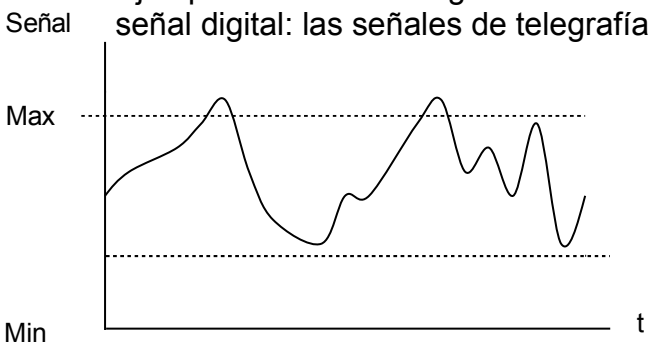
Bloque IV: Electrónica digital

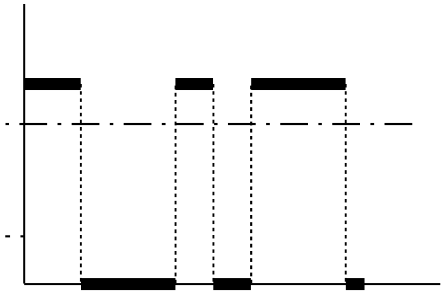
1. Introducción

Una señal **analógica** es aquella que puede tomar infinitos valores para representar la información. En cambio en una señal **digital** se utiliza sólo un número finito de valores.

Ejemplo de señal analógica. La señal de lectura de una cinta vídeo. Ejemplo de señal digital: las señales de telegrafía que usan el código Morse.



Señal analógica



Señal digital

Los circuitos digitales son aquellos que comunican y procesan información de tipo digital.

2. Sistemas de numeración y códigos

Un número está constituido por una sucesión de dígitos ordenados de izquierda a derecha. Nuestra cultura utiliza un sistema de numeración que cumple con esta condición: esta constituido por 10 dígitos (0,1,2,3,...9). Se dice que nuestro sistema de numeración es de **base 10** (la base de un sistema es el número de posibles dígitos que se utiliza) A nuestro sistema de numeración se le denomina también **sistema decimal**.

Los circuitos digitales utilizan un sistema de numeración cuya **base es 2** y por ello se dice que es **binario**.

Cualquier información que deba ser tratada por un circuito digital debe ser codificada previamente al sistema que él entiende, el binario.

Un número se representa en un sistema de base b mediante un desarrollo en forma polinómica:

$$N = a_n b^n + a_{n-1} b^{n-1} + \dots a_1 b^1 + a_o b^o + a_{n-1} b^{n-1}$$

donde **b** es la **base**

$\mathbf{a_i}$ son los **coeficientes** que representan las cifras del número

Ejemplos: Sistema decimal

b = 10
a_i = 0, 1, 2,3,... 9

El número 7423,56 se puede representar como

$$7423{,}56 = 7\cdot 10^3 + 4\cdot 10^2 + 2\cdot 10^1 + 3\cdot 10^0 + 5\cdot 10^{-1} + 6\cdot 10^{-2}$$

Sistema binario

b= 2
a_i = 0, 1

El número 1101,101 se representa como

$$1101{,}101 = 1\cdot 2^3 + 1\cdot 2^2 + 0\cdot 2^1 + 1\cdot 2^0 + 1\cdot 10^{-1} + 0\cdot 10^{-2} + 1\cdot 10^{-3}$$

¿Por qué utilizan los circuitos digitales un sistema binario? Porque el más adecuado es aquel que precisa del menor número de componentes básicos para su realización, ya que de esta manera el coste del circuito resulta mínimo.

3. Sistema binario

En este sistema sólo existen dos dígitos, el 0 y el 1. Esta unidad mínima se denomina bit.

Expresemos el equivalente decimal de un número binario. Ejemplo $101101{,}11_{(2)}$

$101101{,}11_{(2)}$ =

$$1\cdot 2^5 + 0\cdot 2^4 + 1\cdot 2^3 + 1\cdot 2^2 + 0\cdot 2^1 + 1\cdot 2^0 + 1\cdot 10^{-1} + 1\cdot 10^{-2} n = 32 + 0 + 8 + 4 + 0 + 1 + 0{,}5 + 0{,}25 = 45{,}75_{10}$$

¿Cómo se realiza la operación inversa (pasar de un número decimal entero a binario)?

Se realizan sucesivas divisiones por 2 hasta que el cociente sea inferior a 2

Ejemplo: pasar el número 45 a binario

Número	Cociente	Resto
45:2	22	1
22:2	11	0
11:2	5	1
5:2	2	1
2:2	1	0

El bit que tengamos por último en el cociente se denomina **bit más significativo**

Así pues $45_{(10)} = 101101_{(2)}$

Si el número decimal no es entero, se multiplica **la parte fraccionaria** por dos, la parte fraccionaria del resultado es multiplicada por dos y así sucesivamente hasta que no se obtenga nueva fracción.

