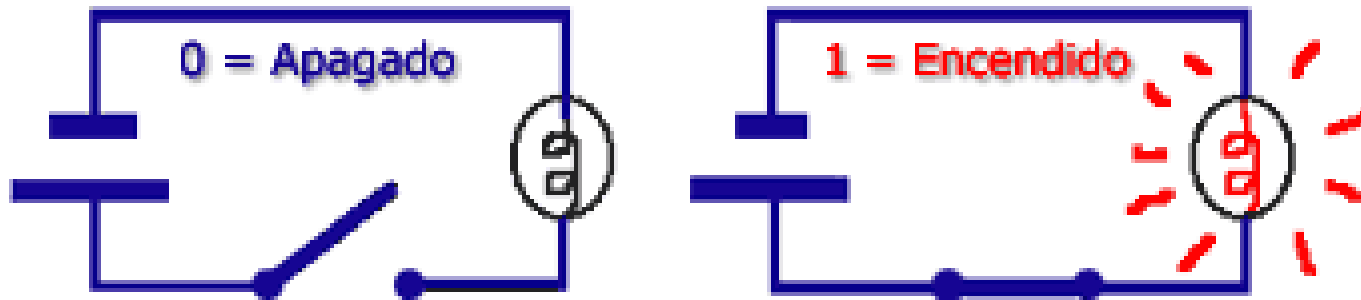


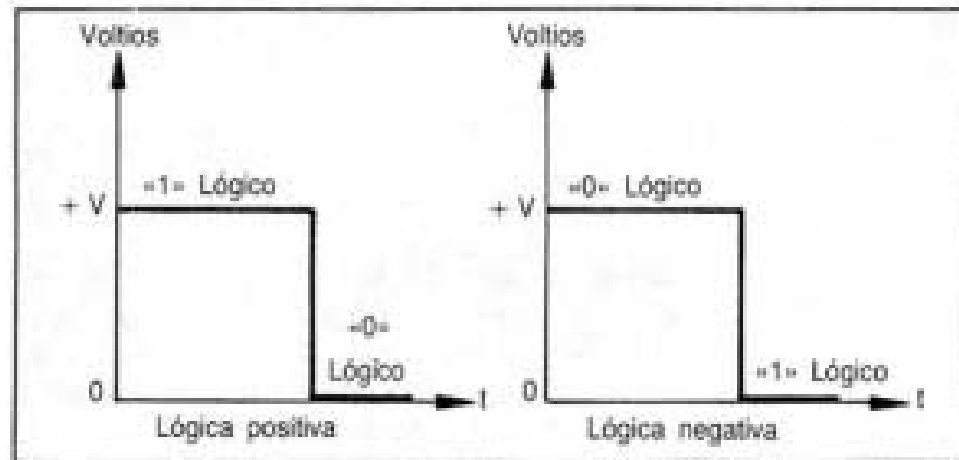
TEMA 1

SISTEMAS Y CÓDIGOS DE NUMERACIÓN. ALGEBRA DE BOOLE

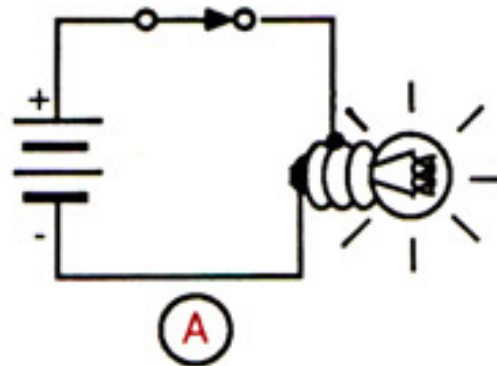
Uno de los motivos para utilizar el sistema binario en electrónica es que permite representar el estado de los componentes de un circuito eléctrico para representar matemáticamente el funcionamiento del circuito.

La idea básica del código binario

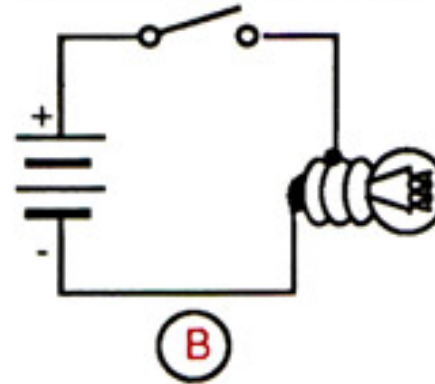




Interrupor cerrado = 1



Interrupor abierto = 0



1. SISTEMAS DE NUMERACIÓN

- a) Sistema decimal
- b) Sistema binario
- c) Sistema octal
- d) Sistema hexadecimal

1. Sistemas de numeración

A. Sistema decimal.

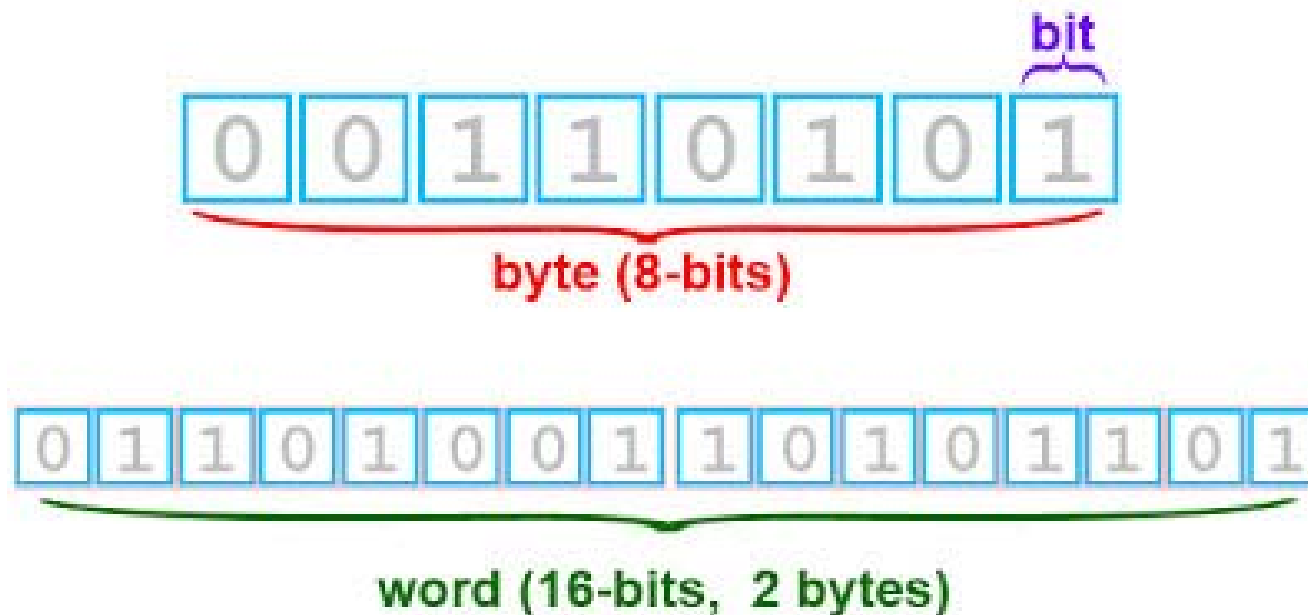
Es aquel sistema de numeración que tiene como base diez dígitos. Es decir su base es 10. Los pesos de los dígitos son potencia de 10.

$$2009 = 2000 + 9 = 2 \times 1000 + 9 \times 1 = 2 \times 10^3 + 9 \times 10^0 = 2009$$

B. Sistema binario.

Es aquel sistema de numeración que nos permite representar cualquier número utilizando sólo dos dígitos 0 y 1. Los pesos de los dígitos son potencias de 2.

