

ELECTRÓNICA DIGITAL. CIRCUITOS COMBINACIONALES

por Aurelio Gallardo

29 - Octubre - 2023



Electrónica Digital. Circuitos Combinacionales. By Aurelio Gallardo Rodríguez, Is Licensed Under A Creative Commons Reconocimiento-NoComercial-CompartirIgual 4.0 Internacional License.

Índice General

1. INTRODUCCIÓN	2
2. SISTEMAS DE NUMERACIÓN	2
2.1. EXPRESAR UN NÚMERO BINARIO EN SU EQUIVALENTE DECIMAL	2
2.2. EXPRESAR UN NÚMERO DECIMAL EN SU EQUIVALENTE BINARIO	3
2.3. SISTEMA HEXADECIMAL	4
2.4. SISTEMA OCTAL	4
3. CÓDIGOS BINARIOS	4
3.1. REPRESENTACIÓN BCD	4
3.2. OTROS CÓDIGOS	5
4. ÁLGEBRA DE BOOLE	5
4.1. LÓGICA DE NIVELES	5
4.2. VARIABLES Y FUNCIONES LÓGICAS. LAS TABLAS DE LA VERDAD	6
4.3. OPERACIONES BÁSICAS EN EL ÁLGEBRA DE BOOLE	7
4.3.1. SUMA LÓGICA: PUERTAS OR	7
4.3.2. PRODUCTO LÓGICO: PUERTAS AND	7
4.3.3. NEGACIÓN O COMPLEMENTACIÓN: PUERTA NOT	8
4.4. POSTULADOS, PROPIEDADES Y TEOREMAS DEL ÁLGEBRA BOOLEANA	8
4.4.1. PROPIEDADES	8
4.4.2. POSTULADOS	8
4.4.3. TEOREMAS	9

4.5. OTRAS FUNCIONES LÓGICAS	9
4.5.1. PUERTA NAND (NEGATIVA AND)	9
4.5.2. PUERTA NOR (NEGATIVA OR)	9
4.5.3. PUERTA XOR (EXCLUSIVA OR)	10
4.6. FUNCIONES LÓGICAS DE VARIAS VARIABLES	10
4.6.1. SUMA DE PRODUCTOS	10
4.6.2. PRODUCTOS DE SUMAS	11
4.7. SIMPLIFICACIÓN DE FUNCIONES.	12
4.8. IMPLEMENTAR FUNCIONES CON PUERTAS NAND O NOR	14
4.9. CIRCUITOS COMBINACIONALES INTEGRADOS	14
4.9.1. CODIFICADOR O ENCODER	15
4.9.2. DECODIFICADOR	15
4.9.3. MULTIPLEXADOR O MULTIPLEXOR	16
4.9.4. CIRCUITO SUMADOR ARITMÉTICO	17
4.9.5. OTROS	17

A. VOCABULARIO	17
-----------------------	-----------

1. Introducción

Una primera clasificación de los circuitos digitales implica establecer las diferencias entre un circuito combinacional y otro secuencial.

- 1.- En un circuito **combinacional**, en todos y cada uno de los instantes de tiempo, las salidas son función exclusivamente de las entradas.
- 2.- En un circuito **secuencial**, las salidas son función de las entradas y del historial que hayan tomado las salidas en instantes anteriores.

Un recurso genial para trabajar este tema lo encontramos en: <https://angelmicelti.github.io/4ESO/EDI/index.html>. Además de información encontraremos simuladores, tanto online como descargables y ejecutables.

2. Sistemas de numeración

Los circuitos digitales están formados por componentes físicos que son capaces de trabajar con dos estados o valores. Estos estados pueden asimilarse matemáticamente a un sistema de numeración binario o base 2. Por lo tanto, cualquier información que deba ser tratada por un sistema digital debe ser codificada previamente al sistema binario.

Esto significa que cualquier número puede expresarse como una combinación de «0» y «1» en el sistema binario, de forma análoga a como cualquier número es representado en el sistema decimal con los dígitos del 0 al 9 (diez dígitos).

Por ejemplo, el número 453.25 es un número decimal. Podríamos decir de él que se forma con 4 centenas, 5 decenas, 3 unidades, 2 décimas y 5 centésimas. Ese número se expresa perfectamente con los dígitos del 0 al 9, dígitos que adquieren un significado «mayor» cuanto más a la izquierda se escriben y «menor» cuanto más a la derecha. No solemos pensarlo, pero en realidad al escribir ese número en formato decimal, que nos es tan habitual, estamos expresando el número en función de diversas potencias de diez, cuyos exponentes vienen dados por el lugar que ocupa el dígito (forma polinómica). De derecha a izquierda de la coma se numeran empezando por el 0. A la derecha de la coma, se empieza a numerar por el -1, -2, etc.

$$453,25 = 4 \cdot 10^2 + 5 \cdot 10^1 + 3 \cdot 10^0 + 2 \cdot 10^{-1} + 5 \cdot 10^{-2}$$

Y si tuviésemos el número 1101,01 en formato binario, tendríamos que expresarlo en función de las potencias del número 2.

$$1101,01 = 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 + 0 \cdot 2^{-1} + 1 \cdot 2^{-2}$$

En general, si necesito representar un número **N** en una base **b**, necesitaré unos coeficientes (dígitos) a_i que se posicionan en el lugar i -ésimo.

$$N = a_n \cdot b^n + a_{n-1} \cdot b^{n-1} + \dots + a_2 \cdot b^2 + a_1 \cdot b^1 + a_0 \cdot b^0 + a_{-1} \cdot b^{-1} + \dots + a_{-m} \cdot b^{-m} \quad (1)$$

Llamamos **código binario natural** o **sistema binario**, al sistema de representación numérica que utiliza dos dígitos, el 0 y el 1. A cada uno de estos símbolos lo denominamos bit.

