

# **Tema1 . Introducción a la asignatura Electrónica Digital**

## **Grado de robótica - USC**

**Prof. Juan J. Pombo García**

# Contenidos

Tema 1: Introducción al procesamiento digital de la información

Tema 2: Puertas lógicas y su implementación física: familias lógicas

Tema 3: Lenguajes de descripción hardware orientados a la síntesis lógica

Tema 4: Lógica combinacional

Tema 5: Lógica combinacional programable

Tema 6: Lógica secuencial

Tema 7: Memorias RAM y CAM

Tema 8: Lógica secuencial programable

## Competencias fundamentales

A final de curso deberíamos ser capaces de:

- Comprender y aplicar con claridad aspectos de lógica y álgebra Booleana (fundamental)
- Manejar las representaciones numéricas digitales y entender y saber utilizar los sistemas de cálculo que se usan en un circuito digital
- Diseñar y construir sistemas electrónicos digitales de complejidad media, para dar solución a problemas de procesamiento de señales
- Entender y diseñar máquinas de estados digitales con diferentes tecnologías.
- Comprender las bases de funcionamiento de un computador digital.

## **Bibliografía básica y complementaria**

### **Bibliografía Básica:**

Floyd. Thomas L. Fundamentos de sistemas digitales. 11ª edición.  
Pearson-Prentice Hall, Madrid 2016

J. Mira, A.E. Delgado, S. Dormido, M.A. Canto. Electrónica digital.  
Editorial Sainz y Torres, S.L., 2001.

### **Bibliografía en inglés:**

Floyd. Thomas L. Digital Fundamentals. 11ª edición. Pearson  
Educational Limited.

# Evaluación de la asignatura

Prácticas: **15%** de la nota final. Carácter obligatorio.

Tareas entregables: **25%** de la nota final. Será obligatorio entregar el 80% de las asignación con una calidad aceptable, que demuestre que fueron realizadas de manera personal, no fraudulenta, y con un nivel de desarrollo mínimo para ser considerados.

Examen final: Será presencial y supondrá el **60%** de la nota final

# **Tema 1: Introducción al procesamiento Digital de la Información**

- **Introducción general**
- **Variables y operadores lógicos**
- **Funciones lógicas: formas canónicas**
- **Representaciones completas con operadores NAND y NOR**
- **Minimización de funciones**

# Introducción

La electrónica ha cambiado nuestras vidas en el último siglo. Pero la electrónica en sí también ha cambiado mucho en las últimas décadas.

Son muchas las facetas de la electrónica: electrónica analógica, de potencia, digital ...

La electrónica digital es un **subconjunto** de TODO lo que se puede hacer de forma electrónica, pero a lo largo de las últimas décadas ha avanzado hasta convertirse en la solución electrónica más cotidiana.

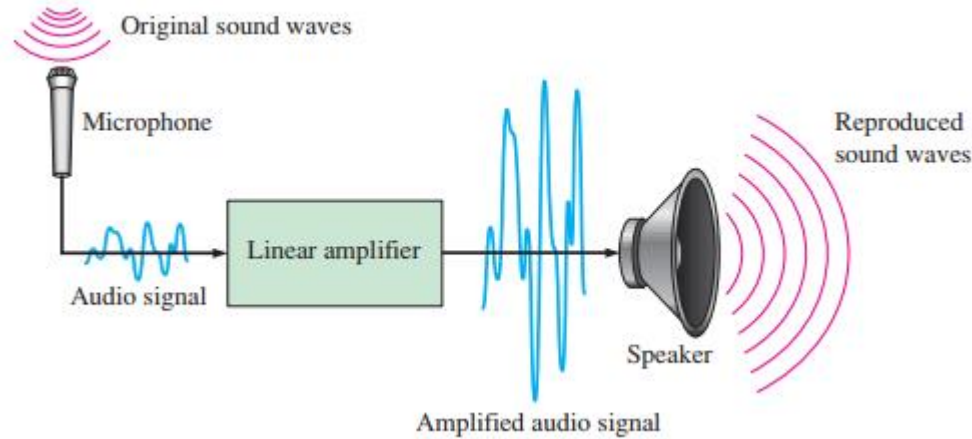
La tecnología digital se aplica a multitud de escenarios hoy en día: multimedia, comunicaciones, sistemas de navegación, control de maquinaria y procesos industriales, instrumental médico, ....

La característica fundamental que la define es que cualquier representación de un dato, o una magnitud está representada por un dígito, o una combinación de dígitos, 0's y 1's, que a nivel electrónico son especialmente **sencillos** de manejar.

Por ejemplo: un transistor bipolar que opera en región de saturación o corte, cambiando de forma eficiente según su polarización.

# Introducción. Mundo analógico vs digital

Un caso de electrónica analógica



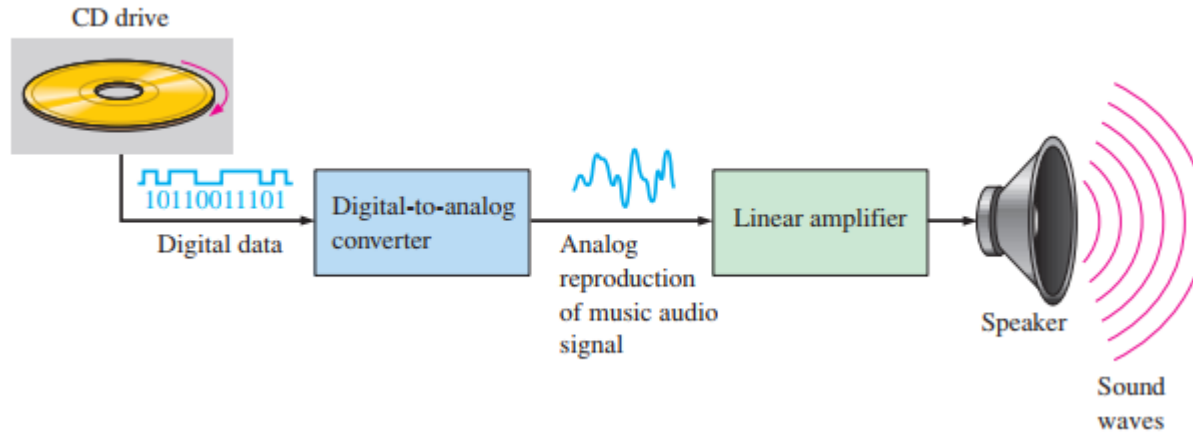
¡No es necesaria ninguna transformación al mundo digital para hacer esto!



# Introducción. Mundo analógico vs digital

El caso más común actualmente:

La electrónica digital y la analógica se integran para dar una solución conjunta.



# Introducción. Situémonos:

## El mundo digital y el analógico

Todas las señales son analógicas en el mundo real (sonido, presión, potencia de una onda de radio,... )

Señal física -> sensor/transductor -> magnitud eléctrica (ej. voltaje)

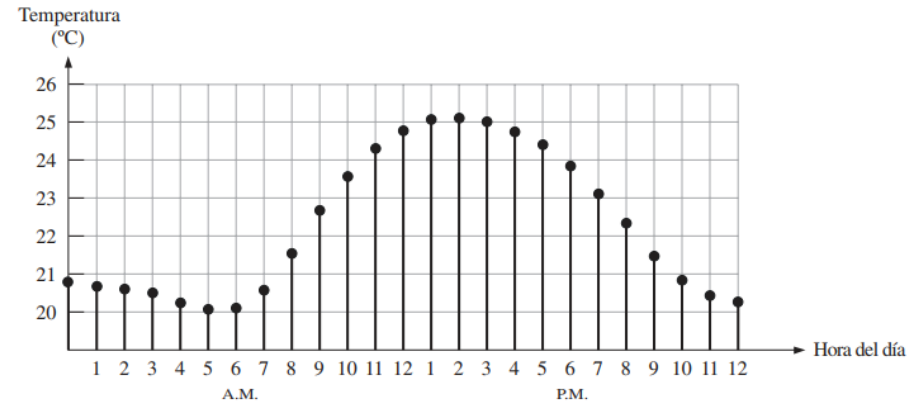
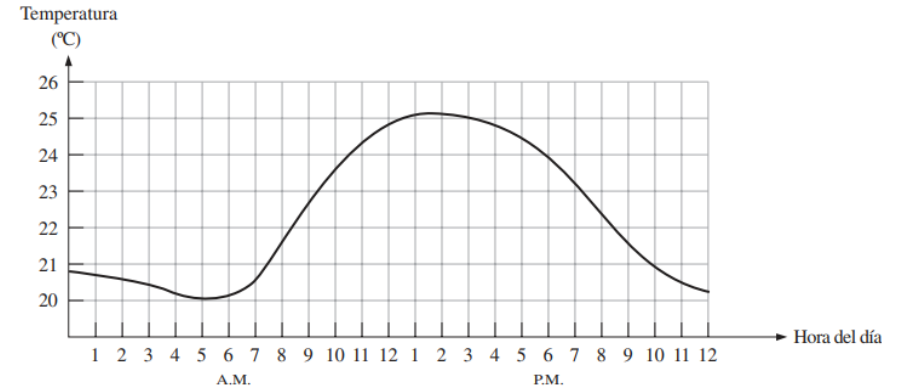
Ejemplo:

Sensado de temperatura.

