UNIVERSIDAD POLITECNICA DE VALENCIA

etsinf

DISCA

Fundamentos de computadores

TEMA 2. PRINCIPIOS DEL DISEÑO DIGITAL

Objetivos

- Conocer las funciones lógicas y su representación
- Diseñar circuitos lógicos sencillos
- Fundamentos del Álgebra de Boole
- Métodos de simplificación. Mapas de Karnaugh

Bibliografía

Introducción a los Computadores.

J. Sahuquillo y otros.

Ed. SP-UPV, 1997 (ref. 97.491)

Recursos de aprendizaje

FCO

- Poliformat, sección “Recursos”
  - Ejercicios sin solución.
  - Soluciones a los ejercicios.
  - **Entrenador de Karnaugh.**
  - Exámenes de años anteriores.

- Poliformat, sección “Lessons”
  - Módulo 2: Principios de diseño digital.
    - » *Tablas de verdad.*
    - » *Puertas lógicas.*

DISCA

Índice

- Introducción
- Funciones lógicas y tablas de verdad
- Puertas lógicas
- Análisis de circuitos
- Álgebra de Boole
- Formas canónicas de representar una función lógica
- Simplificación de funciones lógicas
  - Mapas de Karnaugh

Introducción

- Transistor
  - Unidad física mínima de diseño digital
- Puerta lógica
  - Unidad lógica mínima de diseño digital
- Circuito combinacional
  - Las salidas sólo dependen del valor de las entradas en el momento actual
    - Ejemplo. Selección de la bebida en una máquina de café

Introducción

- Circuito secuencial
  - Las salidas dependen del valor actual de las entradas y de la secuencia de valores anteriores (historia) del circuito.
    - Ejemplo. Monedero de una máquina de café
- Unidad funcional
  - Suma de pequeños circuitos que realizan una función definida

Índice

- Introducción
- Funciones lógicas y tablas de verdad
- Puertas lógicas
- Análisis de circuitos
- Álgebra de Boole
- Formas canónicas de representar una función lógica
- Simplificación de funciones lógicas
  - Mapas de Karnaugh

Funciones lógicas y tablas de verdad FCO

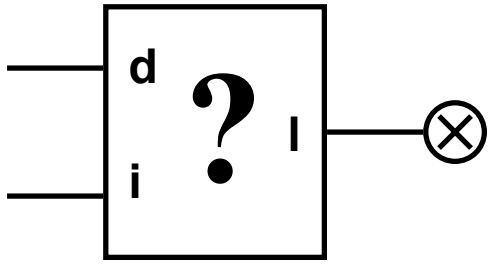
- Función lógica
  - Expresión formal del comportamiento de un circuito lógico
  - Permite determinar la salida del circuito en función de las entradas
  - Aridad = número de variables lógicas de entrada
  - Valoración = una de las combinaciones de valores de las entradas

Funciones lógicas y tablas de verdad (ii) FCO

- Tabla de verdad
  - Forma tabular de expresar una función lógica
  - Para cada entrada o salida se asigna una columna
  - Para cada valoración se asigna una fila
  - Entradas a la izquierda, salidas a la derecha
  - Valoraciones siguiendo la numeración binaria

Tablas de verdad. Ejemplos

- Luz interior de un coche
  - A partir de dos entradas ***d***, ***i*** (puertas derecha e izquierda), diseñad un circuito que encienda una luz ***l*** cuando alguna de las puertas esté abierta



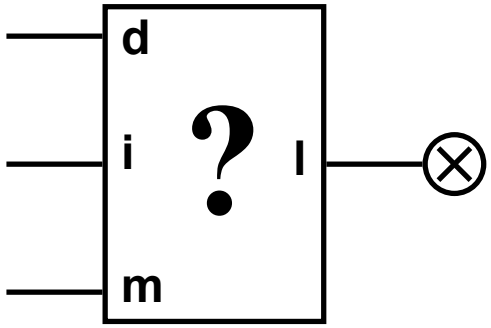
Símbolo Lógico

d	i	l
0	0	0
0	1	1
1	0	1
1	1	1

Tabla de verdad

Tablas de verdad. Ejemplos (ii)

- Luz interior de un coche (ii)
  - Añadir una entrada ***m*** de encendido manual: Si la entrada ***m*** está activada (***m=1***) encender la luz independientemente del estado (abierto/cerrado) de las puertas



m	d	i	l
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

m	d	i	l
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	X	X	1

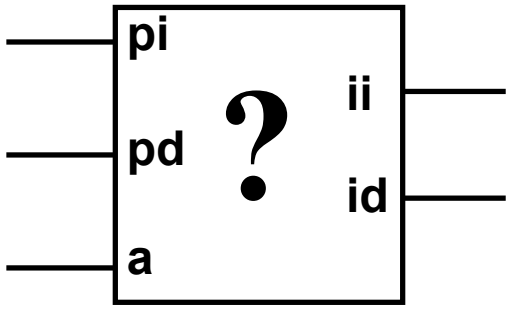
Tabla de verdad reducida

Tablas de verdad. Ejemplos (iii)

- Funciones con entradas indiferentes
  - Aquellas combinaciones de valores de entrada para las que no importa el valor de la salida, por
    - tratarse de una combinación de las entradas para la que no se ha especificado el comportamiento del circuito
    - o tratarse de una combinación de las entradas que es imposible
  - En la tabla de verdad, la salida para estas valoraciones es X

Tablas de verdad. Ejemplos (iv)

- Intermitentes de un coche
  - A partir de 3 entradas: palanca a la izquierda (***pi***), palanca a la derecha (***pd***) y avería (***a***), generar las salidas que activen los intermitentes izquierdo (***ii***) y derecho (***id***)

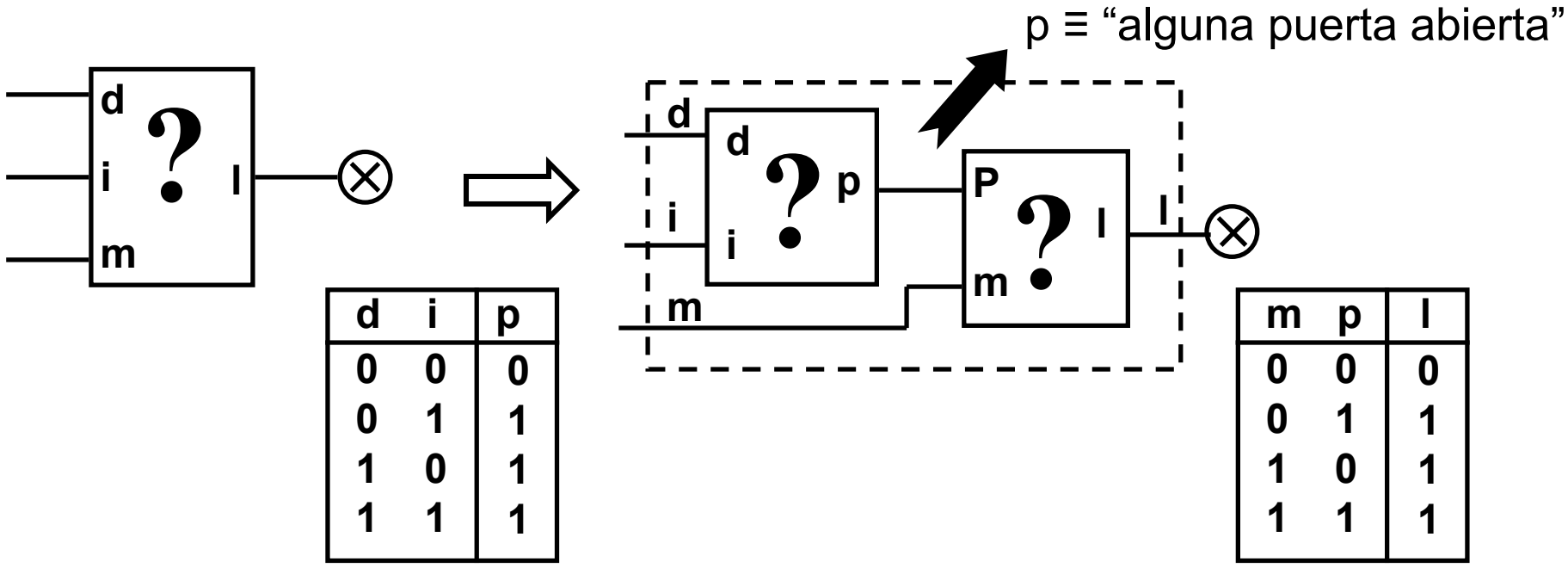


a	pi	pd	ii	id
0	0	0	0	0
0	0	1	0	1
0	1	0	1	0
0	1	1	X	X
1	0	0	1	1
1	0	1	1	1
1	1	0	1	1
1	1	1	X	X

Tabla de verdad
de una función
con entradas indiferentes

Composición de funciones FCO

- Función compuesta. Aquella en la que la salida de una (sub)función es utilizada como entrada de otra
- Ejemplo: Luz interior de coche con encendido manual



d	i	p
0	0	0
0	1	1
1	0	1
1	1	1

m	p	l
0	0	0
0	1	1
1	0	1
1	1	1

Índice

- Introducción
- Funciones lógicas y tablas de verdad
- Puertas lógicas
- Análisis de circuitos
- Álgebra de Boole
- Formas canónicas de representar una función lógica
- Simplificación de funciones lógicas
  - Mapas de Karnaugh