2.1. Expresar un número binario en su equivalente decimal

Se utiliza una expresión polinómica; a veces nos ayudamos de una tabla. Por ejemplo, para el número binario 10111101,11

Comprobar en la web: <https://www.calculvivo.com/conversion-binario-decimal>

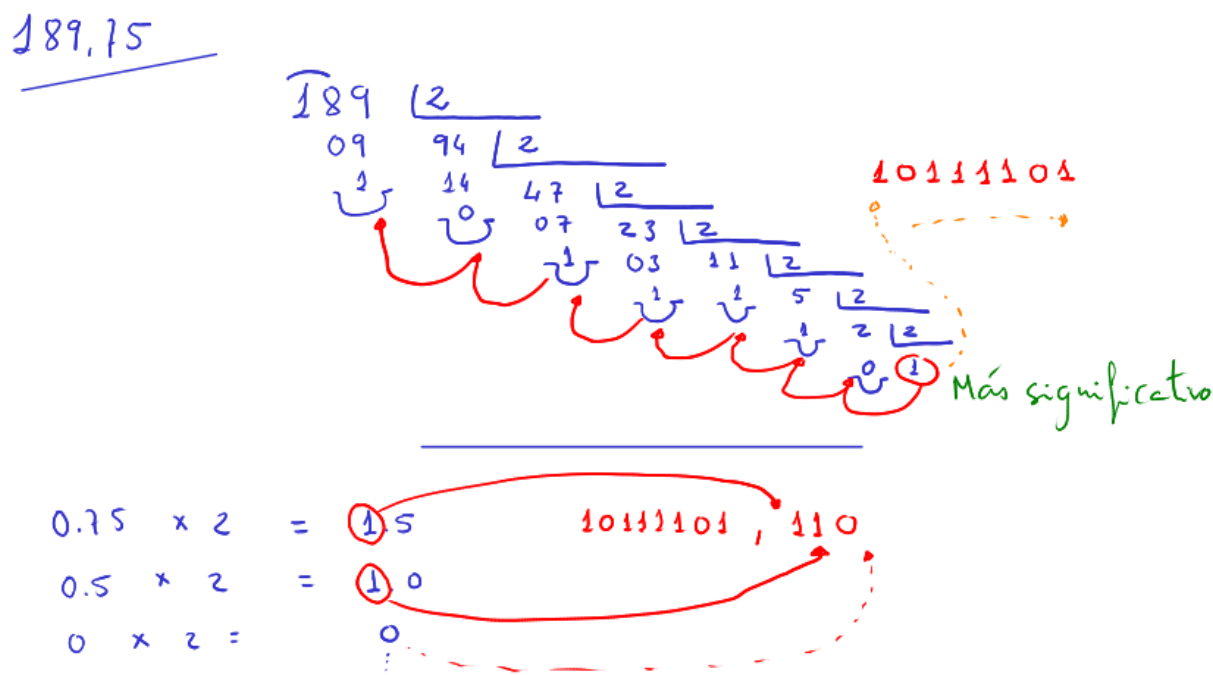


FIGURA (1). Cálculo de número binario a partir del número decimal

Bits	1	0	1	1	1	1	0	1	, 1	1
Posición	7	6	5	4	3	2	1	0	-1	-2
Factor	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}
Número	128	0	32	16	8	4	0	1	0.5	0.25

$$10111101,11_2 = 2^7 + 2^5 + 2^4 + 2^3 + 2^2 + 2^0 + 2^{-1} + 2^{-2} = 128 + 32 + 16 + 8 + 4 + 1 + 0,5 + 0,25 = 189,75_{(10)}$$

2.2. Expresar un número decimal en su equivalente binario

Haremos el proceso con el mismo número, para comprobar el resultado.

La parte entera se obtiene dividiendo entre dos sucesivamente. El bit más significativo (el de más a la izquierda) es el último cociente, y los demás son los restos, tomados de derecha a izquierda.

La parte decimal se obtiene multiplicando sucesivamente por dos y tomando la parte entera. Hasta que el resultado no sea más que cero u hasta obtener la precisión requerida.

Ejercicios: calcular el paso de número decimal a binario del 235.186

Reflexiona: ¿cómo se sabe que un número es par o impar en binario? ¿Cómo se puede multiplicar por 2 en binario?

Importante

- ✓ Si tengo un número binario de n bits, podré representar los números del 0 al número $2^n - 1$, en total 2^n números.
- ✓ Si quiero representar el número 2^n , necesitaré $n+1$ bits. Un «1» como bit más significativo y n ceros.
- ✓ Por ejemplo, si tengo 10 bits, puedo representar los números entre el 0 y el 1023 ($2^{10} - 1 = 1023$) que son 1024 números. El número 1024 es $1024_{(10)} = 1000000000_2$

2.3. Sistema hexadecimal

En este sistema, utilizamos dieciséis dígitos: del 0 al 9 y de la A a la F. La correspondencia de los primeros 16 números enteros positivos decimales con la numeración binaria y hexadecimal es:

\mathbb{Z}^+	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Bin	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
Hex	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

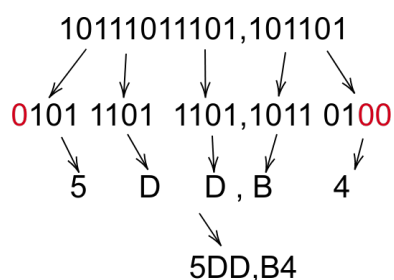


FIGURA (2). binario a hexadecimal

Para pasar un número binario a hexadecimal, se agrupan los bits en grupos de 4; a la izquierda a partir de la coma para la parte entera y a la derecha para la parte decimal. Y se sustituye cada grupo de 4 por su símbolo. Para la conversión contraria, se hace el procedimiento contrario.

Para convertir de números decimal a hexadecimal o viceversa, se convierte de forma intermedia a binarios. De hexadecimal a decimal se puede usar la expresión binómica con base 16.

$$127F_{16} = 1 \cdot 16^3 + 2 \cdot 16^2 + 7 \cdot 16^1 + F \cdot 16^0 = 1 \cdot 4096 + 2 \cdot 256 + 7 \cdot 16 + 15 \cdot 1 = 4735_{10}$$

El sistema hexadecimal se utiliza en el mundo de la informática para expresar números binarios muy grandes de forma más compacta y sencilla. Los ordenadores pueden manejar números de 64 o de 32 bits y es fácil equivocarse usando números binarios de esa longitud.

Calculadora de conversión: <https://www.disfrutalasmaticas.com/numeros/binario-decimal-hexadecimal-conversor.html>

2.4. Sistema Octal

El sistema octal, como el hexadecimal, convierte los números binarios a un sistema de números más compacto; esta vez de tres en tres dígitos. Se trabaja de forma análoga pero en grupos de tres. La base es el número 8.

\mathbb{Z}^+	0	1	2	3	4	5	6	7
Bin	0000	0001	0010	0011	0100	0101	0110	0111
Oct	0	1	2	3	4	5	6	7

3. Códigos binarios

Ya hemos visto el código binario natural. A cada número decimal le asigno un número binario usando los símbolos 0 y 1 de forma alterna. Sin embargo, hay otras formas de representar números usando códigos binarios.