Digitalización consiste en:

- 1) “Discretizar” el instante de tiempo al que corresponden
- 2) “Digitalizar” el valor de las temperaturas

**¡En las figuras a la derecha sólo está representado uno de los 2 procesos!**



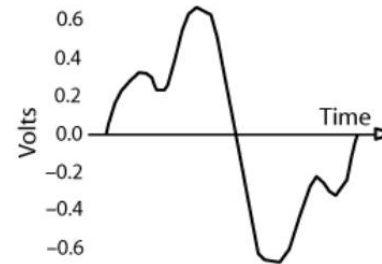
# Introducción. Muestreo.

Ejemplo aplicado al sonido

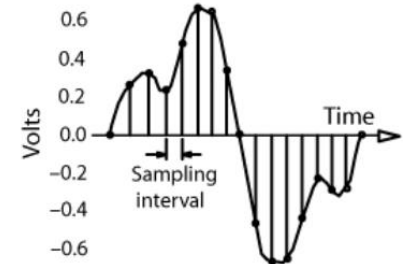
Típica frecuencia de muestreo = 44 kHz

Fases:

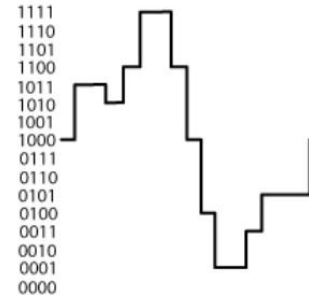
- 1) En el caso del sonido normalmente el proceso comienza con un filtro “pasa-baja” que elimina las frecuencias >22 kHz (filtro *anti-aliasing*)
- 2) Conversión ADC
- 3) Codificación digital de cada muestra
- 4) Grabación del resultado en el medio digital: memoria RAM, disco duro, flash memory, CD/DVD, ...



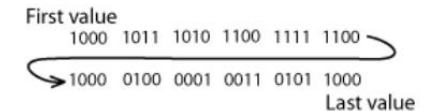
(A) The audio waveform enters the A/D converter.



(B) The voltage is measured or sampled at regular intervals.



(C) The voltage measurements are converted to binary numbers.



(D) The numbers are stored on the recording medium.

# Introducción

Adquisición del dato analógico.

2 fases:

- Muestreo digital (“*sampling*”)
- Cuantificación (“*quantization*”)



Pérdidas de información:

En el muestreo **no tiene por qué haber pérdida de información** si la frecuencia es suficiente (teorema de Nyquist-Shannon). El teorema demuestra y determina la frecuencia de muestreo necesaria para poder reconstruir posteriormente una señal con total precisión (**Teorema:** cualquier señal limitada en banda de frecuencia puede ser representada y es matemáticamente reconstruible con total precisión si fue muestreada al menos al doble del ancho de banda que abarca la señal original)

En la cuantificación sí que puede haber pérdidas de información irreversibles, porque, en realidad, se está asignando a una determinada magnitud el código más próximo de la tabla de códigos. (El teorema de Shannon asume precisión infinita en la representación digital, el teorema solo habla de la parte de muestreo desde el punto de vista matemático)

En el almacenamiento de información, si es digital, no hay pérdidas. En la transmisión, se pueden evitar.

# Introducción. Muestreo.

Selección de las características del muestreo.

Son varias las restricciones que aparecen:

Para solucionar, o al menos minimizar, cualquier error de cuantificación podríamos introducir codificaciones de **altísima resolución**.

Pero introducir esta resolución nos influiría en:

Complejidad de computación de la representación, y por tanto:

Tiempo mínimo entre cada muestra

**Máximo “sampling rate”**

**Mayor coste**

**Mayor consumo de potencia ,....**

Solución ideal: compromiso entre todos estos factores.

La electrónica asociada a esta aplicación está en constante evolución todavía hoy en día.

# Introducción

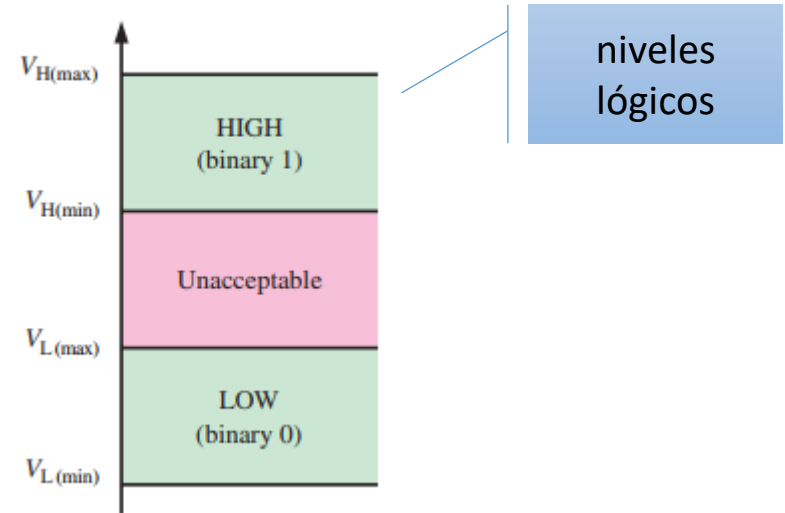
- Algunas de las ventajas de los sistemas Digitales:
  - Mayor facilidad de almacenamiento de la información.
  - Facilidad de transmisión de la información
  - Los datos se pueden replicar y procesar con mayor facilidad.
  - Se puede trabajar con gran precisión y fiabilidad
  - Los circuitos integrados son “fáciles” de fabricar. Más económicos.
  - Los sistemas son relativamente fáciles de escalar.

# Introducción. El código digital.

Para el manejo por un computador de los valores discretos usamos codificaciones que llamamos “binarias” porque sólo utilizan 2 dígitos (*bits*)

A nivel electrónico lo que se usa es 2 niveles de tensión de una señal analógica pero de forma “umbralizada”, es decir sólo se considerará estar en un de 2 estados : el alto, o el bajo (HIGH/LOW), el 0 o el 1, el TRUE/FALSE.

El trabajar con 2 niveles de esta manera tiene aspectos interesantes desde el punto de vista de la ingeniería de construcción de circuitos: uno de ellos es la inmunidad al ruido eléctrico.

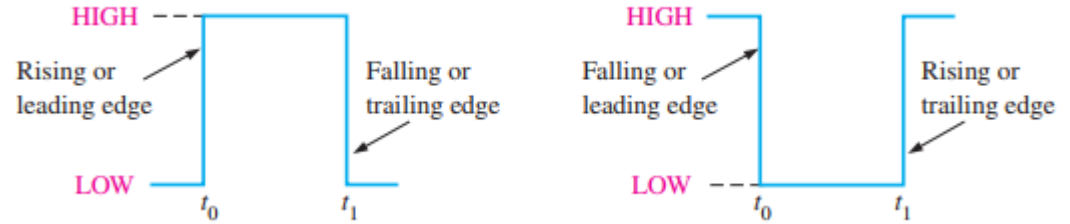


Otros conceptos:  
lógica negativa  
*switching logic*

# Introducción

Transmisión digital de la información

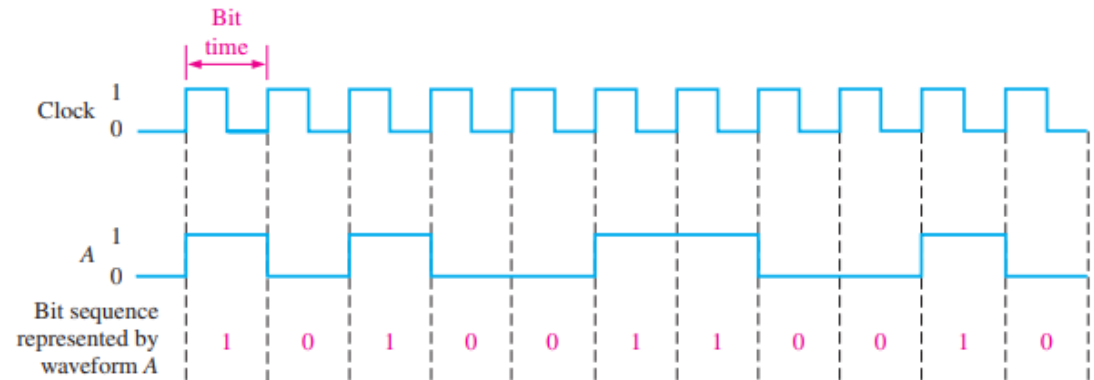
Tren de pulsos (tensión o corriente)



¿Si quiero transmitir 2 “0’s” / 2 “1’s” consecutivos?

