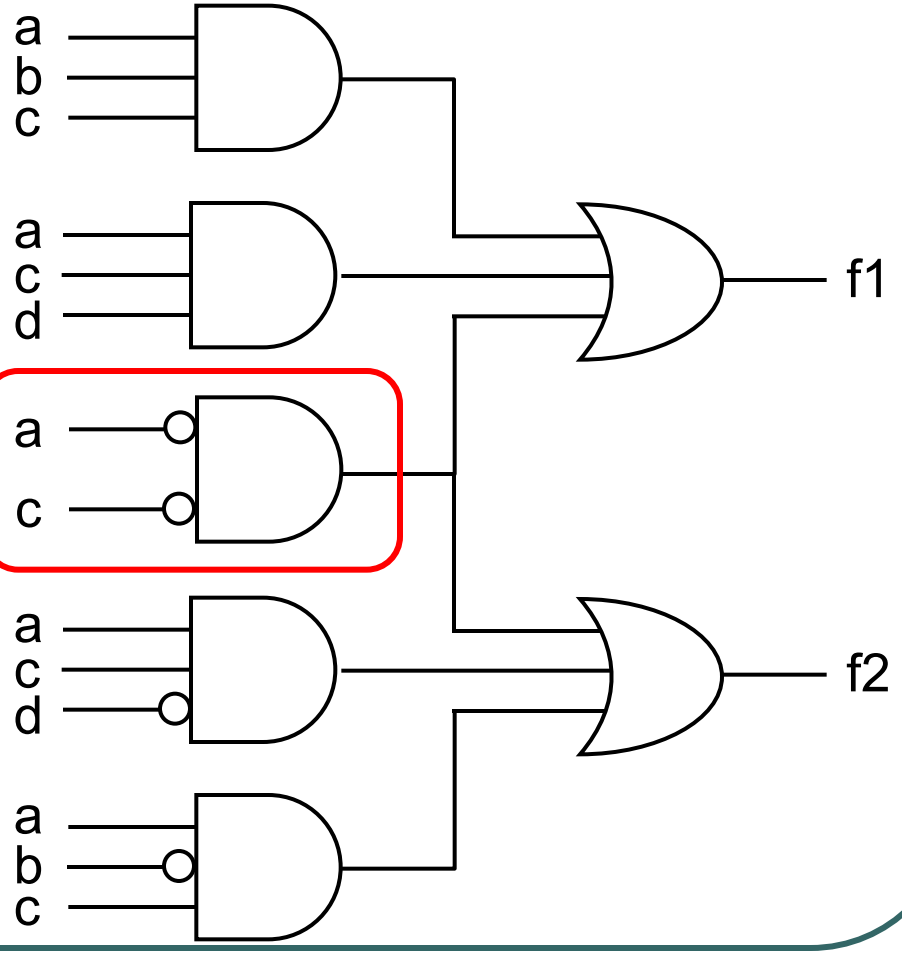
Funciones múltiples

- En los circuitos digitales se implementan generalmente funciones múltiples: varias funciones a la vez o una función de múltiples salidas
- Las funciones múltiples se pueden implementar de forma óptima al considerarlas conjuntamente
  - Se pueden compartir términos o partes comunes para ahorrar lógica
- La descomposición de funciones múltiples de manera que se maximicen los términos comunes es difícil
  - Los algoritmos son difíciles de aplicar manualmente
  - Generalmente lo haremos por inspección