3.1. Representación BCD

En la representación **BCD** (Binary-Coded Decimal o Decimal codificado en binario), los dígitos del sistema decimal de 0 al 9 se representa por un código de cuatro dígitos. Y posteriormente cualquier número se conforma a esa representación.

El primer código BCD que vamos a ver es el **BCD natural**, en el que los dígitos del 0 al 9 se representan con su número binario natural (4 bits). Así, por ejemplo, el número 385 se representa como:

Número decimal	Representación en código binario	BCD natural
385.75	110000001.110	0011 1000 0101.0111 0101

El segundo es el **código Aiken BCD**. En este código se utilizan representaciones para los dígitos del 0 al 4 como en el binario natural, y del 5 al 9 con las últimas posibles combinaciones con 4 dígitos.

El tercer código es el **Exceso-a-tres BCD**. El código usado para representar los dígitos del 0 al 3 sería el código natural binario más tres.

Decimal	0	1	2	3	4	5	6	7	8	9
BCD natural	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001
BCD Aiken	0000	0001	0010	0011	0100	1011	1100	1101	1110	1111
BCD Exceso a tres	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100

3.2. Otros códigos

- ✓ **Código de paridad:** utilizo el código binario natural o BCD natural, por ejemplo, y añado un bit a la derecha (menos significativo) que será «1» si el número de unos que aparece en el código es impar, y un 0 si es par. Por ejemplo, el número $5_{(10)} = 0101_{(2)} = 01010$. El número $7_{(10)} = 0111_{(2)} = 01011$. Este código, normalmente se aplica en transmisión de datos con números más grandes y sirve para detectar que hayan existido o no errores en la transmisión. Forma parte de un conjunto de códigos de corrección y detección de errores; otros de esta familia es el **biquinario**.
- ✓ **Códigos progresivos:** cada combinación difiere en la anterior en un solo bit. La veremos posteriormente en el tema.
- ✓ **Códigos ASCII:** representación de símbolos alfanuméricos con códigos binarios. Los hay de 7 bits y extendidos de 8 bits. La evolución de este código ha dado lugar a otros códigos como el **UTF-8 (Unicode)**.

4. Álgebra de Boole

En 1847 el matemático inglés George Boole desarrolló un álgebra cuyo objetivo consiste en representar las formas de razonamiento lógico, sistematizarlas y profundizar en el conocimiento de sus mecanismos. En 1948 Claude Shanon adoptó el álgebra de Boole para el estudio de los circuitos **que contienen elementos que adoptan dos estados estables**.

El álgebra de Boole opera con variables que adquieren solo dos valores, designados por «0» y «1». Estos dos valores en realidad no son números, sino símbolos que significan:

- ⇒ 0 ► estado bajo, o apagado (luces), o abierto (interruptor). También llamado «LOW». En programación, también «falso».
- ⇒ 1 ► estado alto, o encendido (luces), o cerrado (interruptor). También llamado «HIGH». En programación, también «verdadero».

4.1. Lógica de niveles

Un circuito digital, en el fondo, no es más que un circuito electrónico que trabaja a dos valores de tensión determinados. Por ejemplo, en los circuitos de la familia TTL se trabaja con los valores de referencia 0 Voltios y 5 Voltios.

Cuando se corresponde el valor «0» del álgebra de Boole con el valor más bajo, 0 Voltios, y el valor «1» con el más alto, 5 Voltios, se dice que estamos trabajando con **lógica positiva**. Si lo hacemos al contrario, el valor «0» corresponde a 5 Voltios, y el «1» a 0 Voltios, estamos trabajando con **lógica negativa**.

Será más normal trabajar con lógica positiva, aunque el uso de la lógica negativa puede simplificar mucho algunos cálculos.

Otra familia de circuitos puede trabajar con otras tensiones. Por ejemplo, -15V y 15V, 0V y 3V, etc.

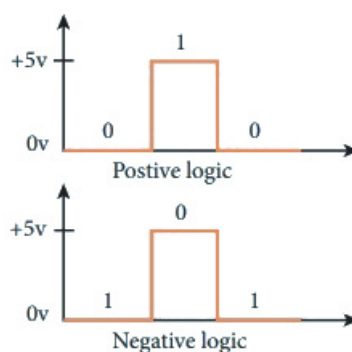


FIGURA (3). Lógicas positiva y negativa

4.2. Variables y funciones lógicas. Las tablas de la verdad

Podemos definir una **proposición**, en términos sencillos, como una oración gramatical que expresa una acción que el sujeto de ella podrá hacer o no. Por ejemplo, si le digo a alguien que «pulse el botón verde», esa persona podrá decidir hacerlo o no hacerlo.

Imaginemos que a la proposición «pula el botón verde» le asigno la variable **a**. Esta proposición podrá ser verdadera (si el sujeto pulsa el botón) o falsa (si no lo pulsa), con lo cual la variable **a** tomará en el primer caso el valor «1» y en el segundo el valor «0».

Las proposiciones suelen combinarse. En la siguiente frase: «enchufa el cable y acciona la llave de contacto» hay una unión gramaticalmente copulativa. Normalmente se expresa una tercera proposición, consecuencia de las otras dos, que depende de ellas: «enchufa el cable y acciona la llave de contacto, y se encenderá el motor».

	a	b	s
a	0	0	0
0	0	1	0
1	1	0	0
	1	1	1

CUADRO (1). Tablas de la verdad de una y dos variables

En este caso, puedo asignar una variable **a** a la primera frase, una variable **b** a la segunda frase, y un resultado **s** a la consecuencia. Tanto **a** como **b** tomarán dos valores de forma independiente: el operario a cargo del motor podrá elegir entre «enchufar o no el cable», o «accionar o no la llave de contacto». Pero está claro que solo si lleva a cabo las dos acciones se «encenderá el motor». Por lo tanto...

⇒ Si «enchufo el cable» ($a=1$) y «acciono la llave» ($b=1$), el resultado es «se encenderá el motor» ($s=1$)

⇒ En cualquier otro caso, o no enchufo el cable o no acciono la llave, o ambas; el motor no se encenderá ($s=0$)

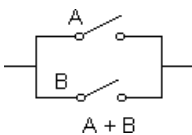
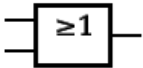

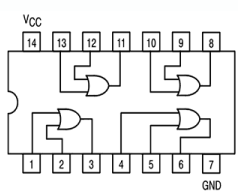
Las distintas combinaciones que pueden darse se pueden resumir en forma de tabla. Estas tablas se conocen como «**tablas de la verdad**». En el caso de la tabla anterior con dos variables y un resultado, existen cuatro combinaciones (2^2) posibles para el resultado. En general, si tengo una tabla con **n** proposiciones de entrada, o **n** variables independientes, tendré 2^n posibles combinaciones para el resultado.