Puertas lógicas

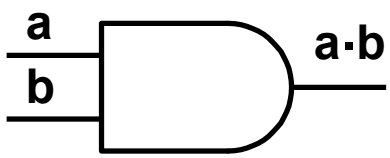
- Puerta lógica. Circuito electrónico que implementa una función lógica elemental

- Tipos
  - Básicos: AND, OR, NOT
  - Otras: XOR
  - Con salida negada: NAND, NOR, XNOR

- Tecnologías. Base física de construcción
  - TTL, CMOS

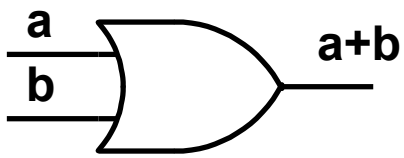
Puertas lógicas (ii)

- AND
  - Producto lógico (“y”)
  - Ampliable



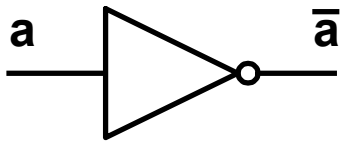
b	a	a·b
0	0	0
0	1	0
1	0	0
1	1	1

- OR
  - Suma lógica (“o”)
  - Ampliable



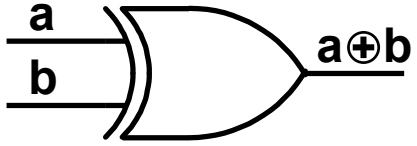
b	a	a+b
0	0	0
0	1	1
1	0	1
1	1	1

- NOT
  - Negación lógica (“no”)
  - No ampliable



a	ā
0	1
1	0

- XOR
  - OR Exclusiva
  - No ampliable

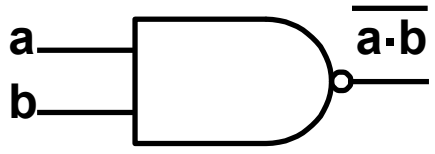


b	a	a⊕b
0	0	0
0	1	1
1	0	1
1	1	0

Puertas lógicas con salida negada

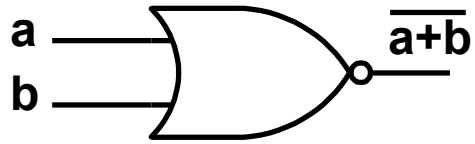
FCO

- NAND = NOT (AND)
  - Ampliable



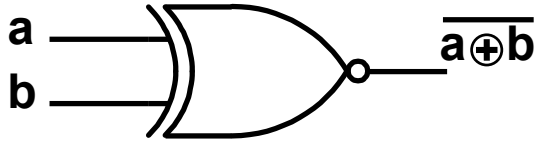
b	a	$\overline{a \cdot b}$
0	0	1
0	1	1
1	0	1
1	1	0

- NOR = NOT (OR)
  - Ampliable



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

- XNOR = NOT (XOR)
  - No ampliable



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

DISCA

Puertas lógicas. Tecnologías

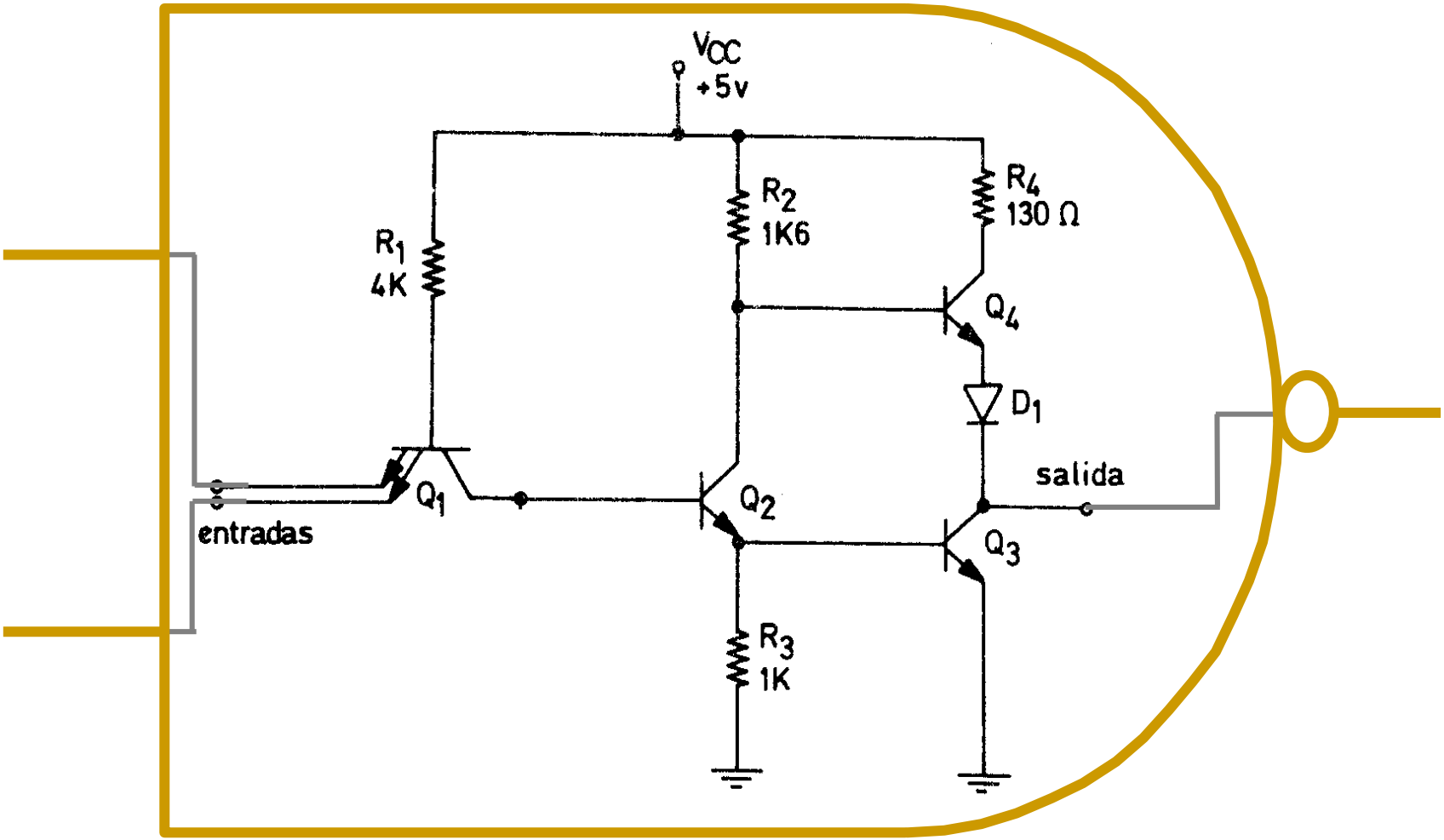
- Cada tecnología de construcción emplea diferentes tipos de elementos físicos (transistores) y tensiones para representar los valores lógicos “0” y “1”

- TTL = Transistor-Transistor Logic
  - Basada en transistores bipolares
  - Alta velocidad, alto consumo, difícil integración

- CMOS = Complementary Metal Oxide Semiconductor
  - Basada en transistores MOSFET
  - Menor velocidad, bajo consumo, alta escala de integración