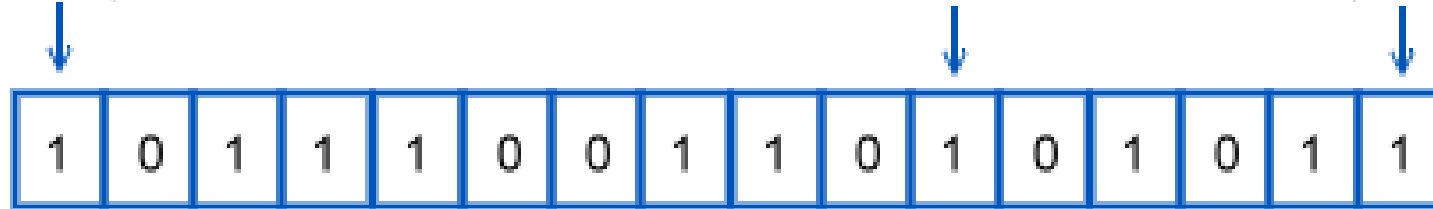
Se llama **bit** (acrónimo de Binary digit) a un dígito del sistema de numeración binario



Most Significant Bit
(MSB)

Bit

Least Significant Bit
(LSB)



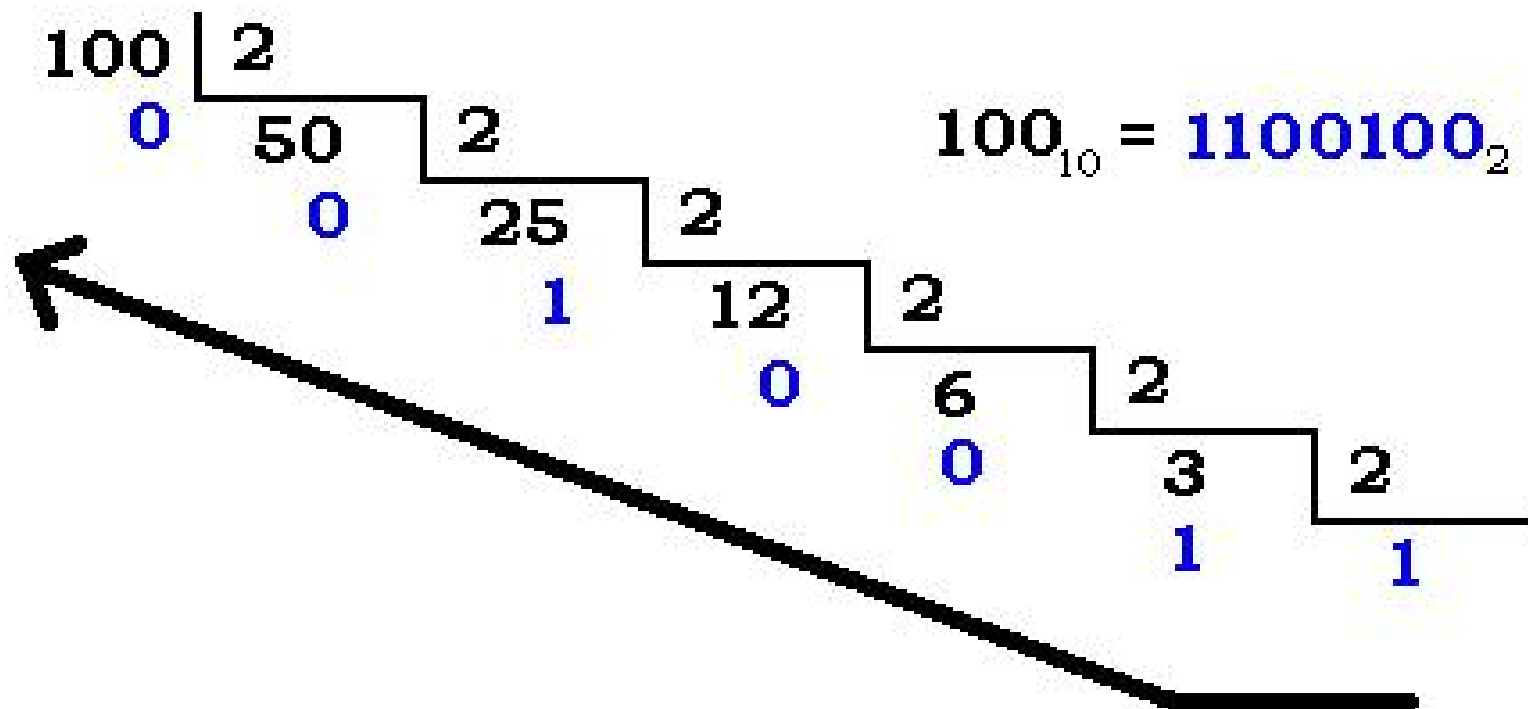
Conversión binario-decimal

$$101001 = 1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 2^5 + 2^3 + 2^0 = 41$$

Conversión decimal-binario

785 | 2
18 392
05 19 | 2
① 12 196 | 2
① 12 98 | 2
① 12 49 | 2
① 12 24 | 2
① 12 12 | 2
① 6 6 | 2
① 3 3 | 2
① 1 1

785 = 1100010001₂



Conversión Binario \rightarrow Decimal * Método rápido

1 0 0 1 0 1

↓ ↓ ↓ ↓ ↓ ↓

32 16 8 4 2 1

Sumamos sólo los que tienen un “1”.

Por tanto: 1 0 0 1 0 1 = 32 + 4 + 1 = 37

156₁₀

128	64	32	16	8	4	2	1
-----	----	----	----	---	---	---	---

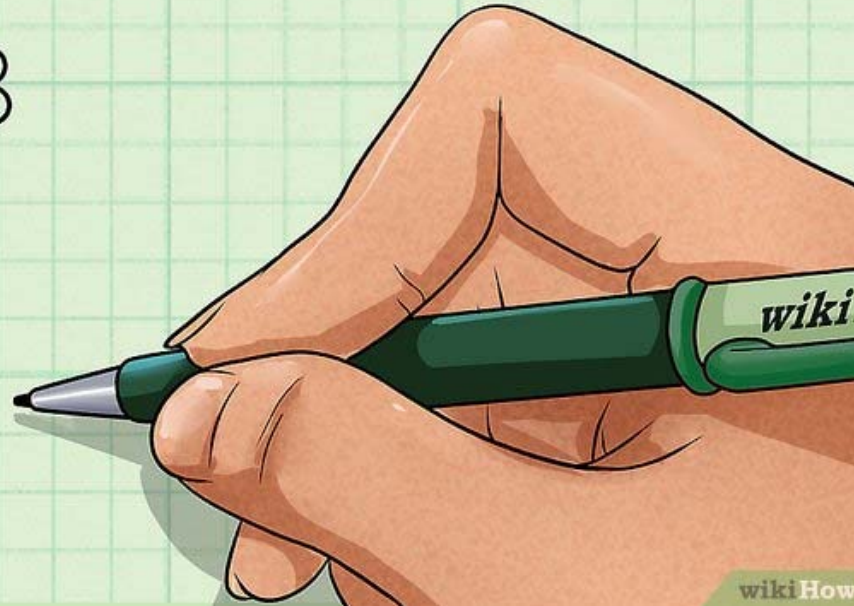
1 0 0 1 1 1 0 0

$$156 - 128 = 28$$

$$28 - 16 = 12$$

$$12 - 8 = 4$$

$$4 - 4 = 0$$




C. Sistema octal.

Es aquel sistema de numeración que nos permite representar cualquier número utilizando sólo ocho dígitos 0, 1, 2, 3, 4, 5, 6 y 7. **Los pesos de los dígitos son potencias de 8.**

$$352_8 = 3 \times 8^2 + 5 \times 8^1 + 2 \times 8^0 = 3 \times 64 + 5 \times 8 + 2 \times 1 = 234_{10}$$




$$\begin{array}{r} 563 \\ 03 \end{array} \begin{array}{r} \overline{)8} \\ 70 \end{array} \begin{array}{r} \overline{)8} \\ 8 \end{array} \begin{array}{r} \overline{)8} \\ 0 \end{array} \begin{array}{r} \overline{)8} \\ 1 \end{array}$$

③ ⑥ ① ①

$$563 \equiv 1063_8$$

$$\underline{563 \equiv 1000110011}$$




Conversión octal-binario, binario-octal

$$\boxed{101}\boxed{001}\boxed{011}_2 = 513_8$$

5 1 3




7 5 0

$$750_8 = \boxed{111}\boxed{101}\boxed{000}_2$$




Conversión hexadecimal-binario, binario-hexadecimal

$$\boxed{1010}\boxed{0111}\boxed{0011}_2 = A73_{16}$$

A 7 3

1 F 6

$$1F6_{16} = \boxed{0001}\boxed{1111}\boxed{0110}_2$$

2. REPRESENTACIÓN DE LOS NÚMEROS BINARIOS EN COMA FIJA Y EN COMA FLOTANTE.-

Los números reales binarios que se representen en coma fija están formados por dos partes separadas por una coma o un punto, que ocupa una posición fija, para lo que tendrá un número fijo de bits tanto la parte entera como la parte decimal.