Además, en matemáticas, cuando tengo variables independientes y una variable dependiente de ellas, puedo tener una función. Por ejemplo, si tengo la expresión $p = 3 \cdot x + \log(y)$ tengo una función $p(x,y)$ que dependen de x e y según la expresión algebraica anterior, de forma que a una combinación de la variable x e y , puedo asignar un valor a la p .

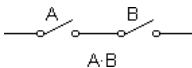
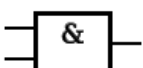

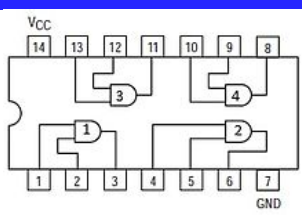
Nuestras tablas de la verdad podrán tener una función asociada en el álgebra de Boole. En la tabla anterior de dos variables se puede asociar la función $s = a \cdot b$, o producto lógico (no confundir con el producto aritmético).

4.3. Operaciones básicas en el álgebra de Boole

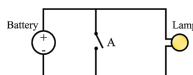
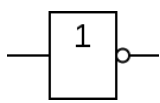

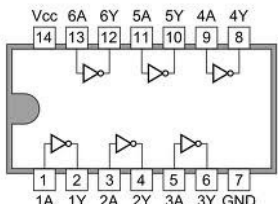
4.3.1. Suma lógica: puertas OR

Función	Tabla de la verdad	Circuito equivalente con interruptores															
$s = a + b$	<table border="1"> <thead> <tr> <th>A</th><th>B</th><th>Q</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>1</td></tr> </tbody> </table>	A	B	Q	0	0	0	0	1	1	1	0	1	1	1	1	
A	B	Q															
0	0	0															
0	1	1															
1	0	1															
1	1	1															
Símbolos lógicos	Circuito 7432 familia TTL																
 																	

4.3.2. Producto lógico: puertas AND

Función	Tabla de la verdad	Circuito equivalente con interruptores															
$s = a \cdot b$	<table border="1"> <thead> <tr> <th>A</th><th>B</th><th>Q</th></tr> </thead> <tbody> <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> </tbody> </table>	A	B	Q	0	0	0	0	1	0	1	0	0	1	1	1	
A	B	Q															
0	0	0															
0	1	0															
1	0	0															
1	1	1															
Símbolos lógicos	Circuito 7408 familia TTL																
 																	

4.3.3. Negación o complementación: puerta NOT

Función	Tabla de la verdad	Circuito equivalente con interruptores						
$s = \bar{a}$ $s = a'$	<table> <tr> <th>Q</th> <th>Q'</th> </tr> <tr> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> </tr> </table>	Q	Q'	0	1	1	0	
Q	Q'							
0	1							
1	0							
Símbolos lógicos		Circuito 7404 familia TTL						
		<p>SN 7404</p> 						

Estas puertas / funciones lógicas son las más básicas del álgebra de Boole. Cualquier otra función lógica puede expresarse como una combinación de estas tres funciones lógicas. Por lo tanto, se dice que **el conjunto de puertas OR, AND y NOT es un conjunto completo**.

4.4. Postulados, propiedades y teoremas del álgebra booleana

4.4.1. Propiedades

- ⇒ Conmutativa de la suma lógica: $a + b = b + a$
- ⇒ Conmutativa del producto lógico: $a \cdot b = b \cdot a$
- ⇒ Asociativa de la suma lógica: $a + b + c = (a + b) + c$
- ⇒ Asociativa del producto lógico: $a \cdot b \cdot c = a \cdot (b \cdot c)$
- ⇒ Distributiva del producto respecto de la suma: $a \cdot (b + c) = a \cdot b + a \cdot c$
- ⇒ **Distributiva de la suma respecto del producto:** $a + b \cdot c = (a + b) \cdot (a + c)$

4.4.2. Postulados

Nota: en el álgebra de Boole hay simetría. Si un postulado es cierto, el simétrico (cambiando suma por producto y «0» por «1») también es cierto.

- ⇒ $a + 1 = 1$, y también $a \cdot 0 = 0$
- ⇒ $a + 0 = a$, y también $a \cdot 1 = a$
- ⇒ $a + a = a$, y también $a \cdot a = a$ (ley de idempotencia)
- ⇒ $a + a' = 1$, y también $a \cdot a' = 0$. Lo puedo escribir como $a + \bar{a} = 1$, $a \cdot \bar{a} = 0$
- ⇒ $(a')' = a$. Lo puedo escribir como $\bar{\bar{a}} = a$

4.4.3. Teoremas

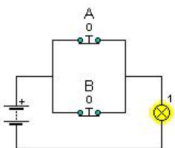
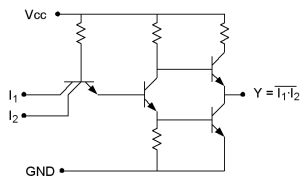
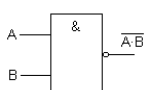

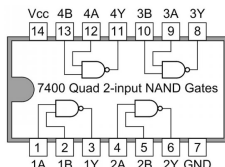
⇒ Leyes de absorción: $a + a \cdot b = a$, y por simetría $a \cdot (a + b) = a$

⇒ $a + \bar{a} \cdot b = a + b$, y por simetría $a \cdot (\bar{a} + b) = a \cdot b$

⇒ **Leyes de Morgan:** $\overline{(a + b)} = \bar{a} \cdot \bar{b}$, y por simetría $\overline{(a \cdot b)} = \bar{a} + \bar{b}$

4.5. Otras funciones lógicas

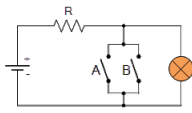
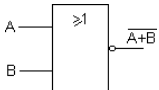

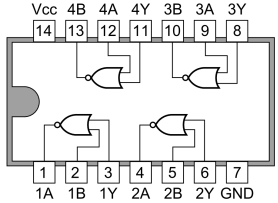
4.5.1. Puerta NAND (negativa AND)

Función	Tabla de la verdad	Circuito equivalente con interruptores	NAND TTL															
$s = \overline{a \cdot b}$ $s = \overline{a} + \overline{b}$	<table border="1"> <thead> <tr> <th>A</th> <th>B</th> <th>S</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>	A	B	S	0	0	1	0	1	1	1	0	1	1	1	0		
A	B	S																
0	0	1																
0	1	1																
1	0	1																
1	1	0																
Símbolos lógicos		Circuito 7400 familia TTL																
 																		

La puerta lógica NAND es muy importante, por las siguientes razones:

1. Forma por sí misma **un conjunto completo**. Con puertas NAND se puede representar cualquier función lógica. También se puede decir de ella que es una **puerta universal**.
2. Constructivamente es la puerta más fácil de fabricar dentro de la familia TTL. Consta de al menos cuatro transistores (uno multiemisor) y cuatro resistencias.

