Universidad Rey Juan Carlos

Tema 7: Electrónica digital

Índice

Analógico vs. Digital. Representación de la información digital: Sistemas de numeración. Códigos binarios: Magnitud y Signo, C2.

Lógica binaria. Álgebra de Boole.

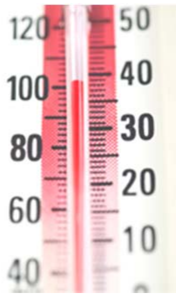
Especificación de sistemas digitales: Formas canónicas. Simplificación mediante mapas de Karnaugh.

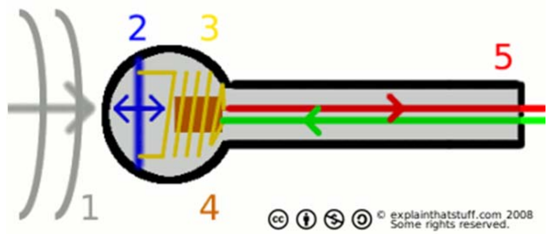
Implementación de sistemas digitales mediante puertas lógicas

Sistemas Analógicos

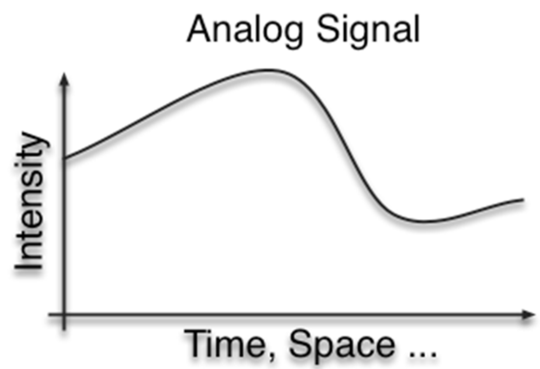
✓ En un sistema analógico, la representación de la información se realiza mediante magnitudes físicas que pueden tomar un espectro continuo de valores.

- En la naturaleza, las magnitudes físicas suelen ser analógicas: espectro de luz, sonido, energía, etc. Tienen una variación continua.
- Ejemplos: relojes analógicos, termómetros de mercurio, micrófonos de audio…





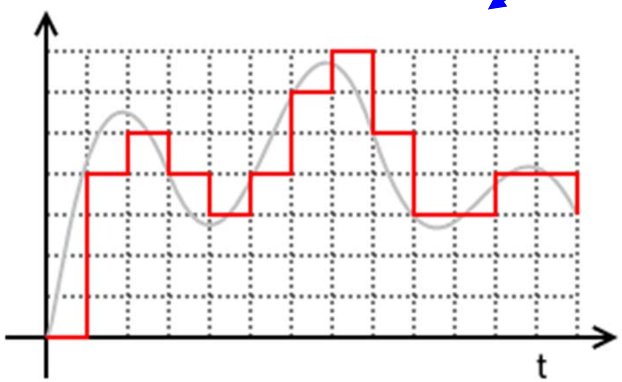
- Un sistema analógico hace una "copia continua" (análoga) de una magnitud (p.e. la temperatura) a otra (p.e. una señal eléctrica de tensión):

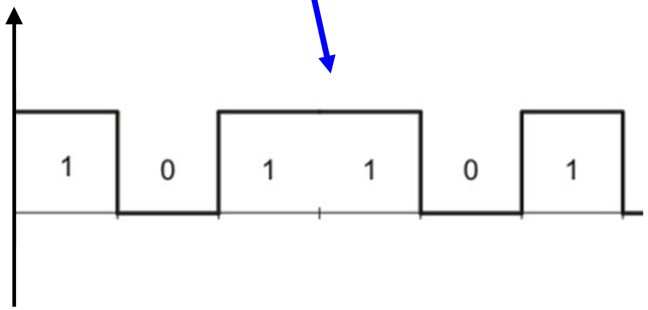


Sistemas Digitales

✓ En un sistema digital, la información se procesa y transmite mediante magnitudes físicas que sólo pueden tomar un conjunto discreto de valores.

- Pueden ser Multivaluados o Bivaluados (binarios)





Ejemplo: termómetros analógicos vs. digital

- Las señales digitales presentan muchas ventajas:
  - Son menos sensibles a las perturbaciones: mayor calidad de la señal.
  - Se almacenan mejor en soportes electrónicos y magnéticos.
  - La evolución de la electrónica permite que los sistemas digitales sean veloces, precisos y baratos.

Información digital. Sistemas de numeración

Sistema de numeración: conjunto de reglas y signos para representar los números.

Un sistema de representación numérica es un sistema consistente en:

- un conjunto ordenado de símbolos (dígitos o cifras).
- un conjunto de reglas bien definidas para las operaciones aritméticas de suma, resta, multiplicación, división, etc.

Números: secuencia de dígitos que pueden tener parte entera y parte fraccionaria, ambas separadas por una coma.

$$\mathbf{(N)_r = [(parte\ entera)\ ,\ (parte\ fraccionaria)]_r}$$

Base (r): nº en que se fundamenta el sistema de numeración. Especifica el nº de dígitos o cardinal de dicho conjunto ordenado.

Sistemas de numeración posicionales

Sistema posicional: **cada dígito tiene un valor distinto dependiendo de su posición.**

Así, un número N en base r se representa de la siguiente manera:

$(\mathbf{N})_{\mathbf{r}} = (a_{p-1}\ a_{p-2}\ \ldots\ a_1\ a_0\ ,\ a_{-1}\ a_{-2}\ \ldots\ a_{-q})_r$ ó

$\mathbf{N}\lfloor_{\mathbf{r}} = a_{p-1}\ a_{p-2}\ \ldots\ a_1\ a_0\ ,\ a_{-1}\ a_{-2}\ \ldots\ a_{-q}\lfloor_r$ ó

$\mathbf{N} = a_{p-1}\ a_{p-2}\ \ldots\ a_1\ a_0\ ,\ a_{-1}\ a_{-2}\ \ldots\ a_{-q}$ si se sobreentiende que está en base r

Donde:

- $\mathbf{a_i}$ son los dígitos o cifras que constituyen el número,
- **p** es el número de dígitos enteros,
- **q** es el número de dígitos fraccionarios,
- $\mathbf{a_{p-1}}$ es el dígito más significativo,
- $\mathbf{a_{-q}}$ es el dígito menos significativo.

Al ser N un número en base r, sus dígitos deben situarse entre 0 y r-1, es decir:

$$0 \le a_i \le r-1, \forall i \text{ con } -q \le i \le p-1$$

Sistemas de numeración: peso y valor

Cada dígito del número es más significativo que el que se encuentra a su derecha, ya que su peso es mayor. En notación polinómica o polinomial, el número se expresa como la suma del valor de cada dígito:

- Se define el peso de cada dígito a_i, que depende de su posición, como r^i
- El valor de cada dígito es $a_i \cdot r^i$

$$N = \sum_{i=-q}^{p-1} a_i \cdot r^i$$

Ejemplo: $(N)_r = (a_{p-1}\ a_{n-2} \ldots a_1\ a_0\ , a_{-1}\ a_{-2} \ldots a_{-q})_r$ Dígitos

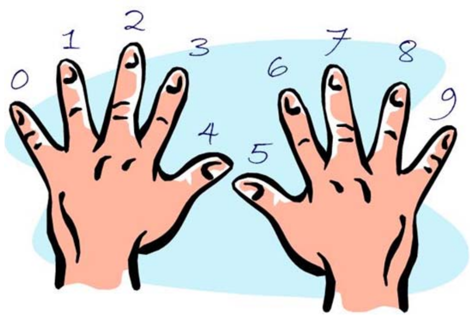
$$(1283)_{10} = (a_3 = 1\ a_2 = 2\ a_1 = 8\ a_0 = 3,\ a_{-i} = 0)_{10} =$$

$$= 1\times10^3 + 2\times10^2 + 8\times10^1 + 3\times10^0 = 1000 + 200 + 80 + 3$$

Pesos = base$^{\text{posición}}$ Valores

Sistemas de numeración en electrónica digital

- ✓ En electrónica digital la representación de la información se realiza utilizando el sistema binario (de base 2), debido a:
  - ➢ Los dispositivos físicos para almacenar y procesar información capaces de representar dos estados son mas simples, rápidos y económicos.
  - ➢ Las operaciones aritméticas con números binarios son muy simples.



- ✓ Conviene, no obstante, conocer bien los siguientes sistemas:
  - ➢ **Sistema binario, BIN:** base r = 2. Cifras 0, 1. Se denominan **bits (binary digits)**
  - ➢ **Sistema octal, OCT:** base r = 8. Cifras 0, 1, 2, 3, 4, 5, 6, 7
  - ➢ **Sistema decimal, DEC:** base r = 10. Cifras 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.
  - ➢ **Sistema hexadecimal, HEX:** base r = 16. Cifras 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

Códigos binarios

Números positivos (sin signo)

- Binario natural (o puro)

Números negativos (con signo)

- Bit de signo + magnitud
- ~~Complemento a 1~~
- Complemento a 2

S MMMMMMM

MSB LSB

~~Números reales~~

- ~~Coma flotante (Estándar IEEE 754)~~

Ancho de palabra y rango representable

- ✓ **Muy importante:** para no aumentar innecesariamente la complejidad de un sistema digital, se suele fijar el número de bits con el que trabaja. A ese número se le conoce como **ancho de palabra**, o **anchura de palabra (n).**
- ✓ Con n bits se pueden representar un número limitado de valores (2^n configuraciones distintas). Se conoce como rango representable.
- ✓ Las palabras de anchura más común reciben un nombre específico:
  - Nibble: palabra de 4 bits. Cada nibble se puede representar con un dígito hexadecimal (de 0 a F).
  - Byte u octeto: palabra de 8 bits. Compuesta por 2 nibbles. Se puede representar con dos dígitos hexadecimales.
  - Word (o palabra): Palabra de 16 bits. Compuesta por 2 Bytes, ó 4 nibbles.
  - Dword (doble palabra): 32 bits. Compuesta por 4 Bytes, 8 nibbles.
  - Qword (palabra cuádruple): 64 bits, 8 Bytes, 16 nibbles.