Funciones múltiples: Ejemplo

$$f1(a,b,c,d) = \bar{a}\bar{c} + abc + acd$$

$$f2(a,b,c,d) = \bar{a}\bar{c} + a\bar{b}c + ac\bar{d}$$



☑ *Términos comunes*

Funciones múltiples: Ejemplo

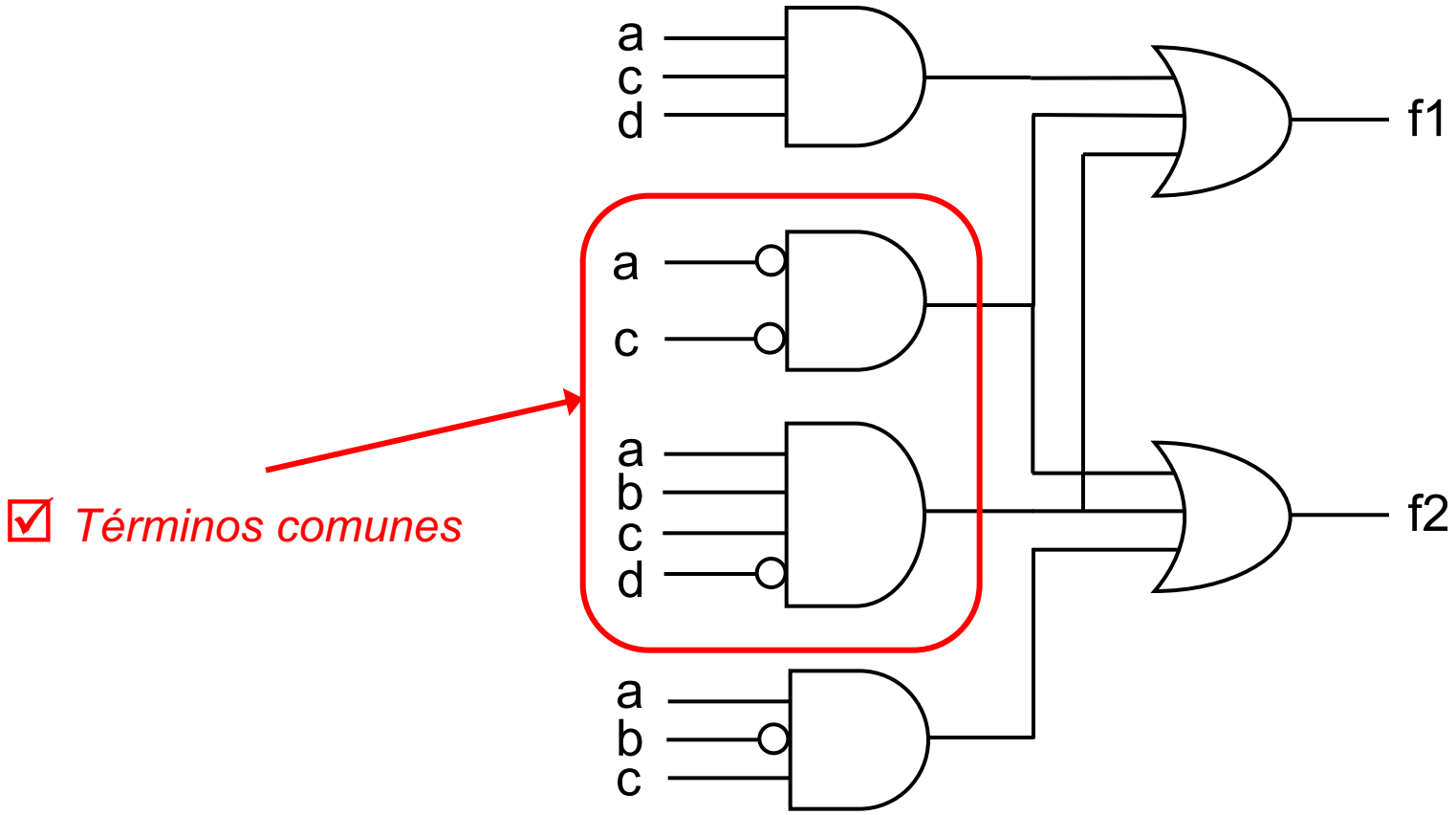
- Es posible encontrar más términos comunes

$$f1(a,b,c,d) = \bar{a}\bar{c} + abc + acd = \bar{a}\bar{c} + abc\bar{d} + acd$$

$$f2(a,b,c,d) = \bar{a}\bar{c} + a\bar{b}c + ac\bar{d} = \bar{a}\bar{c} + abc\bar{d} + a\bar{b}c$$