Ejemplo: $0{,}36_{(10)}$

Número	Resultado	Dígito fraccionario	
0,36·2	**0**,72	Primer dígito fraccionario	0
0,72·2	**1**,44	Segundo dígito fraccionario	1
0,44·2	**0**,88	Tercer dígito fraccionario	0
0,88·2	**1**,76	Cuarto dígito fraccionario	1
0,76·2	**1**,52	Quinto dígito fraccionario	1
0,52·2	**1**,04	Sexot dígito fraccionario	1

Así pues $0{,}36_{(10)} = 0{,}010111_{(2)}$ …

4. Códigos binarios

Un código es una representación de cantidades de tal forma que a cada una de éstas se les asigna una combinación de símbolos determinados y viceversa.

Hay dos códigos binarios básicos, aunque existen más:

a) Código BCD (Decimal Codificado en Binario)

Debido a que con n cifras binarias o bits se pueden obtener 2n combinaciones diferentes. ¿Cómo? Se representa por separado cada una de sus cifras. El número de bits necesarios para representar cada cifra es de cuatro y, por lo tanto, podemos efectuar 24=16 combinaciones.

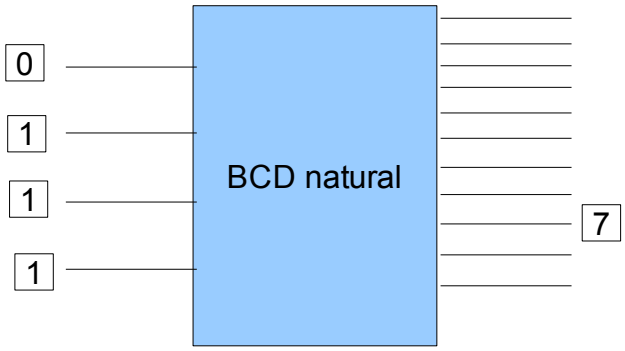
El más utilizado el BCD natural

Decimal	BCD natural
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

El dispositivo que emplea este código tiene cuatro entrada y diez s

DECIMAL	BCD 8421	Exceso 3
0	0000	0011
1	0001	0100
2	0010	0101
3	0011	0110
4	0100	0111
5	0101	1000
6	0110	1001
7	0111	1010
8	1000	1011
9	1001	1100

www.unicrom.com



Al código BCD natural se le denomina también código 8421

DECIMAL	Aiken
	2421
0	0000
1	0001
2	0010
3	0011
4	0100
5	1011
6	1100
7	1101
8	1110
9	1111

www.unicrom.com

Ejemplo: 25=> 2(0010) 5(0101) 2=0·8+0·4+1·2+0·1 5=0·8+1·4+0·2+1·1

Otros códigos son el código **Aiken o 2421**

Ejemplo 25 => 2 (0010) 5(1011) 2=0·2+0·4+1·2+0·1 5=2·1+0·4+1·2+1·1

y el **exceso tres.** Se suma a cada dígito 3 y luego se pasa a binario cada cifra.

Ejemplo 25=> 2 (0101) 5(1000) 2=>2+3=>5=>0101 5=>5+3=>8=>1000

Otros códigos conocidos, aunque más complejos son:

- El código **Hamming** (7 bits), corrector de errores
- El código **ASCII** para representar letras, símbolos, números,...

b) Sistema Hexadecimal

Es un sistema de numeración muy empleado en microprocesadores.

Tiene **base 16.**

- Los diez primero dígitos son los dígitos decimales del 0 al 9
- Los seis últimos son las letras del alfabeto de la A a la F

La equivalencia entres sistema hexadecimal y decimal es

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

Para realizar la conversión de un número hexadecimal en decimal, se resuelve el polinomio equivalente, sabiendo que

base b = 16
coeficientes ai = 0, 1, 2, 3, 4, ….15

Ejemplo: 2EF será...

$$\mathbf{2}\cdot 16^2 + 14\cdot 16^1 + 15\cdot 16^0 = 2\cdot 256 + 14\cdot 16 + 15\cdot 1$$

E F

2EF = 512 + 224 + 15 = 751

Para pasar del código decimal al hexadecimal se realizan sucesivas divisiones por 16, hasta que el último cociente sea menor que 16. Ejemplo: 751 a hexadecimal.

Número	Cociente	Resto	
751:16	46	15	F
46:16	2	14	E
		2	Bit más significativo

Luego $751_{(10)} = 2EF_{(16)}$

Para pasar de binario a hexadecimal, se pasa primero a decimal y desde ahí a hexadecimal. Y viceversa para lograr el objetivo inverso.