Códigos binarios: Magnitud y signo

S	MMMMMMM

1. **Definición:** primer bit de **signo**, n-1 restantes **magnitud** del número en binario puro.
2. **Conversión a base 10:** $A = (1 - 2 \cdot a_{n-1}) \cdot \sum_{i=0}^{n-2} a_i \cdot 2^i$
3. **Representación de números intuitiva:**
   - **nº positivos:**

0	MMMMMMM

   - **nº negativos:**

1	MMMMMMM

Signo	Magnitud o Valor Absoluto			
0	1	0	1	1
1	n-1			

Signo	Magnitud o Valor Absoluto			
1	1	0	1	1
1	n-1			

4. **Rango representable:** $[-(2^{n-1}-1), 2^{n-1}-1]$.
   Ambigüedad en el cero: 0000 – 1000
5. **Cálculo del opuesto:** cambiar el bit de signo
6. **Extensión de signo:** se desplaza a la izquierda el bit de signo, y los huecos nuevos se rellenan con bits a 0.
7. **Aritmética:** No es intuitiva, hay que tener en cuenta los signos.

Códigos binarios. Complemento a 2 (C2)

1. Se define COMPLEMENTO A LA BASE o COMPLEMENTO A 2 de un número N como C2(N) = 2^n-N, donde “n” es la anchura de palabra.

Con n = 4, $C_2(1010) = 10000 - 1010 = 0110$
Con n = 5, $C_2(10100) = 2^5 - 10100 = 01100$

$$\begin{array}{r} 100000 \\ -\ \ 10100 \\ \hline 01100 \end{array}$$

ATAJO: para calcular el C2 de N se pueden copiar los bits empezando por la derecha hasta que aparezca el primer 1 (inclusive) y negar los bits restantes (permutar 0 por 1 y 1 por 0)

Si n = 4, $C_2(1010) = 0110$
Si n = 5, $C_2(10100) = 01100$
Si n = 16, $C_2(0010010101001110) = 1101101010110010$

2. Conversión a base 10: Se asigna un peso negativo -2^{n-1} al bit más significativo, a_{n-1} (MSB).

$$A = -2^{n-1} \cdot a_{n-1} + \sum_{i=0}^{n-2} a_i \cdot 2^i$$

$A_{\lfloor C2} = 00011101_{\lfloor C2}$, $A = 1x2^0+0x2^1+1x2^2+1x2^3+1x2^4+0x2^5+0x2^6-0x2^7 = 29_{\lfloor 10}$

$B_{\lfloor C2} = 11001011_{\lfloor C2}$, $B = 1x2^0+1x2^1+0x2^2+1x2^3+0x2^4+0x2^5+1x2^6-1x2^7 = -53_{\lfloor 10}$

Códigos binarios. Complemento a 2

3. Representación de números

- **nº positivos:** Igual que en magnitud y signo

0	MMMMMMM

- **nº negativos: C_2 (N). Complemento a la base** del número positivo N. Bit de signo negativo, magnitud no está en binario!

Ejemplo (número positivo): representar $A = 29_{\lfloor 10}$ en C2 con n = 8 bits

$A_{\lfloor C2} = A_{\lfloor MS} = 00011101_{\lfloor C2}$

Ejemplo (número negativo): representar $B = -53_{\lfloor 10}$ en C2 con n = 8 bits

- Primero lo representamos en positivo: $-B_{\lfloor C2} = 00110101_{\lfloor C2}$
- Después calculamos el C2: $B_{\lfloor C2} = C2(-B_{\lfloor C2}) = 11001011_{\lfloor C2}$
- También podemos calcularlo directamente con la definición y después pasarlo a binario:

$$B_{\lfloor C2} = 2^8-53 = 256-53 = 203 = 11001011_{\lfloor C2}$$

Códigos binarios. Complemento a 2

4. Rango representable en C2: $[-2^{n-1}, 2^{n-1}-1]$

Ejemplo: Con 4 bits (n = 4), el rango es $-2^3 \leq x \leq 2^3-1 \Rightarrow -8 \leq x \leq 7$.

$0111\rfloor_{C2} = +7\rfloor_{10}$ $\quad$ $1000\rfloor_{C2} = -8\rfloor_{10}$

Ejemplo: Con 1 Byte (n = 8), el rango es $-2^7 \leq x \leq 2^7-1 \Rightarrow -128 \leq x \leq 127$.

$01111111\rfloor_{C2} = +127\rfloor_{10}$ $\quad$ $10000000\rfloor_{C2} = -128\rfloor_{10}$

5. Cálculo del opuesto (cambio de signo): se lleva a cabo mediante la complementación, es decir, mediante el cálculo del C2 del número

Ejemplo: cambiar de signo el número $A\rfloor_{C2} = 00011101\rfloor_{C2}$, n = 8,

$-A\rfloor_{C2} = C2(A\rfloor_{C2}) = 11100011\rfloor_{C2}$

Ejemplo: cambiar de signo el número $B\rfloor_{C2} = 11001011\rfloor_{C2}$, n = 8,

$-B\rfloor_{C2} = C2(B\rfloor_{C2}) = 00110101\rfloor_{C2}$

Códigos binarios. Complemento a 2

6. Extensión de signo: se replica el bit de signo hacia la izquierda.

Ejemplo: extender X = $100110_{\lfloor C2}$ de 6 a 8 bits $\underline{1}00110$

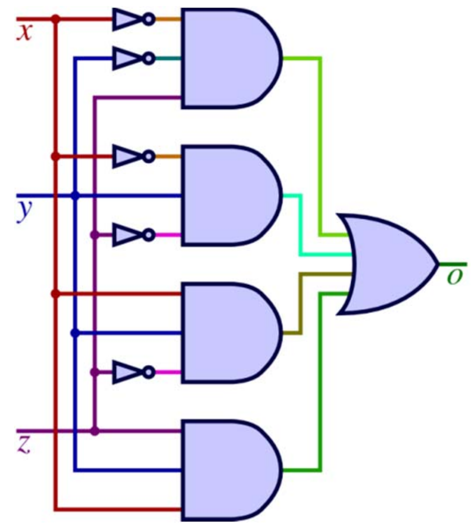
11100110

Extender X = $010011_{\lfloor C2}$ de 6 a 8 bits $\underline{0}10011$

00010011

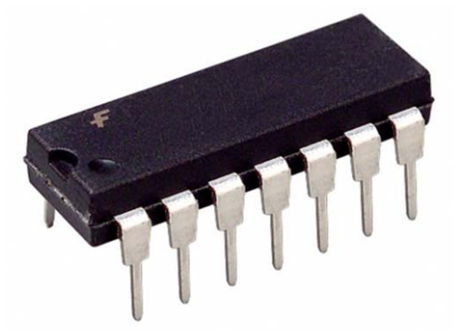
Conceptos: lógica binaria, álgebra de Boole

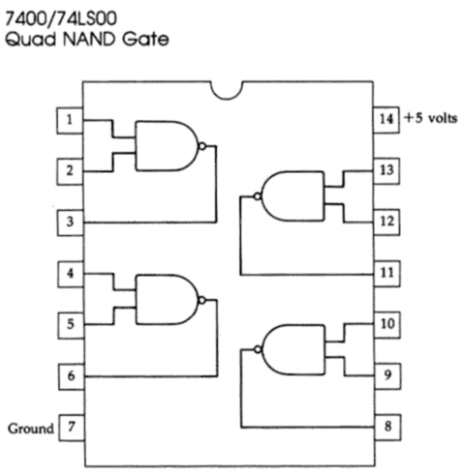
- ✓ La lógica binaria (lógica booleana o de conmutación) es la que trabaja con un conjunto de elementos binarios (0,1) y las operaciones lógicas AND, OR, NOT, etc. Tiene estructura matemática de Álgebra de Boole y es la base de la electrónica digital y los computadores.
- ✓ Los dispositivos o circuitos lógicos digitales (que forman la lógica digital) ejecutan físicamente esas operaciones lógicas.
  - Dispositivos combinacionales: no tienen capacidad de almacenar datos (memoria).
    - Puertas lógicas: AND, OR, NOT…
    - Módulos complejos: codificadores, multiplexores…
  - Dispositivos secuenciales: pueden almacenar datos (tienen memoria). No se ven en esta asignatura.
    - Biestables o flip-flops. Almacenan 1 bit.
    - Módulos complejos: registros, memorias…

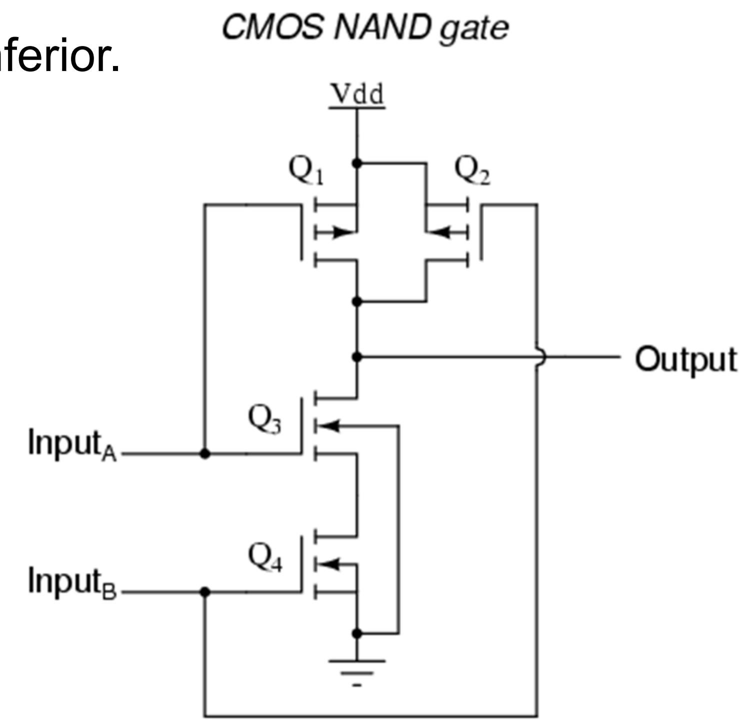


Tecnología en una puerta lógica

- ✓ ¿De qué está hecha una puerta lógica? Aunque queda fuera del objetivo de esta asignatura, conviene saber que se construyen utilizando unos componentes electrónicos denominados transistores.
- ✓ Hay varias tecnologías (familias) para fabricar un transistor, destacando:
  - ➢ TTL: transistores rápidos, pero consumo elevado.
  - ➢ CMOS: algo más lentos, pero consumo muy inferior. Alta densidad de integración en un chip. Es la más utilizada actualmente.