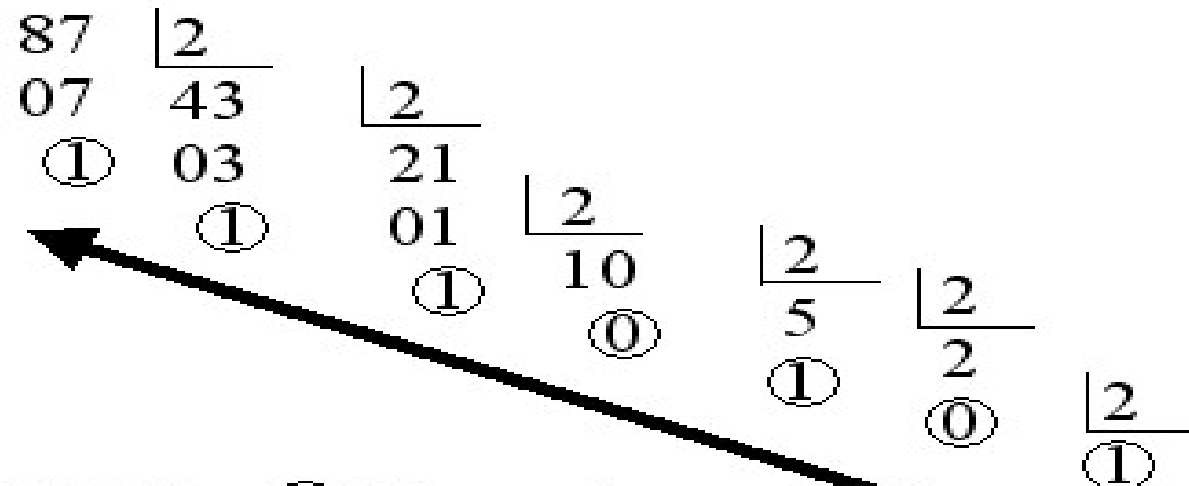
$$1010.10_2 = 2^3 \cdot 1 + 2^2 \cdot 0 + 2^1 \cdot 1 + 2^0 \cdot 0 + 2^{-1} \cdot 1 + 2^{-2} \cdot 0 = 10'5$$

1×2^3	1×2^2	0×2^1	1×2^0		1×2^{-1}	0×2^{-2}	1×2^{-3}	1×2^{-4}
1	1	0	1	.	1	0	1	1
8	4	0	1		0.5	0	0.125	0.0625

Binary point

$$8 + 4 + 0 + 1 + 0.5 + 0 + 0.125 + 0.0625 = 13.6875 \text{ (Base 10)}$$

87.36



0.36 X 2 = ①.72
 0.72 X 2 = ①.44
 0.44 X 2 = ①.88
 0.88 X 2 = ①.76
 0.76 X 2 = ①.52

87.36 = 1010111.01011 ...₂

3. CÓDIGOS DE NUMERACIÓN

A. Códigos ponderados

B. Códigos no ponderados

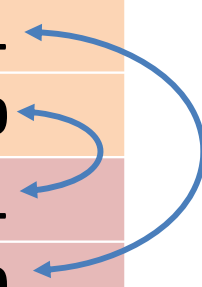
C. Códigos detectores y correctores de información

CÓDIGOS DE NUMERACIÓN

A. Códigos ponderados

DECIMAL	BCD NATURAL	BCD AIKEN
	$P_1P_2P_3P_4$	$P_1P_2P_3P_4$
	8 4 2 1	2 4 2 1
0	0 0 0 0	0 0 0 0
1	0 0 0 1	0 0 0 1
2	0 0 1 0	0 0 1 0
3	0 0 1 1	0 0 1 1
4	0 1 0 0	0 1 0 0
5	0 1 0 1	1 0 1 1
6	0 1 1 0	1 1 0 0
7	0 1 1 1	1 1 0 1
8	1 0 0 0	1 1 1 0
9	1 0 0 1	1 1 1 1

DECIMAL	BCD AIKEN
	$P_1 P_2 P_3 P_4$
	2 4 2 1
0	0 0 0 0
1	0 0 0 1
2	0 0 1 0
3	0 0 1 1
4	0 1 0 0
5	1 0 1 1
6	1 1 0 0
7	1 1 0 1
8	1 1 1 0
9	1 1 1 1



La razón de la codificación Aiken es la de conseguir simetría entre ciertos números.

Ver la simetría en el código Aiken correspondiente a los decimales: 4 y 5, 3 y 6, 2 y 7, 1 y 8, 0 y 9.

Ejemplo: 3 (0011) y 6 (1100).

El código Aiken es muy útil para realizar operaciones de resta y división

B. Códigos no ponderados

DECIMAL	BCD EXCESO 3
0	0011
1	0100
2	0101
3	0110
4	0111
5	1000
6	1001
7	1010
8	1011
9	1100

DECIMAL	JOHNSON DE 5 BITS
0	00000
1	00001
2	00011
3	00111
4	01111
5	11111
6	11110
7	11100
8	11000
9	10000

DECIMAL	GRAY
0	0000
1	0001
2	0011
3	0010
4	0110
5	0111
6	0101
7	0100
8	1100
9	1101
10	1111
11	1110
12	1010
13	1011
14	1001
15	1000

Un código binario es **continuo** si sus combinaciones de bits o palabras código correspondiente a números decimales consecutivos difieren en un solo bit, es decir, son adyacentes.

Si además se cumple que la última combinación es adyacente a la primera, se denomina **cíclico**.

El código GRAY ó reflejado es binario continuo y cíclico.

El código de Gray de n bits, se forma a partir del código de Gray de $n-1$ bits, reflejando éste a partir de una línea horizontal y rellenando por encima de la línea con ceros a la izquierda, y por debajo con unos.

Es adecuado para realización de contadores con registros de desplazamiento.

DECIMAL	GRAY
0	00
1	01
2	11
3	10

DECIMAL	GRAY
0	000
1	001
2	011
3	010
4	110
5	111
6	101
7	100

El **código exceso 3** al igual que el código Aiken cumple con la misma característica de simetría.

Ver la simetría en el código exceso 3 correspondiente a los decimales: 4 y 5, 3 y 6, 2 y 7, 1 y 8, 0 y 9

Es un código muy útil en las operaciones de resta y división.

Código Johnson se utiliza en el control de sistemas digitales sencillos de muy alta velocidad por ser continuo y cíclico

En el **código gray** solo un bit puede cambiar a la vez, es utilizado para obtener funciones lógicas de Minterminos y Maxterminos, para circuitos secuenciales se caracteriza porque cambia un solo bit por conteo.

El código ASCII – www.elCodigoASCII.com.ar

sigla en inglés de American Standard Code for Information Interchange
(Código Estadounidense Estándar para el Intercambio de Información)

Caracteres de control ASCII			
DEC	HEX	Simbolo ASCII	
00	00h	NULL	(carácter nulo)
01	01h	SOH	(inicio encabezado)
02	02h	STX	(inicio texto)
03	03h	ETX	(fin de texto)
04	04h	EOT	(fin transmisión)
05	05h	ENQ	(enquiry)
06	06h	ACK	(acknowledgement)
07	07h	BEL	(timbre)
08	08h	BS	(retroceso)
09	09h	HT	(tab horizontal)
10	0Ah	LF	(salto de línea)
11	0Bh	VT	(tab vertical)
12	0Ch	FF	(form feed)
13	0Dh	CR	(retorno de carro)
14	0Eh	SO	(shift Out)
15	0Fh	SI	(shift In)
16	10h	DLE	(data link escape)
17	11h	DC1	(device control 1)
18	12h	DC2	(device control 2)
19	13h	DC3	(device control 3)
20	14h	DC4	(device control 4)
21	15h	NAK	(negative acknowle.)
22	16h	SYN	(synchronous idle)
23	17h	ETB	(end of trans. block)
24	18h	CAN	(cancel)
25	19h	EM	(end of medium)
26	1Ah	SUB	(substitute)
27	1Bh	ESC	(escape)
28	1Ch	FS	(file separator)
29	1Dh	GS	(group separator)
30	1Eh	RS	(record separator)
31	1Fh	US	(unit separator)
127	20h	DEL	(delete)

Caracteres ASCII imprimibles								
DEC	HEX	Simbolo	DEC	HEX	Simbolo	DEC	HEX	Simbolo
32	20h	espacio	64	40h	@	96	60h	`
33	21h	!	65	41h	A	97	61h	a
34	22h	"	66	42h	B	98	62h	b
35	23h	#	67	43h	C	99	63h	c
36	24h	\$	68	44h	D	100	64h	d
37	25h	%	69	45h	E	101	65h	e
38	26h	&	70	46h	F	102	66h	f
39	27h	'	71	47h	G	103	67h	g
40	28h	(72	48h	H	104	68h	h
41	29h)	73	49h	I	105	69h	i
42	2Ah	*	74	4Ah	J	106	6Ah	j
43	2Bh	+	75	4Bh	K	107	6Bh	k
44	2Ch	,	76	4Ch	L	108	6Ch	l
45	2Dh	-	77	4Dh	M	109	6Dh	m
46	2Eh	.	78	4Eh	N	110	6Eh	n
47	2Fh	/	79	4Fh	O	111	6Fh	o
48	30h	0	80	50h	P	112	70h	p
49	31h	1	81	51h	Q	113	71h	q
50	32h	2	82	52h	R	114	72h	r
51	33h	3	83	53h	S	115	73h	s
52	34h	4	84	54h	T	116	74h	t
53	35h	5	85	55h	U	117	75h	u
54	36h	6	86	56h	V	118	76h	v
55	37h	7	87	57h	W	119	77h	w
56	38h	8	88	58h	X	120	78h	x
57	39h	9	89	59h	Y	121	79h	y
58	3Ah	:	90	5Ah	Z	122	7Ah	z
59	3Bh	;	91	5Bh	[123	7Bh	{
60	3Ch	<	92	5Ch	\	124	7Ch	
61	3Dh	=	93	5Dh]	125	7Dh	}
62	3Eh	>	94	5Eh	^	126	7Eh	~
63	3Fh	?	95	5Fh	-	elCodigoASCII.com.ar		