Binario → Decimal → Hexadecimal
Hexadecimal → Decimal → Binario

Ejemplo: Binario a hexadecimal

Ejemplo: 111 10101101

Primero hacemos grupos de cuatro bits hacia la izquierda, comenzando por la cifra situada a la derecha de la coma. Si el último grupo está incompleto, añadimos ceros a la izquierda

0111	1010	1101	Binario
7	10	13	Decimal
7	A	D	Hexadecimal

Para pasar de hexadecimal a binario: Ejemplo: 4DF

4	D	F	Hexadecimal
4	13	15	Decimal
0100	1101	1111	Binario

Como los ceros a la izquierda no cuenta... $4DF_{(16)} = 10011011111_{(2)}$

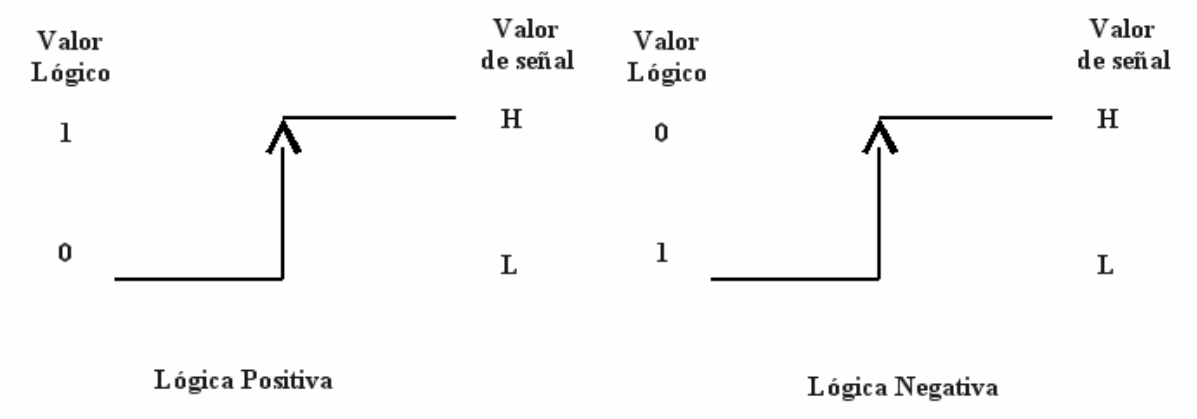
5. Algebra de Boole

Desarrollado a mediado del siglo XIX por el inglés George Boole, cuyo objetivo era representar las formas de razonamiento lógico.

El álgebra de Boole maneja dos variables: verdadero o falso, cero y uno, abierto y cerrado, encendido o apagado,... Esta variables se llaman booleanas.

En el algebra de Boole aplicada a los circuitos digitales se pueden distinguir dos tipos de lógica o niveles, que establece una correspondencia entre los niveles de tensión y los elementos de información binaria.

1. **Lógica positiva**: al nivel de tensión más elevado se le asigna el estado 1 y al nivel de tensión más bajo se le asigna el estado 0
2. **Lógica negativa**: La asignación es a la inversa. Nivel más alto, estado 0 y nivel más bajo, estado 1



6. Operaciones básica con el álgebra de Boole

Definamos el concepto de **función lógica**.

Una función lógica es aquella función cuyos valores son binarios y dependen de una expresión algebraica formada por una serie de variables binarias relacionadas entre sí por determinadas relaciones.

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

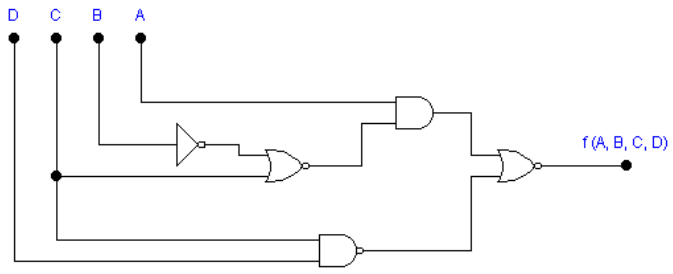
tanto las variables a, b y c sólo pueden tomar dos valores: cero y uno.

– La función vale 1 si a y b vale 1 o si c vale 1, o bien, si cumple ambas condiciones a la vez.

– La función vales 0 si a a o b valen 0 y c vale 0 , o si las tres variables valen 0 a la vez.

El comportamiento de las funciones lógicas se expresa mediante las denominadas **tablas de verdad**, que está constituidas por dos zonas bien diferenciadas, la **zona de entrada**, donde se representan todas las posibles combinaciones de las variables de

entrada, y una **zona de salida** en la que se indica el valor de la función lógica para cada combinación.



Esquema gráfico de un circuito digital que representa una zona de entrada de 4 variables (A,B,C,D) y la zona de salida (función lógica).

A las operaciones básicas del álgebra de Boole cuando se implementan mediante circuitos electrónicos se les acostumbra a llamar puertas lógicas.

El número de combinaciones posibles en una tabla de verdad de n entradas es 2^n.