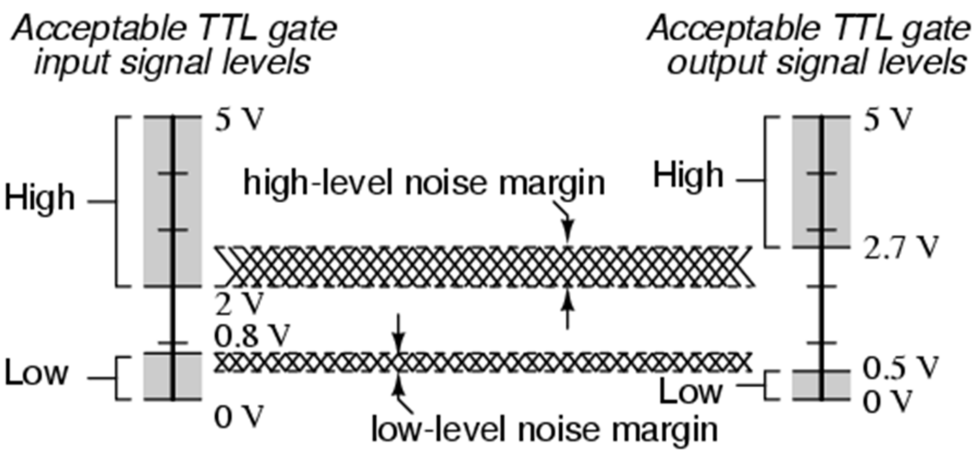






¿Cómo se representan el 0 y el 1 en realidad?

- ✓ Como las puertas lógicas están fabricadas con dispositivos electrónicos, el 0 y el 1 lógico deben representarse con una magnitud eléctrica.
- ✓ Se usan valores de tensión eléctrica (que se mide en voltios, V):
  - ➢ En TTL el 0 lógico se representa con 0 V (nivel bajo) y el 1 lógico se representa con 5 V (nivel alto).
  - ➢ En CMOS el 0 lógico se representa con 0 V (nivel bajo) y el 1 lógico se representa con un valor en el rango de 1.8 a 18 V (nivel alto).



⇨**INCISO:** En realidad todos los dispositivos trabajan dentro de unos márgenes de tensión, por lo que una pequeña perturbación de la tensión no les afecta. Es una de las claves de la tecnología digital.

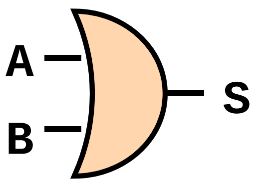
En nuestra aproximación a los circuitos digitales…

Me puedo olvidar que tengo transistores, voltios, ...

subo el nivel de abstracción ("me abstraigo")

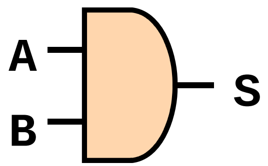
Utilizo puertas lógicas para diseñar mis circuitos

Puerta OR



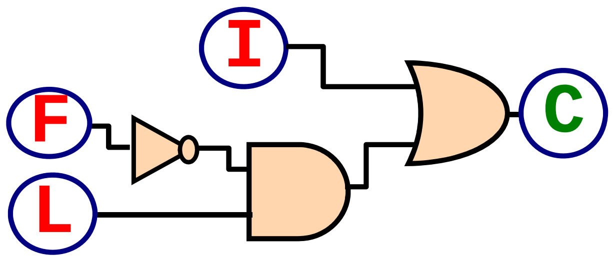
A	B	S
0	0	0
0	1	1
1	0	1
1	1	1

Puerta AND



A	B	S
0	0	0
0	1	0
1	0	0
1	1	1

Las entradas y las salidas son ceros y unos



Conecto señales y puertas para formar mi circuito

Las reglas vienen dadas por el álgebra de Boole

Independiente de la electrónica del circuito

Álgebra de Boole: definición

Un álgebra de Boole bivaluada es un conjunto B que cumple que:

1. $\forall$ a $\in$ B , a = 0 ó a = 1.

2. Todo elemento tiene un complementario (función NOT, $\bar{a}$).

- **NOT**: negación lógica o complementación.
- A veces se representa como a', ~a, ¬a

a	$f(a)=\bar{a}$
0	1
1	0

3. La operación producto lógico ("·" , AND) se define como:

- **AND**: producto lógico, intersección o conjunción.

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

4. La operación suma lógica ("+", OR) se define como:

- **OR**: suma lógica, unión o disyunción.

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

5. La operación AND tiene precedencia sobre la OR.

Álgebra de Boole: operaciones

- Otras operaciones usuales
  - XOR o EOR (suma lógica exclusiva o diferencia simétrica)
  - NOR (suma lógica complementada)
  - NAND (producto lógico complementado)
  - XNOR (suma lógica exclusiva complementada o equivalencia).

XOR

a	b	$f(a,b)=a\oplus b$
0	0	0
0	1	1
1	0	1
1	1	0

NOR

a	b	$f(a,b)=\overline{a+b}$
0	0	1
0	1	0
1	0	0
1	1	0

NAND

a	b	$f(a,b)=\overline{a\cdot b}$
0	0	1
0	1	1
1	0	1
1	1	0

XNOR

a	b	$f(a,b)=\overline{a\oplus b}$
0	0	1
0	1	0
1	0	0
1	1	1

Álgebra de Boole: Teoremas y propiedades

Propiedad asociativa	a+(b+c) = (a+b)+c = a+b+c	a·(b·c) = (a·b)·c = a·b·c
Propiedad conmutativa	a+b = b+a	a·b = b·a
Propiedad distributiva	a+(b·c) = (a+b) · (a+c)	a·(b+c) = a·b + a·c
Elemento neutro	0+a=a	1·a=a
	1+a=1	0·a=0
Teoremas de identidad	a+a'=1	a·a'=0
Teoremas de idempotencia	a+a=a	a·a=a
Teorema de involución	(a')'=a	
Teoremas de absorción	a+a·b = a	a·(a+b) = a
	a+a'·b=a+b	a·(a'+b)=a·b
Teoremas del consenso	a·b+a'·c= a·b+a'·c+b·c	(a+b)·(a'+c)= (a+b)·(a'+c)·(b+c)
Leyes de De Morgan	(a+b)' = a'·b'	(a·b)' = a'+b'

Especificación de un sistema electrónico digital

El comportamiento de un sistema electrónico digital se puede describir usando:

1. Funciones lógicas, booleanas o funciones de conmutación (FC): una FC describe el valor que toma cada una de las salidas de un circuito digital para todas las posibles configuraciones binarias que puedan presentarse en sus entradas
   - Tablas de verdad: representan los valores adoptados por las FC de forma extensiva:
     - Tienen una columna por cada variable, más una adicional para el valor de la función.
     - Tienen una fila por cada posible combinación de valores de las variables

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

2. Expresiones lógicas, booleanas o expresiones de conmutación (EC): esta alternativa usa “ecuaciones”, es decir, cadenas de texto en las que aparecen símbolos (variables binarias, denominadas literales), constantes (0 y 1) y los operadores binarios NOT (-), OR (+) y AND (·).

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

Formas canónicas de las EC

➲ Todas las expresiones de conmutación (EC), independientemente de su forma, pueden convertirse en cualquiera de las dos formas canónicas.

➲ **Formas canónicas**, formas normales o formas estándares de una función booleana son expresiones booleanas de la función que verifican:

- ❖ **Primera forma canónica, primera forma normal o forma normal disyuntiva**: es una expresión de una función booleana compuesta por una suma de minitérminos (minterms).
- ❖ **Segunda forma canónica, segunda forma normal o forma normal conjuntiva**: es una expresión de una función booleana compuesta por un producto de maxitérminos (maxterms).

Minitérminos y maxitérminos

- **Minitérmino (minterm):** término producto que contiene todas las variables de la función
  - Ejemplo: f(a,b,c)

    SÍ son minitérminos: $\bar{a}\cdot\bar{b}\cdot\bar{c}$ $\bar{a}\cdot b\cdot\bar{c}$ $\bar{a}\cdot b\cdot c$ $a\cdot\bar{b}\cdot c$ $a\cdot b\cdot c$

    NO son minitérminos: $\bar{a}\cdot\bar{b}$ $\bar{b}\cdot c$ $\bar{a}\cdot c$ $a\cdot\bar{b}$ $a\cdot c$

- **Maxitérmino (maxterm):** término suma que contiene todas las variables de la función.
  - Ejemplo: f(a,b,c)

    SÍ son maxitérminos: $\bar{a}+\bar{b}+\bar{c}$ $\bar{a}+b+\bar{c}$ $\bar{a}+b+c$ $a+\bar{b}+c$ $a+b+c$

    NO son maxitérminos: $\bar{a}+\bar{b}$ $\bar{b}+c$ $\bar{a}+c$ $a+\bar{b}$ $a+c$

Primera forma canónica (1FC)

- Los minitérminos se nombran con subíndices (m_i), donde i es un número obtenido tras pasar a base 10 el número binario formado al sustituir ordenadamente las variables afirmadas por 1 y las negadas por 0.
  - Ejemplo: f(a,b,c), minitérmino $a \cdot \overline{b} \cdot c = m_5$
- Cada minitérmino está asociado a una fila de la tabla de verdad de la función lógica correspondiente.
- La **primera forma canónica o 1FC** es una expresión de una función booleana compuesta por una suma de minitérminos
- La expresión en 1FC es única para cada función.

a	b	c	d	Minitérmino	m_i
0	0	0	0	$\overline{a} \cdot \overline{b} \cdot \overline{c} \cdot \overline{d}$	m_0
0	0	0	1	$\overline{a} \cdot \overline{b} \cdot \overline{c} \cdot d$	m_1
0	0	1	0	$\overline{a} \cdot \overline{b} \cdot c \cdot \overline{d}$	m_2
0	0	1	1	$\overline{a} \cdot \overline{b} \cdot c \cdot d$	m_3
0	1	0	0	$\overline{a} \cdot b \cdot \overline{c} \cdot \overline{d}$	m_4
0	1	0	1	$\overline{a} \cdot b \cdot \overline{c} \cdot d$	m_5
0	1	1	0	$\overline{a} \cdot b \cdot c \cdot \overline{d}$	m_6
0	1	1	1	$\overline{a} \cdot b \cdot c \cdot d$	m_7
1	0	0	0	$a \cdot \overline{b} \cdot \overline{c} \cdot \overline{d}$	m_8
1	0	0	1	$a \cdot \overline{b} \cdot \overline{c} \cdot d$	m_9
1	0	1	0	$a \cdot \overline{b} \cdot c \cdot \overline{d}$	m_{10}
1	0	1	1	$a \cdot \overline{b} \cdot c \cdot d$	m_{11}
1	1	0	0	$a \cdot b \cdot \overline{c} \cdot \overline{d}$	m_{12}
1	1	0	1	$a \cdot b \cdot \overline{c} \cdot d$	m_{13}
1	1	1	0	$a \cdot b \cdot c \cdot \overline{d}$	m_{14}
1	1	1	1	$a \cdot b \cdot c \cdot d$	m_{15}

Primera forma canónica (1FC)

La expresión en 1FC de una función booleana es la suma de los minitérminos asociados a las filas que valen 1 en la tabla de verdad.

