

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:
 - $B = \{0, 1\}$
 - $\Phi = \{+, \bullet\}$

Postulados del Álgebra de Boole

- Ley de composición interna
 - $\forall a, b \in B \Rightarrow a + b \in B, a \cdot b \in B$
- Elementos neutros
 - $\forall a \in B \Rightarrow \exists$ elementos neutros (0 y 1 respectivamente)
 $a + 0 = a$
 $a \cdot 1 = a$
- Propiedad conmutativa
 - $\forall a, b \in B \Rightarrow$
 $a + b = b + a$
 $a \cdot b = b \cdot a$
- Propiedad distributiva
 - $\forall a, b, c \in B \Rightarrow$
 $a + b \cdot c = (a + b) \cdot (a + c)$
 $a \cdot (b + c) = a \cdot b + a \cdot c$

Postulados del Álgebra de Boole

- Elemento inverso o complementario

- $\forall a \in B \Rightarrow \exists \bar{a} \in B$

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

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

Propiedades fundamentales del Álgebra de Boole

- Dualidad: Toda ley válida tiene una dual, que se obtiene cambiando $0 \leftrightarrow 1$ y $+ \leftrightarrow \cdot$

- Idempotencia

- $\forall a \in B \Rightarrow$
 $a + a = a$
 $a \cdot a = a$

- Demostración:

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

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

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	$a \cdot b$
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

- $\forall a \in B \Rightarrow \overline{\overline{a}} = a$

- Absorción

- $\forall a, b \in B \Rightarrow \begin{aligned} a + ab &= a \\ a(a+b) &= a \end{aligned}$

- Demostración:

- $$a + ab = a \cdot 1 + ab = a(1 + b) = a \cdot 1 = a$$

- Propiedad asociativa

- $\forall a, b, c \in B \Rightarrow \begin{aligned} (a + b) + c &= a + (b + c) \\ (a \cdot b) \cdot c &= a \cdot (b \cdot c) \end{aligned}$

Propiedades fundamentales del Álgebra de Boole

- Leyes de De Morgan:

- $\forall a, b \in B \Rightarrow$
$$\overline{a + b} = \bar{a} \bar{b}$$
$$\overline{a \cdot b} = \bar{a} + \bar{b}$$

- Demostración:

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

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

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

Funciones y expresiones booleanas

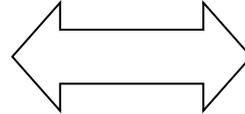
- Definiciones:
 - Una variable lógica o booleana es cualquier elemento $x \in B = \{0, 1\}$
 - Un literal es una variable negada o sin negar
 - Función lógica o booleana:

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

Representación de funciones lógicas

- Expresión
- Tabla de verdad

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

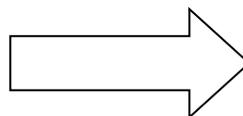


a	b	f(a,b)
0	0	0
0	1	1
1	0	1
1	1	1

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

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

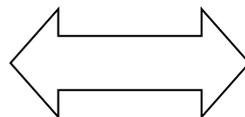


a	b	c	f
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

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:

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



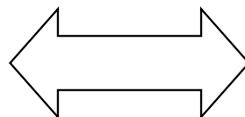
a	b	c	f
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

- 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

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

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

- Demostración

$$\begin{aligned} x_i = 0 \Rightarrow f(x_1, x_2, \dots, x_n) &= 1 \cdot f(x_1, \dots, 0, \dots, x_n) + 0 \cdot f(x_1, \dots, 1, \dots, x_n) = \\ &= f(x_1, \dots, 0, \dots, x_n) \end{aligned}$$

$$\begin{aligned} x_i = 1 \Rightarrow f(x_1, x_2, \dots, x_n) &= 0 \cdot f(x_1, \dots, 0, \dots, x_n) + 1 \cdot f(x_1, \dots, 1, \dots, x_n) = \\ &= f(x_1, \dots, 1, \dots, x_n) \end{aligned}$$

- La otra forma se demuestra por dualidad

Corolario del Teorema de Expansión de Shannon

- Aplicando recursivamente el Teorema:

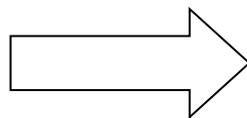
$$\begin{aligned} f(a,b,c) &= \bar{a}f(0,b,c) + af(1,b,c) = \\ &= \bar{a}(\bar{b}f(0,0,c) + bf(0,1,c)) + a(\bar{b}f(1,0,c) + bf(0,1,c)) = \\ &= \bar{a}\bar{b}f(0,0,c) + \bar{a}bf(0,1,c) + a\bar{b}f(1,0,c) + abf(0,1,c) = \\ &= \bar{a}\bar{b}\bar{c}f(0,0,0) + \bar{a}\bar{b}cf(0,0,1) + \bar{a}b\bar{c}f(0,1,0) + \bar{a}bcf(0,1,1) + \\ &\quad + a\bar{b}\bar{c}f(1,0,0) + a\bar{b}cf(1,0,1) + ab\bar{c}f(1,1,0) + abc f(1,1,1) = \\ &= \sum_3 m_i k_i \end{aligned}$$

- 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 minterminos 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

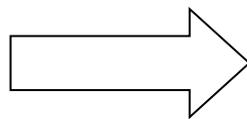


$$\begin{aligned}
 f(a,b,c) &= \sum_3 (0,2,5) = \sum_3 m(0,2,5) = \\
 &= \bar{a}\bar{b}\bar{c} + \bar{a}b\bar{c} + a\bar{b}c
 \end{aligned}$$

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



$$\begin{aligned}
 f(a,b,c) &= \prod_3 (1,3,4,6,7) = \prod_3 M(1,3,4,6,7) = \\
 &= (a + b + \bar{c})(a + \bar{b} + \bar{c})(\bar{a} + b + c) \\
 &\quad (\bar{a} + \bar{b} + c)(\bar{a} + \bar{b} + \bar{c})
 \end{aligned}$$

CUIDADO: al contrario que los mintérminos!