Falta algo más para que el dato transmitido se pueda recuperar: la señal de reloj

Esto introduce el concepto de “período de bit”



cronograma

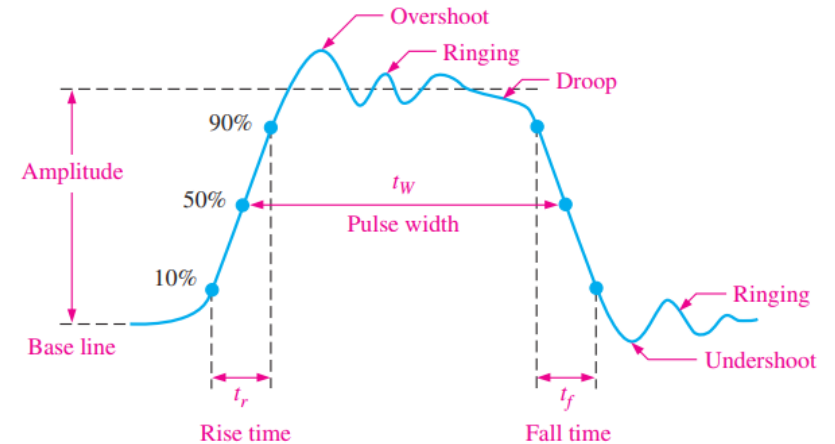


# Introducción

La transmisión nunca es tan ideal: los circuitos reales no son puramente resistivos, tienen comportamientos derivados de las componentes capacitivas e inductivas.

Aparecen fenómenos como:

sobreimpulso, rizado, tiempos de subida de flanco, tiempos de bajada, ....



# Introducción. Sistemas de numeración.

Muchas aplicaciones las podremos resolver únicamente utilizando estados codificables por 1 bit (abierto/cerrado , encendido/apagado, detecta/no detecta, ... ),

pero en general necesitaremos representar las magnitudes “analógicas” de la vida real en el mundo digital, o bien representar 1 estado de N posibles,  $N \neq 2$ .

Necesitamos algo más:

Los valores numéricos se **codifican** para su manejo computacional mediante códigos binarios.

Dedicaremos una parte de nuestro tiempo a lo largo del curso a este aspecto:

- Diferentes codificaciones de números enteros
- Codificaciones para números flotantes
- Codificaciones especiales para limitar errores durante las transmisiones de datos
- etc...

# Introducción. Sistemas de numeración.

Sistema de numeración decimal (base 10 / radix-10)

Ej: 
$$3586 = 6 \times 10^0 + 8 \times 10^1 + 5 \times 10^2 + 3 \times 10^3 = 6 + 80 + 500 + 3000 = 3586$$

Ej2: 3586.265

Parte entera como la de arriba, parte fraccionaria:

$$265 = 2 \times 10^{-1} + 6 \times 10^{-2} + 5 \times 10^{-3} = 0.2 + 0.06 + 0.005 = 0.265$$

El sistema de numeración binario estándar funciona de manera análoga aunque es un sistema de numeración radix-2 / base 2

Por supuesto aquí quedan cosas por codificar, números fraccionarios, números negativos, números de “coma flotante”.

En próximos temas abordaremos estos aspectos con mayor detalle.

Al igual que en el caso binario se ha ido desarrollando un **completísimo y altamente eficiente** sistema aritmético para esta representación numérica.

*"Sólo hay 10 tipos de personas en el mundo: Los que entienden binario y los que no lo hacen."*

decimal	binary	conversion
0	0	$0 (2^0)$
1	1	$1 (2^0)$
2	10	$1 (2^1) + 0 (2^0)$
3	11	$1 (2^1) + 1 (2^0)$
4	100	$1 (2^2) + 0 (2^1) + 0 (2^0)$
5	101	$1 (2^2) + 0 (2^1) + 1 (2^0)$
6	110	$1 (2^2) + 1 (2^1) + 0 (2^0)$
7	111	$1 (2^2) + 1 (2^1) + 1 (2^0)$
8	1000	$1 (2^3) + 0 (2^2) + 0 (2^1) + 0 (2^0)$
9	1001	$1 (2^3) + 0 (2^2) + 0 (2^1) + 1 (2^0)$
10	1010	$1 (2^3) + 0 (2^2) + 1 (2^1) + 0 (2^0)$

# Introducción. De los semiconductores a los computadores.

La electrónica digital se puede abordar en diversos niveles de complejidad. A lo largo de este curso abordaremos un poco de cada uno de ellos:

- Los transistores son fabricados a partir de materiales semiconductores organizados de determinadas maneras
- Las puertas lógicas se construyen a partir de transistores
- Las funciones lógicas se construyen a partir de puertas lógicas
- Los Flip-flops se construyen a partir de lógica
- Los circuitos secuenciales y los contadores se desarrollan basándose en los flip-flops
- Los microprocesadores se diseñan a partir de aglutinar circuitos secuenciales
- Los computadores y controles se basan en los microprocesadores

Número de transistores en Intel Core i7 6 núcleos: ¡731 millones!

# Variables y operadores lógicos

## Puertas lógicas:

- Las puertas lógicas son los bloques básicos de los circuitos digitales.
- La forma en la que operan se formaliza a través del álgebra Booleana

## Variables lógicas:

Variables que toman 2 valores: TRUE/FALSE , ON/OFF, 1/0

## Expresiones lógicas:

Combinaciones de variables lógicas y operadores lógicos

## *Un ejemplo:*

Encender una caldera de gas:

Variables: b=bloqueo de chimenea

c=termostato indica frío

p=llama piloto

v=comando abrir válvula de gas

$$v = (\text{NOT } b) \text{ AND } c \text{ AND } p$$

Si no hay bloqueo y la casa está fría y la llama piloto está encendida -> abrir la válvula del gas

# Variables y operadores lógicos

## Representación de las funciones lógicas.

### Varios modos:

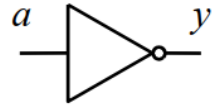
- Símbolos para las puertas
- Tablas de verdad
- Algebra Booleana (George Boole, lógico y matemático Irlandés, aprox. 1850)

La idea con la que Boole inició su trabajo fue poder representar simbólicamente razonamientos lógicos y poder operar con ellos con un formalismo matemático similar al de cualquier otra álgebra, (una idea extraña para la época). Hasta mediados del siglo XX no se le encontró una aplicación práctica real. Pero después fue la base para los trabajos de Shannon en codificación y John von Neumann en arquitectura de computadores.

A continuación mostraremos la distinta forma para los operadores lógicos más comunes y sus puertas lógicas correspondientes.

# Operadores lógicos: puerta NOT

Symbol



Truth-table

$a$	$y$
0	1
1	0

Boolean

$$y = \bar{a}$$

También llamada puerta inversora/inversor

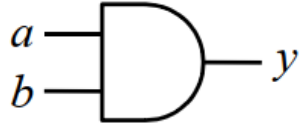
El circulito colocado a la salida de una función es indicador de que la función es la invertida (“complementaria”)

Propiedad básica:  $\bar{\bar{A}} = A$

0 = LOW = FALSE  
1 = HIGH = TRUE

# Operadores lógicos: puerta AND

Symbol



Truth-table

<i>a</i>	<i>b</i>	<i>y</i>
0	0	0
0	1	0
1	0	0
1	1	1

Boolean

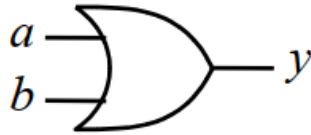
$$y = a.b$$

Operador “.” en álgebra Booleana.