Ejemplo: $f(a,b,c) = \overline{a} \cdot (b+c) + a \cdot \overline{c}$

Calculando su tabla de verdad se obtiene lo siguiente:

a	b	c	a'	b+c	a'·(b+c)	c'	a·c'	f(i)
0	0	0	1	0	0	1	0	0
0	0	1	1	1	1	0	0	1
0	1	0	1	1	1	1	0	1
0	1	1	1	1	1	0	0	1
1	0	0	0	0	0	1	1	1
1	0	1	0	1	0	0	0	0
1	1	0	0	1	0	1	1	1
1	1	1	0	1	0	0	0	0

Entonces: $f(a,b,c) = m_1 + m_2 + m_3 + m_4 + m_6 = \sum_{5} m(1, 2, 3, 4, 6)$

Segunda forma canónica (2FC)

- Los maxitérminos se nombran con subíndices (M_i), donde i es un número obtenido tras pasar a base 10 el número binario formado al sustituir ordenadamente las variables afirmadas por 0 y las negadas por 1.
  - Ejemplo: f(a,b,c), maxitérmino $a+\overline{b}+c=M_2$
- Cada maxitérmino está asociado a una fila de la tabla de verdad de la función lógica correspondiente
- **Segunda forma canónica o 2FC** es una expresión de una función booleana compuesta por una suma de maxitérminos
- La expresión en 2FC es única para cada función

a	b	c	d	Maxitérmino	M_i
0	0	0	0	$a+b+c+d$	M_0
0	0	0	1	$a+b+c+\overline{d}$	M_1
0	0	1	0	$a+b+\overline{c}+d$	M_2
0	0	1	1	$a+b+\overline{c}+\overline{d}$	M_3
0	1	0	0	$a+\overline{b}+c+d$	M_4
0	1	0	1	$a+\overline{b}+c+\overline{d}$	M_5
0	1	1	0	$a+\overline{b}+\overline{c}+d$	M_6
0	1	1	1	$a+\overline{b}+\overline{c}+\overline{d}$	M_7
1	0	0	0	$\overline{a}+b+c+d$	M_8
1	0	0	1	$\overline{a}+b+c+\overline{d}$	M_9
1	0	1	0	$\overline{a}+b+\overline{c}+d$	M_{10}
1	0	1	1	$\overline{a}+b+\overline{c}+\overline{d}$	M_{11}
1	1	0	0	$\overline{a}+\overline{b}+c+d$	M_{12}
1	1	0	1	$\overline{a}+\overline{b}+c+\overline{d}$	M_{13}
1	1	1	0	$\overline{a}+\overline{b}+\overline{c}+d$	M_{14}
1	1	1	1	$\overline{a}+\overline{b}+\overline{c}+\overline{d}$	M_{15}

Segunda forma canónica (2FC)

La expresión en 2FC de una función booleana es el producto de los maxitérminos asociados a las filas que valen 0 en la tabla de verdad.

Ejemplo: $f(a,b,c) = \overline{a} \cdot (b + c) + a \cdot \overline{c}$

Calculando su tabla de verdad habíamos obtenido:

a	b	c	f(i)
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

$$f(a,b,c) = M_0 \cdot M_5 \cdot M_7 = \prod_3 M(0,5,7)$$

Valores indiferentes o redundancias (*don't care*)

- En algunos sistemas digitales reales, hay ciertas combinaciones de las variables de entrada que no pueden producirse nunca.
- En estos casos, la salida que pudiera producir el sistema ante dichas combinaciones de entrada es irrelevante, puesto que nunca se va a dar el caso.
- Las combinaciones imposibles de entrada se denominan **indiferencias, valores indiferentes, redundancias**, y en la tabla de verdad se representan con el símbolo **X (o d)**.
- Si aparece **X (o d)** en una o varias filas de una tabla, nos daría exactamente igual sustituirla por un 1 ó por un 0.

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

Ejemplo: función que dice si un número en BCD es par.

$$f(a,b,c,d) = \sum_{5(11)} m(0,2,4,6,8) + X(10,11,12,13,14,15)$$

$$f(a,b,c,d) = \prod_{5(11)} M(1,3,5,7,9) \cdot X(10,11,12,13,14,15)$$

Universidad Rey Juan Carlos

Formas canónicas: resumen

- ✓ Las formas canónicas se pueden extraer directamente de la tabla de verdad

- ✓ Primera forma canónica (1FC): suma de minitérminos asociados a las filas con valor 1.

$$f(a,b,c,d)=\sum_{8} m(0,3,4,5,10,11,14,15)$$

- ✓ Segunda forma canónica (2FC): producto de maxitérminos asociados a las filas con valor 0.

$$f(a,b,c,d)=\prod_{8} M(1,2,6,7,8,9,12,13)$$

- ✓ Las formas canónicas son únicas para cada función: una función tiene una única expresión en 1FC y una única expresión en 2FC.

- ✓ La 1FC y la 2FC de una función son equivalentes.

Simplificación de funciones lógicas

- Dado que existen múltiples circuitos para implementar una función lógica dada, lo mejor es utilizar el circuito más adecuado para cada situación.

- Criterios posibles para manipular las expresiones lógicas:
  - Obtener el circuito más barato reduciendo el número de términos.
  - Obtener el circuito más rápido.
  - Obtener el circuito formado por menos circuitos integrados de un tipo.
  - Obtener un circuito sin valores transitorios no deseados (azares, *glitches*).

- Simplificación: proceso que conduce a reducir el número de literales y términos de una función lógica.
  - Se puede simplificar expresiones de conmutación mediante manipulaciones algebraicas (proceso manual, costoso).
  - Métodos gráficos: **Veitch-Karnaugh** (proceso manual, sencillo para pocas variables).

Método de Veitch-Karnaugh

- Inventado por Veitch a principios de los años 50, y perfeccionado por Karnaugh
- Se basa en construir unos diagramas adecuados para simplificar gráficamente.
- Diagrama (mapa, tabla) de Veitch-Karnaugh para una función de *n* variables: tabla rectangular de 2^n celdas, cada una de las cuales está asociada a una combinación de variables (y a una fila de la tabla de verdad).
  - En cada casilla habrá un 1 ó un 0, dependiendo de la fila de la tabla de verdad asociada:

a \ b	0	1
0	1 [0]	0 [1]
1	1 [2]	1 [3]

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

- Propiedad principal: cada casilla es **adyacente a todas sus vecinas en horizontal y vertical**, es decir, entre una casilla y su vecina sólo difiere el valor de una variable.

Método de Veitch-Karnaugh. 2 variables

- El mapa tiene 4 casillas, cada una asociada a una combinación de los valores de las variables.
- Cada casilla tiene 2 vecinas.
- En cada casilla se ha añadido el nº de la fila de la tabla de verdad asociada a dicha casilla, así como la combinación de variables que la corresponde.

a \ b	0	1
0	$\bar{a}\cdot\bar{b}$ 0 f(0)	$\bar{a}\cdot b$ 1 f(1)
1	$a\cdot\bar{b}$ 2 f(2)	$a\cdot b$ 3 f(3)

- Vecindades:
  - Casilla 0: 1 y 2.
  - Casilla 1: 0 y 3.
  - Casilla 2: 0 y 3.
  - Casilla 3: 1 y 2.

Método de Veitch-Karnaugh. 2 variables

Ejemplo:

Tabla de verdad

a	b	f(i)
0	0	1
0	1	0
1	0	1
1	1	1

Función: $f(a,b)=m_0+m_2+m_3=M_1$

Mapa de V-K:

a \ b	0	1
0	1 (0)	0 (1)
1	1 (2)	1 (3)

Universidad Rey Juan Carlos

Método de Veitch-Karnaugh. 3 variables

➲ Ahora el mapa tiene 8 casillas, y cada casilla tiene 3 vecinas.

a \ bc	00	01	11	10
0	$\bar{a}\cdot\bar{b}\cdot\bar{c}$ 0 f(0)	$\bar{a}\cdot\bar{b}\cdot c$ 1 f(1)	$\bar{a}\cdot b\cdot c$ 3 f(3)	$\bar{a}\cdot b\cdot\bar{c}$ 2 f(2)
1	$a\cdot\bar{b}\cdot\bar{c}$ 4 f(4)	$a\cdot\bar{b}\cdot c$ 5 f(5)	$a\cdot b\cdot c$ 7 f(7)	$a\cdot b\cdot\bar{c}$ 6 f(6)

➲ Vecindades:

- Casilla 0: 1, 2 y 4.
- Casilla 1: 0, 3 y 5.
- Casilla 2: 0, 3 y 6.
- Casilla 3: 1, 2 y 7.
- Casilla 4: 0, 5 y 6.
- Casilla 5: 1, 4 y 7.
- Casilla 6: 2, 4 y 7.
- Casilla 7: 2, 4 y 7.