ASCII extendido											
DEC	HEX	Simbolo	DEC	HEX	Simbolo	DEC	HEX	Simbolo	DEC	HEX	Simbolo
160	A0h	á	192	C0h	Ł	224	E0h	Ó	256	100h	À
161	A1h	í	193	C1h	ł	225	E1h	ô	257	101h	Á
162	A2h	ó	194	C2h	Ł	226	E2h	Ô	258	102h	Â
163	A3h	ú	195	C3h	ł	227	E3h	Õ	259	103h	Ã
164	A4h	ñ	196	C4h	Ł	228	E4h	ö	260	104h	Ä
165	A5h	Ñ	197	C5h	ł	229	E5h	Ö	261	105h	Å
166	A6h	ª	198	C6h	Ł	230	E6h	µ	262	106h	Æ
167	A7h	º	199	C7h	Ł	231	E7h	þ	263	107h	Ç
168	A8h	¿	200	C8h	Ł	232	E8h	ß	264	108h	È
169	A9h	®	201	C9h	Ł	233	E9h	Ü	265	109h	É
170	AAh	¬	202	CAh	Ł	234	EAh	Ù	266	10Ah	Ê
171	ABh	½	203	CBh	Ł	235	EBh	Ú	267	10Bh	Ë
172	ACH	¼	204	CCh	Ł	236	ECh	Ý	268	10Ch	Ï
173	ADh	¡	205	CDh	Ł	237	EDh	Ý	269	10Dh	Ï
174	Aeh	«	206	CEh	Ł	238	EEh	·	270	10Eh	Ï
175	Afh	»	207	CFh	Ł	239	EFh	·	271	10Fh	Ï
176	B0h	⋮	208	D0h	Đ	240	F0h	±	272	110h	Ï
177	B1h	⋮	209	D1h	Đ	241	F1h	±	273	111h	Ï
178	B2h	⋮	210	D2h	Đ	242	F2h	±	274	112h	Ï
179	B3h	⋮	211	D3h	Đ	243	F3h	±	275	113h	Ï
180	B4h	⋮	212	D4h	Đ	244	F4h	±	276	114h	Ï
181	B5h	⋮	213	D5h	Đ	245	F5h	±	277	115h	Ï
182	B6h	⋮	214	D6h	Đ	246	F6h	±	278	116h	Ï
183	B7h	⋮	215	D7h	Đ	247	F7h	±	279	117h	Ï
184	B8h	⋮	216	D8h	Đ	248	F8h	±	280	118h	Ï
185	B9h	⋮	217	D9h	Đ	249	F9h	±	281	119h	Ï
186	BAh	⋮	218	DAh	Đ	250	FAh	±	282	120h	Ï
187	BBh	⋮	219	DBh	Đ	251	FBh	±	283	121h	Ï
188	BCh	⋮	220	DCh	Đ	252	FCh	±	284	122h	Ï
189	BDh	⋮	221	DDh	Đ	253	FDh	±	285	123h	Ï
190	BEh	⋮	222	DEh	Đ	254	FEh	±	286	124h	Ï
191	BFh	⋮	223	DFh	Đ	255	FFh	±	287	125h	Ï

C. Códigos detectores y correctores de información

7 bits de datos	byte con bit de paridad	
	par	impar
0000000	0000000 0	0000000 1
1010001	1010001 1	1010001 0
1101001	1101001 0	1101001 1
1111111	1111111 1	1111111 0

Código Hamming 7.4

	Posición						
	1	2	3	4	5	6	7
	p_1	p_2	m_1	p_3	m_2	m_3	m_4
Mensaje original en BCD			0		1	0	0
Paridad par en 1, comprueba 3, 5, 7 implica $p_1=1$	1	0	0	1	1	0	0
Paridad par en 2, comprueba 3, 6, 7 implica $p_2=0$	1	0	0	1	1	0	0
Paridad par en 4, comprueba 5, 6, 7 implica $p_3=1$	1	0	0	1	1	0	0

Código autocorrector de errores simples: Debe poseer la cualidad corregir errores simples, por lo que la distancia entre las palabras del código debe ser de 3.

4. ALGEBRA DE BOOLE.-

Es un conjunto de elementos A que pueden tomar dos valores perfectamente diferenciados (0 y 1), relacionados por dos operaciones binarias denominadas suma lógica (+) y producto lógico (*), y una operación unitaria complemento (')

La operación "+" se define de la siguiente manera:

$$0 + 0 = 0$$

$$0 + 1 = 1$$

$$1 + 0 = 1$$

$$1 + 1 = 1$$

La operación "." se define así:

$$0 \cdot 0 = 0$$

$$0 \cdot 1 = 0$$

$$1 \cdot 0 = 0$$

$$1 \cdot 1 = 1$$

Postulados

Para un **Álgebra de Boole** $(A, +, \cdot)$, definimos los siguientes postulados o axiomas:

1. La existencia de un elemento identidad para las dos operaciones:

$$\mathbf{X + 0 = X}$$

$$\mathbf{X \cdot 1 = X}$$

2. Propiedad conmutativa.

$$\mathbf{X + Y = Y + X}$$

$$\mathbf{X \cdot Y = Y \cdot X}$$

3. Cada operación es distributiva respecto a la otra.

$$\mathbf{X \cdot (Y + Z) = (X \cdot Y) + (X \cdot Z)}$$

$$\mathbf{X + (Y \cdot Z) = (X + Y) \cdot (X + Z)}$$

4. Existencia del elemento complementario.

$$\mathbf{X + X' = 1}$$

$$\mathbf{X \cdot X' = 0}$$

Teoremas y propiedades

Teorema 1.

$$X + 1 = 1$$

$$X \cdot 0 = 0$$

Teorema 2.

$$X + X = X$$

$$X \cdot X = X$$

Este teorema es conocido por el "**Teorema de la idempotencia**".

Teorema 3.

$$(X')' = X$$

Este teorema se conoce con el nombre de "**ley de involución**".

Teorema 4.

$$X + XY = X$$

$$X(X + Y) = X$$

Este teorema se conoce con el nombre de "**ley de absorción**".

Teorema 5.

En un algebra de Boole, las operaciones "+" y "·" son asociativas.

$$X + (Y + Z) = (X + Y) + Z$$

$$X \cdot (Y \cdot Z) = (X \cdot Y) \cdot Z$$

Teorema 6.

$$(X + Y)' = X' \cdot Y'$$

$$(X \cdot Y)' = X' + Y'$$

Este teorema, es conocido con el nombre de "**leyes de Morgan**".

5. Funciones lógicas. Formas canónicas de una función

Definimos **función lógica** a aquella función matemática compuesta por variables binarias relacionadas entre sí por las operaciones suma lógica "+" y/o producto lógico ".".

Tabla de verdad

Sea la función lógica: $F(a, b, c) = \bar{a} b \bar{c} + a . b . \bar{c}$

abc	F
000	0
001	0
010	1
011	0
100	0
101	0
110	1
111	0

Formas canónicas de una función

minitérminos: términos en los que las variables lógicas están relacionadas entre sí mediante el producto lógico " \cdot ", y éstos a su vez están sumados. *(Vulgarmente se dice que una función compuesta por minitérminos es una "función suma de productos").*

maxitérminos: términos en los que las variables lógicas están relacionadas entre sí mediante "+", y éstos a su vez están multiplicados. *(Vulgarmente se dice que una función compuesta por maxitérminos es una "función producto de sumas").*

abc	F
000	0
001	0
010	1
011	0
100	0
101	0
110	1
111	1

$$F(a, b, c) = ab'c + abc' + abc$$

$$F(a, b, c) = (a+b+c)(a+b+c')(a+b'+c')(a'+b+c)(a'+b+c')$$

FORMA NUMÉRICA DE REPRESENTAR UNA FUNCIÓN.-

$$F(a, b, c) = \sum (2, 6, 7)$$

abc	F
000	0
001	0
010	1
011	0
100	0
101	0
110	1
111	1

$$F(a, b, c) = a'bc' + abc' + abc$$

010 110 111

minitérminos

$$F(a, b, c) = (a+b+c)(a+b+c')(a+b'+c')(a'+b+c)(a'+b+c')$$

000 001 011 100 101

maxitérminos

Las funciones booleanas se tienen que simplificar al máximo, para diseñar los circuitos con el menor número de componentes electrónicos, y esta simplificación la podemos realizar de dos maneras diferentes:

- **Utilizando las propiedades y teoremas del Algebra de Boole. Se denomina método analítico de simplificación de funciones.** Hay que manejar muy bien estas propiedades para poder eliminar la mayor cantidad de términos y variables.
- **Utilizando el método de Karnaugh.** Es un método gráfico que si lo aplicamos bien, nos garantiza que obtendremos la función más simplificada posible, a partir de una tabla de verdad.