# Operadores lógicos: puerta OR

Symbol



Truth-table

<i>a</i>	<i>b</i>	<i>y</i>
0	0	0
0	1	1
1	0	1
1	1	1

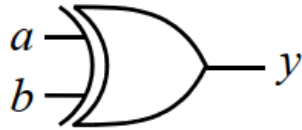
Boolean

$$y = a + b$$

Operador “+” en álgebra Booleana

# Operadores lógicos: puerta OR EXCLUSIVO (XOR)

Symbol



Truth-table

<i>a</i>	<i>b</i>	<i>y</i>
0	0	0
0	1	1
1	0	1
1	1	0

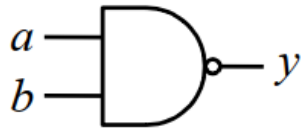
Boolean

$$y = a \oplus b$$

Operador “ $\oplus$ ” en álgebra Booleana

# Operadores lógicos: puerta NAND

Symbol



Truth-table

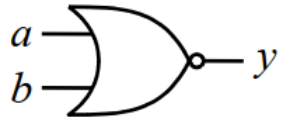
<i>a</i>	<i>b</i>	<i>y</i>
0	0	1
0	1	1
1	0	1
1	1	0

Boolean

$$y = \overline{a.b}$$

# Operadores lógicos: puerta NOR

Symbol



Truth-table

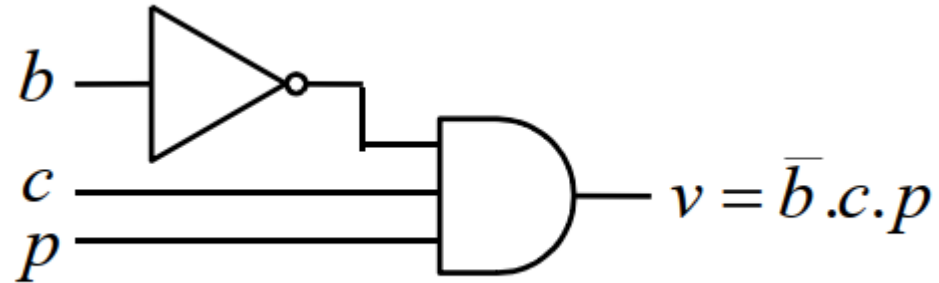
$a$	$b$	$y$
0	0	1
0	1	0
1	0	0
1	1	0

Boolean

$$y = \overline{a + b}$$

## Ejemplo anterior

Si no hay bloqueo y la casa está fría y la llama piloto está encendida -> abrir la válvula del gas



Las puertas lógicas se conectan para formar un circuito "combinacional"

# Álgebra Booleana

Introduciremos las leyes básicas de álgebra Booleana.

A partir de aquí podremos diseñar el primer tipo de circuitos digitales que analizaremos: los *circuitos combinacionales*

Circuito combinacional:

Aquel que no guarda estados previos. La salida es, en todo momento, únicamente función de las entradas.

# Álgebra Booleana. Formas básicas

Algunas operaciones básicas:

OR

$$a + 0 = a$$

$$a + a = a$$

$$a + 1 = 1$$

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

AND

$$a \cdot 0 = 0$$

$$a \cdot a = a$$

$$a \cdot 1 = a$$

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

Precedencia de operadores: AND precede a OR

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

# Álgebra Booleana. Propiedades

Propiedades básicas del álgebra de Boole:

Conmutativa  $a + b = b + a$   
 $a \cdot b = b \cdot a$

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

¡NOVEDAD!

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

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



# Álgebra Booleana. Propiedades

Propiedades básicas del álgebra de Bool (II)

Absorción (*absorption rule*)

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

¡NOVEDAD!

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

¡NOVEDAD!

Demostración:

$$\begin{aligned} \overset{a}{a} \cdot 1 + a \cdot c &= (\text{distrib.} \rightarrow) \\ a \cdot (\underbrace{1 + c}_{=1}) &= a \end{aligned}$$

$$\begin{aligned} \overset{a}{a} + (0 \cdot c) &= \\ \underline{a} + (\underbrace{0 \cdot c}_{=0}) &= a \end{aligned}$$

# Álgebra Booleana. Ejemplos

Demostrar las siguientes expresiones:

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

$$\cancel{a} \cdot \bar{\cancel{a}} + \underline{a} \underline{b} = a \cdot b$$

Handwritten red proof for  $a(\bar{a} + b) = a \cdot b$ . The expression  $a\bar{a} + ab$  is shown. A red line is drawn through  $a\bar{a}$ , and a red circle is drawn around  $\bar{a}$  with a red '0' below it, indicating that  $a\bar{a} = 0$ . The term  $ab$  is underlined, and the final result  $a \cdot b$  is also underlined.

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

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

Handwritten red proof for  $a + (\bar{a} \cdot b) = a + b$ . The expression  $(a + \bar{a}) \cdot (a + b)$  is shown. A red bracket is drawn under  $a + \bar{a}$  with the text "X OR x" written below it. A red line is drawn through the bracket, and a red '1' is written below it, indicating that  $a + \bar{a} = 1$ . The final result  $a + b$  is underlined twice.

OR

Truth-table

a	b	y
0	0	0
0	1	1
1	0	1
1	1	1

AND

Truth-table

a	b	y
0	0	0
0	1	0
1	0	0
1	1	1

# Álgebra Booleana. Ejemplos de operación.

Muy a menudo se pueden arreglar expresiones expandiendo cada uno de los términos hasta que todos los términos contienen una estancia de cada variable (o su complemento)

Vamos a usarlo para probar por otra vía la regla de absorción:

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

Expansión:  $a \cdot c + a \cdot \bar{c} + \cancel{a \cdot c} = a \cdot c + a \cdot \bar{c}$

$= a \cdot (\underbrace{c + \bar{c}}_1) = a$

*ya tengo este término No aporta nada*

OR

Truth-table

a	b	y
0	0	0
0	1	1
1	0	1
1	1	1

AND

Truth-table

a	b	y
0	0	0
0	1	0
1	0	0
1	1	1

# Álgebra Booleana. Ejemplos

Simplificar:

$$x \cdot y + \bar{y} \cdot z + x \cdot z + x \cdot y \cdot z$$

$$x \cdot y \cdot \bar{z} + x \cdot y \cdot \bar{\bar{z}} + \bar{x} \cdot \bar{y} \cdot z + \bar{\bar{x}} \cdot \bar{y} \cdot z + \cancel{x \cdot y \cdot z} + \cancel{x \cdot \bar{y} \cdot z} + \cancel{x \cdot y \cdot z}$$

$$= \underbrace{x \cdot y \cdot \bar{z}} + \underbrace{x \cdot y \cdot \bar{\bar{z}}} + \underbrace{x \cdot \bar{y} \cdot z} + \underbrace{\bar{x} \cdot \bar{y} \cdot z} =$$

$$= x \cdot y \cdot (\underbrace{\bar{z} + \bar{\bar{z}}}_{1}) + (\underbrace{x + \bar{x}}_{1}) \cdot (\bar{y} \cdot z) =$$

$$= \boxed{x \cdot y + \bar{y} \cdot z}$$

OR

Truth-table

a	b	y
0	0	0
0	1	1
1	0	1
1	1	1

AND

Truth-table

a	b	y
0	0	0
0	1	0
1	0	0
1	1	1

# Álgebra Booleana. Teorema de DeMorgan

2 formas:  $\overline{a + b + c + \dots} = \bar{a} \cdot \bar{b} \cdot \bar{c} \dots$

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

Muy útil: permite cambiar los operadores de OR a AND, o viceversa.