Método de Veitch-Karnaugh. 3 variables

Ejemplo:

Tabla de verdad

Función: $f(a,b)=m_1+m_2+m_3+m_4+m_6=M_0 \cdot M_5 \cdot M_7$

a	b	c	f(i)
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Mapa de V-K:

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

Método de Veitch-Karnaugh. 4 variables

➲ Ahora el mapa tiene 16 casillas, y cada una tiene 4 vecinas.

ab \ cd	00	01	11	10
00	$\bar{a}\cdot\bar{b}\cdot\bar{c}\cdot\bar{d}$ 0 f(0)	$\bar{a}\cdot\bar{b}\cdot\bar{c}\cdot d$ 1 f(1)	$\bar{a}\cdot\bar{b}\cdot c\cdot d$ 3 f(3)	$\bar{a}\cdot\bar{b}\cdot c\cdot\bar{d}$ 2 f(2)
01	$\bar{a}\cdot b\cdot\bar{c}\cdot\bar{d}$ 4 f(4)	$\bar{a}\cdot b\cdot\bar{c}\cdot d$ 5 f(5)	$\bar{a}\cdot b\cdot c\cdot d$ 7 f(7)	$\bar{a}\cdot b\cdot c\cdot\bar{d}$ 6 f(6)
11	$a\cdot b\cdot\bar{c}\cdot\bar{d}$ 12 f(12)	$a\cdot b\cdot\bar{c}\cdot d$ 13 f(13)	$a\cdot b\cdot c\cdot d$ 15 f(15)	$a\cdot b\cdot c\cdot\bar{d}$ 14 f(14)
10	$a\cdot\bar{b}\cdot\bar{c}\cdot\bar{d}$ 8 f(8)	$a\cdot\bar{b}\cdot\bar{c}\cdot d$ 9 f(9)	$a\cdot\bar{b}\cdot c\cdot d$ 11 f(11)	$a\cdot\bar{b}\cdot c\cdot\bar{d}$ 10 f(10)

Vecindades:

- Casilla 0: 1, 2, 4 y 8.
- Casilla 1: 0, 3, 5 y 9.
- Casilla 2: 0, 3, 6 y 10.
- Casilla 3: 1, 2, 7 y 11.
- Casilla 4: 0, 5, 6 y 12.
- Casilla 5: 1, 4, 7 y 13.
- Casilla 6: 2, 4, 7 y 14.
- Casilla 7: 2, 4, 7 y 15.
- Casilla 8: 0, 9, 10 y 12.
- Casilla 9: 1, 8, 11 y 13.
- Casilla 10: 2, 8, 11 y 14.
- Casilla 11: 3, 9, 10 y 15.
- Casilla 12: 4, 8, 13 y 14.
- Casilla 13: 5, 9, 12 y 15.
- Casilla 14: 6, 10, 12 y 15.
- Casilla 15: 7, 11, 13 y 14.

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

$$f(a,b,c,d) = \sum_4 m(0,3,4,5,10,11,14,15) = \prod_4 M(1,2,6,7,8,9,12,13)$$

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

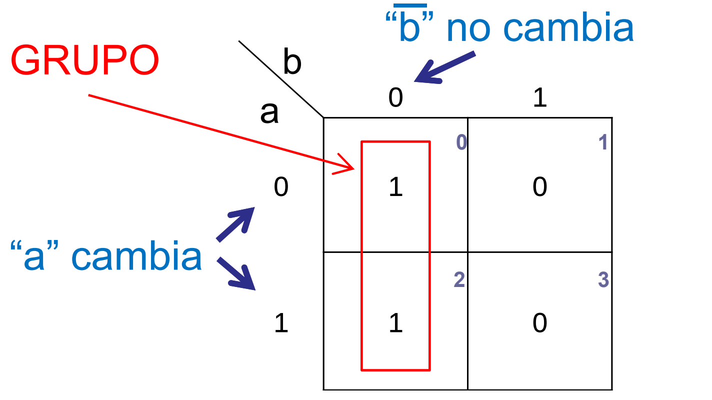
Simplificación por V-K con minitérminos (1FC)

- **Agrupación de 1s** pertenecientes a celdas adyacentes. El objetivo es maximizar el tamaño de los grupos y minimizar el nº de grupos.
  - Un grupo puede contener 1, 2, 4, 8, 16 celdas (potencias de 2)
  - Cada celda de un grupo tiene que ser adyacente a una o más celdas del mismo grupo, pero no todas las celdas del grupo tiene que ser adyacentes entre sí.
    - Grupos de dos celdas: cada celda 1 adyacencia
    - Grupos de cuatro celdas: cada celda 2 adyacencias….
  - Cada 1 del mapa debe estar incluido en al menos en un grupo.

- Dentro de cada grupo, para obtener la expresión (término producto) se eliminan las variables que cambian.

- Cuando se han obtenido todos los términos producto mínimos se suman para obtener la expresión "**suma de términos producto" mínima**.

Función en 1FC: $f(a,b)=m_0+m_2$



Función simplificada: $f(a,b)=\overline{b}$

Ejemplo, parte 1: simplificar por minitérminos la función f(a,b,c,d) cuya tabla de verdad se indica a continuación.

Construimos el mapa de V-K:

ab \ cd	00	01	11	10
00	X	0	0	1
01	0	X	0	1
11	1	0	1	0
10	1	0	0	1

a	b	c	d	f
0	0	0	0	X
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	X
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

Paso 1: tomar los 1 que no puedan formar subcubos con otras casillas.

Paso 1: tomar los 1 que no puedan formar subcubos con otras casillas.

ab \ cd	00	01	11	10
00	X	0	0	1
01	0	X	0	1
11	1	0	$\boxed{1}$	0
10	1	0	0	1

$\boxed{P1 = a \cdot b \cdot c \cdot d}$

Paso 2: tomar los 1 que sólo puedan formar subcubos de 2 casillas $\Rightarrow$

Paso 2: tomar los 1 que sólo puedan formar subcubos de 2 casillas.

ab \ cd	00	01	11	10
00	X	0	0	1
01	0	X	0	1
11	1	0	1	0
10	1	0	0	1

$\boxed{P1 = a \cdot b \cdot c \cdot d}$

$\boxed{P2 = a \cdot \overline{c} \cdot \overline{d}}$

¿Repetir paso 2? $\Rightarrow$

Paso 2: tomar los 1 que sólo puedan formar subcubos de 2 casillas.

ab \ cd	00	01	11	10
00	X	0	0	1
01	0	X	0	1
11	1	0	1	0
10	1	0	0	1

$\boxed{P1 = a \cdot b \cdot c \cdot d}$

$\boxed{P2 = a \cdot \overline{c} \cdot \overline{d}}$

$\boxed{P3 = \overline{a} \cdot c \cdot \overline{d}}$

Paso 3: tomar los 1 que no estén tomados y sólo puedan formar subcubos de 4 (y no de 8)

Paso 3: tomar los 1 que no estén tomados y sólo puedan formar subcubos de 4 y no de 8.

ab \ cd	00	01	11	10
00	X	0	0	1
01	0	X	0	1
11	1	0	1	0
10	1	0	0	1

$\boxed{P1 = a \cdot b \cdot c \cdot d}$

$\boxed{P2 = a \cdot \overline{c} \cdot \overline{d}}$

$\boxed{P3 = \overline{a} \cdot c \cdot \overline{d}}$

$\boxed{P4 = \overline{b} \cdot \overline{d}}$

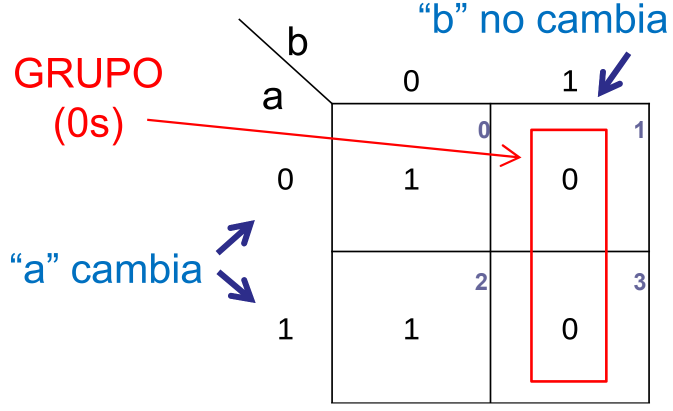
Paso 4: no ha lugar porque todos los 1 ya están cogidos en algún subcubo. Ya hemos terminado. Sumamos los términos producto resultantes:

$$\boxed{f_1(a,b,c,d) = a \cdot b \cdot c \cdot d + a \cdot \overline{c} \cdot \overline{d} + \overline{a} \cdot c \cdot \overline{d} + \overline{b} \cdot \overline{d}}$$

Simplificación por V-K con maxitérminos (2FC)

- Se realiza de forma parecida a como se hace con los minitérminos, con las siguientes diferencias:
  - Los grupos están formados por casillas con valor 0 ó X (en los minitérminos se toman las casillas con 1 ó X).
  - Al escribir el término simplificado, la complementación de las variables es la contraria:
    - Las variables que irían complementadas en minitérminos van sin complementar en maxitérminos y las variables que irían sin complementar en minitérminos van complementadas en maxitérminos.
- La función simplificada resultante es un producto de sumas (POS, PdS) (en minitérminos era una suma de productos).