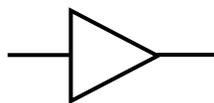
Puertas lógicas

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

- Identidad

$$z = a$$

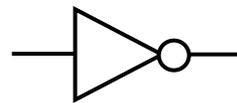
a	a
0	0
1	1



- Puerta NOT o inversor

$$z = \bar{a}$$

a	\bar{a}
0	1
1	0

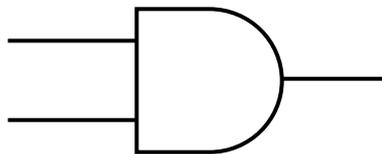


Puertas AND y OR

- Puerta AND

$$z = a \cdot b$$

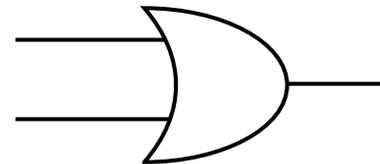
a	b	a•b
0	0	0
0	1	0
1	0	0
1	1	1



- Puerta OR

$$z = a + b$$

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

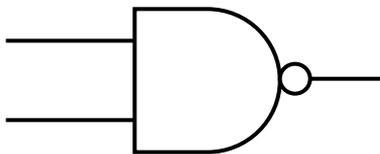


Puertas NAND y NOR

- Puerta NAND

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

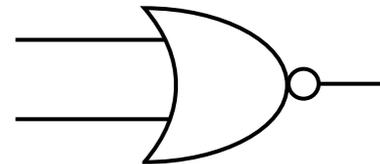
a	b	$\overline{a \cdot b}$
0	0	1
0	1	1
1	0	1
1	1	0



- Puerta NOR

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

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

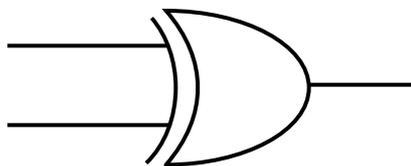


Puertas XOR y XNOR

- Puerta XOR (OR-Exclusiva)

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

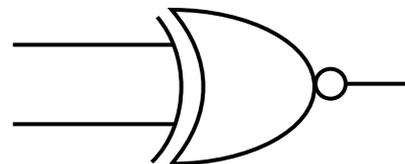
a	b	$a \oplus b$
0	0	0
0	1	1
1	0	1
1	1	0



- 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

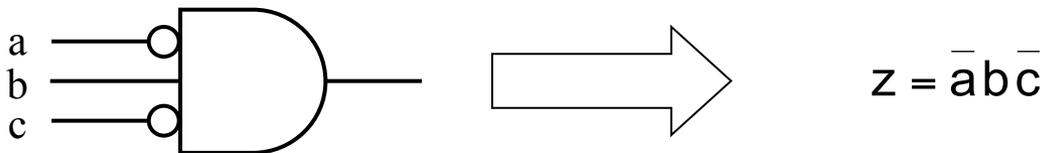


Generalización a n entradas

Puerta	Valor de la salida	
	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 n° par de entradas = 1	Hay un n° impar de entradas = 1
XNOR	Hay un n° impar de entradas = 1	Hay un n° par de entradas = 1

Otros símbolos

- Un círculo en una entrada o una salida indica negación



Tecnologías digitales

- Las puertas lógicas son circuitos electrónicos
- El nivel lógico (0 o 1) se representa mediante un nivel de tensión
- Generalmente se utiliza “lógica positiva”
 - Tensión alta (5V, 3.3V, 2.5 V, etc) → 1
 - Tensión baja (0V) → 0
- Existen muchas tecnologías, según la forma en que se realizan las puertas lógicas y las características que se obtienen

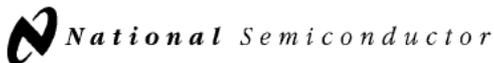
Familias lógicas

- El conjunto de componentes digitales básicos, tales como puertas lógicas y otros que estudiaremos a lo largo del curso, se conoce popularmente como *Serie o Familia 74*
- Existen numerosas subfamilias:
 - Según el rango de temperaturas de operación:
 - Serie 74: 0° a 70°
 - Serie 54: -55° a 125°
 - Según la tecnología utilizada:
 - LS
 - ALS
 - F
 - HC
 - AHC
 - G
 -

Familias lógicas

- Designación de componentes:
 - <Serie><Subfamilia><Componente>
- Ejemplo: 74HC00
 - Serie 74: rango de temperaturas convencional
 - Subfamilia HC (High speed CMOS)
 - Componente 00: 4 puertas NAND de 2 entradas
- Importante: las subfamilias no son compatibles entre sí
 - No se deben mezclar componentes de distintas subfamilias en un circuito

Hojas de catálogo



June 1989

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

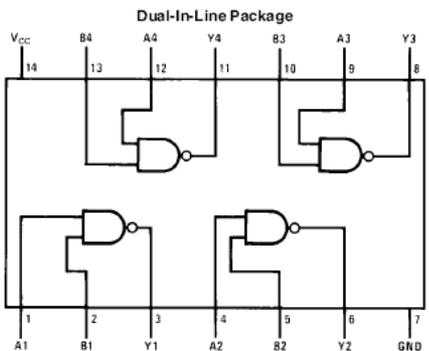
General Description

This device contains four independent gates each of which performs the logic NAND function.

Features

- Alternate Military/Aerospace device (54LS00) is available. Contact a National Semiconductor Sales Office/Distributor for specifications.