Normalmente las formas canónicas en minitérminos y maxitérminos **no** son las expresiones más simplificadas.

6. SIMPLIFICACION DE FUNCIONES LOGICAS.-

1. Método analítico de simplificación de funciones:

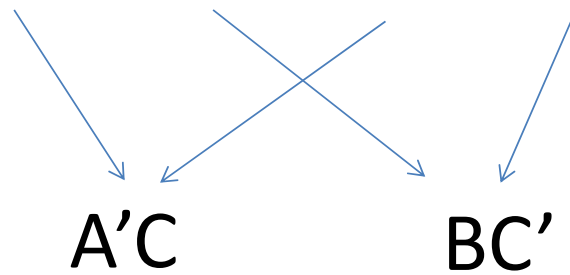
Es aquel que utiliza las propiedades y teoremas del Álgebra de Boole para realizar las simplificaciones. Es decir no es un método mecánico sino que hay que basarse en la experiencia y el conocimiento del Álgebra de Boole.

2. El Método de Karnaugh

Es un método de simplificación de funciones mecánico; es decir, no hay que tener presente ninguna ley matemática presente. Nos permitirá simplificar funciones con dos, tres, cuatro, ...variables de una forma sencilla.

1. Método analítico de simplificación de funciones:

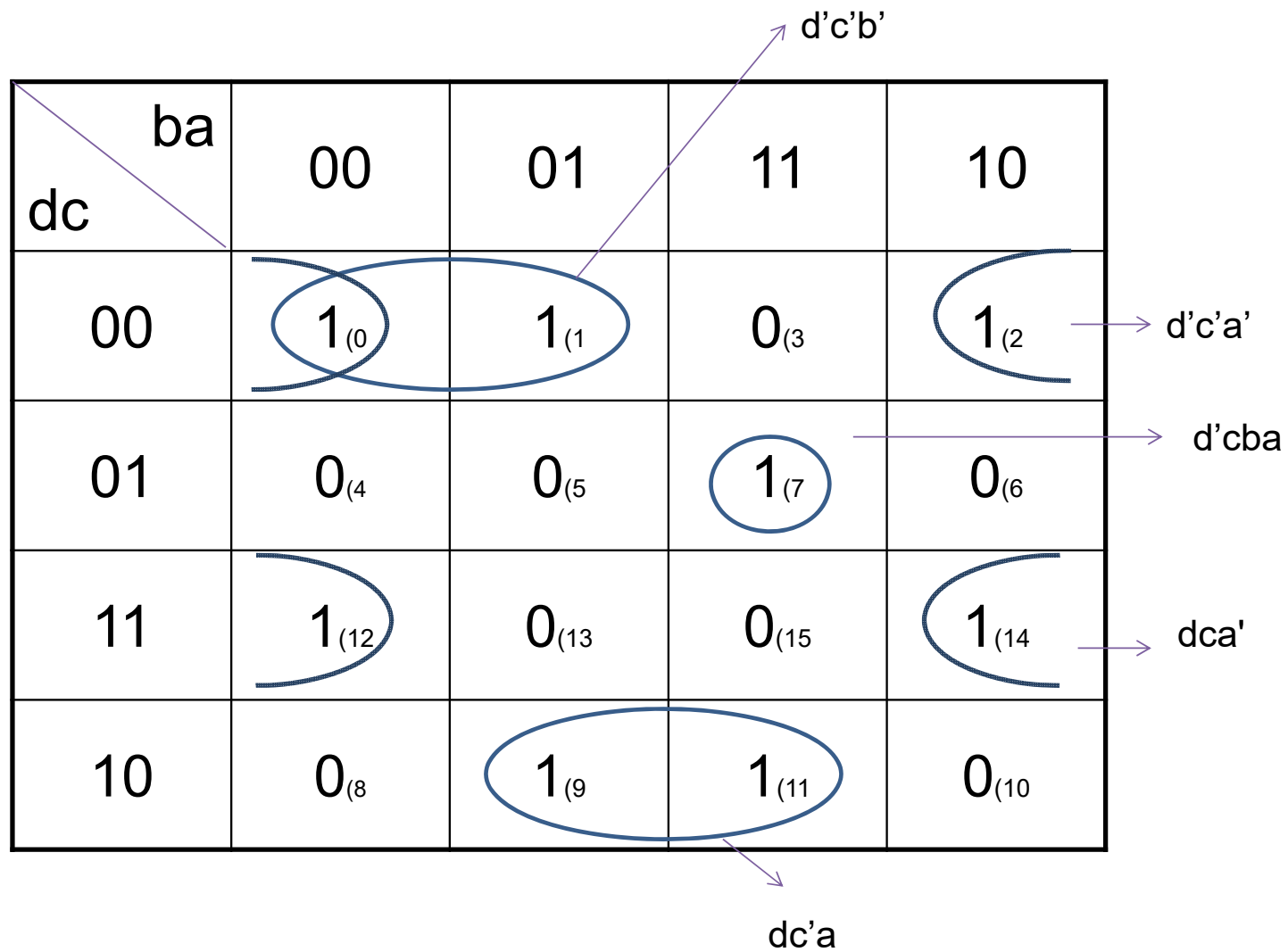
$$F(A, B, C) = A'B'C + A'BC' + A'BC + ABC'$$



2. El Método de Karnaugh

$$F(A, B, C) = A'B'C + A'BC' + A'BC + ABC'$$

<div>AB</div> <div>C</div>	00	01	11	10
0	0 ₍₀₎	1 ₍₂₎	0 ₍₆₎	0 ₍₄₎
1	1 ₍₁₎	1 ₍₃₎	0 ₍₇₎	0 ₍₅₎



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

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

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

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

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

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

5. Puertas lógicas

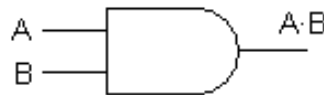
5.Puertas lógicas

Una **puerta lógica** es un dispositivo electrónico básico cuya función es la de implementar funciones lógicas.

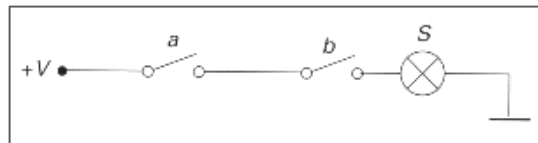
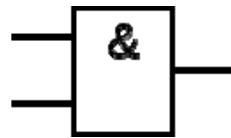
Puerta AND

Esta puerta implementa la operación "." del Algebra de Boole.