4.5.2. Puerta NOR (negativa OR)

Función	Tabla de la verdad	Circuito equivalente con interruptores															
$s = \overline{a + b}$ $s = \bar{a} \cdot \bar{b}$	<table border="1"> <thead> <tr> <th>A</th><th>B</th><th>Q</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>	A	B	Q	0	0	1	0	1	0	1	0	0	1	1	0	
A	B	Q															
0	0	1															
0	1	0															
1	0	0															
1	1	0															
Símbolos lógicos	Circuito 7402 familia TTL																
 																	

La puerta NOR también forma un **conjunto completo**, y es una **puerta universal**.

4.5.3. Puerta XOR (exclusiva OR)

Función	Tabla de la verdad	Circuito equivalente con interruptores															
$s = a \oplus b$ $s = \bar{a} \cdot b + a \cdot \bar{b}$	<table border="1"> <thead> <tr> <th>A</th><th>B</th><th>Q</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>	A	B	Q	0	0	0	0	1	1	1	0	1	1	1	0	
A	B	Q															
0	0	0															
0	1	1															
1	0	1															
1	1	0															
Símbolos lógicos	Circuito 7486 familia TTL																

La puerta XOR es una puerta también importante porque compara dos entradas de un bit. Si ambas valen lo mismo, la salida es «0». Si son distintas, «1».

4.6. Funciones lógicas de varias variables

Para obtener una función lógica cualquiera el procedimiento pasa por construir su tabla de la verdad. Por ejemplo, obtendremos la expresión de la función lógica en la que para tres variables, su salida es como en la de la tabla:

término	a	b	c	s
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	1
4	1	0	0	1
5	1	0	1	0
6	1	1	0	1
7	1	1	1	1

CUADRO (2). Ejemplo de función lógica de tres variables

4.6.1. Suma de productos

Si observamos la tabla, la salida es «1» cuando tenemos los términos 1, 3, 4, 6 y 7.

- ⇒ En el término 1, la salida es «1» si la combinación de a y b y c es [0,0,1]. En este caso, el producto $a' \cdot b' \cdot c$ valdría «1».
- ⇒ En el término 3, la salida es «1» si la combinación de a y b y c es [0,1,1]. En este caso, el producto $a' \cdot b \cdot c$ valdría «1».

- ⇒ En el término 4, la salida es «1» si la combinación de a y b y c es $[1,0,0]$. En este caso, el producto $a \cdot b' \cdot c'$ valdría «1».
- ⇒ En el término 6, la salida es «1» si la combinación de a y b y c es $[1,1,0]$. En este caso, el producto $a \cdot b \cdot c'$ valdría «1».
- ⇒ En el término 7, la salida es «1» si la combinación de a y b y c es $[1,1,1]$. En este caso, el producto $a \cdot b \cdot c$ valdría «1».

Y la función que buscamos será «1» si se produce la combinación 1, la 3, la 4 la 6 o la 7.

Luego la función de salida s puede expresarse como:

$$s = a' \cdot b' \cdot c + a' \cdot b \cdot c + a \cdot b' \cdot c' + a \cdot b \cdot c' + a \cdot b \cdot c \quad (2)$$

Cuando tenga las combinaciones correctas la función valdrá 1, porque uno de los productos lógicos será 1, y sumados lógicamente con los otros términos producirá un 1 a la salida. Si tengo una combinación que produce un 0 en la función, ninguno de los términos valdrá 1, todos serán ceros. Por lo tanto al sumar todos los ceros obtendré un cero a la salida.

Esta forma de expresar la salida de una función lógica, como la suma lógica de todos los productos que dan «1» se denomina **primera forma canónica**. Cada producto se denomina **mintérmino** o **minterm**. También podemos expresar simbólicamente la expresión de la salida s como suma de mintérminos de la forma:

$$s = \sum_3 (1, 3, 4, 6, 7) = m_1 + m_3 + m_4 + m_6 + m_7 \quad (3)$$

4.6.2. Productos de sumas

¿Hay simetría en el álgebra de Boole? Ya lo vimos en un apartado anterior. Pues en la forma de expresar una salida, tenemos otra posibilidad simétrica. En vez de fijarnos en los productos en los que la función es 1, nos podemos fijar en las sumas en las que la función es 0.

- ⇒ En el término 0, la salida es «0» si la combinación de a b c es $[0,0,0]$. En este caso, la suma $a + b + c$ valdría «0».
- ⇒ En el término 2, la salida es «0» si la combinación de a b c es $[0,1,0]$. En este caso, la suma $a + b' + c$ valdría «0».
- ⇒ En el término 5, la salida es «0» si la combinación de a b c es $[1,0,1]$. En este caso, la suma $a' + b + c'$ valdría «0».

Y la función que buscamos será «0» si se produce la combinación 0, la 2, y la 5. Entonces la función se puede expresar como:

$$s = (a + b + c) \cdot (a + b' + c) \cdot (a' + b + c') = M_0 \cdot M_2 \cdot M_5 \quad (4)$$

Si cualquiera de los términos de la expresión 4 es cero, da igual el valor de los otros que la salida será cero. Esta forma de expresar la salida de una función lógica, como el producto lógico de los términos que dan «0» se denomina **segunda forma canónica**. Cada sumando es un **maxtérmino** o un **maxterm**.

$$s = \prod_3 (0, 2, 5) \quad (5)$$

Economía en las expresiones lógicas.

Cuando quiero construir un circuito lógico, necesitaré un determinado número de puertas lógicas que tendré que fabricar o comprar. Además, los circuitos lógicos no dejan de ser circuitos electrónicos que hay que conectar a una fuente de tensión y consumen energía eléctrica. Cuantas más puertas lógicas necesite, más energía gastaré.

En el ejemplo de la función anterior, ¿cuántas puertas necesitaré?

- ⇒ En la expresión 2 aparece la complementación de las variables a , b y c (a' , b' y c'), luego necesitaré tres puertas NOT.
- ⇒ Y hay 5 minterminos, que suponen diez multiplicaciones; o necesito diez puertas AND de dos entradas o cinco puertas AND de tres entradas.
- ⇒ Y hay cuatro sumas, lo que supone al menos cuatro puertas OR de dos entradas, o una de cinco entradas.
- ⇒ Considerando que tengo puertas de dos entradas, necesito tres NOT, diez AND y cuatro OR. En total 17 circuitos.