Ejemplo: El esquema anterior tiene 4 entradas (o variables de entrada, A, B, C, D), es decir, la tabla de verdad correspondiente tiene $2^4 = 16$ posibles combinaciones.

Las funciones lógicas más importantes, consideradas **básicas** son:

– Función suma lógica o función unión
– Función producto lógico o función intersección
– Función complemento o función negación

FUNCION SUMA LOGICA O UNIÓN

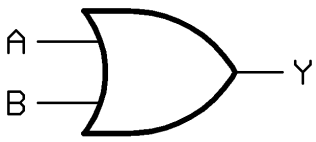
Es una función de dos variables de entrada. La suma toma el valor 1 si cualquiera de variables toma el valor 1

$$S = a + b$$

cuya tabla de verdad correspondiente es...

Entradas		Salida
A	B	S
0	0	0
0	1	1
1	0	1
1	1	1

El circuito digital que realiza esta operación se denomina puerta lógica y, en concreto, en este caso, la puerta lógica que se corresponde a esta función lógica es



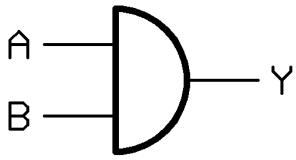
FUNCION PRODUCTO O INTERSECCION

Se representa como $S = a \cdot b$

El producto toma el valor 1 cuando a y b presentan un 1. La tabla de verdad es

Entradas		Salida
A	B	S
0	0	0
0	1	0
1	0	0
1	1	1

El circuito electrónico equivalente, o puerta lógica equivalente, es una puerta AND o puerta Y.



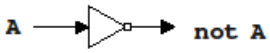
FUNCION COMPLEMENTO O NEGACION

Es una variable de una sola entrada y se representa por $S = \bar{a}$

y responde a la siguiente tabla de verdad

Entrada	Salida
A	S
0	1
1	0

La puerta lógica que resuelve la operación lógica se denomina puerta NOT o puerta NO



A	not A
0	
1	

Propiedades del álgebra de Boole

Cumple las propiedades de involución, idempotencia, conmutativa, asociativa, distributiva, existe elemento neutro y opuesto.

Propiedad de idempotencia.

$a.a = a \qquad a+a = a$

Ley de involución.

$\overline{\overline{a}} = a$

Propiedad conmutativa

$a + b = b + a$

$a \cdot b = b \cdot a$

Propiedad asociativa

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

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

Propiedad distributiva

Respecto al producto a . (b + c) = a . b + a . c

Respecto a la suma a + (b . c) = (a + b) . (a + c)

Existencia de elemento neutro

Para la suma: a + 0 = a

Para el producto a . 1 = a

Existencia de elemento opuesto

Para la suma $\overline{a}$ + a = 1

Para el producto a . a = 0

PUERTAS LOGICAS COMPLEJAS O UNIVERSALES

PUERTA NOR

Es una función inversa a la puerta OR. El resultado es la negación de la función suma. NOR significa NOT OR. Su expresión es

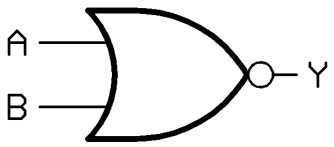
$$S=\overline{a+b}$$

que a través de un teorema llamado de Moran equivale a $S=\overline{a}\cdot\overline{b}$

La tabla de verdad equivalente es

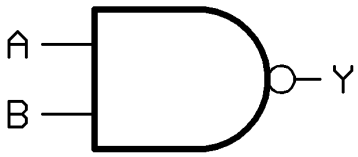
Entradas		Salida
A	B	S
0	0	1
0	1	0
1	0	0
1	1	0

La puerta NOR se representa como



PUERTA NAND

Es la puerta AND seguida de una negación. NAND significa NOT AND.



Su expresión algebraica es $S=\overline{a\cdot b}$

aplicando el teorma de Morgan $S=\overline{a}+\overline{b}$

PUERTA O-EXCLUSIVA

También llamada puerta EXOR. Se define como aquella que presenta a su salida el valor 1 cuando ambas variables son distintas y 0 si son iguales.

Su expresión es $S=a\circ b$

La tabla de verdad es

Entradas		Salida
A	B	S
0	0	0
0	1	1
1	0	1
1	1	0

Para realizar circuitos electrónicos que realicen estas operaciones, los fabricantes de componentes electrónicos construyen **circuitos integrados** basados en transistores, en cuyo interior implementan varias puertas. Las patillas del mismo constituirán las entradas, salidas y alimentación.