Connection Diagram



Order Number 54LS00DMQB, 54LS00FMQB, 54LS00MQB, DM54LS00J, DM54LS00W, DM74LS00M or DM74LS00N
See NS Package Number E20A, J14A, M14A, N14A or W14B

TL/F/6439-1

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

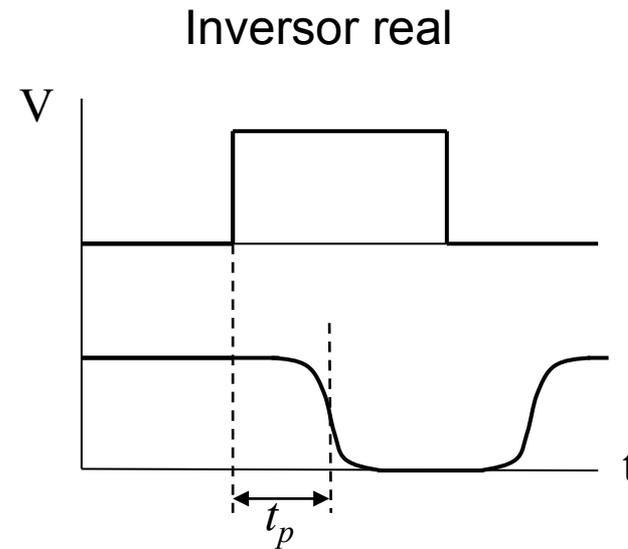
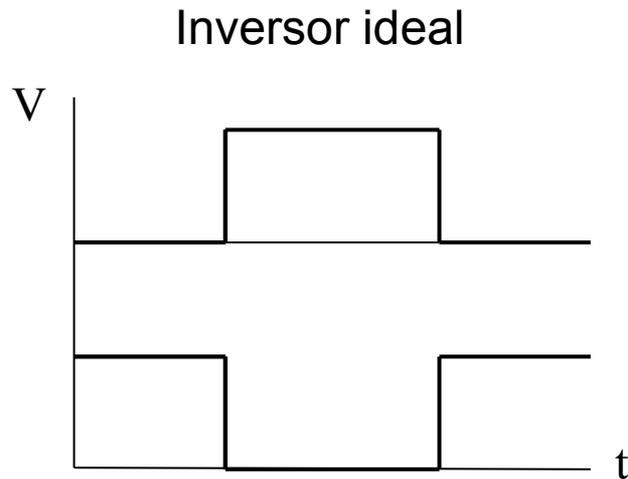
L = Low Logic Level

Características de las tecnologías digitales

- Principales características:
 - Margen de temperaturas de operación
 - Tensión de alimentación
 - Margen de ruido (intervalos de tensiones que se asocian a un nivel lógico determinado)
 - Retardo de conmutación
 - Consumo
 - Otros
- Cada tecnología o subfamilia presenta valores diferentes respecto a estos parámetros

Retardos

- Las puertas lógicas no conmutan instantáneamente



- El retardo limita la velocidad de operación del circuito

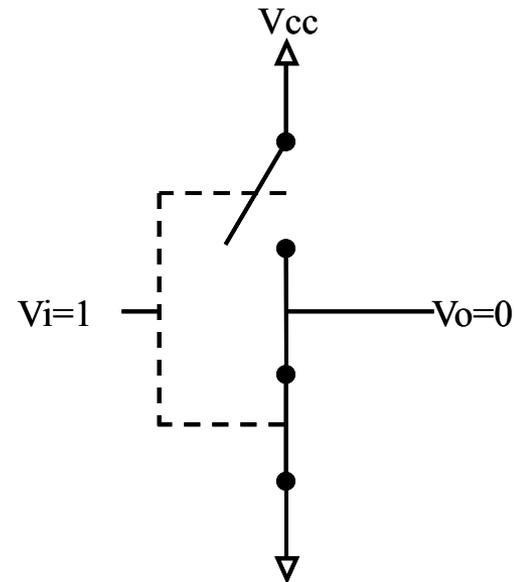
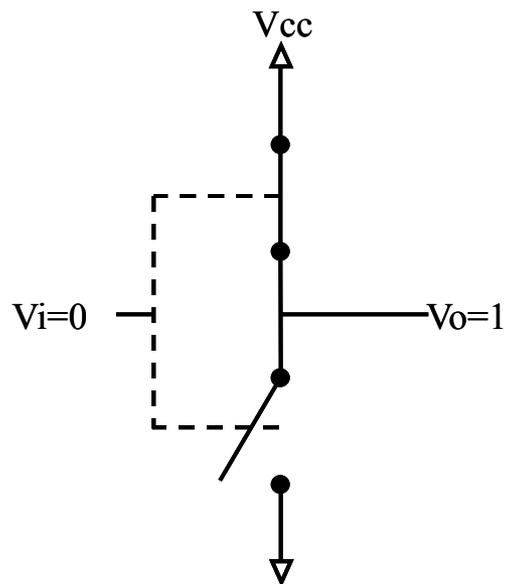
Consumo

- Las puertas lógicas consumen energía:
 - Estática: la que se consume por tener alimentada la puerta lógica, sin cambiar los valores lógicos
 - Dinámica: la que se consume al conmutar
- En la tecnología CMOS (la más utilizada actualmente), el consumo estático es muy pequeño. Sin embargo,
 - Los circuitos modernos pueden llegar a tener más de 10^8 puertas lógicas!
 - El consumo dinámico es proporcional a la frecuencia de conmutación
- El consumo es un problema importante:
 - La energía consumida se transforma en calor, que hay que disipar. Si el circuito consume mucho, puede ser difícil disipar el calor
 - En dispositivos portátiles, el tamaño y el peso de la batería es limitado

Tecnología CMOS

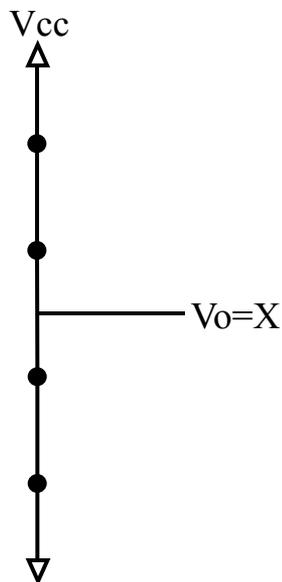
- La tecnología CMOS (Complementary Metal Oxide Semiconductor) es la tecnología más utilizada en la actualidad
- Basada en:
 - Transistores MOS: interruptores controlados por tensión
 - Complementarios: cada transistor o interruptor tiene su complementario, de manera que si un interruptor está abierto su complementario está cerrado y viceversa