-
- ⇒ En la expresión 4 también aparece la complementación de las tres variables a , b y c (a' , b' y c'). Sigo necesitando tres puertas NOT.
 - ⇒ Hay tres sumas, en total seis puertas OR de dos entradas.
 - ⇒ Hay dos multiplicaciones, dos puertas AND de dos entradas.
 - ⇒ En total necesito tres NOT, seis OR y dos AND. En total 11 circuitos.

Así que, desde un punto de vista económico, es *más recomendable en este caso construir la función por el método de minterminos*, en la segunda forma canónica.

¿Y no sería genial obtener métodos que me simplificara aún más la función lógica y necesitar menos puertas para representar una función? Eso es lo que estudiaremos en el siguiente apartado.

4.7. Simplificación de funciones.

Hay tres métodos para simplificar funciones. El primero es usar directamente los postulados y teoremas del álgebra de Boole para simplificar los términos. El segundo, como el Quine-McCluskey, es un método numérico usado informáticamente para la reducción de términos. Por ejemplo, usando el **primer método**...

$$\begin{aligned}
 s &= abc + ab'c + abc' \\
 s &= ab(c + c') + ab'c = ab + ab'c = a(b + b'c) \\
 b + b'c &= b + c \\
 s &= a(b + c)
 \end{aligned}$$

El tercero consiste en el método gráfico de las tablas de Karnaugh. Se puede simplificar con este método funciones de hasta seis variables, pero nosotros lo haremos con funciones de tres o cuatro variables.

Ejemplo de mapa de KarnaughMapas de Karnaugh en el simulador web: <http://www.32x8.com/var4.html>**1º) Obtengo la tabla de la verdad de la función**

	A	B	C	D	Y
0	0	0	0	0	0
1	0	0	0	1	0
2	0	0	1	0	0
3	0	0	1	1	1
4	0	1	0	0	0
5	0	1	0	1	0
6	0	1	1	0	0
7	0	1	1	1	1
8	1	0	0	0	1
9	1	0	0	1	1
10	1	0	1	0	0
11	1	0	1	1	0
12	1	1	0	0	1
13	1	1	0	1	1
14	1	1	1	0	0
15	1	1	1	1	0

«A» es el bit más significativo, «D» el menos significativo e «Y» la salida

2º) Dispongo los valores en un tabla

		CD			
		00	01	11	10
AB	00				
	01				
	11				
	10				

A la izquierda en columna los dos bits más significativos AB, en el orden 00-01-11-10.

En la fila superior los bits CD en el orden 00-01-11-10

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

Pondré el valor de la función en la celda que corresponde al mintermino correspondiente

		CD			
		00	01	11	10
AB	00	0	0	1	0
	01	0	0	1	0
	11	1	1	0	0
	10	1	1	0	0

Representación circuito con puertas lógicas ⇨

3º) Hacemos grupos

		CD			
		00	01	11	10
AB	00	0	0	1	0
	01	0	0	1	0
	11	1	1	0	0
	10	1	1	0	0

Agrupamos los «1», primero en grupos de 8. Si no hay de 4, y si no de 2, y por último los «1» sueltos.

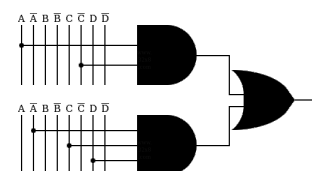
En el grupo rosa observamos que A siempre vale 1 y C siempre vale 0. B y D varían. Luego la función tendrá el término **AC'**

En el grupo verde A siempre es cero, C y D valen 1. B varía. Luego la función tendrá el término **A'CD**

La función simplificada es:

$$s = a \cdot c' + a' \cdot c \cdot d$$

$$s = a \cdot \bar{c} + \bar{a} \cdot c \cdot d$$



Cuando simplificamos una función por Karnaugh debemos tener en cuenta lo siguiente ⁽¹⁾:


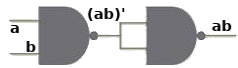
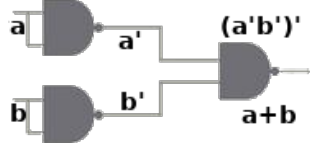
- Siempre se consiguen los grupos con mayor número de «1» posibles, **aunque se solapen**.
- **Las esquinas, y las partes superior e inferior de la tabla forman grupos**.
- Es posible resolver un mapa de Karnaugh **agrupando ceros**. Se consigue una expresión en productos de sumas.
- Algunas tablas presentan funciones válidas distintas, dependiendo de agrupaciones diferentes pero posibles.

La resolución de mapas de Karnaugh es un aspecto importante de este tema, que se tratará ampliamente en los problemas. Ayudarse del simulador servirá para corregir y observar los fallos cometidos en los ejercicios: <http://www.32x8.com/var4.html>


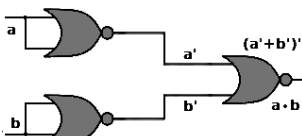
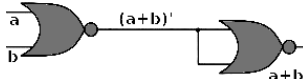
¹Sería conveniente en el examen indicar si vamos a usar «'» para complementar o la barra superior « $\bar{}$ »

4.8. Implementar funciones con puertas NAND o NOR

Las puertas NAND o NOR, por sí mismas forman un conjunto completo. Cualquier función puede expresarse como una combinación exclusivamente de puertas NAND o de puertas NOR. De hecho, las tres puertas básicas pueden obtenerse como combinación de una o de otra. Para las puertas NAND

	NOT \Rightarrow NAND NOT	AND \Rightarrow NAND AND	OR \Rightarrow NAND OR
PUERTAS			
FUNCIÓN	$s = (a \cdot a)' = a'$	$s = ((a \cdot b)')' = a \cdot b$	$s = (a' \cdot b')' = (a')' + (b')' = a + b$

Para las puertas NOR...

	NOT \Rightarrow NOR NOT	AND \Rightarrow NOR AND	OR \Rightarrow NOR OR
PUERTAS			
FUNCIÓN	$s = (a + a)' = a'$	$s = (a' + b')' = (a')' \cdot (b')' = a \cdot b$	$s = ((a + b)')' = a + b$

¿Cómo se puede implementar un circuito cualquiera por...?