7. Obtención de la función lógica a partir de la tabla de verdad

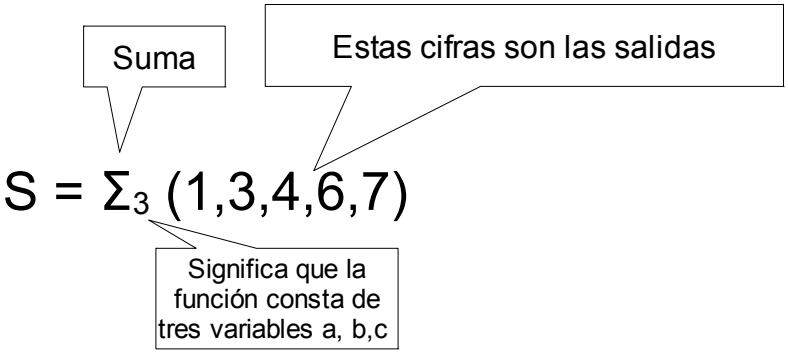
Hay dos formas de representar una función lógica mediante dos modalidades distintas:

- **Primera forma canónica** o suma de productos. **MIMTERMS**
- **Segunda forma canónica** o producto de sumas **MAXTERMS**

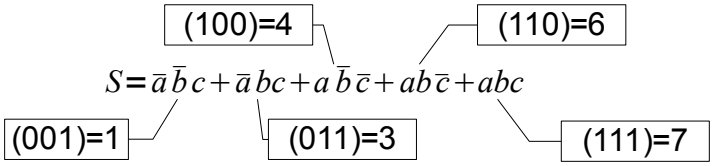
Una forma canónica de una función es todo producto de sumas o toda suma de productos en las que aparecen las variables.

En la primera forma, la función es la suma de todos los productos lógicos que dan salida 1, asignando el estado 0 a la variable inversa y al estado 1 la variable directa.

Ejemplo Sea la función



Así, esta función se representa como

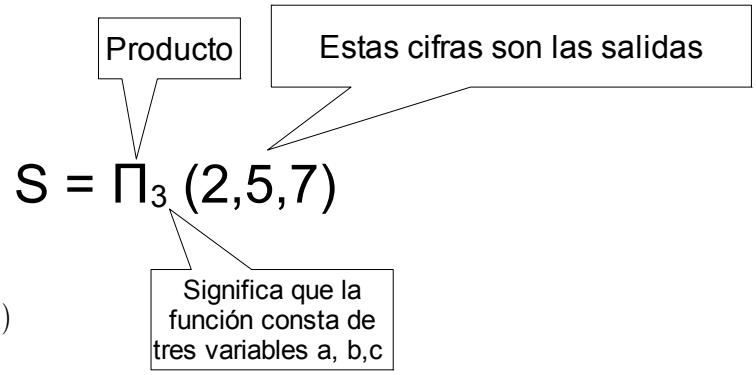


Ejemplo de tabla de equivalencia entre cada mimterm y cada producto para tres variables (a,b,c)

n	m	a	b	c
0	$m_0 = \bar{a}\,\bar{b}\,\bar{c}$	0	0	0
1	$m_1 = \bar{a}\,\bar{b}\,c$	0	0	1
2	$m_2 = \bar{a}\,b\bar{c}$	0	1	0
3	$m_3 = \bar{a}\,bc$	0	1	1
4	$m_4 = a\bar{b}\,\bar{c}$	1	0	0
5	$m_5 = a\bar{b}\,c$	1	0	1
6	$m_6 = ab\bar{c}$	1	1	0
7	$m_7 = abc$	1	1	1

- En la segunda forma canónica, la función es el producto de todas las sumas lógicas, que dan salida 0

Ejemplo: Sea la función



Así

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

n	M	a	b	c
0	$M_0 = a+b+c$	0	0	0
1	$M_1 = a+b+\bar{c}$	0	0	1
2	$M_2 = a+\bar{b}+c$	0	1	0
3	$M_3 = a+\bar{b}+\bar{c}$	0	1	1
4	$M_4 = \bar{a}+b+c$	1	0	0
5	$M_5 = \bar{a}+b+\bar{c}$	1	0	1
6	$M_6 = \bar{a}+\bar{b}+c$	1	1	0
7	$M_7 = \bar{a}+\bar{b}+\bar{c}$	1	1	1

Esta tabla establece la correspondencia entre los maxterms y cada una las sumas para tres variables.

Ejemplo: Obtener las formas canónicas de la función que viene en la tabla de verdad.

A	B	C	S	Minterm
0	0	0	0	m_0
0	0	1	0	m_1
0	1	0	1	m_2
0	1	1	1	m_3
1	0	0	1	m_4
1	0	1	1	m_5
1	1	0	1	m_6
1	1	1	0	m_7

La primera forma canónica pone la función como la suma de los productos lógicos cuyos coeficientes dan como **salida 1** (marcados en amarillo).