Función en 2FC: $f(a,b)=M_1 \cdot M_3$



Función simplificada: $f(a,b)=\overline{b}$

➲ Ejemplo, parte 2: simplificar por maxitérminos la misma función f(a,b,c,d), cuya tabla de verdad era:

a	b	c	d	f
0	0	0	0	X
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	X
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

Construimos el mapa de V-K:

ab \ cd	00	01	11	10
00	X	0	0	1
01	0	X	0	1
11	1	0	1	0
10	1	0	0	1

Paso 1: tomar los 0 que no puedan formar subcubos con otras casillas

Paso 1: tomar los 0 que no puedan formar subcubos con otras casillas.

ab \ cd	00	01	11	10
00	X	0	0	1
01	0	X	0	1
11	1	0	1	$\boxed{0}$
10	1	0	0	1

$$\boxed{S1=\overline{a}+\overline{b}+\overline{c}+d}$$

Paso 2: tomar los 0 que sólo puedan formar subcubos de 2 casillas $\Rightarrow$

Paso 2: tomar los 0 que sólo puedan formar subcubos de 2 casillas ⇒ NO HAY NINGUNO.

ab \ cd	00	01	11	10
00	X	0	0	1
01	0	X	0	1
11	1	0	1	$\boxed{0}$
10	1	0	0	1

$$\boxed{S1=\overline{a}+\overline{b}+\overline{c}+d}$$

Paso 3: tomar los 0 que no estén tomados y sólo puedan formar subcubos de 4 (y no de 8) ⇒

Paso 3: tomar los 0 que no estén tomados y sólo puedan formar subcubos de 4 (y no de 8).

ab \ cd	00	01	11	10
00	X	0	0	1
01	0	X	0	1
11	1	0	1	0
10	1	0	0	1

$$\boxed{S1 = \bar{a} + \bar{b} + \bar{c} + d}$$

$$\boxed{S2 = b + \bar{d}}$$

¿Repetir paso 3? $\Rightarrow$

Paso 3: tomar los 0 que no estén tomados y sólo puedan formar subcubos de 4 (y no de 8).

ab \ cd	00	01	11	10
00	X	0	0	1
01	0	X	0	1
11	1	0	1	0
10	1	0	0	1

$$\boxed{S1 = \overline{a} + \overline{b} + \overline{c} + d}$$

$$\boxed{S2 = b + \overline{d}}$$

$$\boxed{S3 = a + \overline{d}}$$

¿Repetir paso 3? $\Rightarrow$

Paso 3: tomar los 0 que no estén tomados y sólo puedan formar subcubos de 4 (y no de 8).

ab \ cd	00	01	11	10
00	X	0	0	1
01	0	X	0	1
11	1	0	1	0
10	1	0	0	1

$S1 = \overline{a} + \overline{b} + \overline{c} + d$

$S2 = b + \overline{d}$

$S3 = a + \overline{d}$

$S4 = a + c$

¿Repetir paso 3? $\Rightarrow$

Paso 3: tomar los 0 que no estén tomados y sólo puedan formar subcubos de 4 (y no de 8).

ab \ cd	00	01	11	10
00	X	0	0	1
01	0	X	0	1
11	1	0	1	0
10	1	0	0	1

$S1 = \overline{a} + \overline{b} + \overline{c} + d$

$S2 = b + \overline{d}$

$S3 = a + \overline{d}$

$S4 = a + c$

$S5 = c + \overline{d}$

Paso 4: no ha lugar, porque todos los 0 ya están cogidos en algún subcubo. Ya hemos terminado. Hacemos el producto de los términos suma resultantes:

$$\boxed{f_2(a,b,c,d) = (\overline{a} + \overline{b} + \overline{c} + d) \cdot (b + \overline{d}) \cdot (a + \overline{d}) \cdot (a + c) \cdot (c + \overline{d})}$$

Puertas Lógicas

Puertas lógicas: dispositivos electrónicos capaces de implementar operadores lógicos

Para cada operación lógica (AND, OR, NOT, XOR, NAND, NOR, XNOR) existe la correspondiente puerta lógica que la materializa.

Puerta **INVERSOR (NOT)**

- Puerta **AND**
- Puerta **OR**

Puertas básicas (las tres forman un conjunto universal)

- Puerta **NAND**
- Puerta **NOR**

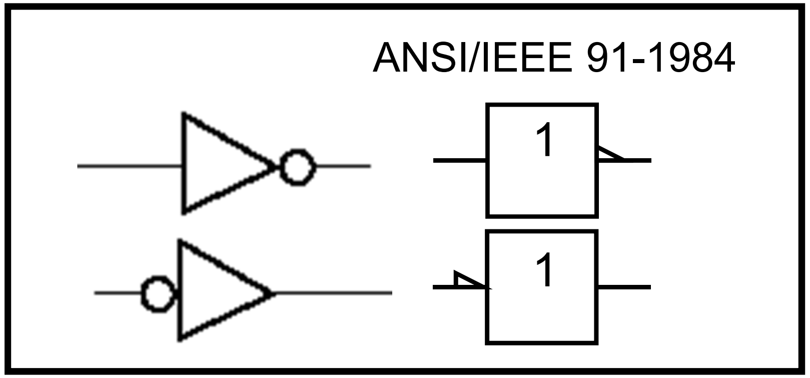
Puertas universales (cada una es conjunto universal)

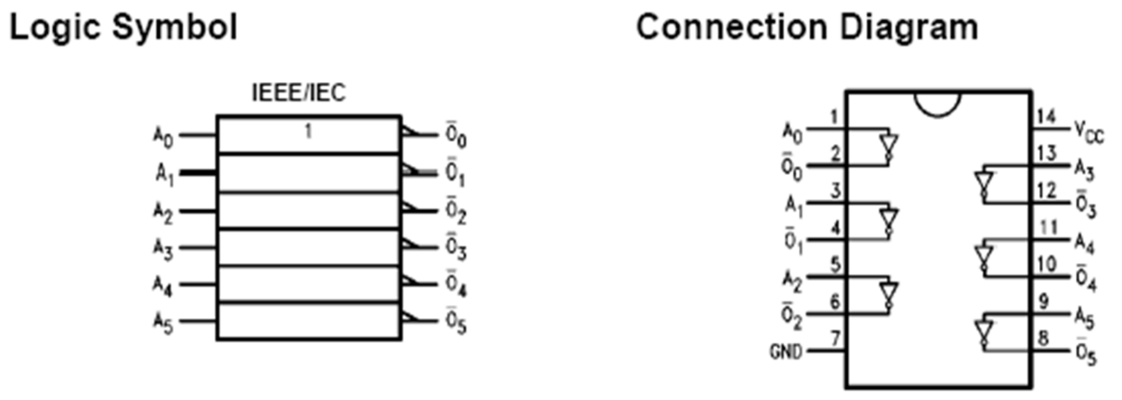
- Puerta **XOR**
- Puerta **XNOR**

Puerta NOT o inversor

- Realiza la operación lógica de **INVERSIÓN** o **COMPLEMENTACIÓN**: cambia un nivel lógico al nivel opuesto.
- Expresión lógica: $S = \overline{A}$
- Tabla de verdad:

A	S
0	1
1	0



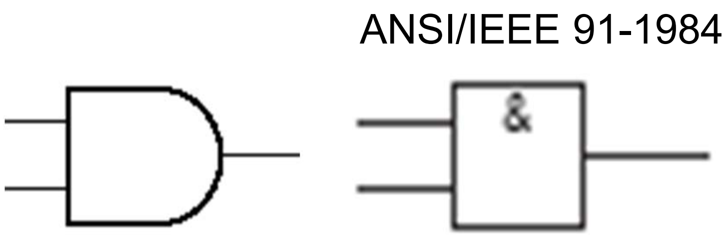


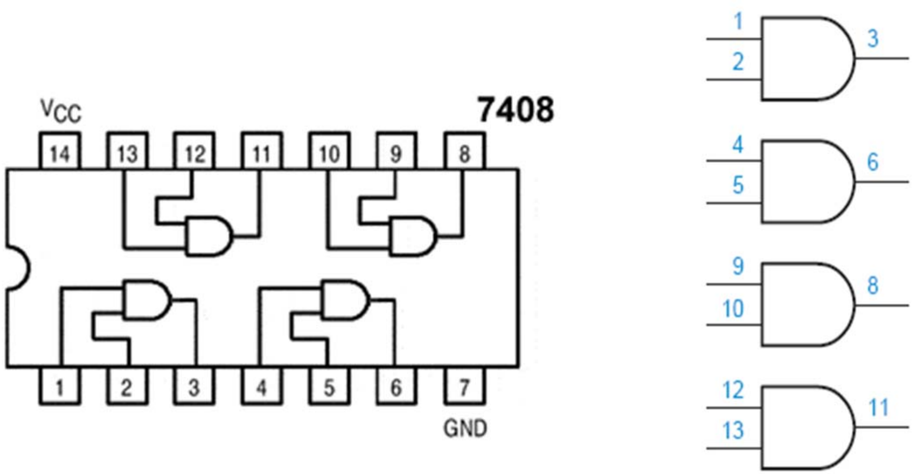
Circuito comercial: 74x04

Puerta AND

- Realiza la operación lógica de **MULTIPLICACIÓN LÓGICA**
- Expresión lógica: $S = A \cdot B$
- Tabla de verdad (dos entradas):

A	B	S
0	0	0
0	1	0
1	0	0
1	1	1





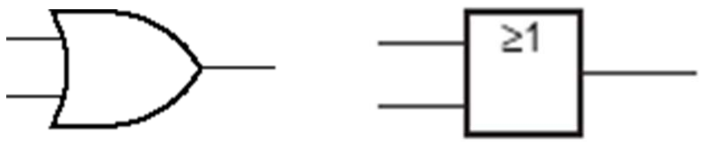
Circuito comercial: 7408 (TTL), 4081 (CMOS)

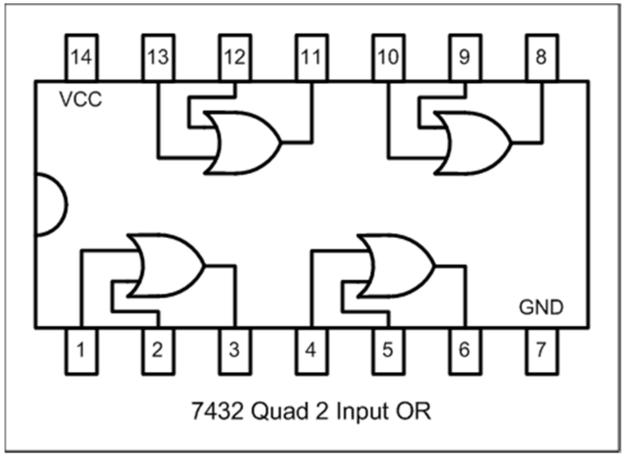
Puerta OR

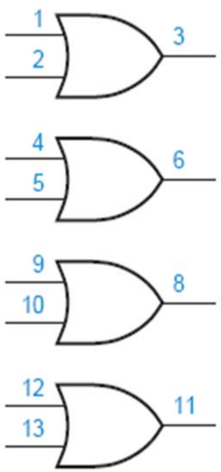
- Realiza la operación lógica de **SUMA LÓGICA**
- Expresión lógica: $S = A + B$
- Tabla de verdad:

A	B	S
0	0	0
0	1	1
1	0	1
1	1	1

ANSI/IEEE 91-1984



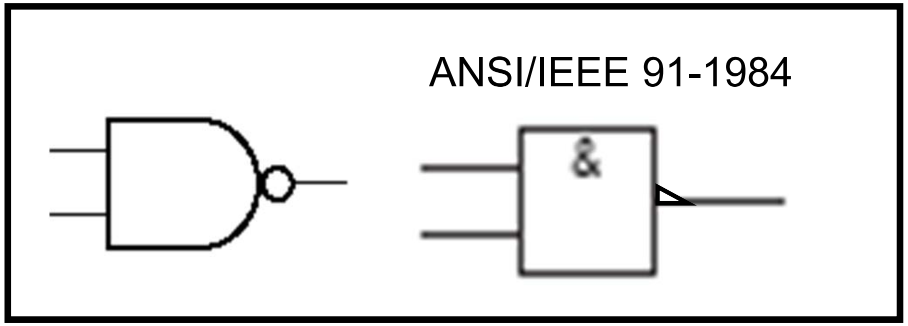




Circuito comercial: 7432 (TTL), 4071 (CMOS)

Puerta NAND

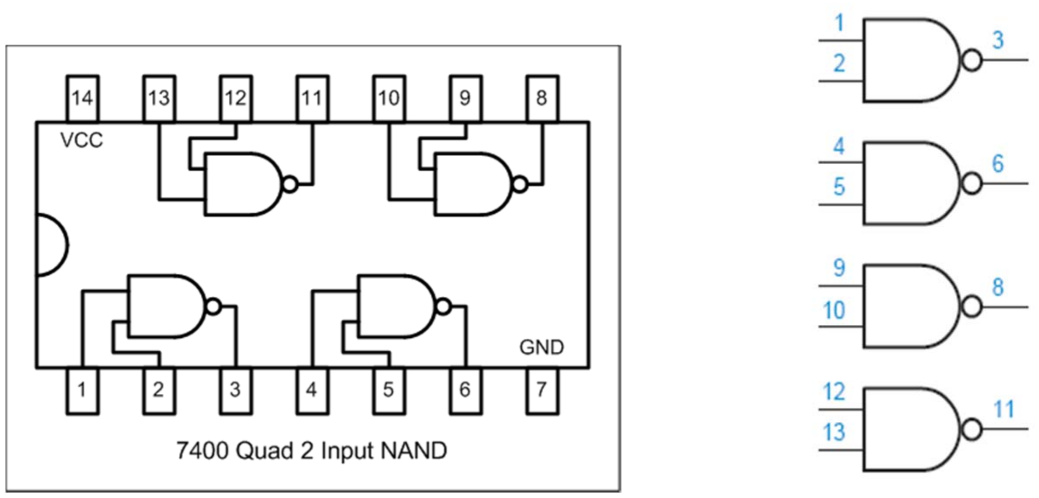
- Realiza la operación lógica de **NOT-AND : una función AND con salida complementada**



- Expresión lógica: $S = \overline{A \cdot B}$

- Tabla de verdad:

A	B	S
0	0	1
0	1	1
1	0	1
1	1	0

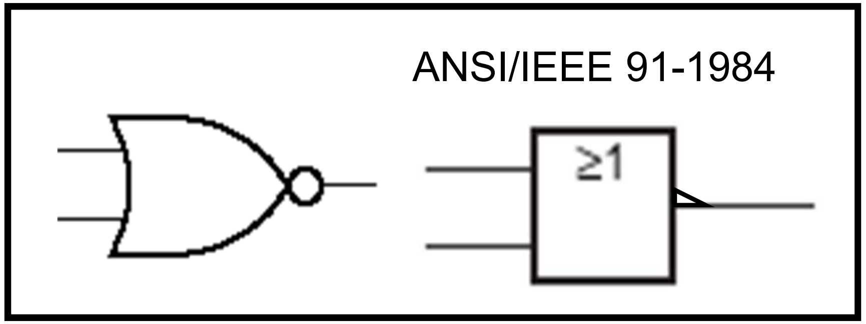


Circuito comercial: 7400 (TTL), 4011 (CMOS)

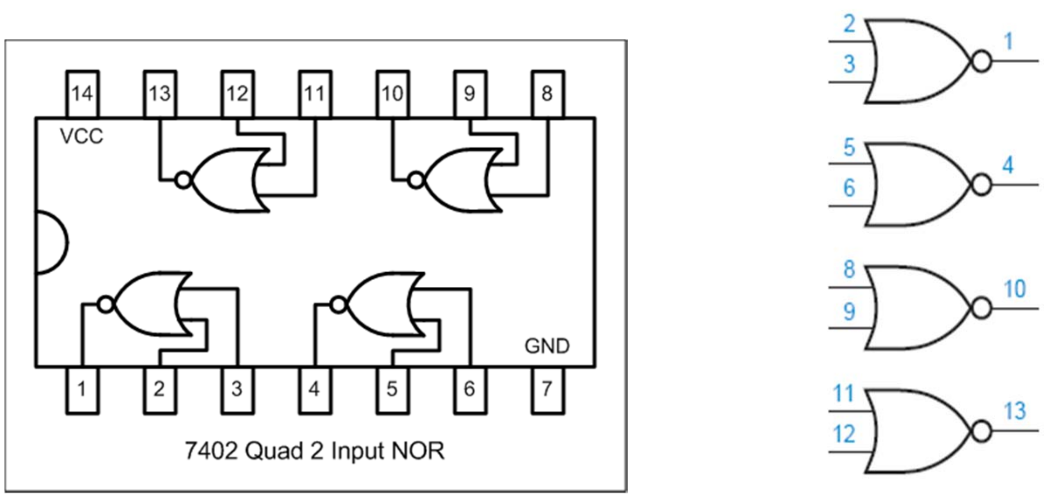
- **Puerta universal:** las puertas NAND pueden generar cualquiera de las puertas básicas NOT, AND, OR.

Puerta NOR

- Realiza la operación lógica de **NOT-OR : una función OR con salida complementada**.
- Expresión lógica: $S = \overline{A + B}$
- Tabla de verdad:



A	B	S
0	0	1
0	1	0
1	0	0
1	1	0

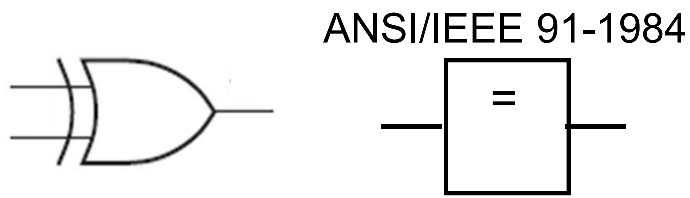


Circuito comercial: 7402 (TTL), 4001 (CMOS)

- **Puerta universal:** las puertas NOR pueden generar cualquiera de las puertas básicas NOT, AND, OR.

Puerta XOR (OR-exclusiva)

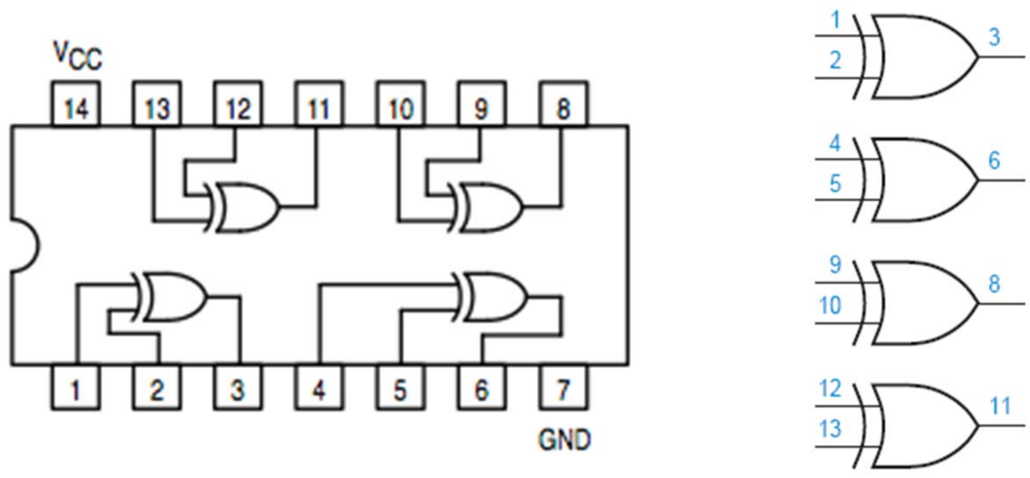
- La salida de una puerta OR-exclusiva se pone a nivel alto sólo cuando hay un nº impar de entradas a nivel alto. En el caso particular de una puerta con dos entradas, la salida estará a nivel ALTO cuando las entradas tengan niveles lógicos opuestos.