Esquema físico de una NAND TTL

FCO



DISCA

Puertas lógicas. Integración. FCO

National Semiconductor

54LS00/DM54LS00/DM74LS00
Quad 2-Input NAND Gates

Function Table

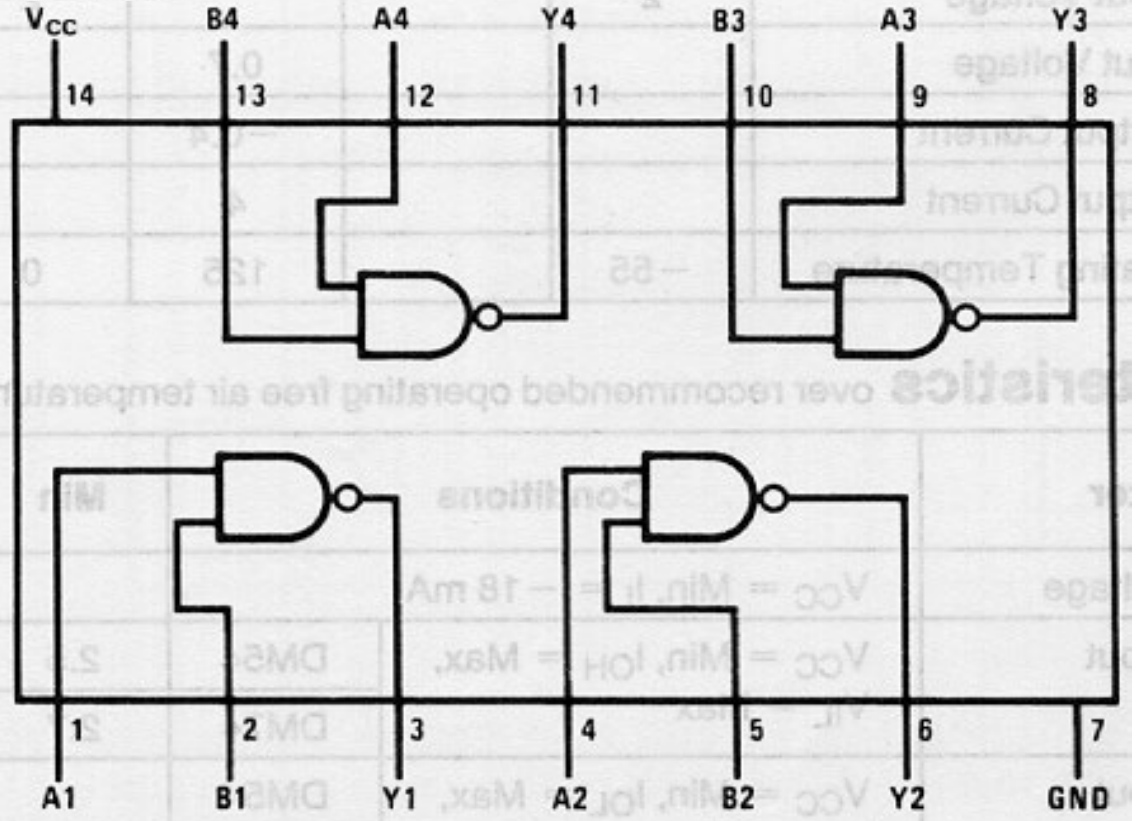
$$Y = \overline{AB}$$

Inputs		Output
A	B	Y
L	L	H
L	H	H
H	L	H
H	H	L

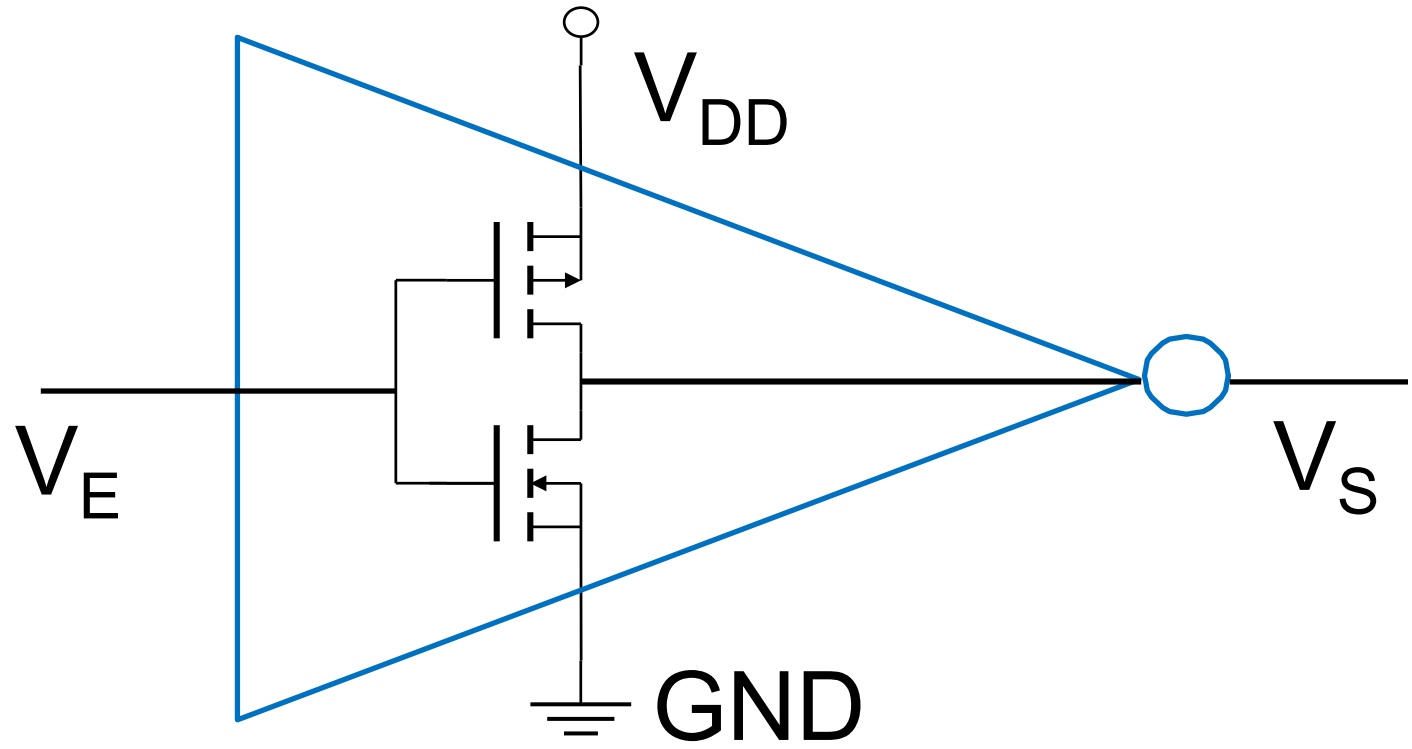
H = High Logic Level

L = Low Logic Level

Dual-In-Line Package



Esquema físico de un inversor CMOS

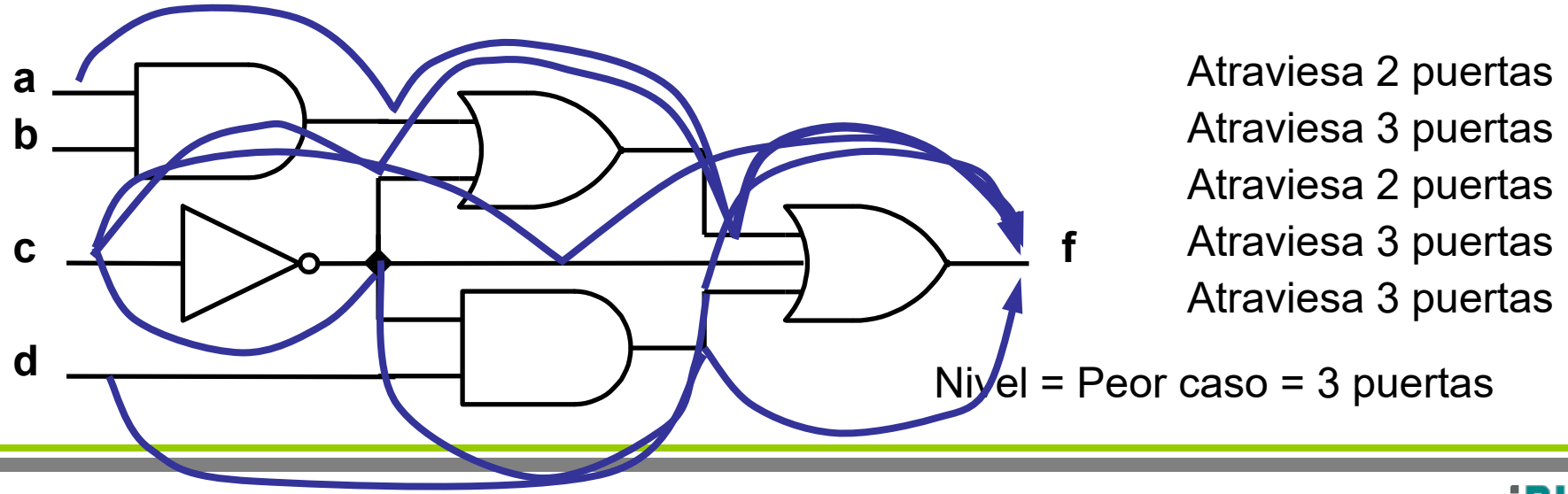


Índice

- Introducción
- Funciones lógicas y tablas de verdad
- Puertas lógicas
- Análisis de circuitos
- Álgebra de Boole
- Formas canónicas de representar una función lógica
- Simplificación de funciones lógicas
  - Mapas de Karnaugh

Nivel de un circuito lógico

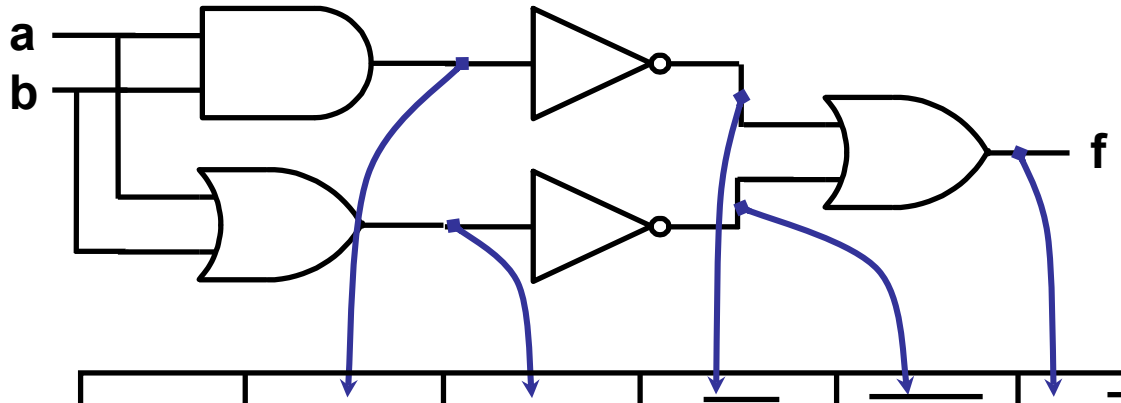
- Nivel
  - Número de puertas que hay que atravesar en el peor de los casos desde las entradas a las salidas del circuito
  - Es una indicación del retardo del circuito
  - Cada puerta tiene un retardo T
  - Nivel 0 = entradas



Análisis de circuitos

- Dado un circuito, se trata de obtener su función lógica y tabla de verdad
  - Función lógica: componiendo las subfunciones correspondientes a cada punto del circuito
  - Tabla de verdad: calculando la salida para todas las posibles combinaciones de entrada