Así, esta función lógica será f(A,B,C) = m_2 + m_3 + m_4 + m_5 + m_6

que, finalmente, se representa como

$$f(A,B,C)=\bar{a}\,b\,\bar{c}+\bar{a}\,bc+a\,\bar{b}\,\bar{c}+a\,\bar{b}\,c+ab\,\bar{c}$$

$$010+011+100+101+110$$

Cada cero se corresponde con una variable con negación y cada uno con una variable sin negación

La segunda forma canónica se obtiene como producto de las sumas de los términos cuyo resultado sea nulo en la tabla de verdad

A	B	C	S	Maxterm
0	0	0	0	m_0
0	0	1	0	m_1
0	1	0	1	m_2
0	1	1	1	m_3
1	0	0	1	m_4
1	0	0	1	m_5
1	1	0	1	m_6
1	1	1	0	m_7

En este caso, la función será...

$$f(A,B,C) = m_0 \cdot m_1 \cdot m_7$$

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

Representación mediante puertas

8. Simplificación de funciones

Su objetivo es hacer más fáciles las operaciones que se van a efectuar y que el coste de los circuitos digitales sea el mínimo posible.

Existe un procedimiento gráfico conocido como mapa de Karnaugh y que se aplica a funciones con un número relativamente pequeño de variables no superior a seis. Este método se basa en la determinación, a partir de la tabla de verdad, de otra tabla, denominada tabla de Karnaugh, que se construye situando como entradas todas las posibles combinaciones de las variables de las que depende la función que se intenta simplificar, de manera que al pasar de una columna o de una fila a la contigua sólo cambie de valor una variable.

Para dos variables

A \ B	0	1
0	0	2
1	1	3

Para tres variables

C \ AB	00	01	11	10
0	0	2	6	4
1	1	3	7	5

Para cuatro variables

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

Cada cuadrícula de la tabla corresponde a una combinación de las variables cuya representación decimal se suele indicar en el ángulo inferior derecho de cada casilla.

Para representar una función en el mapa de Karnaugh partiendo de la primera forma canónica basta con escribir un 1 en las cuadrículas correspondientes a los términos que

estén presentes. Las demás casillas se dejan en blanco. Ejemplo:

Sea la función f(A,B,C) = $m_2 + m_3 + m_4 + m_5$

El mapa de Karnaugh correspondiente es:

C \ AB	00	01	11	10
0	0	1 (2)	6	1 (4)
1	1	1 (3)	7	1 (5)

Sabemos que la primera forma canónica es una suma de productos y que dos cuadrículas adyacentes no difieren entre sí más que en el valor de una variable.

Por tanto, en este ejemplo, si consideramos los términos m_2 y m_3, vemos que la función toma el valor 1, independientemente de los valores (0 o 1) que pueda tomar la variabla C. Por lo tanto, como esta variable no afecta, podemos prescindir de ella y teniendo en cuenta que

$$m_2 = \bar{a}\, b\, \bar{c}\,;\; m_3 = \bar{a}\, b\, c$$

la suma de ambos términos equivale a

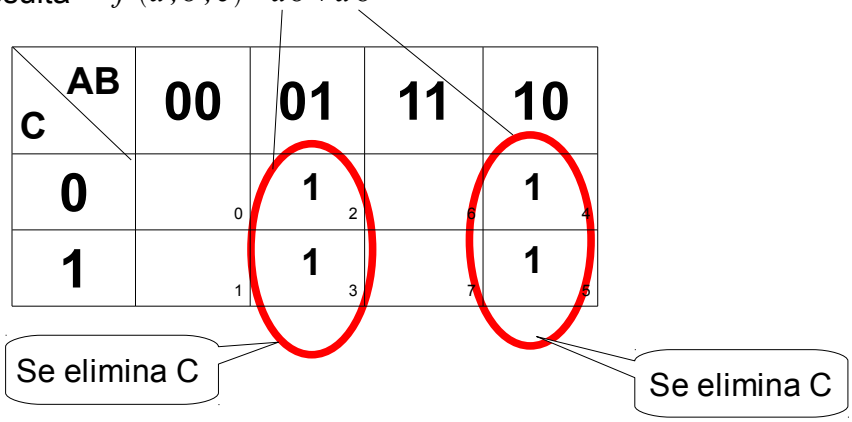
$$m_2 + m_3 = \bar{a}\, b\, (\bar{c} + c) = \bar{a}\, b$$

De una forma general, esto se simplifica en el mapa de Karnaugh estableciendo asociaciones de dos 2^n términos, siendo n el número de variables de las que depende la función. Así, en el caso n= 3, pueden realizarse agrupaciones de dos, cuatro u ocho términos, y no es válido, por ejemplo, agrupar seis términos aunque sean adyacentes. Cada asociación debe contener el número posible de cuadros, y el número de asociaciones debe ser mínimo.

En el ejemplo que estamos considerando, las asociaciones son

- Casillas 2 y 3: eliminan la variable C. El resultado es $\bar{a}\, b$
- Casillas 4 y 5: eliminan también la variable C. El resultado es $a\, \bar{b}$

con lo cual resulta $f(a,b,c) = \bar{a}\, b + a\, \bar{b}$



NOTA: Existe la posibilidad de que quede algún 1 aislado, sin posibilidad de reducción

con ningún término adyacente.