Cuando la expresión de partida no está negada se puede utilizar de esta otra manera:

$$\begin{aligned} a + b + c + \dots &= \overline{\bar{a} \cdot \bar{b} \cdot \bar{c} \dots} \\ a \cdot b \cdot c \dots &= \overline{\bar{a} + \bar{b} + \bar{c} + \dots} \end{aligned}$$

Usando  $\overline{\bar{A}} = A$ ,

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

# Álgebra Booleana. Teorema de DeMorgan

Demostración: Por tablas de verdad es obvio lo siguiente:

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

OR

Truth-table

a	b	y
0	0	0
0	1	1
1	0	1
1	1	1

AND

Truth-table

a	b	y
0	0	0
0	1	0
1	0	0
1	1	1

Por tanto para 2 variables el teorema se cumple. Para más variables se puede comprobar que también sigue funcionando sin más que considerar un caso generalizado de 2 términos:

Ej para 3:  $\overline{(\overline{a+b}) + c} = \overline{x} \cdot \overline{c} = \overline{a+b} \cdot (\overline{c}) = \overline{a} \cdot \overline{b} \cdot \overline{c}$  „ y así sucesivamente para más términos

*(sustit. de nuevo)* *Con 2 ya lo hemos demostrado...*

# Álgebra Booleana. Teorema de DeMorgan

Ejemplo de aplicación.

Simplificar:

$$a \cdot \bar{b} + a \cdot (\overline{b+c}) + b(\overline{b+c})$$

$$\begin{aligned} &= a \cdot \bar{b} + a \cdot (\underbrace{\bar{b} \cdot \bar{c}}_{\text{DeM}}) + b(\underbrace{\bar{b} \cdot \bar{c}}_{\text{DeM}}) = \\ &= \underline{a \cdot \bar{b}} + \underline{a \cdot \bar{b} \cdot \bar{c}} + \cancel{b \cdot \bar{b} \cdot \bar{c}} = a \cdot (\underbrace{\bar{b} + \bar{b} \bar{c}}_{\bar{b}}) = \underline{\underline{a \cdot \bar{b}}} \end{aligned}$$

*Handwritten notes: The term  $b \cdot \bar{b} \cdot \bar{c}$  is crossed out with a large 'X' and labeled '0 AND 0'. The expression  $\bar{b} + \bar{b} \bar{c}$  is circled and labeled 'Absorption'.*

# Álgebra Booleana. Teorema de DeMorgan

Ejemplo de aplicación (II)

Simplificar:

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

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

D.M

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

$$= a \cdot b \cdot \underline{c} \cdot c \cdot d + \cancel{a \cdot b \cdot \overline{b} \cdot c \cdot d} + \cancel{a \cdot b \cdot \overline{d} \cdot c \cdot d} + \overline{a} c d + \overline{b} c d =$$

= 0

$$= a \underline{b} c \underline{d} + \overline{a} c \underline{d} + \overline{b} c \underline{d} = (ab + \overline{a} + \overline{b}) \cdot c \cdot d = \underline{\underline{c \cdot d}}$$

D.M

$$\overline{\overline{a}b}$$

= 1



# Álgebra Booleana.

Hasta aquí hemos desarrollado la formulación matemática básica del álgebra booleana.

Todas las funciones las podemos expresar como combinaciones de los operadores  $+$  y  $\cdot$ .

A continuación comenzaremos a ver cómo esto se integra con la electrónica para poder realizar estas funciones en *hardware*, y así poder operar con señales del mundo físico real.

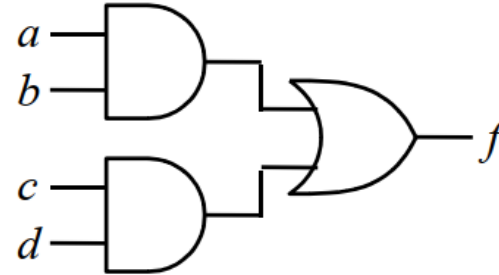
Antes de nada consideraremos un aspecto tecnológico de especial relevancia.

# Álgebra Booleana. Expresar funciones con puertas

Consideremos el siguiente caso de función lógica:

$$f = a.b + c.d$$

Una posible implementación directa sería la siguiente:



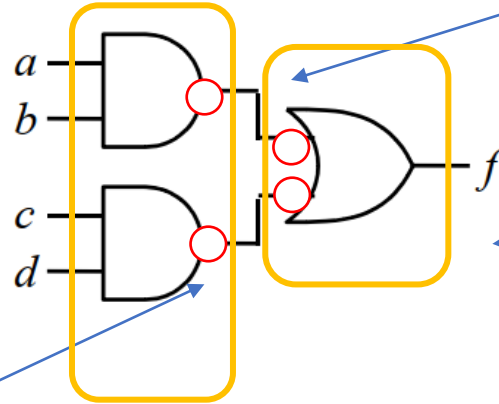
Pero en muchas ocasiones desearemos que el circuito solamente use puertas **NAND** o **NOR**. El motivo es tecnológico, ya que normalmente son más sencillas de implementar y más rápidas.

En estos casos es cuando recurrimos a la aplicación del teorema de **DeMorgan**.

Primero veámoslo de forma gráfica en la siguiente diapositiva.

# Álgebra Booleana. Expresar funciones con puertas NAND

$$f = a.b + c.d$$



2.- Hacemos lo mismo del paso 1 aquí

3.- Estas ya son puertas NAND

1.- Una doble negación como esta no influye en la función lógica

4.- Analicemos esta puerta:

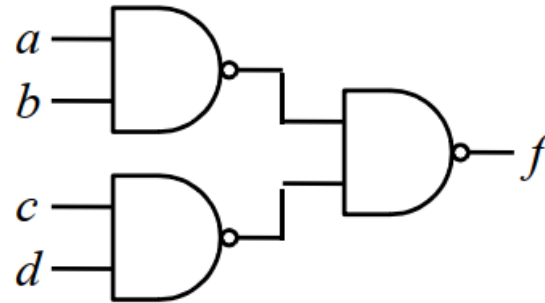
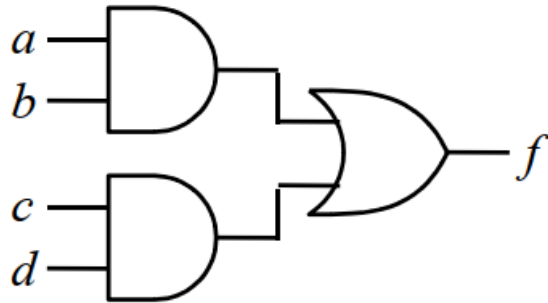
$$\bar{x} + \bar{y} = \overline{xy}$$

Es decir que es equivalente a una NAND



# Álgebra Booleana. Expresar funciones con puertas NAND

$$f = a.b + c.d$$



Ambos circuitos son equivalentes.

La función del ejemplo puede implementarse alternativamente con 3 puertas NAND

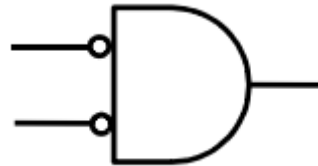
**Cualquier función lógica puede expresarse exclusivamente con puertas NAND**

# Álgebra Booleana. Expresar funciones con puertas NOR

Un análisis similar nos permite encontrar cómo expresar una función lógica compuesta de puertas AND y OR exclusivamente con puertas NOR

Tomo una puerta AND y niego cada una de sus entradas:

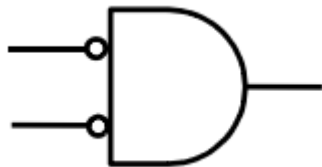
1) Inversor  $a \rightarrow \bar{a}$



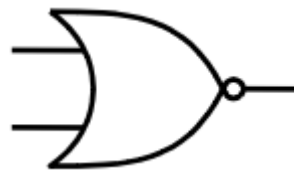
Aplicando de Morgan:

$$\bar{x} \cdot \bar{y} = \overline{x + y}$$

Por tanto:

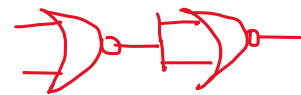
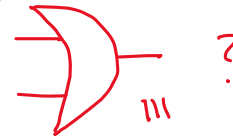


es equivalente a:

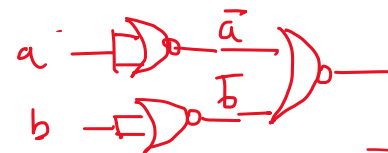


Este resultado me permite construir cualquier expresión lógica con puertas NOR.

2) Puerta OR



3) Puerta AND



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

# Álgebra Booleana. Minimización lógica

En algunos de los ejemplos anteriores hemos visto cómo simplificar una expresión lógica dada.