Análisis de circuitos (ii)



- Ejemplo

Función lógica

b	a	a·b	a + b	$\overline{a \cdot b}$	$\overline{a + b}$	$f = \overline{a \cdot b} + \overline{a + b}$
0	0	0	0	1	1	1
0	1	0	1	1	0	1
1	0	0	1	1	0	1
1	1	1	1	0	0	0

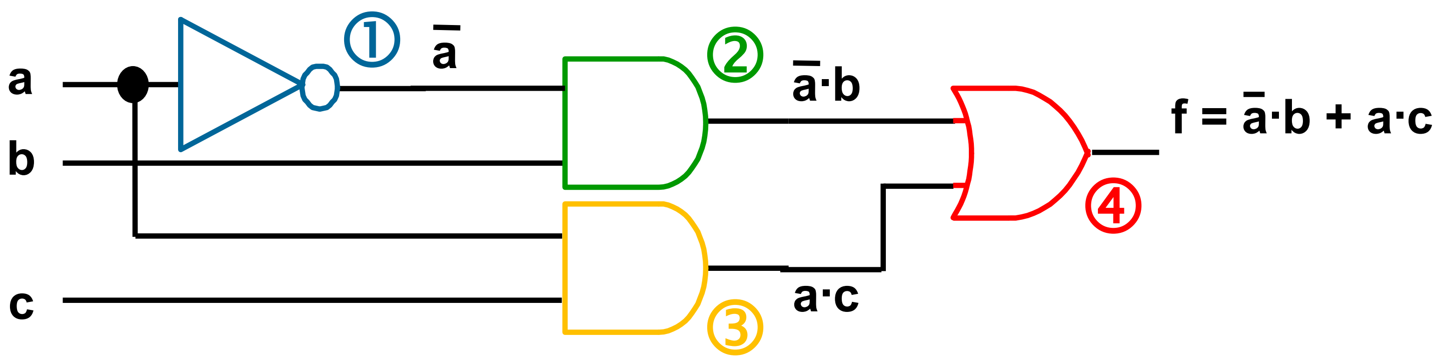
b	a	f
0	0	1
0	1	1
1	0	1
1	1	0

Tabla de verdad

Síntesis de un circuito lógico

- A partir de su función lógica:
  - Añadir e interconectar las puertas en el orden en que se evalúan los términos de la expresión lógica
  - Ejemplo:

$$f = \overline{a} \cdot b + a \cdot c$$



Índice

- Introducción
- Funciones lógicas y tablas de verdad
- Puertas lógicas
- Análisis de circuitos
- Álgebra de Boole
- Formas canónicas de representar una función lógica
- Simplificación de funciones lógicas
  - Mapas de Karnaugh

Álgebra de Boole

- George Boole (s. XIX)
  - Matemático y filósofo inglés
  - Desarrolla una estructura algebraica con dos valores (“verdadero”, “falso”) y dos leyes de composición interna (“y”, “o”)
  - Permite formalizar las reglas del razonamiento lógico
- Claude Shannon (1938, Lab. Bell)
  - Adapta este álgebra a la computación
    - Valores 0 y 1, leyes de composición AND y OR
  - Permite formalizar las reglas de construcción de circuitos digitales

Precedencia (si no hay paréntesis)

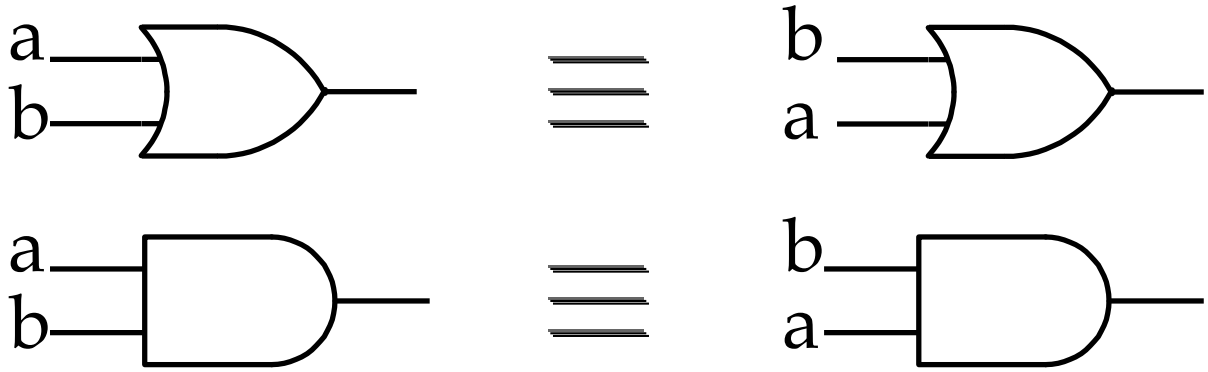
Puerta lógica	Símbolo estándar
NOT	—
AND	·
OR	+
XOR	⊕

Álgebra de Boole. Axiomas

- Conmutatividad

$$a + b = b + a$$

$$a \cdot b = b \cdot a$$



- Distributividad

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

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

Álgebra de Boole. Axiomas (ii)

- Existencia de elemento neutro

$$a + 0 = a$$

$$a \cdot 1 = a$$

- Existencia de elemento complementario

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

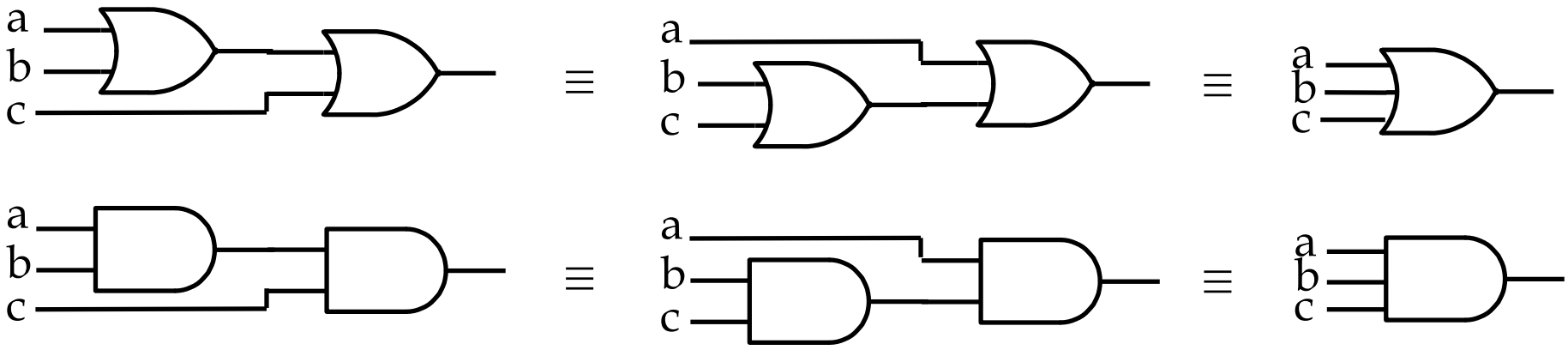
$$a \cdot \overline{a} = 0$$

Álgebra de Boole. Propiedades

- Asociativa

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

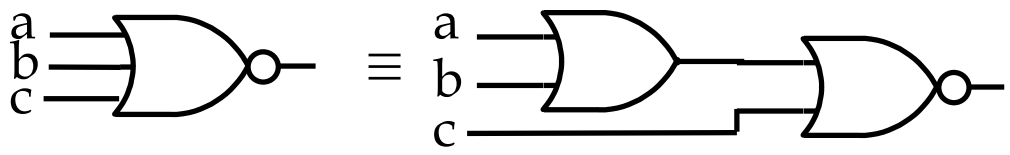
  - Permite construir puertas con mayor número de entradas a partir de puertas más pequeñas:

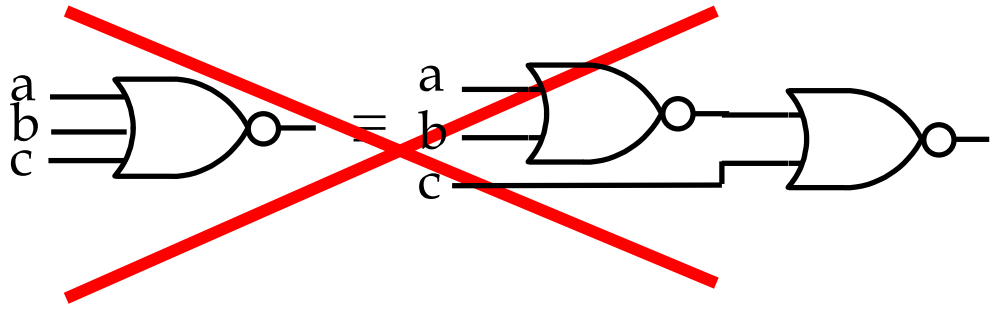


Álgebra de Boole. Propiedades (ii)

- Asociativa (ii)
  - ¡OJO a las puertas con salida negada!:

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

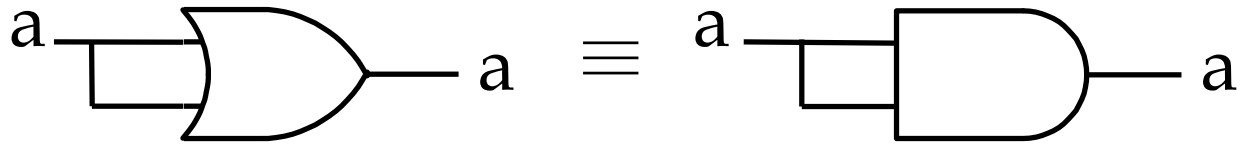




Álgebra de Boole. Propiedades (iii) FCO

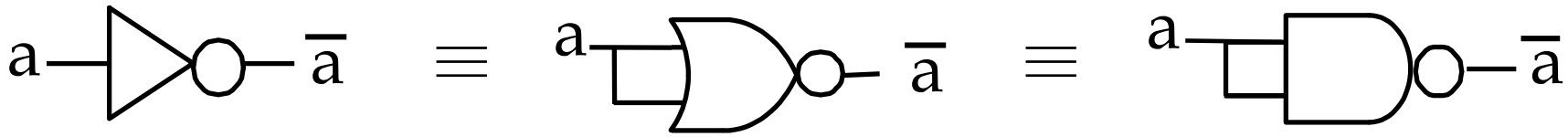
- Idempotencia

$$a + a = a \qquad a \cdot a = a$$



  - Permite construir puertas NOT a partir de NAND o NOR

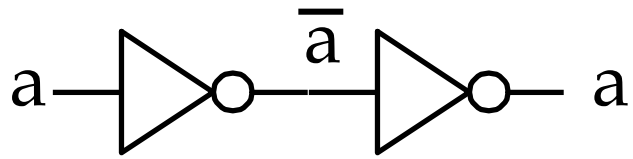
$$\overline{a + a} = \overline{a} = \overline{a \cdot a}$$



Álgebra de Boole. Propiedades (iv)

- Involución

$$\overline{\overline{a}} = a$$



- Leyes de De Morgan

$$\overline{(a + b + \ldots + n)} = \overline{a} \cdot \overline{b} \cdot \ldots \cdot \overline{n}$$

$$\overline{(a \cdot b \cdot \ldots \cdot n)} = \overline{a} + \overline{b} + \ldots + \overline{n}$$

  - ¡OJO!

$$\overline{(a + b)} = \overline{a} + \overline{b}$$

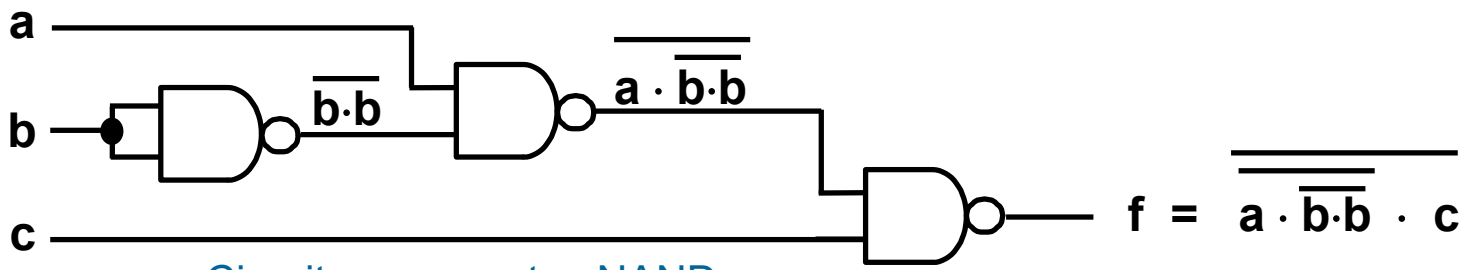
$$\overline{(a + b)} = \overline{a} \cdot \overline{b}$$

Álgebra de Boole. Propiedades (v) FCO

- Aplicando adecuadamente las leyes de De Morgan, cualquier circuito lógico se puede implementar sólo con puertas NAND o NOR.
  - Ejemplo:

INVOLUCIÓN DE MORGAN INVOLUCIÓN IDEMPOTENCIA (este último paso se suele obviar)

$$f = a \cdot \overline{b} + \overline{c} = \overline{\overline{a \cdot \overline{b} + \overline{c}}} = \overline{\overline{a \cdot \overline{b}} \cdot \overline{\overline{c}}} = \overline{\overline{a \cdot \overline{b}} \cdot c} = \overline{\overline{a \cdot \overline{b \cdot b} \cdot c}}$$



Circuito con puertas NAND

Índice

- Introducción
- Funciones lógicas y tablas de verdad
- Puertas lógicas
- Análisis de circuitos
- Álgebra de Boole
- Formas canónicas de representar una función lógica
- Simplificación de funciones lógicas
  - Mapas de Karnaugh

Formas canónicas

- Expresión algebraica única de una función lógica formulada con maxitérminos o minitérminos

- Minitérmino de orden n
  - Producto en el que aparecen las n variables lógicas de entrada
  - Cada variable aparece complementada si su valor es 0
  - Cada valoración da lugar a un minitérmino distinto
  - Los minitérminos se numeran según la cantidad representada por la valoración correspondiente

Formas canónicas (ii)

- Maxitérmino de orden n
  - Suma en la que aparecen las n variables lógicas de entrada
  - Cada variable aparece complementada si su valor es 1
  - Cada valoración da lugar a un maxitérmino distinto
  - Los maxitérminos se numeran según la cantidad representada por la valoración correspondiente

Forma canónica disyuntiva

- Forma canónica disyuntiva o suma de productos
  - Suma de los minitérminos pertenecientes a la función
  - Pertenecen a la función los minitérminos correspondientes a las valoraciones para las que la función vale 1

$$\sum_{listadevblesdelafunción}\left(lista\ numerada\ de\ los\ minitérminos\ de\ la\ función\right)$$

b	a	f	minitérmino	nº
0	0	0	$\overline{b}\cdot\overline{a}$	0
0	1	1	$\overline{b}\cdot a$	1
1	0	0	$b\cdot\overline{a}$	2
1	1	1	$b\cdot a$	3

Forma canónica

$$f=\sum_{b,a}(1,3)=\overline{b}\cdot a+b\cdot a$$

Expresión algebraica equivalente

Forma canónica conjuntiva

- Forma canónica conjuntiva o producto de sumas
  - Producto de los maxitérminos de la función
  - Pertenecen a la función los maxitérminos correspondientes a las valoraciones para las que la función vale 0

$$\prod_{lista\ de\ vbles\ de\ la\ función} (lista\ numerada\ de\ los\ maxitérminos\ de\ la\ función)$$

b	a	f	maxitérmino	nº
0	0	0	$b + a$	0
0	1	1	$b + \overline{a}$	1
1	0	0	$\overline{b} + a$	2
1	1	1	$\overline{b} + \overline{a}$	3

Forma canónica

$$f = \prod_{b,a} (0,2) = (b + a) \cdot (\overline{b} + a)$$

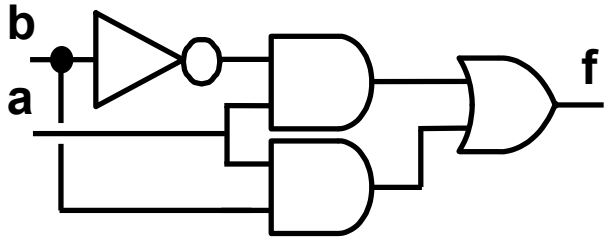
Expresión algebraica equivalente

Formas canónicas. Interés

- Expresión única y compacta de una función lógica.
- Primera aproximación a la síntesis de circuitos a partir de una tabla de verdad:

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

$$f = \sum_{b,a}(1,3) = \bar{b} \cdot a + b \cdot a$$



- Cualquier función lógica puede implementarse mediante un circuito de nivel ≤ 3.

Formas canónicas. Entradas indiferentes FCO

- Formas canónicas para funciones con entradas indiferentes
  - Estas combinaciones se agrupan por separado en sumatorios o productorios del conjunto vacío Φ

	a	pi	pd	id
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	X
4	1	0	0	1
5	1	0	1	1
6	1	1	0	1
7	1	1	1	X

$$id = \sum_{a,\,pi,\,pd}(1,4,5,6) + \sum_{\phi}(3,7)$$

$$id = \prod_{a,\,pi,\,pd}(0,2) \bullet \prod_{\phi}(3,7)$$

Índice

- Introducción
- Funciones lógicas y tablas de verdad
- Puertas lógicas
- Análisis de circuitos
- Álgebra de Boole
- Formas canónicas de representar una función lógica
- Simplificación de funciones lógicas
  - Mapas de Karnaugh

Simplificación

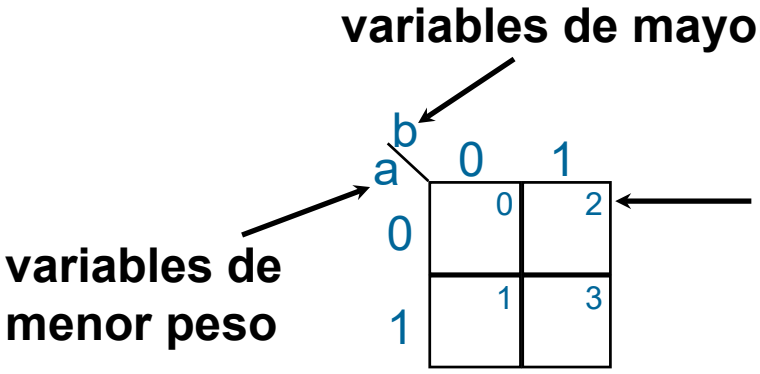
- Simplificar una función
  - Consiste en hallar una expresión algebraica equivalente a la de partida, pero de menor tamaño (menos términos, términos con menos variables)
  - El objetivo es reducir al máximo el circuito con el que se implementa una función lógica
- Metodología
  - Algebraica. Aplicación de axiomas y propiedades del álgebra de Boole
    - Elemento complementario, elemento neutro, distributiva y asociativa
  - Gráfica. Mapas de Karnaugh