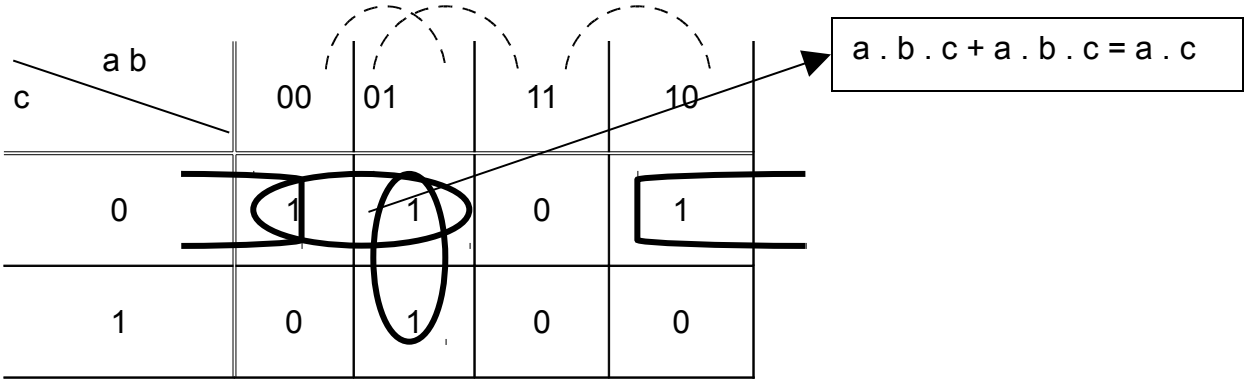
El mismo 1 puede ser utilizado en varias agrupaciones diferentes.

Ejemplo :

Obtener la función lógica más simple para la tabla de la verdad siguiente.

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

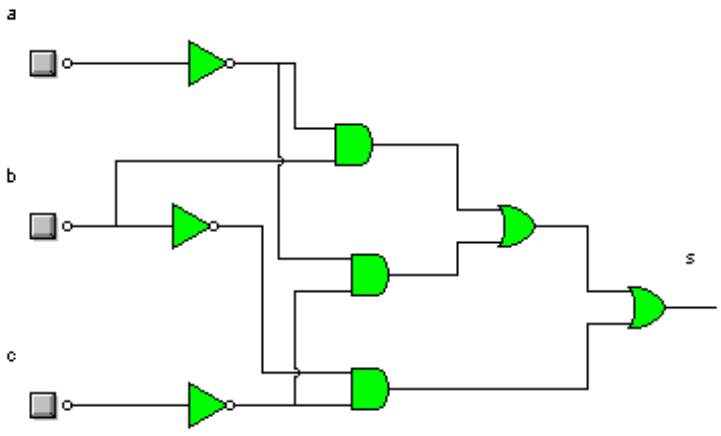
1. Lo siguiente que hacemos es plantear la tabla de Karnaugh, trasladando las combinaciones de la tabla de verdad a esta nueva tabla. Obsérvese como de una columna a otra sólo cambia un bit.



2. A continuación nos fijamos en qué tiene en común cada agrupación y obtenemos la función lógica

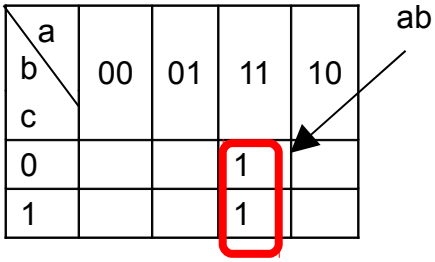
$$S = \bar{b}\,\bar{c} + \bar{a}\,b + \bar{a}\,\bar{c}$$

3. Por último planteamos el esquema o circuito lógico



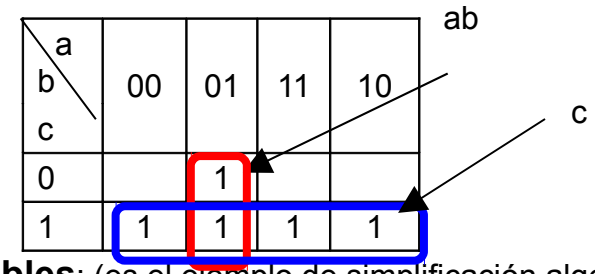
Ejemplo 1 con 3 variables:

f = $ab\bar{c} + abc$



Luego:
f = ab

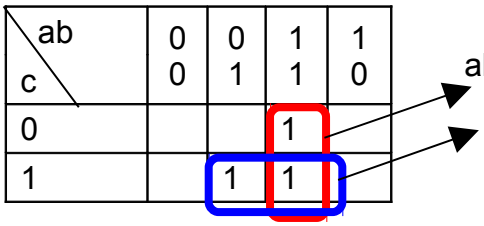
Ejemplo 2 con 3 variables: f = $\bar{a}b\bar{c} + \bar{a}bc + a\bar{b}c + abc + ab\bar{c}$