Inversor CMOS

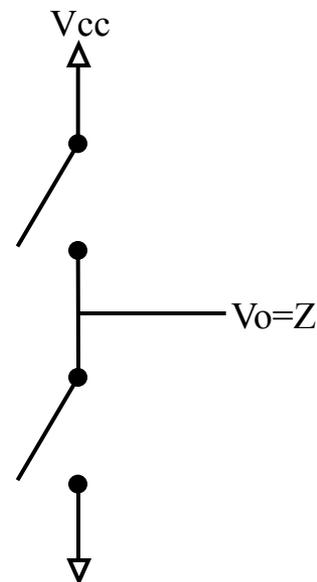


Valores metalógicos

- Hay situaciones que no se corresponden con valores lógicos
 - Cortocircuito (X)

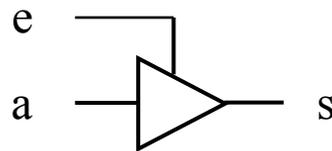


Alta impedancia o triestado (Z)



Buffer triestado

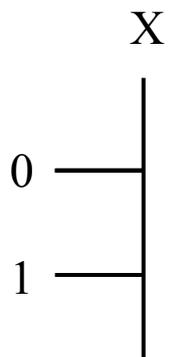
- Un tipo especial de puerta lógica que puede poner su salida en alta impedancia



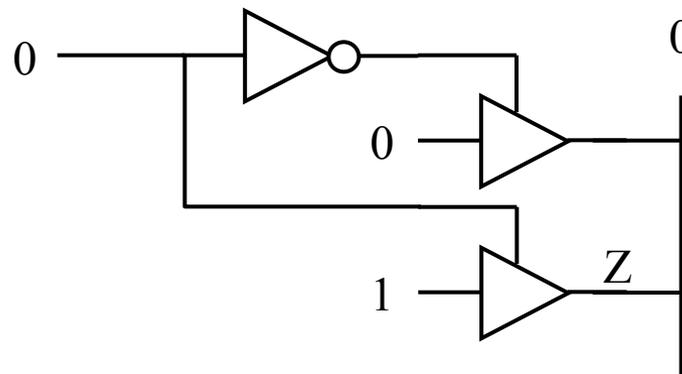
e	a	s
0	0	Z
0	1	Z
1	0	0
1	1	1

Buffer triestado

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



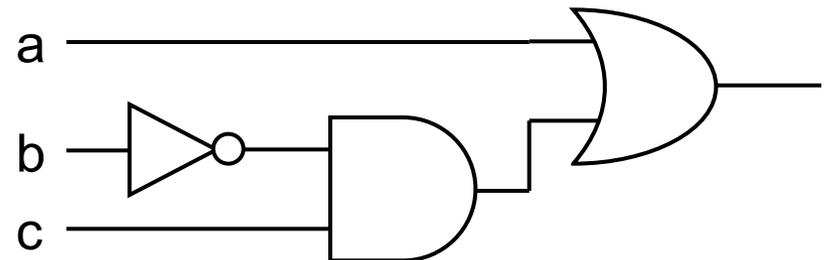
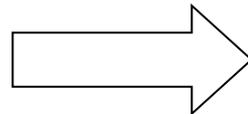
Cortocircuito!



Realización de una función lógica con puertas lógicas

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

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

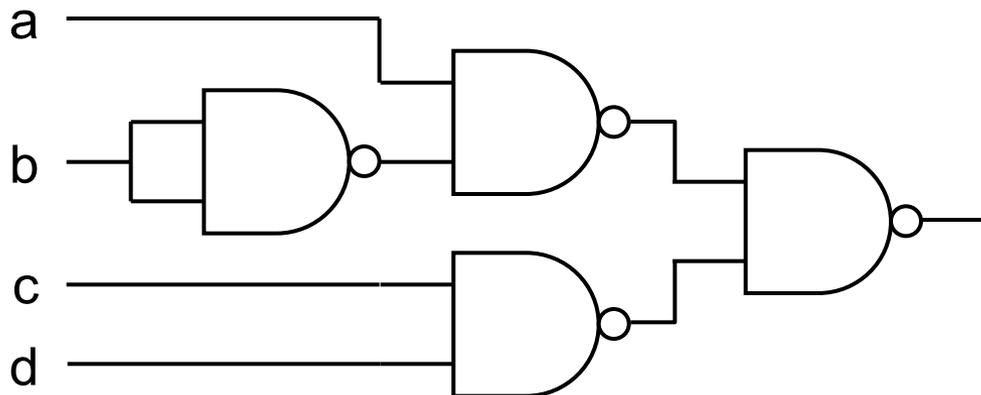


Conjuntos completos

- 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

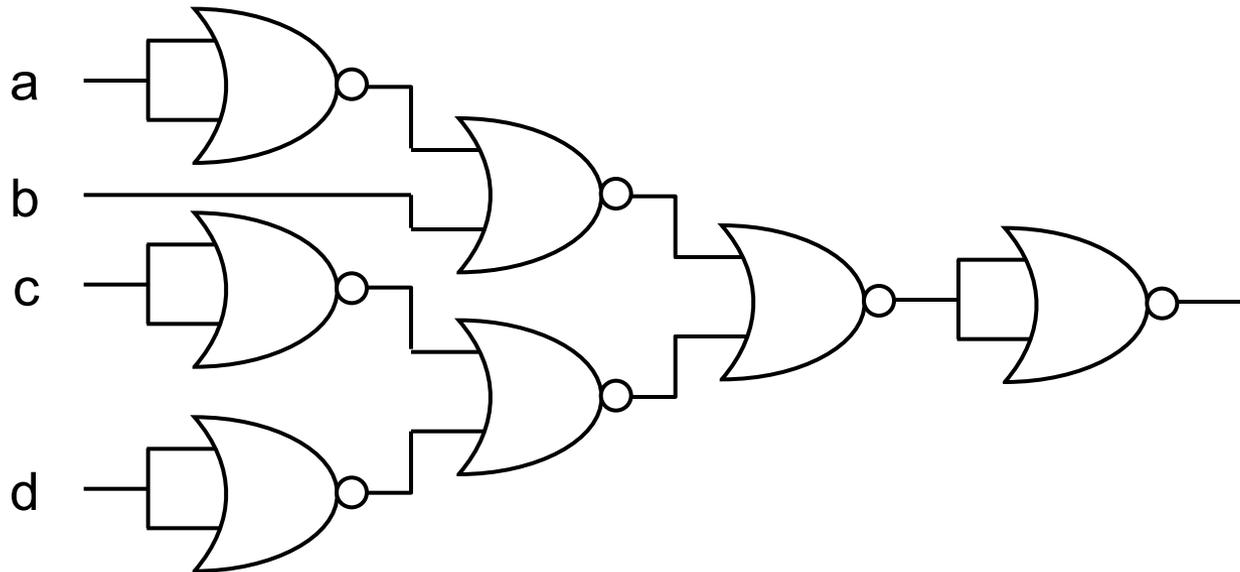
- Aplicación directa de las leyes de De Morgan
- Ejemplo: $f(a,b,c) = a\bar{b} + cd = \overline{\overline{a\bar{b}} + \overline{cd}} = \overline{\overline{a} \cdot b + \overline{cd}} = \overline{\overline{a} \cdot b} \cdot \overline{\overline{cd}} = \overline{\overline{a} \cdot b} \cdot cd$