Esta simplificación llevará a una reducción significativa del número de puertas necesarias para implementar el circuito

Las técnicas más usuales para reducir son:

- Técnicas algebraicas (no siempre se dispone de la expresión de partida)
- Mapas de Karnaugh
- Otras aproximaciones computacionales (usualmente métodos ejecutados por software)

Hasta aprox. 5 variables Karnaugh es un buen método.

# Álgebra Booleana. Minimización lógica

## Minimización mediante mapas de Karnaugh (mapas K )

Tabla de verdad

$x$	$y$	$z$	$f$	minterms
0	0	0	1	$\bar{x}.\bar{y}.\bar{z}$
0	0	1	1	$\bar{x}.\bar{y}.z$
0	1	0	1	$\bar{x}.y.\bar{z}$
0	1	1	1	$\bar{x}.y.z$
1	0	0	0	
1	0	1	0	
1	1	0	0	
1	1	1	1	$x.y.z$

Introducimos el concepto de “minterm” (a veces traducido por ‘minitérmino’):  
Cada una de las operaciones AND entre todas las variables que dan un valor TRUE en la tabla de verdad

En los *minterm* TODAS las variables aparecen (negadas o no)

La función  $f$  acaba expresada como una “**suma de productos**” (SOP)

En este procedimiento no tenemos que partir necesariamente de una expresión lógica para  $f$ .

Forma disyuntiva normalizada:

$$f = \bar{x}.\bar{y}.\bar{z} + \bar{x}.\bar{y}.z + \bar{x}.y.\bar{z} + \bar{x}.y.z + x.y.z$$

# Álgebra Booleana. Minimización lógica

## Minimización mediante mapas de Karnaugh (mapas K )

Tabla de verdad

$x$	$y$	$z$	$f$	minterms
0	0	0	1	$\bar{x}.\bar{y}.\bar{z}$
0	0	1	1	$\bar{x}.\bar{y}.z$
0	1	0	1	$\bar{x}.y.\bar{z}$
0	1	1	1	$\bar{x}.y.z$
1	0	0	0	
1	0	1	0	
1	1	0	0	
1	1	1	1	$x.y.z$

$$f = \bar{x}.\bar{y}.\bar{z} + \bar{x}.\bar{y}.z + \bar{x}.y.\bar{z} + \bar{x}.y.z + x.y.z$$

Estas 2 funciones tienen la misma tabla de verdad: (comprobamos)

$$f = \bar{x} + y.z$$


La idea es obtener un método sistemático para llegar a esta función simplificada.



# Álgebra Booleana. Minimización lógica

## Minimización mediante mapas de Karnaugh (mapas K )

Tabla de verdad

$x$	$y$	$z$	$f$	minterms
0	0	0	1	$\bar{x}.\bar{y}.\bar{z}$
0	0	1	1	$\bar{x}.\bar{y}.z$
0	1	0	1	$\bar{x}.y.\bar{z}$
0	1	1	1	$\bar{x}.y.z$
1	0	0	0	
1	0	1	0	
1	1	0	0	
1	1	1	1	
				$x.y.z$

Alternativamente podríamos buscar una expresión para aquellos casos en los que  $f$  vale FALSE:

$$\bar{f} = x.\bar{y}.\bar{z} + x.\bar{y}.z + x.y.\bar{z}$$

Por De Morgan :

$$f = (\bar{x} + y + z).(\bar{x} + y + \bar{z}).(\bar{x} + \bar{y} + z)$$

A estos se les denomina *maxterms*.

La expresión  $f$  en este caso se dice que está en *Forma Conjuntiva Normalizada (CNF)* o en producto de sumas (POS)

La forma simplificada sería en este caso:

$$f = (\bar{x} + y).(\bar{x} + z)$$

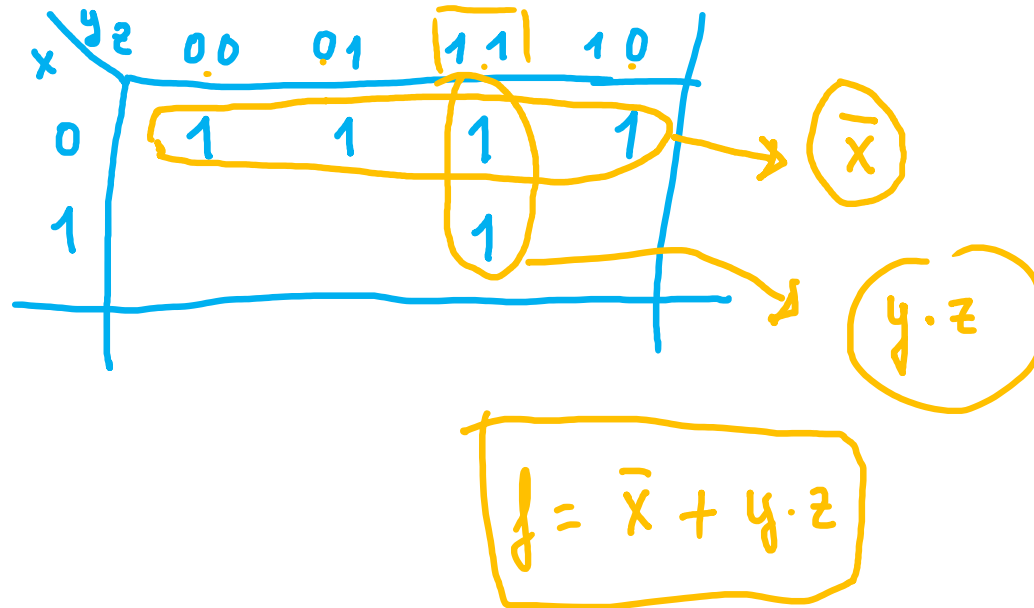
# Álgebra Booleana. Minimización lógica

## Minimización mediante mapas de Karnaugh (mapas K )

Tabla de verdad

$x$	$y$	$z$	$f$	minterms
0	0	0	1	$\bar{x} \cdot \bar{y} \cdot \bar{z}$
0	0	1	1	$\bar{x} \cdot \bar{y} \cdot z$
0	1	0	1	$\bar{x} \cdot y \cdot \bar{z}$
0	1	1	1	$\bar{x} \cdot y \cdot z$
1	0	0	0	
1	0	1	0	
1	1	0	0	
1	1	1	1	$x \cdot y \cdot z$

Los mapas de Karnaugh son básicamente una herramienta visual para simplificar expresiones lógicas.



# Álgebra Booleana. Minimización lógica

## Minimización mediante mapas de Karnaugh (mapas K )

Tabla de verdad

$x$	$y$	$z$	$f$	minterms
0	0	0	1	$\bar{x}.\bar{y}.\bar{z}$
0	0	1	1	$\bar{x}.\bar{y}.z$
0	1	0	1	$\bar{x}.y.\bar{z}$
0	1	1	1	$\bar{x}.y.z$
1	0	0	0	
1	0	1	0	
1	1	0	0	
1	1	1	1	$x.y.z$

Los mapas de Karnaugh son básicamente una herramienta visual para simplificar expresiones lógicas.

		$z$			
		$yz$	00	01	11
$x$	0	1	1	1	1
	1			1	
		$y$			

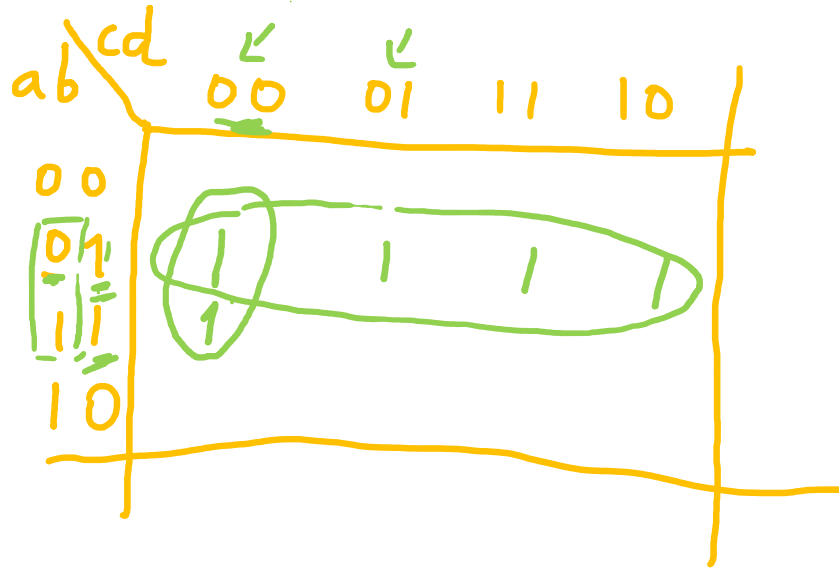
$$f = \bar{x} + y.z$$

# Álgebra Booleana. Minimización lógica

Minimización mediante mapas de Karnaugh (mapas K ). Patrones prototipo.