A) Puertas NAND

- \Rightarrow Se resuelve el mapa de Karnaugh y se da una solución como suma de productos (reducción de minterminos).
- \Rightarrow Se complementa toda la función dos veces.
- \Rightarrow Se resuelve la primera complementación

B) Puertas NOR

- \Rightarrow Se resuelve el mapa de Karnaugh y se da una solución como productos de sumas (reducción de maxtérminos).
- \Rightarrow Se complementa dos veces la función.
- \Rightarrow Se resuelve la primera complementación.

4.9. Circuitos combinacionales integrados

En un circuito combinacional ya sabemos que la o las salidas dependen exclusivamente de sus entradas. Hemos estudiado los circuitos lógicos combinacionales más sencillos, las llamadas puertas lógicas y la forma de implementar circuitos más complicados combinando estas puertas lógicas.

Estudiaremos en este apartado algunos circuitos más complejos con funciones específicas. Normalmente los trataremos como «cajas negras», es decir, estudiaremos sus salidas en función de sus entradas pero no nos molestaremos en averiguar el circuito combinacional intrínseco a los mismos. Todos estos circuitos tienen un equivalente comercial (o más de uno).

4.9.1. Codificador o encoder

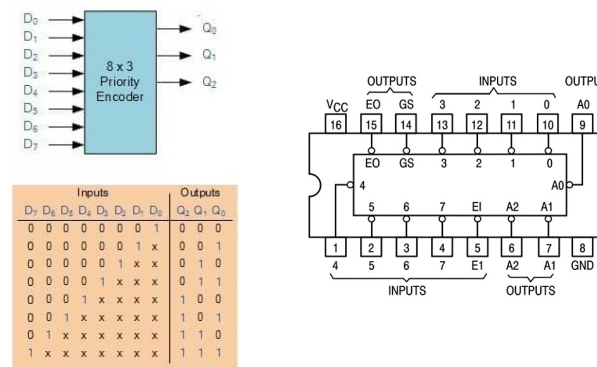


FIGURA (4). Codificador a binario

Es un circuito combinacional que tiene n salidas y 2^n entradas. Las salidas son la señal codificada en binario de la activación de una, y solo una, de las entradas. Las entradas se corresponden con un valor en decimal.

Existen dos variantes: codificador sin prioridad o con prioridad.

- ⇒ **Sin prioridad:** no es posible activar dos o más de dos entradas a la vez. Sólo una.
- ⇒ **Con prioridad:** es posible activar más de una entrada. La salida representa la entrada que se haya activado de mayor peso, de mayor valor decimal.

4.9.2. Decodificador

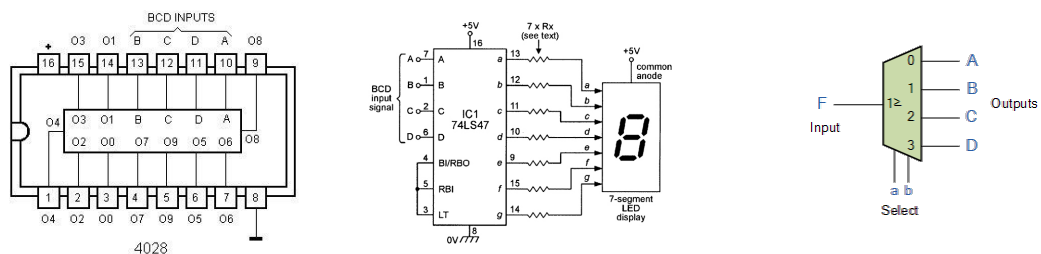


FIGURA (5). Decodificadores y demultiplexador

Contrario al anterior, posee n entradas y 2^n salidas (o algo menor). Básicamente convierten información codificada a otra sin codificar.

- ⇒ Las salidas pueden decodificarse a nivel alto o bajo. A nivel alto se activa la salida con un «1» lógico y las demás son «0». A nivel bajo, se activa la salida con un «0» lógico y las demás salidas están a nivel alto, «1» lógico.
- ⇒ Un decodificador binario puede servir como parte de un circuito lógico complejo de n entradas. Cada salida del decodificador es la activación de un mintermino. Solo necesito poner una puerta OR que sume los minterminos.
- ⇒ Decodificadores importantes son, por ejemplo, el CMOS 4028 o el muy usado decodificador BCB a 7 segmentos 74LS47 o 74LS48. Este último se usa para convertir un código BCD natural (del 0 al 9) al código necesario para activar un **display de barras led de 7 segmentos**, que tienen muchos aparatos, como contadores, relojes, etc. El decodificador BCD a 7 segmentos también se puede clasificar como convertidor de código.
- ⇒ Para saber más: **¿qué es un decodificador?** y **aplicaciones de los decodificadores**

Variante del decodificador: demultiplexador o demultiplexor

Un demultiplexador es un circuito que posee **una entrada de información D**, **n** entradas de control y 2^n salidas.

Imaginemos el caso de un demultiplexador de $n=2$ entradas, cuya tabla de la verdad podemos ver en la (6). En este caso, el valor de la entrada **D**, se traslada a una de las salidas, codificadas en las entradas de control.

Si nos damos cuenta, un demultiplexador de 2 entradas de control y una señal de datos es el mismo concepto que un decodificador $3 \div 8$, teniendo en cuenta de que la señal de entrada considerada como datos (ejemplo, el bit más significativo **A** lo escojo como señal de entrada) me activará el grupo de salidas S_4, S_5, S_6, S_7 o me dará cero en ellas (en la figura (6) en verde). Si considero la señal complementada de **A**, \bar{A} , tendré que considerar las otras salidas (en rojo).

Ver: demultiplexador.

Data Input	Select Inputs		Outputs			
D	S_1	S_0	Y_3	Y_2	Y_1	Y_0
D	0	0	0	0	0	D
D	0	1	0	0	D	0
D	1	0	0	D	0	0
D	1	1	D	0	0	0

A	B	C	S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

FIGURA (6). Demultiplexador 2 to 4

4.9.3. Multiplexador o multiplexor

Un multiplexador es el circuito contrario al demultiplexador. Tiene 2^n entradas, **n** entradas de control y una señal de salida. El multiplexador traslada la señal que tiene una de sus entradas (seleccionada mediante las entradas de control) a la salida. En la figura (7) cuando tengo la entrada de control 00, el valor de la entrada I_0 se traslada a la salida, siendo «0» si $I_0 = 0$ y «1» si $I_1 = 1$. Por lo tanto tendré a la salida el producto $s = a' \cdot b' \cdot I_0$. Asimismo, si tengo la entrada de control 01, el valor de la entrada I_1 se traslada a la salida, siendo «0» si $I_1 = 0$ y «1» si $I_0 = 1$. Por lo tanto tendré a la salida el producto $s = a' \cdot b \cdot I_1$...