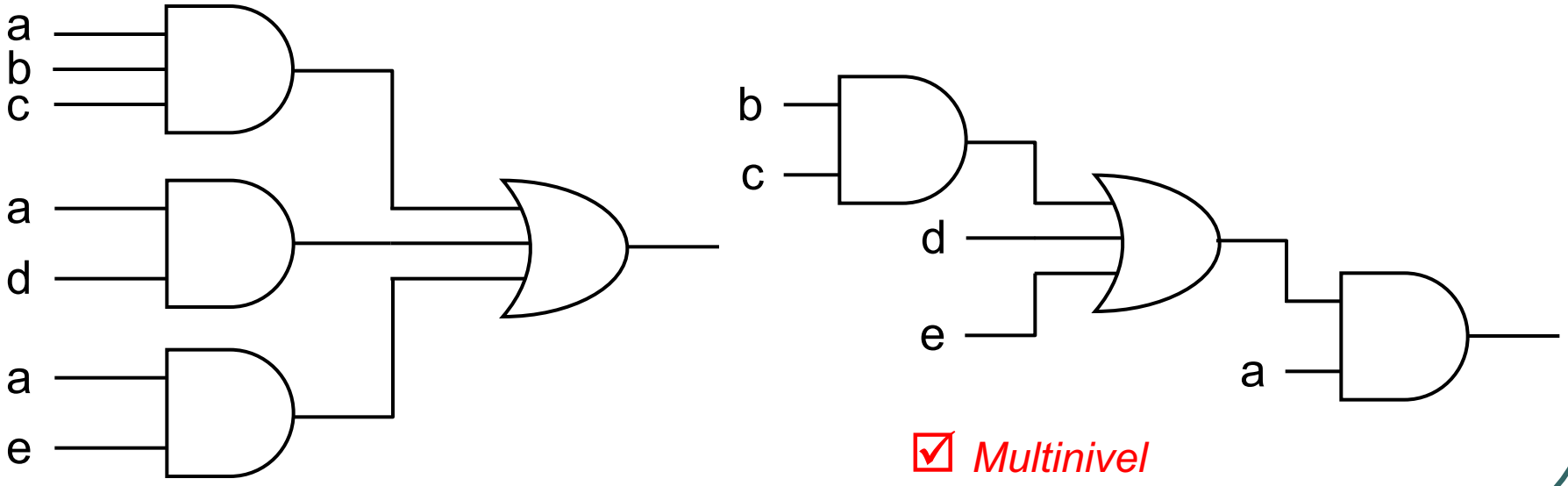
  - Las expresiones de las funciones no son óptimas por separado, pero sí son óptimas en conjunto!
  - Las herramientas de diseño incluyen algoritmos para minimizar funciones múltiples

Funciones múltiples: Ejemplo



Síntesis multinivel

- Si eliminamos la restricción a dos niveles, se pueden encontrar mejores soluciones
  - Se utilizan algoritmos heurísticos, con ayuda de un ordenador
- Ejemplo: $f(a,b,c,d,e) = abc + ad + ae = a(bc + d + e)$



Herramientas de optimización

- Métodos manuales:
  - Sólo en 2 niveles, pocas variables
- Herramientas software
  - Multinivel, múltiples funciones, muchas variables
  - Optimización en área o en retardo
  - Generalmente incorporadas en herramientas de síntesis lógica
- Herramientas de síntesis lógica
  - Funcionan como un compilador, a partir de la descripción del diseño en forma esquemática o mediante un Lenguaje de Descripción de Hardware
  - Optimizan el diseño y generan las puertas lógicas en una tecnología determinada

Referencias

- “Introducción al diseño lógico digital”. J. P. Hayes. Ed. Addison-Wesley
- “Circuitos y sistemas digitales”. J. E. García Sánchez, D. G. Tomás, M. Martínez Iniesta. Ed. Tebar-Flores