- Expresión lógica: $S = A \oplus B$

- Tabla de verdad:

A	B	S
0	0	0
0	1	1
1	0	1
1	1	0

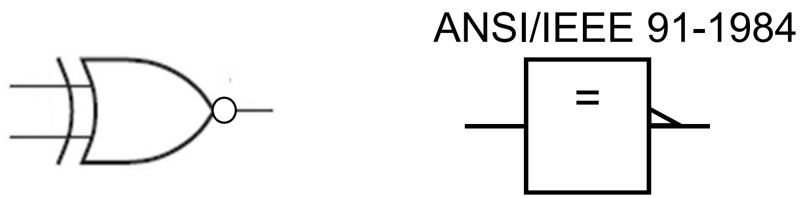


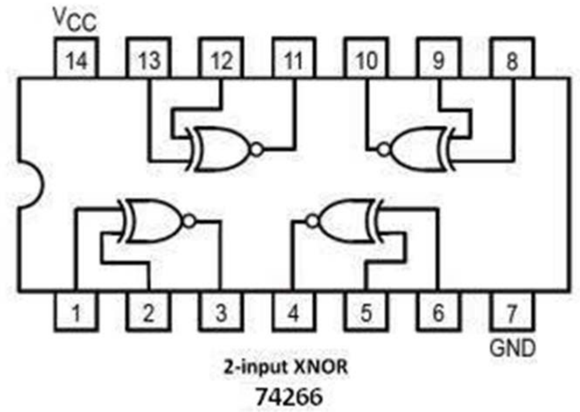
Circuito comercial: 7486 (TTL), 4030-4070 (CMOS)

Puerta XNOR

- Función OR-exclusiva con la salida complementada
- Expresión lógica: $S = \overline{A \oplus B}$
- Tabla de verdad:

A	B	S
0	0	1
0	1	0
1	0	0
1	1	1



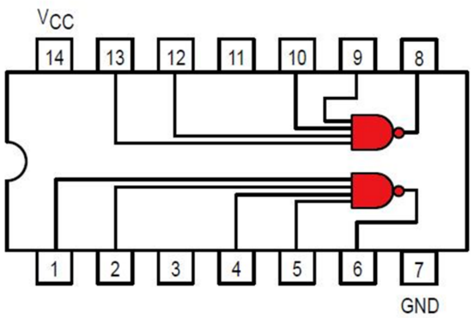


Circuito comercial: 7486 (TTL), 4077 (CMOS)

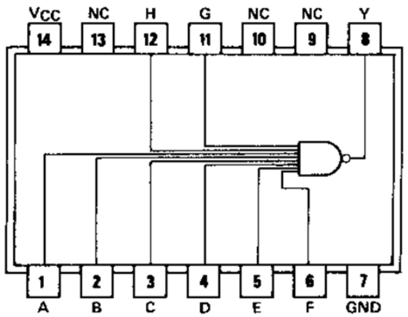
Puertas lógicas de más de dos entradas

- Existen puertas lógicas de más de dos entradas, cuyo comportamiento es equivalente al de dos entradas:

7420 - Dual 4-Input NAND Gate

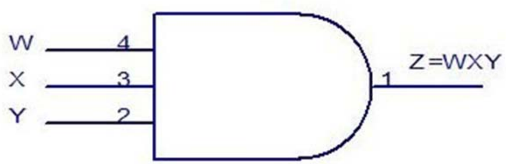


7430



3 Input AND Gate

AND Gate

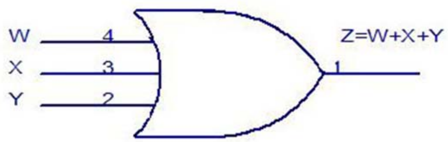


TRUTH TABLE

INPUTS			OUTPUT
W	X	Y	Z
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

3 Input OR Gate

OR Gate



TRUTH TABLE

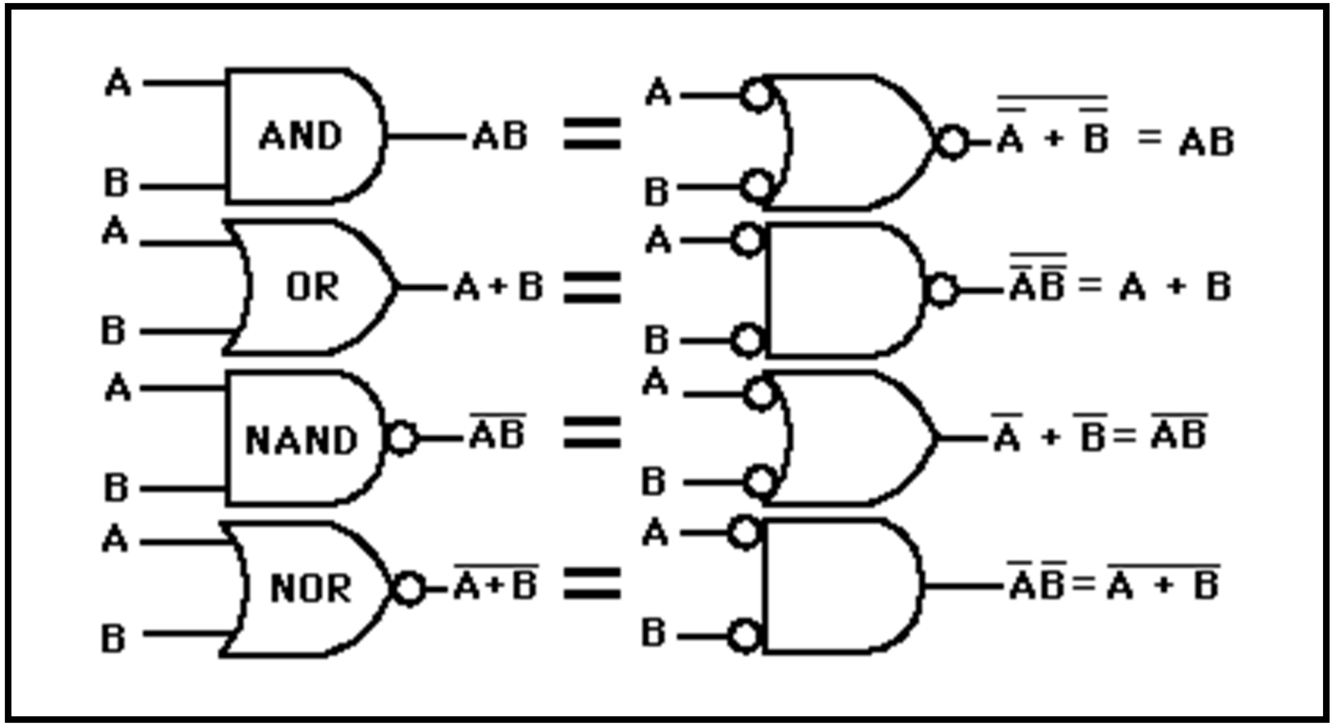
INPUTS			OUTPUT
W	X	Y	Z
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Equivalencia entre puertas lógicas

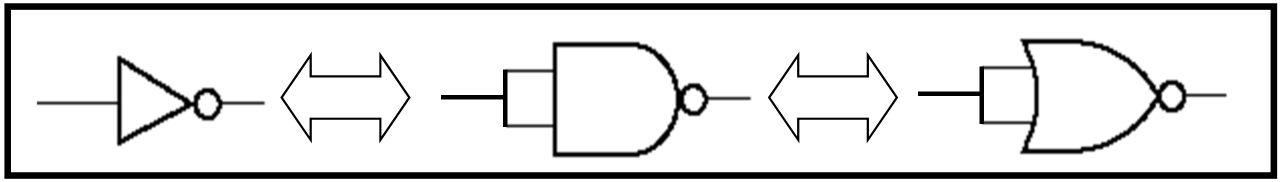
➢ Usando las leyes de Morgan se pueden demostrar las siguientes equivalencias:

$$\overline{A+B}=\overline{A}\cdot\overline{B}$$

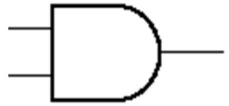
$$\overline{A\cdot B}=\overline{A}+\overline{B}$$



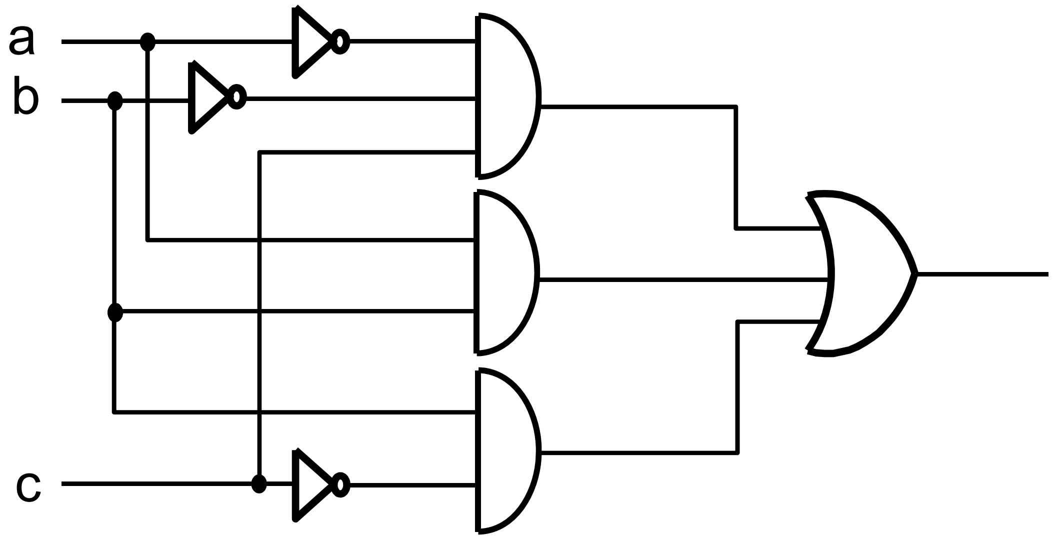
➢ Para la puerta NOT:



Implementación con puertas AND-OR-NOT

- La síntesis o implementación de cualquier sistema digital especificado mediante expresiones de conmutación usando puertas lógicas es directa:
  - Cada negación se implementa con un inversor 
  - Cada operador "·" se implementa con una puerta AND 
  - Cada operador "+" se implementa con una puerta OR 
- Ejemplo:

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



- **Fundamentos de Sistemas Digitales (Digital Fundamentals)**. T.L.Floyd. Prentice-Hall.
- **Fundamentos de computadores**. R. Hermida. Editorial Síntesis, 1998.
- **Sistemas Digitales y Tecnología de Computadores**. J. García Zubía. Paraninfo 2007
- **Problemas resueltos de electrónica digital.** J. García Zubía. Paraninfo 2003
- **Digital Design. Principles & Practices.** J.F. Wakerly. Prentice Hall. Third Edition updated.
- **Electronics: A system Approach.** Fourth Edition. Neil Storey. Prentice Hall.
- **Introducción al diseño lógico digital.** J.P. Hayes, Addison-Wesley

BIBLIOGRAFÍA