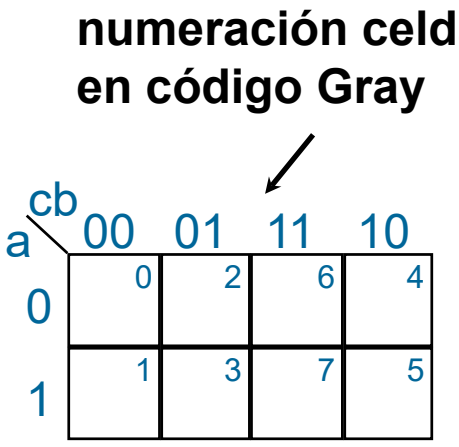
Simplificación por Karnaugh

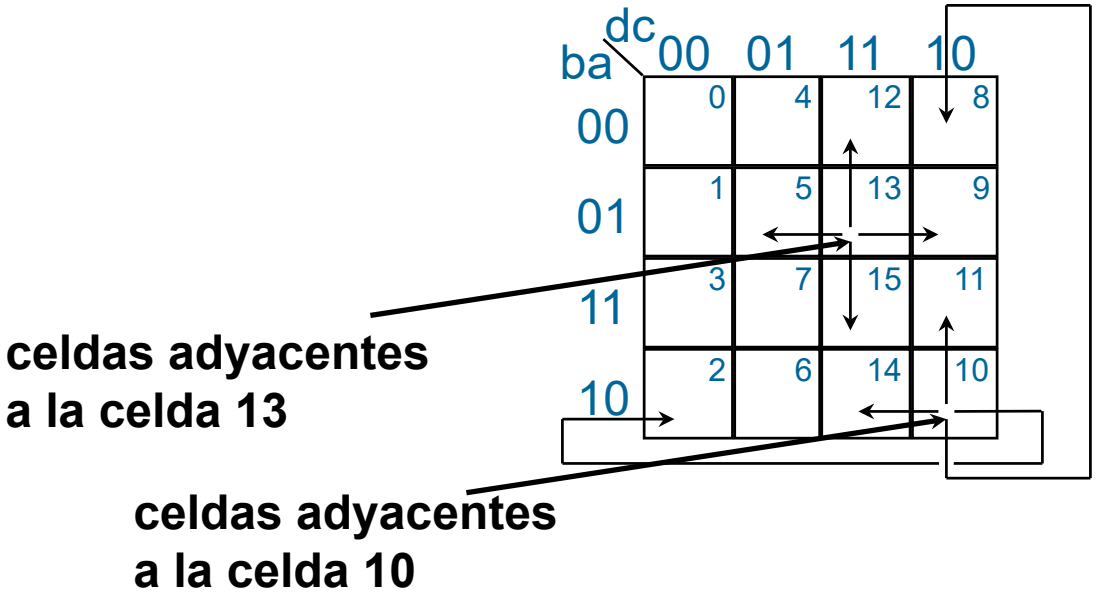
- Mapa de Karnaugh
  - Representación matricial de una tabla de verdad
  - Una celda del mapa de Karnaugh representa una fila de la tabla de verdad
  - En cada celda se coloca el valor de una salida de la función
  - La disposición espacial de las celdas es tal que los términos adyacentes de la función lógica están en celdas adyacentes
  - Dos términos se dicen adyacentes si sus valoraciones difieren en el valor de una sola variable
  - Los bordes del mapa de Karnaugh deben considerarse adyacentes

Simplificación por Karnaugh (ii)

- Mapas para funciones de 2, 3 y 4 variables





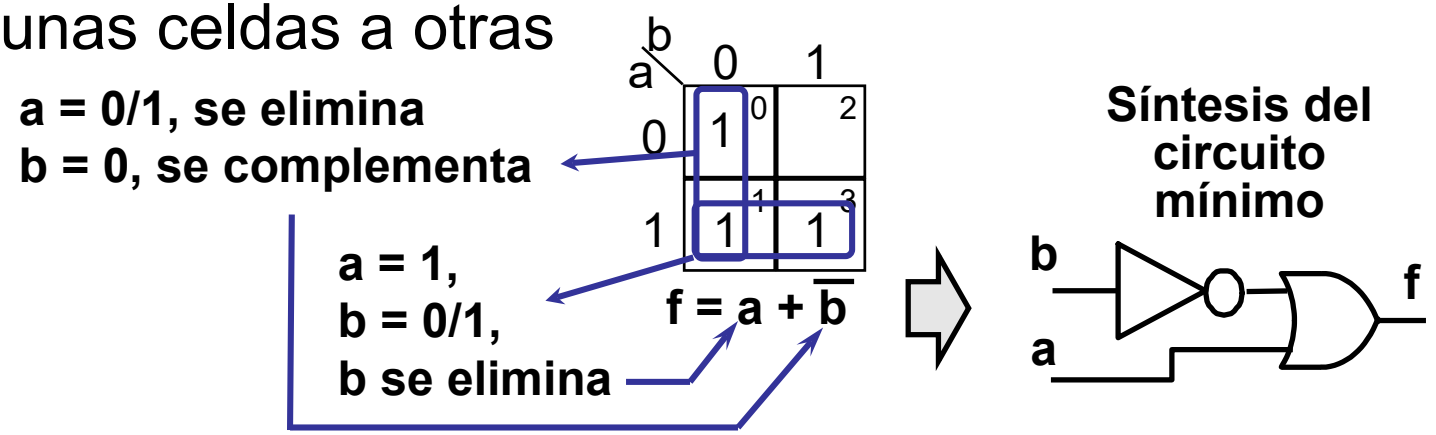


Simplificación por Karnaugh (iii) FCO

- Método de simplificación
  - Agrupar todas las celdas con el mismo valor, en uno o más grupos
  - Cada grupo contendrá un número de celdas adyacentes potencia de 2
  - Hacer los grupos lo más grande posible
  - El número de grupos debe ser mínimo
  - Una celda puede estar en uno o más grupos

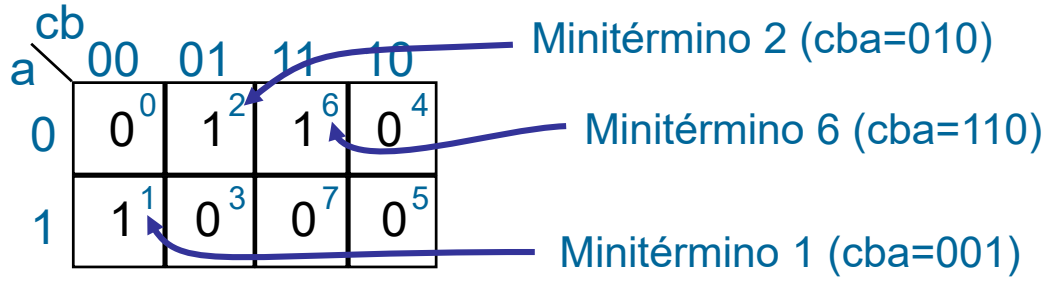
Simplificación por unos. Método FCO

- Agrupar las celdas de valor 1
- Cada grupo representa a un término producto (no minitérmino, puesto que no aparecerán todas las vbles. de la función). Las variables a cero aparecerán complementadas
- Un grupo de 2^k celdas elimina k variables del término resultante, y por tanto tendrá n-k variables
- En cada grupo se eliminan las variables que cambian de valor de unas celdas a otras



Simplificación por unos. Fundamento FCO

– Cada celda a 1 representa un minitérmino que pertenece a la función

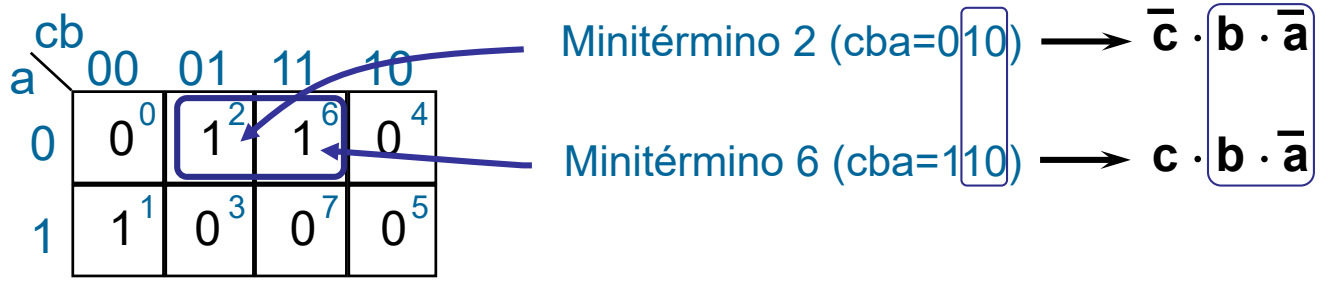


– La función sin simplificar incluiría todos los minitérminos que le pertenecen:

$$\mathbf{f} = \sum_{c,b,a}(1,2,6) = \bar{\mathbf{c}} \cdot \bar{\mathbf{b}} \cdot \mathbf{a} + \bar{\mathbf{c}} \cdot \mathbf{b} \cdot \bar{\mathbf{a}} + \mathbf{c} \cdot \mathbf{b} \cdot \bar{\mathbf{a}}$$