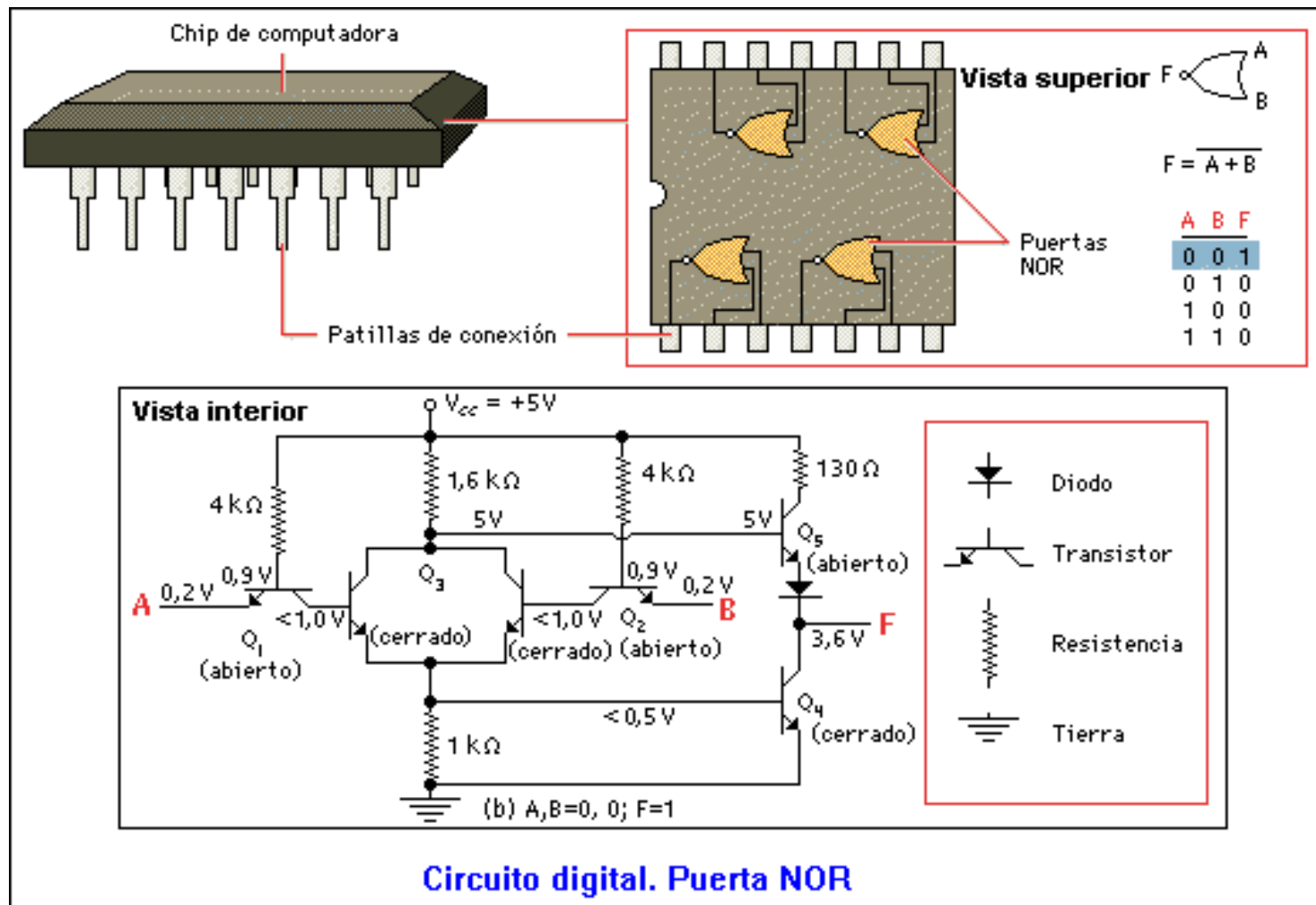
Símbolo ANSI/IEEE
91-1973
Americano



Símbolo ANSI/IEEE
91-1984
Europeo

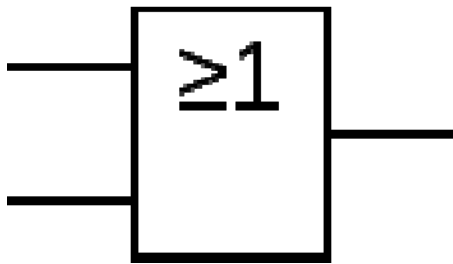
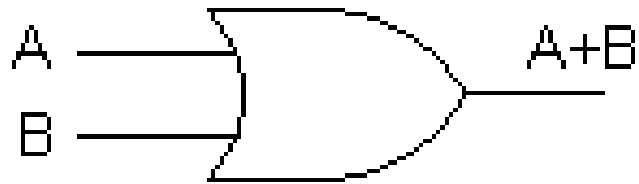


A B	$A \cdot B = F$
0 0	0
0 1	0
1 0	0
1 1	1

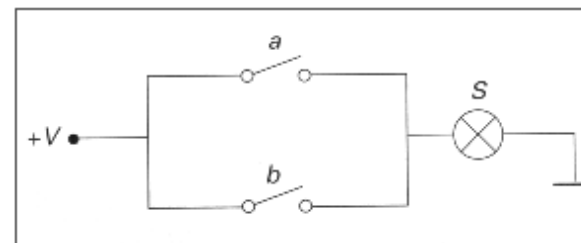


Puerta OR

Esta puerta implementa la operación "+" del Algebra de Boole.

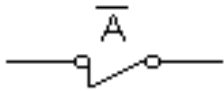


A B	A+B=F
0 0	0
0 1	1
1 0	1
1 1	1

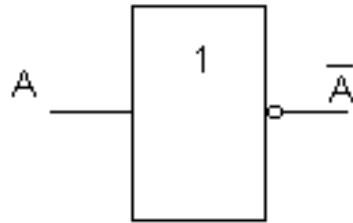


Puerta NOT (Inversor)

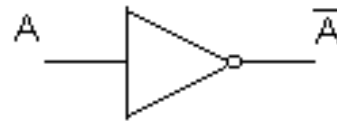
Tiene sólo una entrada y realiza la operación de negación lógica.



a)



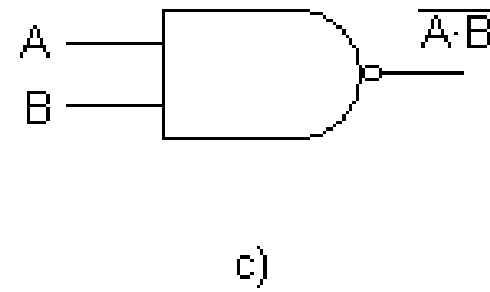
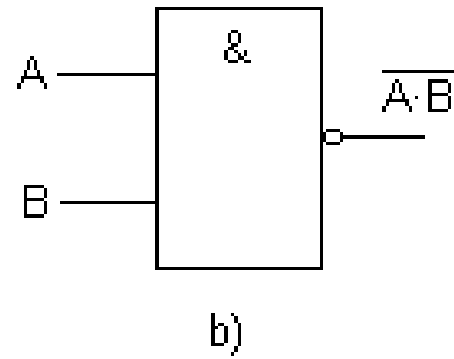
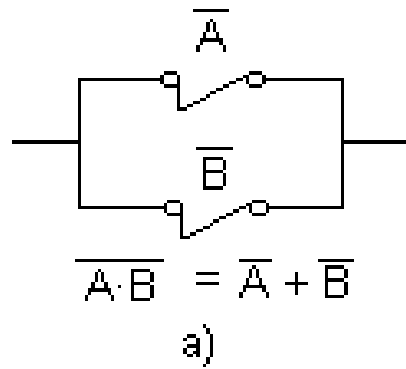
b)



c)

A	$A' = F$
0	1
1	0

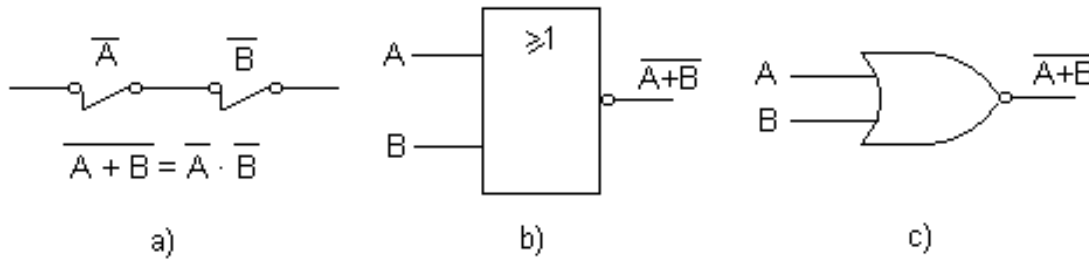
Puerta NAND



A B	$(A \cdot B)' = F$
0 0	1
0 1	1
1 0	1
1 1	0

Puerta NOR

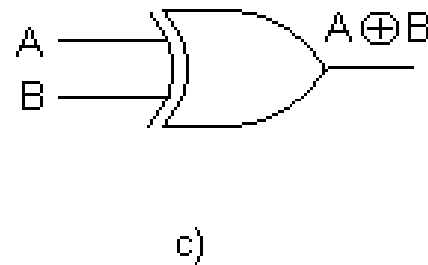
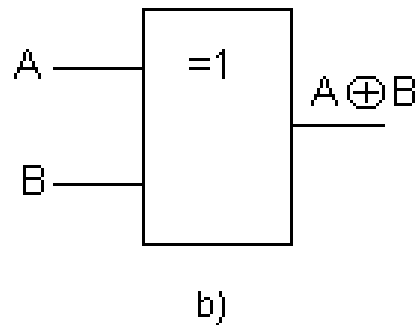
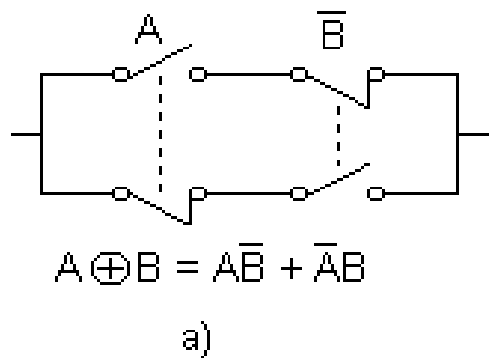
Es una puerta OR negada (NOT-OR). $F = (A + B)'$.