Realización de circuitos con puertas NOR

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

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



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
 - $\forall E, x \in B \Rightarrow Ex + E\bar{x} = E(x + \bar{x}) = E$
 $(E + x)(E + \bar{x}) = E + (x \cdot \bar{x}) = E$ (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:

$$\begin{aligned}
 f(a,b,c) &= \sum_3 (0,1,2,3,7) = \bar{a}\bar{b}\bar{c} + \bar{a}b\bar{c} + a\bar{b}\bar{c} + \bar{a}bc + abc = \\
 &= \underbrace{\bar{a}\bar{b}\bar{c} + \bar{a}b\bar{c}}_{\bar{a}\bar{b}} + \underbrace{a\bar{b}\bar{c} + \bar{a}bc}_{\bar{a}b} + bc = \\
 &= \bar{a} + bc
 \end{aligned}$$

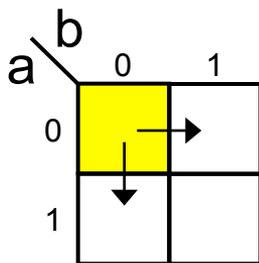
- La observación de las adyacencias puede ser difícil en la práctica

Mapas de Karnaugh

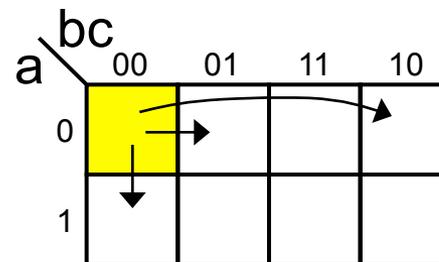
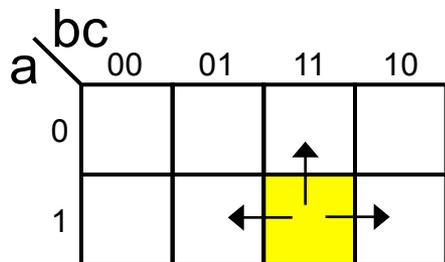
- Mapa que presenta la tabla de verdad de una función de manera que los términos adyacentes son contiguos:
 - Una casilla para cada combinación o término
 - Las casillas se numeran en código Gray
 - En un mapa de n variables, cada casilla tiene n casillas adyacentes que se corresponden con las combinaciones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

Mapas de Karnaugh: adyacencias

- Dos variables

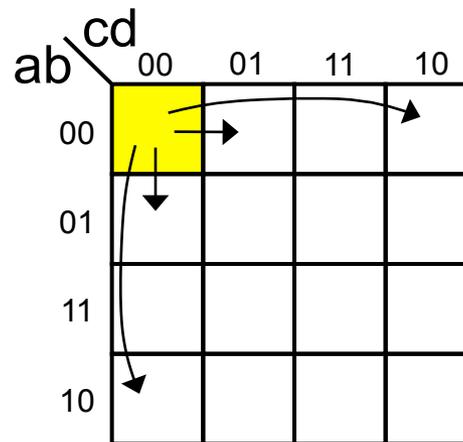
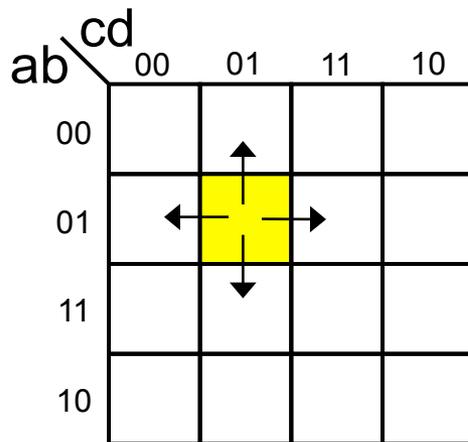


- Tres variables



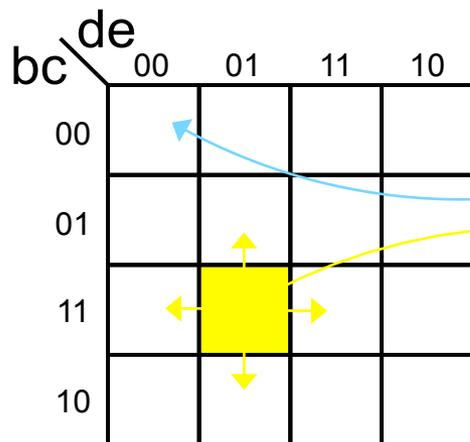
Mapas de Karnaugh: adyacencias

- Cuatro variables

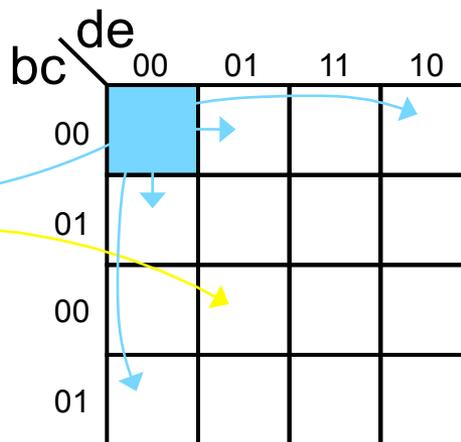


Mapas de Karnaugh: adyacencias

- Cinco variables



$a = 0$



$a = 1$

Mapas de Karnaugh: numeración de las casillas

- Dos variables

	b	0	1
a	0	0	1
	1	2	3

- Cuatro variables

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

- Tres variables

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

Mapas de Karnaugh: numeración de las casillas

- Cinco variables

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

$a = 0$

	de			
bc	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 mintermos o los maxtérminos de la función
- Ejemplo:

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

$$= \prod_3 (4,5,6)$$



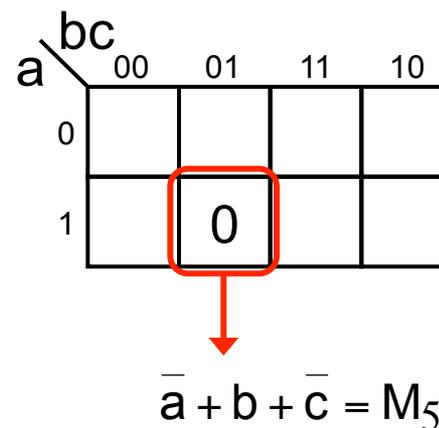
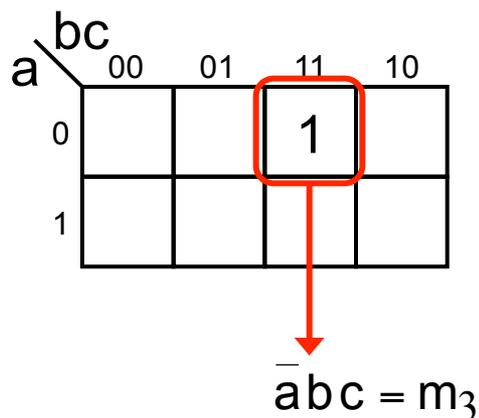
		bc			
		00	01	11	10
a	0	1	1	1	1
	1			1	



		bc			
		00	01	11	10
a	0				
	1	0	0		0

Obtención de una expresión a partir del Mapa de Karnaugh

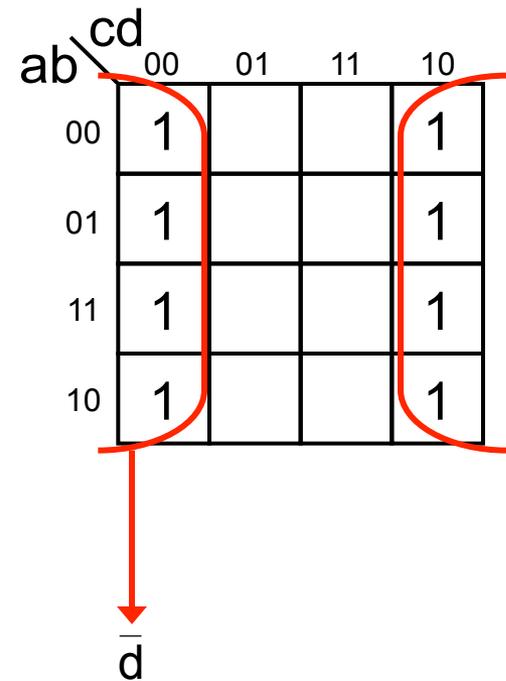
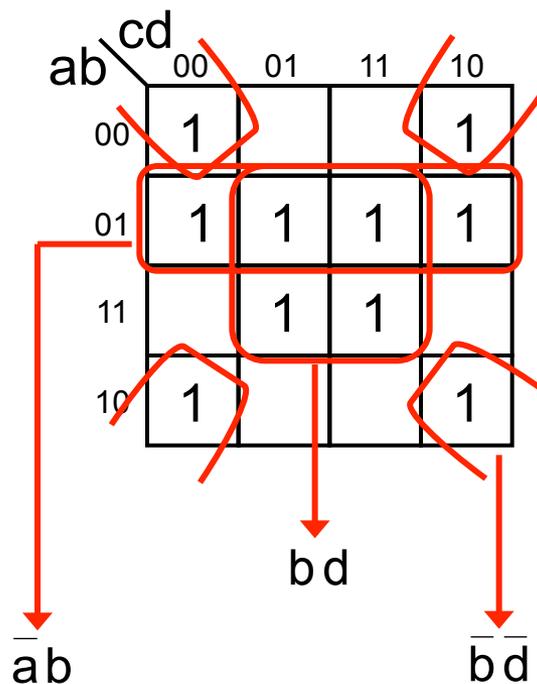
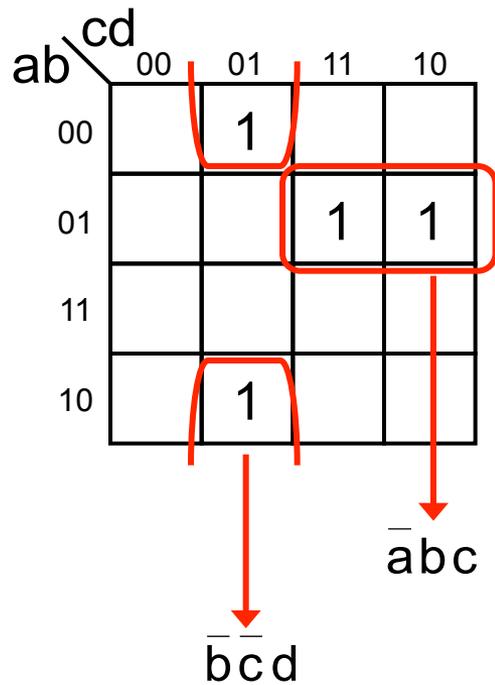
- Se siguen las reglas para mintérminos y maxtérminos
 - Regla para mintérminos
 - 0 → variable negada
 - 1 → variable sin negar
 - Regla para maxtérminos
 - 0 → variable sin negar
 - 1 → variable negada