Obtenemos un mapa de 4 variables para esta expresión:

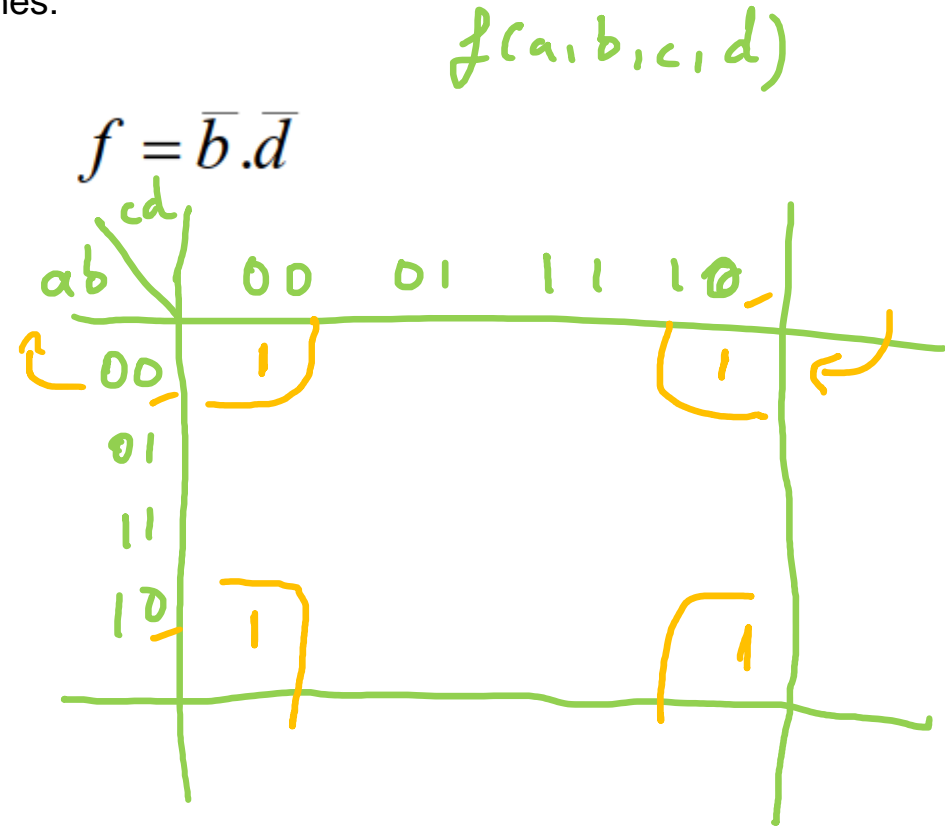
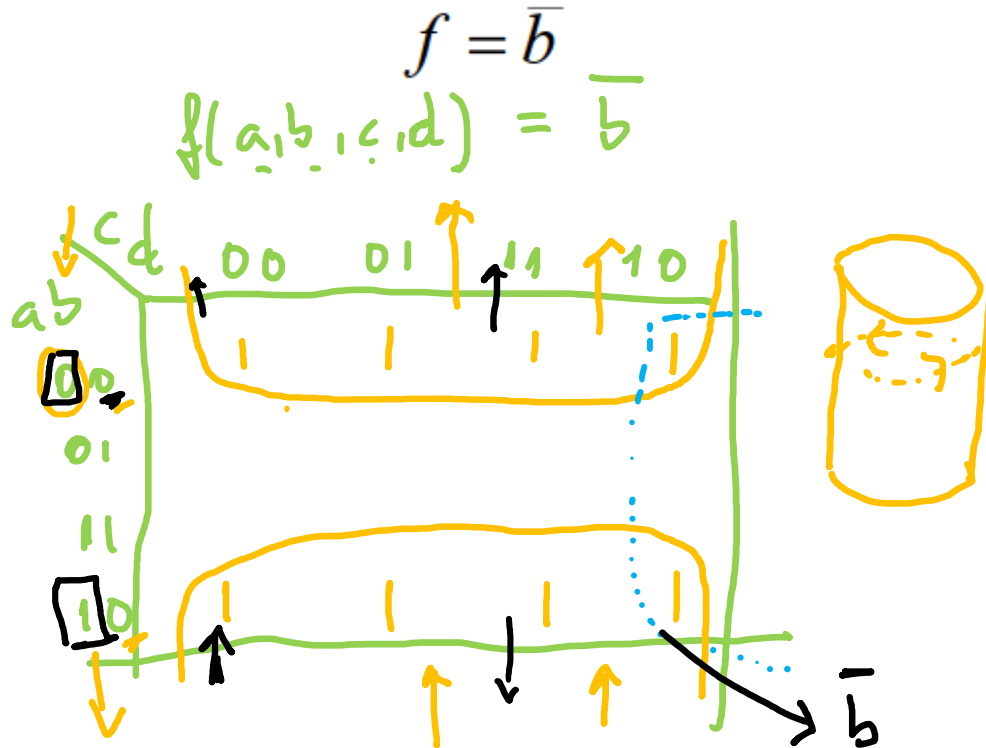
$$f = \bar{a}.b + b.\bar{c}.d$$



# Álgebra Booleana. Minimización lógica

Minimización mediante mapas de Karnaugh (mapas K ). Patrones prototipo.

Obtenemos un mapa de 4 variables para estas otras 2 expresiones:

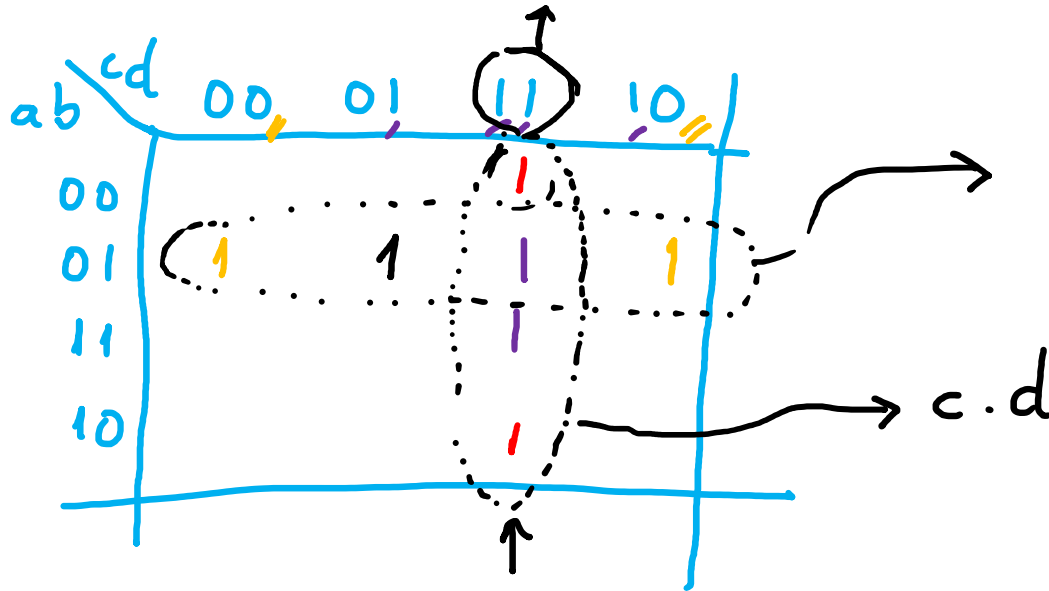


# Álgebra Booleana. Minimización lógica

Minimización mediante mapas de Karnaugh (mapas K)

Ejemplo. Simplificar:

$$f = \bar{a}.b.\bar{d} + b.c.d + \bar{a}.b.\bar{c}.d + c.d$$



$$f = \bar{a}.b + c.d$$

# Álgebra Booleana. Minimización lógica

## Minimización mediante mapas de Karnaugh (mapas K ). Estrategia para las formas POS

Las expresiones en sumas de productos nos llevan por tanto a expresiones que podrían implementarse con un nivel de puertas AND seguido de un segundo nivel de puertas OR.