A B	$(A+B)'=F$
0 0	1
0 1	0
1 0	0
1 1	0

Puerta OR-exclusiva (XOR)

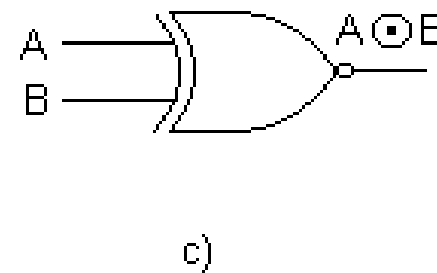
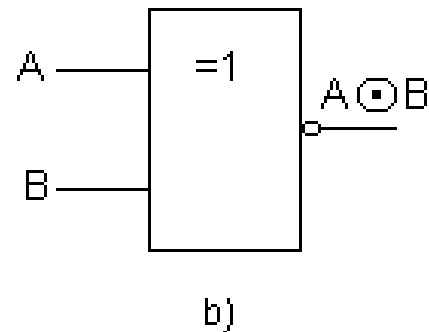
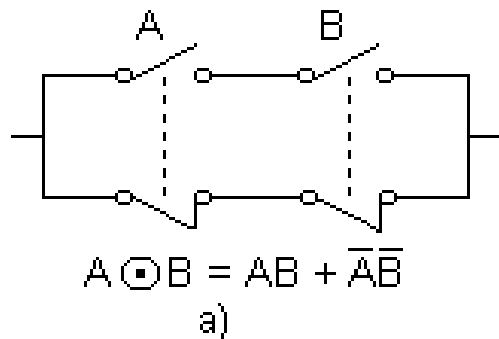
Se puede definir esta puerta como aquella que da por resultado uno, cuando los valores en las entradas son distintos, por ejemplo: 1 y 0, 0 y 1 (en una compuerta de dos entradas). Se obtiene cuando ambas entradas tienen distinto valor.



A B	F
0 0	0
0 1	1
1 0	1
1 1	0

Puerta equivalencia (XNOR)

Se puede definir esta puerta como aquella que proporciona un **1** lógico, sólo si las dos entradas son iguales, esto es, **0** y **0** ó **1** y **1** (2 encendidos o 2 apagados). Sólo es verdadero si ambos componentes tiene el mismo valor lógico



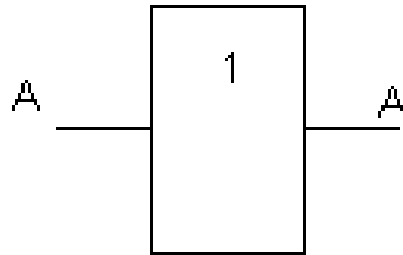
A B	F
0 0	1
0 1	0
1 0	0
1 1	1

Puerta SÍ o *Buffer*

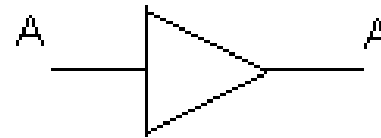
La puerta lógica **SÍ**, realiza la función booleana igualdad. En la práctica se suele utilizar como amplificador de corriente o como seguidor de tensión, para adaptar impedancias (*buffer* en inglés).



a)



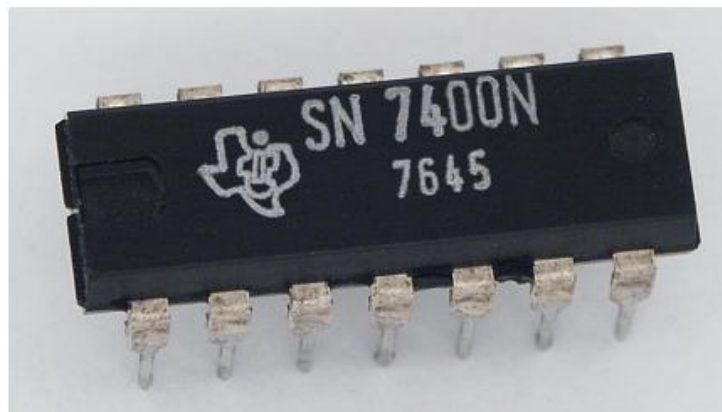
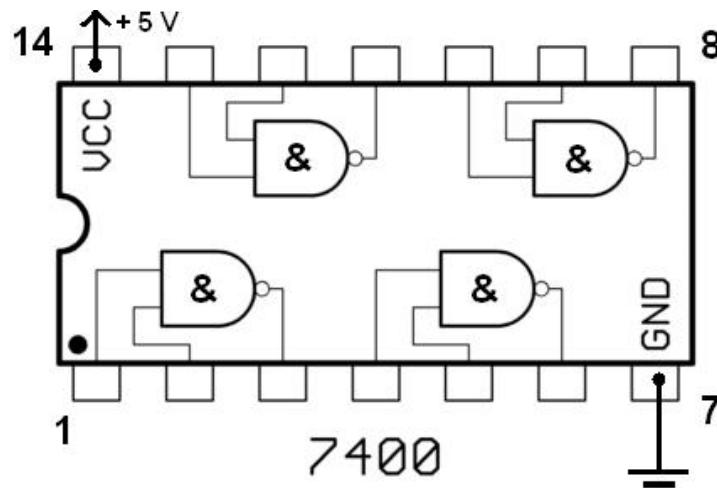
b)



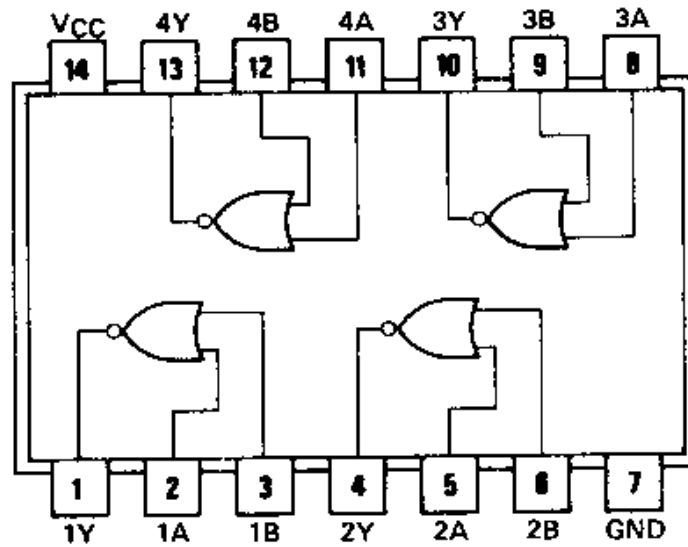
c)