En general a la salida tendré la combinación de todos los posibles productos: $s = a' \cdot b' \cdot I_0 + a' \cdot b \cdot I_1 + a \cdot b' \cdot I_2 + a \cdot b \cdot I_3$. Los términos **ab** son los minterminos de una señal de dos variables, luego podré expresar la salida como

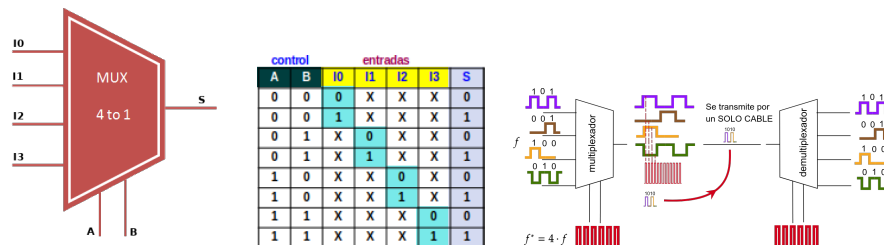
$$s = m_0 \cdot I_0 + m_1 \cdot I_1 + m_2 \cdot I_2 + m_3 \cdot I_3 = \sum_{i=0}^3 m_i \cdot I_i$$


FIGURA (7). Multiplexador 1 to 4

El uso de multiplexores y demultiplexores cobra importancia dentro de la comunicación de datos por cable. Imaginemos que tenemos cuatro señales de teléfono digitales por la que se están transmitiendo datos. La frecuencia a la que se transmiten los datos de esas señales telefónicas es de $f = 40\text{KHz}$, es decir, cada bit de información se transmite con un período de $T = 25\mu\text{s}$.

Usando una señal 4 veces más rápida, de frecuencia $f^* = 160\text{KHz}$, obtengo una señal de período más corto $T^* = 6,25\mu\text{s}$. Aplico esta señal como base de tiempo para generar las señales de control. Como en un ciclo de cambio de las señales de teléfono, las señales de control cambian cuatro veces más rápido, a la salida del multiplexador tengo una señal de frecuencia $4f = 160\text{KHz}$ **con la información del primer bit de las cuatro señales**. Esta información es capaz de viajar **por un único cable**.

Al llegar al destino, las señales se retornan a su forma original aplicando el proceso contrario con un demultiplexor.

Este proceso de **multiplexación** tuvo mucha importancia en el desarrollo de la telefonía comercial. Con él, varios abonados estaban comunicados con una centralita. La centralita era capaz de enviar por un sólo cable una señal que contenía, **a la vez**, las conversaciones de varios abonados.

Dependiendo del material o los materiales de los que estuvieran hechos los cables, de su grosor y otros parámetros, los cables pueden transmitir así un número determinado de conversaciones. Normalmente se hace un cálculo estadístico de cuantos abonados pueden conectar a una centralita para que usen un sólo cable, ya que si tenemos N abonados, normalmente no están hablando todos a la vez y se dimensiona el cable para M conversaciones con $M < N$. Si alguna vez el abonado $M+1$ intenta establecer comunicación, le dará un aviso de error denominado **saturación de líneas**.

4.9.4. Circuito sumador aritmético

Este circuito suma dos entradas, pero no con la suma lógica, sino aritmética. El circuito tiene una salida S , la suma, y una salida C_0 llamada acarreo, que es un bit que vale «0» si la suma no desborda y «1» si desborda (llevarse «1»). Ver animación. La salida S se consigue con una puerta XOR y el acarreo con una puerta AND. Se le denomina *semi-sumador o half-adder*.

A	B	S (suma)	Co (acarreo)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

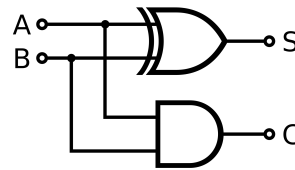


FIGURA (8). Tabla verdad y semi-sumador

¿Por qué al circuito anterior se le denomina «semi»? Es que no está completo... Así sumamos dos bits pero... ¿Y si necesitamos sumar números binarios de n bits? El sumador tendrá que tener en cuenta el acarreo del bit menos significativo anterior. En este caso el circuito tiene tres entradas (los dos bits más el acarreo) y dos salidas (la suma y el acarreo de salida). Y ya se llama *sumador o sumador completo o full-adder*. Ver animación.

Cin	A	B	S (suma)	Cout (acarreo)
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

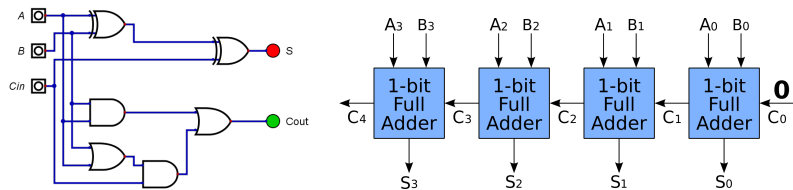


FIGURA (9). Tabla verdad y sumador completo. Implementación.

4.9.5. Otros

Hay otros circuitos combinacionales posibles. Por ejemplo, el comparador, que compara si dos números de n bits A y B cumplen si $A > B$, $A = B$ o $A < B$. Tienen $2n$ entradas digitales y tres salidas, una por cada comparación. Circuito comercial 7485.

También hay substractores, buffers, inversores, etc. https://en.wikipedia.org/wiki/List_of_7400-series_integrated_circuits

A. Vocabulario

Código: sistema de signos que permite comprender y formular mensajes.

Código binario: utiliza los signos 0 y 1

Bit: unidad mínima de información, con dos estados posibles 0 y 1.

Bit más significativo o MSB: bit más a la izquierda, el de mayor peso.

Bit menos significativo o LSB: bit más a la derecha, el de menor peso.

Hexadecimal: código con 16 símbolos, del 0 al 9 y de la A a la F

Octal: código con 8 símbolos, del 0 al 7.

BCD: digital codificado en binario.

ASCII: American Standard Code for Information Interchange

TTL: familia de circuitos integrados transistor-transistor-logic. Otras pueden ser CMOS, DTL, etc.

AND, OR, NOT: puertas lógicas básicas.

NAND,NOR,XOR: otras puertas lógicas.

Forma canónica: función expresada mediante la combinación de todos los términos que la hacen «1» (primera forma canónica) o que la hacen «0» (segunda forma canónica).

Mintérmino: término que se hace «1» en una función lógica, caracterizado por el producto de las variables, complementadas o no.

Maxtérmino: término que se hace «0» en una función lógica, caracterizado por la suma de las variables, complementadas o no.