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

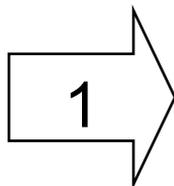


Simplificación mediante Mapas de Karnaugh: Algoritmo

- 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

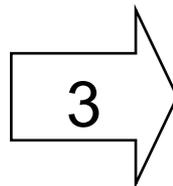
		cd			
		00	01	11	10
ab	00		1		
	01			1	1
	11			1	1
	10			1	



		cd			
		00	01	11	10
ab	00		1		
	01			1	1
	11			1	1
	10			1	



		cd			
		00	01	11	10
ab	00		1		
	01			1	1
	11			1	1
	10			1	



		cd			
		00	01	11	10
ab	00		1		
	01			1	1
	11			1	1
	10			1	

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

$$f(b_3, b_2, b_1, b_0) = \sum_4 (1,3,5,7,9) + \Delta_4(10,11,12,13,14,15) =$$

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

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

	b_1b_0			
b_3b_2	00	01	11	10
00		1	1	
01		1	1	
11	X	X	X	X
10		1	X	X

$$f(b_3, b_2, b_1, b_0) = \bar{b}_3 b_0 + \bar{b}_2 \bar{b}_1 b_0$$

	b_1b_0			
b_3b_2	00	01	11	10
00		1	1	
01		1	1	
11	X	X	X	X
10		1	X	X

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

✓ *Correcto*

Funciones múltiples

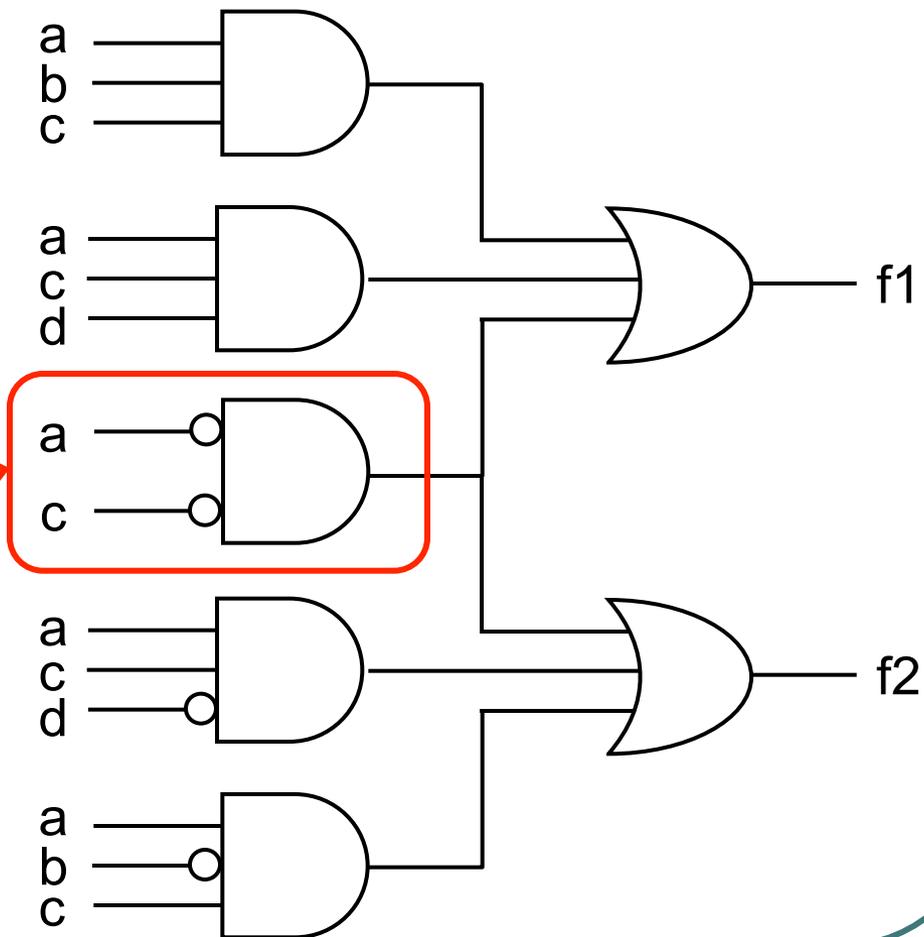
- En los circuitos digitales se implementan generalmente funciones múltiples: varias funciones a la vez o una función de múltiples salidas
- Las funciones múltiples se pueden implementar de forma óptima al considerarlas conjuntamente
 - Se pueden compartir términos o partes comunes para ahorrar lógica
- 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