Simplificación por unos. Fundamento (ii) FCO

– Los grupos de celdas adyacentes detectan minitérminos con un factor común:



– Su suma es simplificable. Algebraicamente sería:

ASOCIATIVA DISTRIBUTIVA ELEM. COMPLEM. ELEM. NEUTRO

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

– Karnaugh obtiene el mismo resultado:

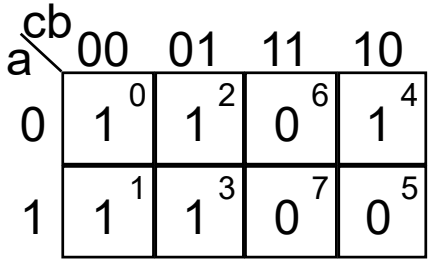
Celda 2 (cba=010)
Celda 6 (cba=110)

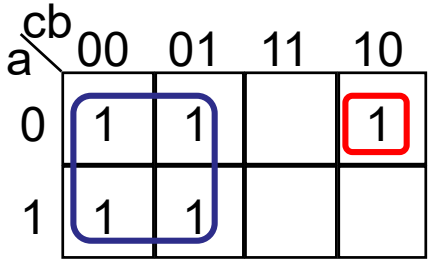
c = 0/1, se elimina
b = 1, se incluye
a = 0, se complementa

$\boxed{b \cdot \bar{a}}$

Simplificación por unos. Ejemplos

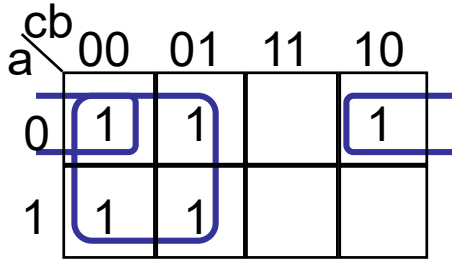
- Ejemplos





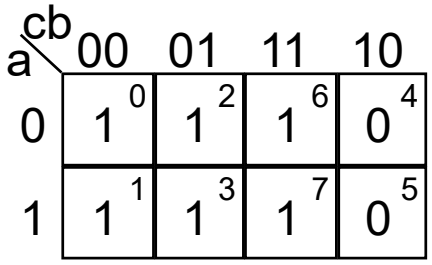
$f = \overline{c} + c \cdot \overline{a} \cdot \overline{b}$

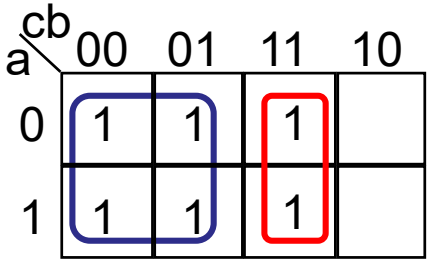
MAL



$f = \overline{c} + \overline{a} \cdot \overline{b}$

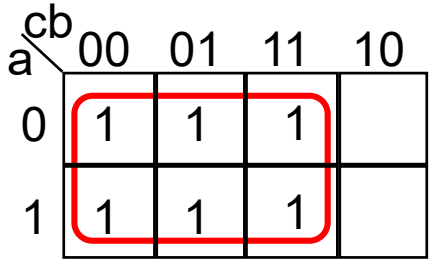
BIEN





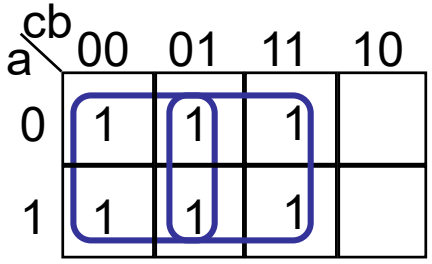
$f = \overline{c} + c \cdot b$

MAL



$f = ?$

MAL

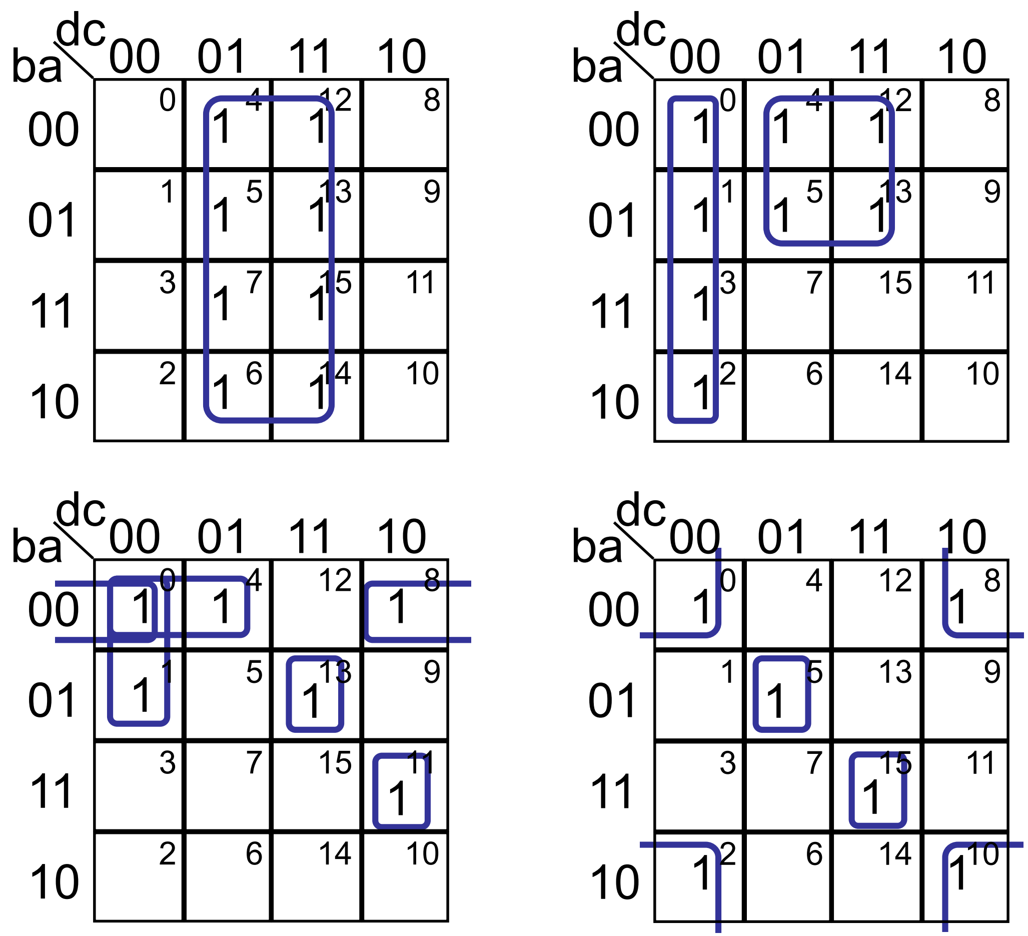


$f = \overline{c} + b$

BIEN

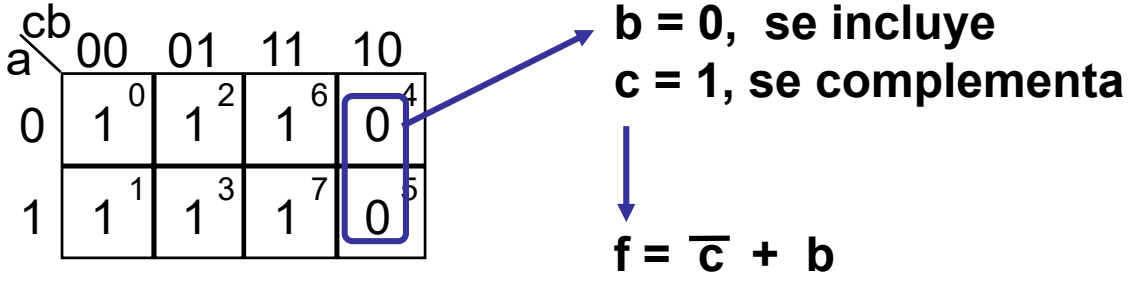
Simplificación por unos. Ejemplos (ii) FCO

- Ejemplos (cont.)