Anteriormente hemos visto que usando la ley de De Morgan es posible convertir siempre esto en puertas NAND.

Si por el contrario usamos “productos de sumas” obtendríamos primero el nivel de puertas OR y en un segundo nivel de conexión, puertas AND.

En este segundo caso la aplicación de De Morgan nos lleva a la posibilidad de implementar la lógica usando exclusivamente puertas NOR.

En el mapa de Karnaugh consiste simplemente en agrupar los “0’s” en la tabla.

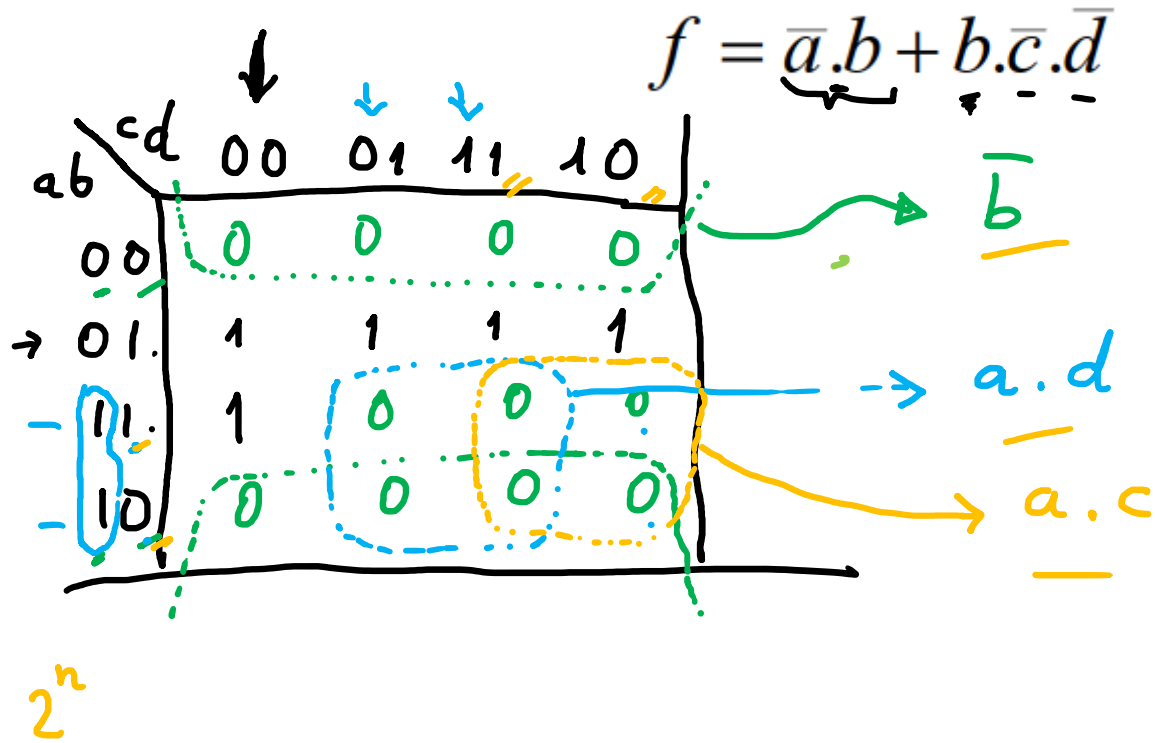
De esta manera lo que estamos haciendo es SIMPLIFICAR LA FUNCION COMPLEMENTARIA.

Al final aplicamos De Morgan y volvemos a complementar, obteniendo la función deseada, y simplificada.

# Álgebra Booleana. Minimización lógica

Minimización mediante mapas de Karnaugh (mapas K ). Simplificación de una expresión en forma POS

Ejemplo. Simplificar la expresión considerada anteriormente:



$$\overline{f} = \overline{b} + a.c + a.d$$

$\downarrow$  D.M

$$\overline{f} = \overline{b} \cdot \overline{a.c} \cdot \overline{a.d} =$$

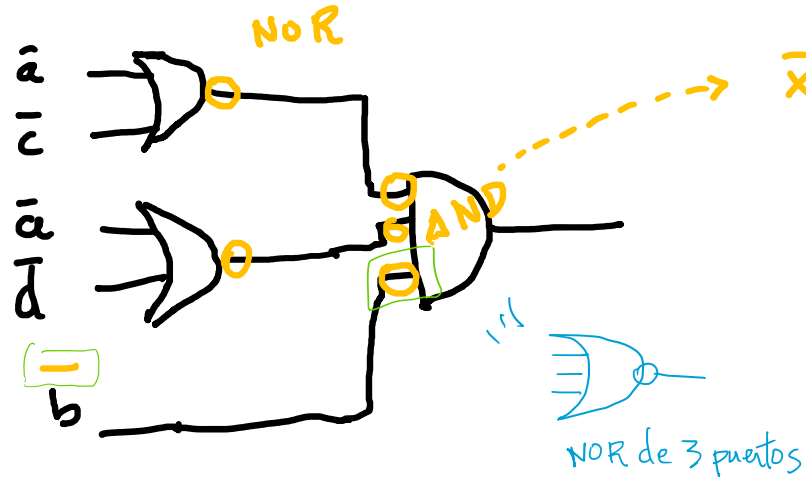
$\downarrow$  D.M

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

$\overline{x+y} = \overline{x} \cdot \overline{y}$



# Álgebra Booleana. Minimización lógica



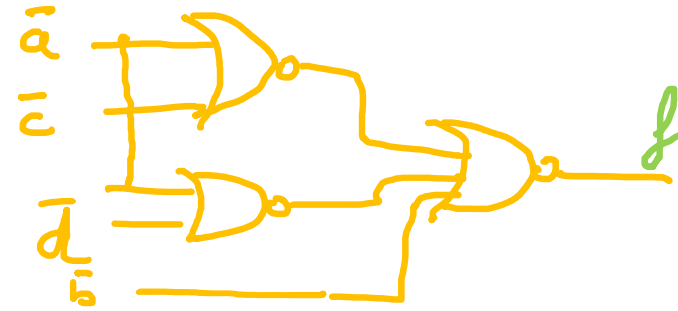
$$\bar{x}_1 \cdot \bar{x}_2 \cdot \bar{x}_3 = \overline{x_1 + x_2 + x_3}$$

NOR

	OR	NOR
00	0	1
01	1	0
10	1	0
11	1	0

Diagram showing a 2-input NOR gate with inputs  $x$  and  $1$ , and output  $\bar{x}$ .

La forma de hacer "NOT" con puertas NOR



# Álgebra Booleana. Minimización lógica

## Minimización mediante mapas de Karnaugh (mapas K ). Empezando por una forma POS

En este caso lo primero que tenemos que hacer para preparar la tabla de Karnaugh es aplicar De Morgan a la expresión que se nos da, y así obtener así  $\bar{f}$ .

Segundo paso: rellenar la tabla con los 0's de la función anterior.

Tercer paso: Rellenar las celdas vacías con 1's : esa es ahora la función  $f$

Por último, simplificar sobre los 1's agrupando normalmente para obtener la forma simplificada de  $f$

# Álgebra Booleana. Minimización lógica

En el ejemplo anterior nos hemos encontrado con que la simplificación/transformación no ha sido especialmente interesante desde el punto de obtener una expresión más simple, que implique menos puertas lógicas, pero tiene otras ventajas, como por ejemplo:

**Convertir la función del ejemplo anterior en SOLO puertas NOR.**

A menudo la simplificación de la forma POS es una buena manera de enfocar el problema de resolver el circuito sólo con puertas NOR.

Lo probamos con el resultado del ejemplo anterior:

# Álgebra Booleana. Minimización lógica

Minimización mediante mapas de Karnaugh (mapas K):

Existiendo términos/condiciones no usados (*don't care*)

Cuando algunas combinaciones, no se pueden dar nunca, podemos aprovecharnos de ellas para simplificar aún más las expresiones.

Ejemplo. Sea la función a simplificar:  $f = \bar{a}.\bar{b}.d + \bar{a}.c.d + a.c.d$

Son términos que no importan:  $\bar{a}.\bar{b}.\bar{c}.\bar{d}, \bar{a}.\bar{b}.c.\bar{d}, \bar{a}.b.\bar{c}.d$