$$f_1(a,b,c,d) = \bar{a}\bar{c} + abc + acd$$

$$f_2(a,b,c,d) = \bar{a}\bar{c} + a\bar{b}c + ac\bar{d}$$

✓ *Términos comunes*



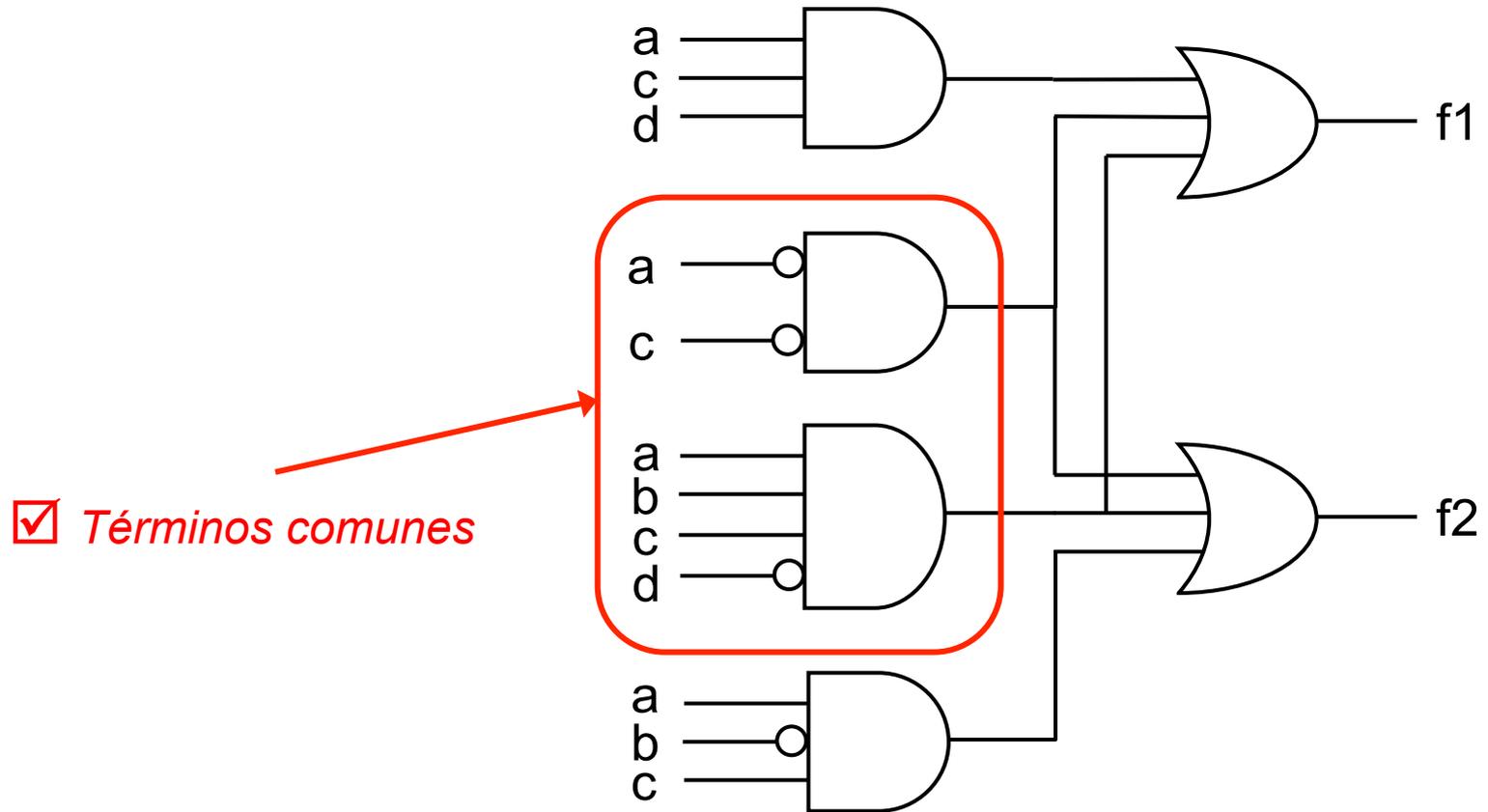
Funciones múltiples: Ejemplo

- Es posible encontrar más términos comunes

$$f_1(a,b,c,d) = \overline{a}\overline{c} + abc + acd = \overline{a}\overline{c} + abcd\overline{d} + acd$$
$$f_2(a,b,c,d) = \overline{a}\overline{c} + a\overline{b}c + ac\overline{d} = \overline{a}\overline{c} + abcd\overline{d} + a\overline{b}c$$

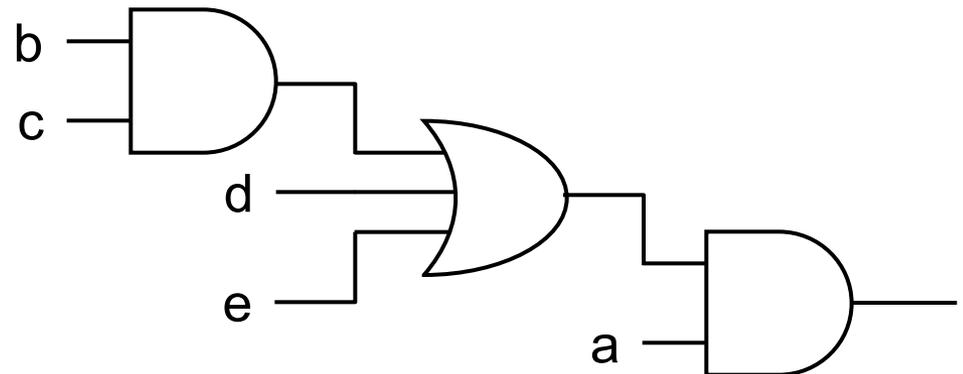
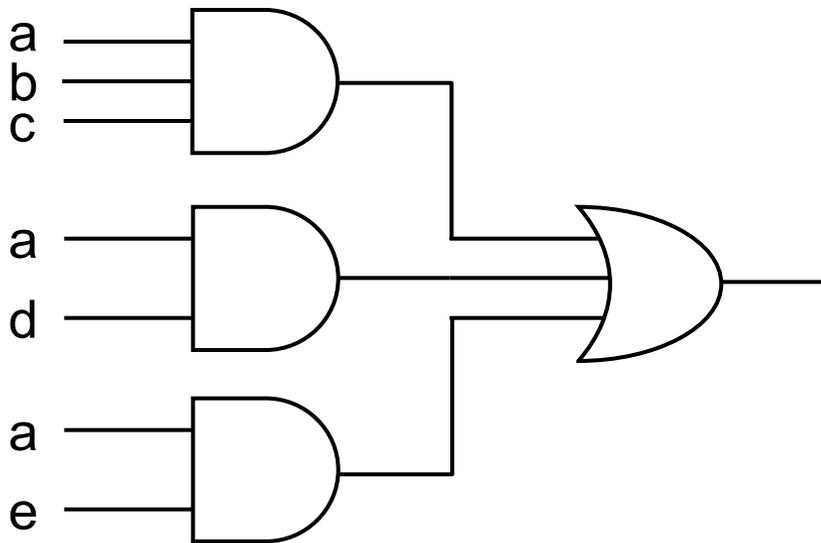
- Las expresiones de las funciones no son óptimas por separado, pero sí son óptimas en conjunto!
- Las herramientas de diseño incluyen algoritmos para minimizar funciones múltiples

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: $f(a,b,c,d,e) = abc + ad + ae = a(bc + d + e)$



✓ *Multinivel*

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