A	
Entrada	Salida
0	0
1	1

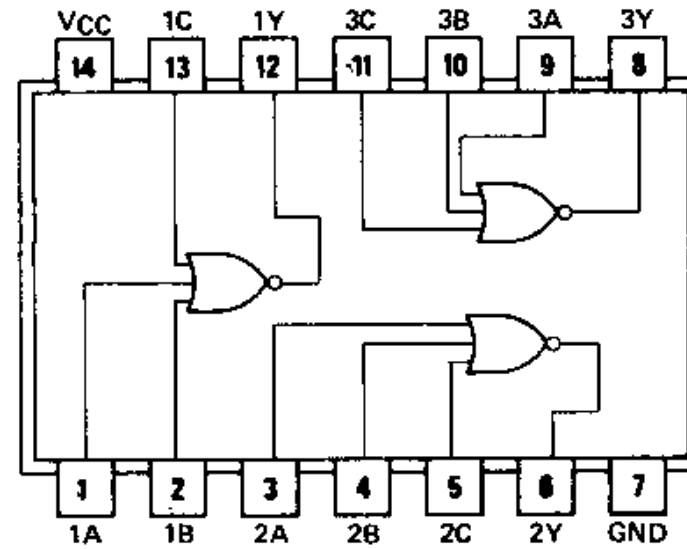
7400: Quad 2-input NAND Gate



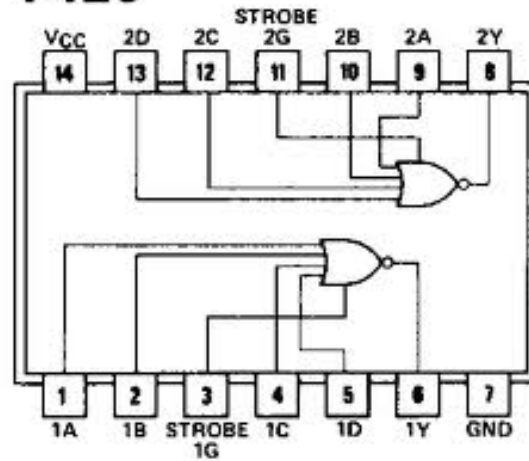
7402



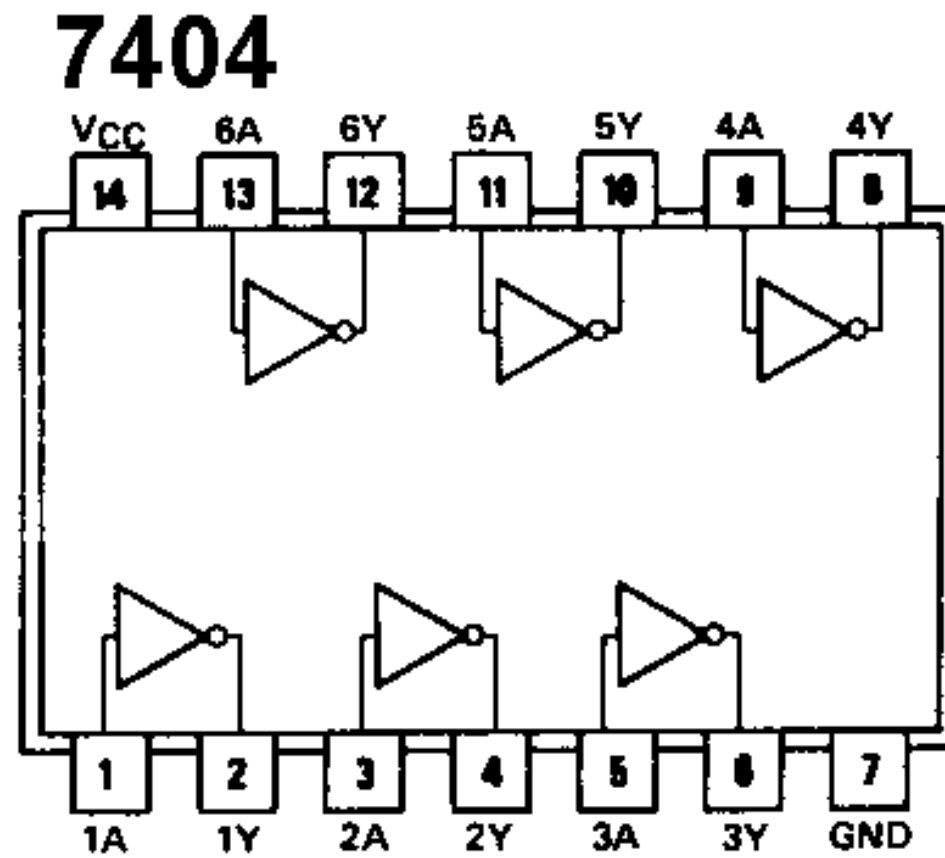
7427


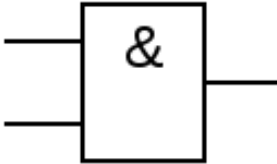
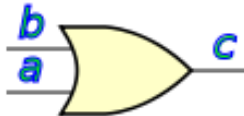
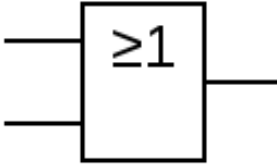
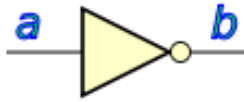
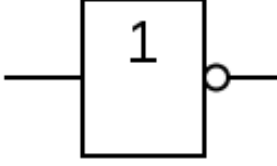


7425



7404: Hex Inverter



Puerta	Operación	Tabla de verdad	Símbolo IEEE	Símbolo IEC
AND	Producto $c = a \cdot b$	b a c		
		0 0 0		
		0 1 0		
		1 0 0		
OR	Suma $c = a + b$	b a c		
		0 0 0		
		0 1 1		
		1 0 1		
NOT	Complemento $b = \bar{a}$	a b		
		0 1		
		1 0		

Puerta	Función lógica	Tabla de verdad	Símbolo IEEE	Símbolo IEC															
NAND	$c = \overline{a \cdot b}$	<table border="1"> <thead> <tr> <th>b</th> <th>a</th> <th>c</th> </tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>0</td></tr> </tbody> </table>	b	a	c	0	0	1	0	1	1	1	0	1	1	1	0		
		b	a	c															
		0	0	1															
		0	1	1															
		1	0	1															
1	1	0																	
NOR	$c = \overline{a + b}$	<table border="1"> <thead> <tr> <th>b</th> <th>a</th> <th>c</th> </tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>0</td></tr> </tbody> </table>	b	a	c	0	0	1	0	1	0	1	0	0	1	1	0		
		b	a	c															
		0	0	1															
		0	1	0															
		1	0	0															
1	1	0																	

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

Tabla de verdad	Función lógica	Símbolo IEEE	Símbolo IEC															
<table border="1"> <thead> <tr> <th>b</th> <th>a</th> <th>c</th> </tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>0</td></tr> </tbody> </table>	b	a	c	0	0	0	0	1	1	1	0	1	1	1	0	$c = a \oplus b = a\bar{b} + \bar{a}b$		
b	a	c																
0	0	0																
0	1	1																
1	0	1																
1	1	0																

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

$c = a \oplus b = a\bar{b} + \bar{a}b$

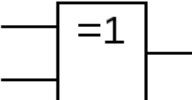


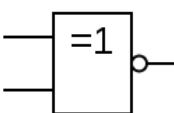
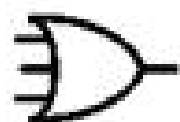


Tabla de verdad	Función lógica	Símbolo IEEE	Símbolo IEC															
<table border="1"> <thead> <tr> <th>b</th> <th>a</th> <th>c</th> </tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>1</td></tr> </tbody> </table>	b	a	c	0	0	1	0	1	0	1	0	0	1	1	1	$c = a \odot b = \overline{a \oplus b} = \bar{a} \cdot \bar{b} + a \cdot b$		
b	a	c																
0	0	1																
0	1	0																
1	0	0																
1	1	1																

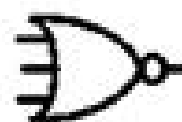
b	a	c			
0	0	1			
0	1	0	$c = a \odot b = \overline{a \oplus b} = \bar{a} \cdot \bar{b} + a \cdot b$		
1	0	0			
1	1	1			

FUNCIONES LÓGICAS BÁSICAS

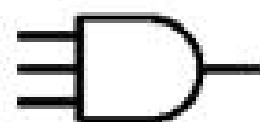
NOMRE	AND - Y	OR - O	XOR O-exclusiva	NOT Inversor	NAND	NOR																																																																																	
SÍMBOLO																																																																																							
SÍMBOLO																																																																																							
TABLA DE VERDAD	<table> <tr><th>a</th><th>b</th><th>z</th></tr> <tr><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>1</td></tr> </table>	a	b	z	0	0	0	0	1	0	1	0	0	1	1	1	<table> <tr><th>a</th><th>b</th><th>z</th></tr> <tr><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>1</td></tr> </table>	a	b	z	0	0	0	0	1	1	1	0	1	1	1	1	<table> <tr><th>a</th><th>b</th><th>z</th></tr> <tr><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>0</td></tr> </table>	a	b	z	0	0	0	0	1	1	1	0	1	1	1	0	<table> <tr><th>a</th><th>z</th></tr> <tr><td>0</td><td>1</td></tr> <tr><td>1</td><td>0</td></tr> </table>	a	z	0	1	1	0	<table> <tr><th>a</th><th>b</th><th>z</th></tr> <tr><td>0</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>0</td></tr> </table>	a	b	z	0	0	1	0	1	1	1	0	1	1	1	0	<table> <tr><th>a</th><th>b</th><th>z</th></tr> <tr><td>0</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>0</td></tr> </table>	a	b	z	0	0	1	0	1	0	1	0	0	1	1	0
a	b	z																																																																																					
0	0	0																																																																																					
0	1	0																																																																																					
1	0	0																																																																																					
1	1	1																																																																																					
a	b	z																																																																																					
0	0	0																																																																																					
0	1	1																																																																																					
1	0	1																																																																																					
1	1	1																																																																																					
a	b	z																																																																																					
0	0	0																																																																																					
0	1	1																																																																																					
1	0	1																																																																																					
1	1	0																																																																																					
a	z																																																																																						
0	1																																																																																						
1	0																																																																																						
a	b	z																																																																																					
0	0	1																																																																																					
0	1	1																																																																																					
1	0	1																																																																																					
1	1	0																																																																																					
a	b	z																																																																																					
0	0	1																																																																																					
0	1	0																																																																																					
1	0	0																																																																																					
1	1	0																																																																																					
EQUIVALENTE EN CONTACTOS																																																																																							
AXIOMA	$z = a \cdot b$	$z = a + b$	$z = \bar{a} \cdot b + a \cdot \bar{b}$	$z = \bar{a}$	$z = \overline{a \cdot b}$	$z = \overline{a + b}$																																																																																	



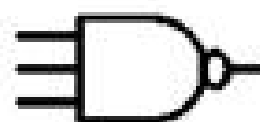
OR



NOR



AND



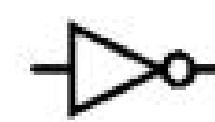
NAND



X-OR



X-NOR



NOT

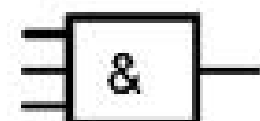
Y los símbolos normalizados según ANSI/IEEE:



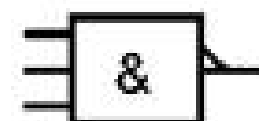
OR



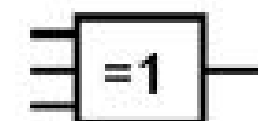
NOR



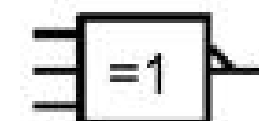
AND



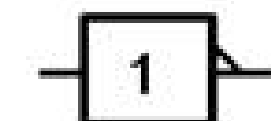
NAND



X-OR



X-NOR



NOT