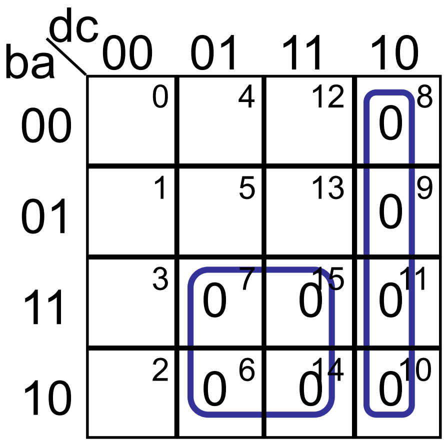
Simplificación por ceros

- Agrupar las celdas de valor cero
- Cada grupo representa un término suma (no maxitérmino, puesto que no aparecerán todas las variables de la función). Las variables de valor uno aparecerán complementadas
- Un grupo de 2^k celdas elimina k variables del término resultante, y por tanto tendrá n-k variables
- En cada grupo se eliminan las variables que cambian de valor de unas celdas a otras

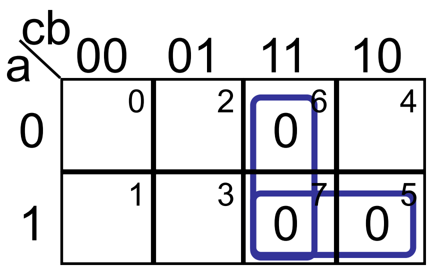


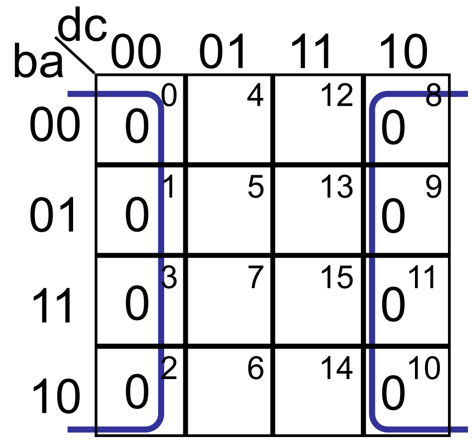
Simplificación por ceros (ii)

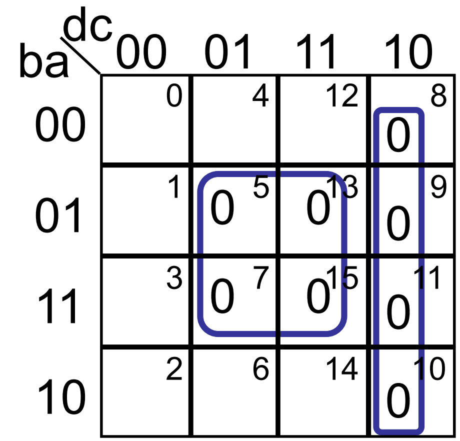
- Ejemplos

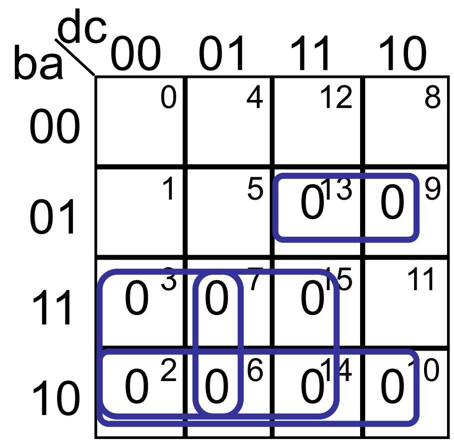


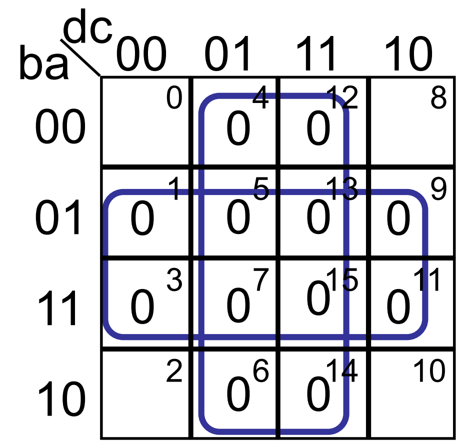
$$f = (\overline{d} + c) \cdot (\overline{c} + \overline{b})$$









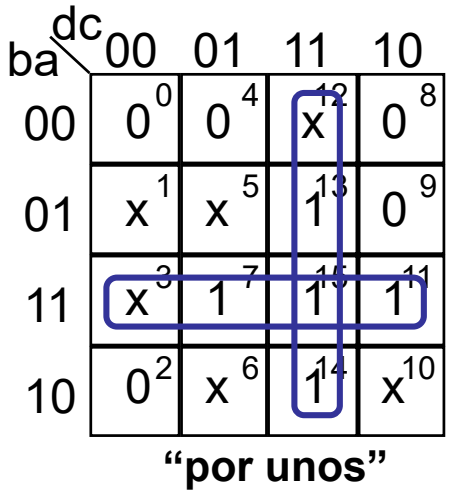


Simplificación. Entradas indiferentes FCO

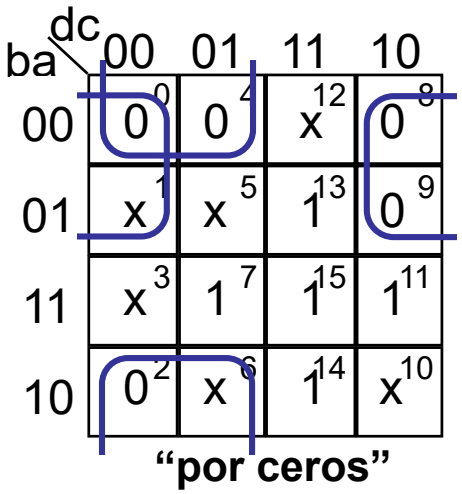
– Las celdas con “x” se toman como si tuvieran valor uno o valor cero, cada una como mejor convenga, para maximizar la simplificación.

d	c	b	a	f
0	0	0	0	0
0	0	0	1	x
0	0	1	0	0
0	0	1	1	x
0	1	0	0	0
0	1	0	1	x
0	1	1	0	x
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	x
1	0	1	1	1
1	1	0	0	x
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

$$f = \sum_{d,c,b,a}(7,11,13,14,15) + \sum_{\phi}(1,3,5,6,10,12) = \prod_{d,c,b,a}(0,2,4,8,9) \bullet \prod_{\phi}(1,3,5,6,10,12)$$



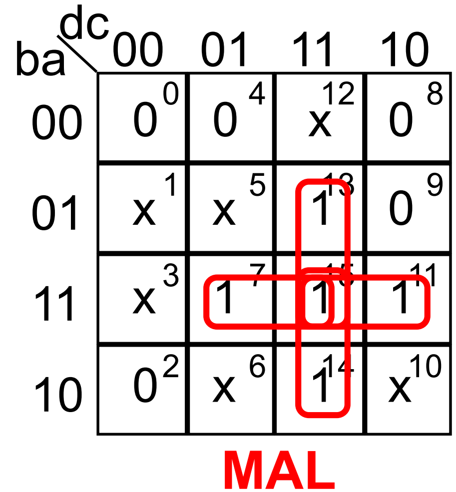
“por unos”

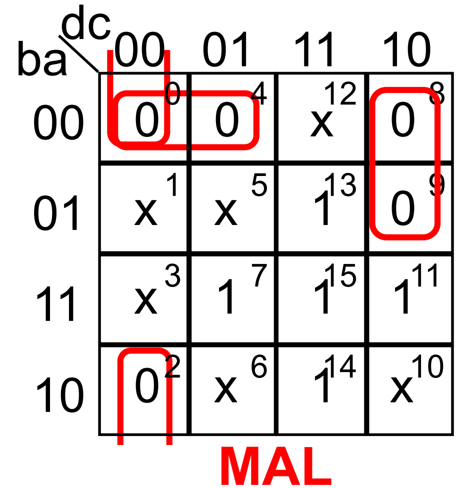


“por ceros”

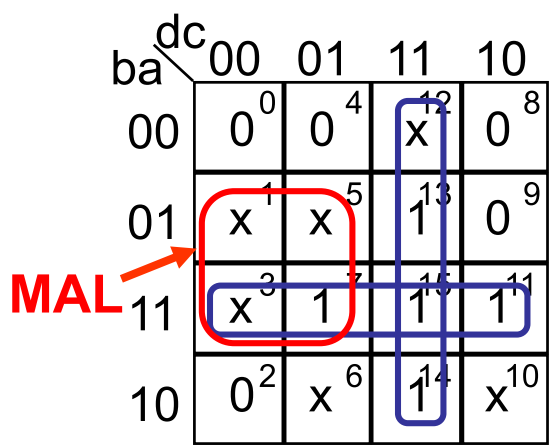
Simplificación. Entradas indiferentes (ii) FCO

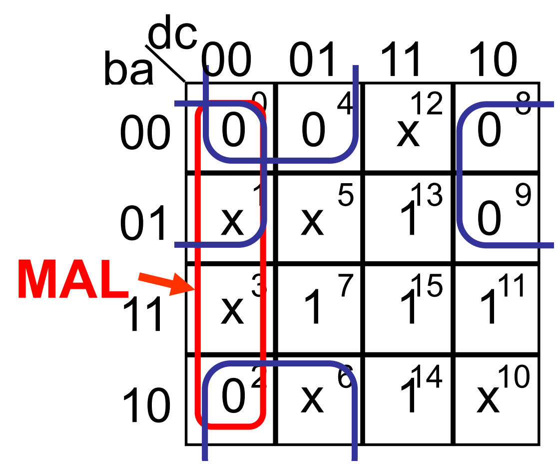
- **Errores comunes**:
  - Tomar todas las "x" por 0 o por 1





  - Hacer grupos con "x" innecesarios





Recursos de aprendizaje FCO

- Poliformat, sección “Recursos”
  - Ejercicios sin solución.
  - Soluciones a los ejercicios.
  - **Entrenador de Karnaugh.**
  - Exámenes de años anteriores.

- Poliformat, sección “Lessons”
  - Módulo 2: Principios de diseño digital.
    - » *Tablas de verdad.*
    - » *Puertas lógicas.*

UNIVERSIDAD POLITECNICA DE VALENCIA

etsinf

DISCA

Fundamentos de computadores

TEMA 2. PRINCIPIOS DEL DISEÑO DIGITAL