Luego:
f = $\bar{a}b$ + c

Ejemplo 3 con 3 variables: (es el ejemplo de simplificación algebraico anterior)

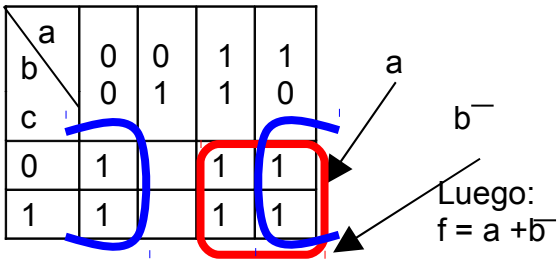
f = $\bar{a}bc + abc + ab\bar{c}$



Luego:
f = ab +bc

Ejemplo 4 con 3 variables:

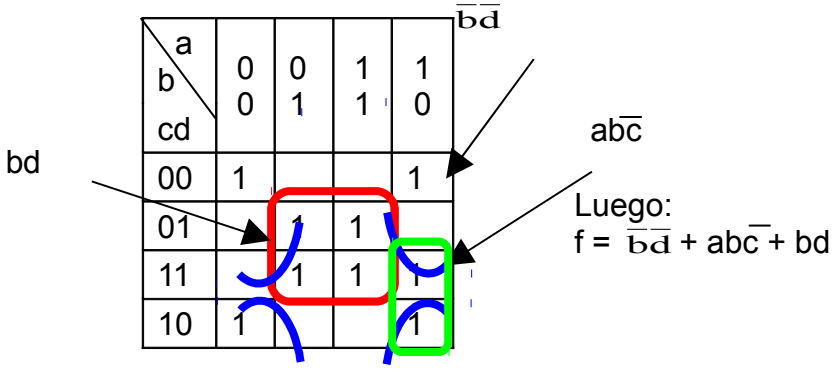
f = $\bar{a}\bar{b}\bar{c} + \bar{a}\bar{b}c + ab\bar{c} + abc + a\bar{b}\bar{c} + a\bar{b}c$



Luego:
f = a +$\bar{b}$

Ejemplo 5 con 4 variables:

f = $\bar{a}b\bar{c}d + \bar{a}bcd + ab\bar{c}d + abcd + \bar{a}\bar{b}\bar{c}d + \bar{a}\bar{b}c\bar{d} + a\bar{b}\bar{c}\bar{d} + a\bar{b}c\bar{d} + a\bar{b}cd$



Luego:
f = $\bar{b}\bar{d}$ + $ab\bar{c}$ + bd

9. Implementación de circuitos con puertas NAND y NOR

Los fabricantes suelen fabricar los circuitos lógicos con puertas NAND y NOR debido a su bajo precio. Para convertir a puertas NAND y NOR hay que usar las reglas de Morgan tantas veces como sea necesario hasta que toda la función se exprese con circuitos negados.

Las leyes de Morgan dicen:

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

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

Ejemplo: Implementar con puertas NAND la función: $S=\bar{a}\,b\,c+a\,\bar{b}\,c+abc$

Aplicamos la doble negación

$$S=\overline{\overline{\bar{a}\,bc+a\,\bar{b}\,c+abc}}=\overline{\overline{(a\bar{b}c)}\cdot\overline{(a\,\bar{b}c)}\cdot\overline{(abc)}}$$

Ejemplo: Implementar con puertas NOR la función $S=\bar{a}\,b\,c+a\,\bar{b}\,c+abc$

Aplicamos la doble negación

$$S=\overline{\overline{\bar{a}\,bc+a\,\bar{b}\,c+abc}}$$

Aplicamos una doble negación a cada término

$$S=\overline{\overline{(\overline{\bar{a}bc})+(\overline{a\,\bar{b}c})+(\overline{abc})}}=\overline{\overline{(a+\bar{b}+barc)}+\overline{(\bar{a}+b+barc)}+\overline{(\bar{a}+\bar{b}+\bar{c})}}$$

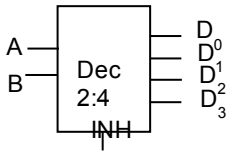
10. Circuitos combinacionales y secuenciales

Un circuito combinacional es aquel sistema cuya salida depende en todo momento de los valores binarios que adoptan las variables de entrada.

Un circuito secuencial es aquel cuya salida en cualquier momento depende no sólo de la entrada al circuito en ese instante determinado, sino también de la evolución que haya experimentado anteriormente, es decir, de la secuencia de entrada a que estuvo sometido.

11. Circuitos combinacionales más comunes

a) Decodificadores: Son circuitos integrados que presentan **n** entradas o líneas de selección y M (un número entero menor o igual a 2^n) salidas o líneas seleccionadas. En este circuito integrado se introduce un número y se activa una y sólo una de las salidas permaneciendo el resto de salidas desactivadas.



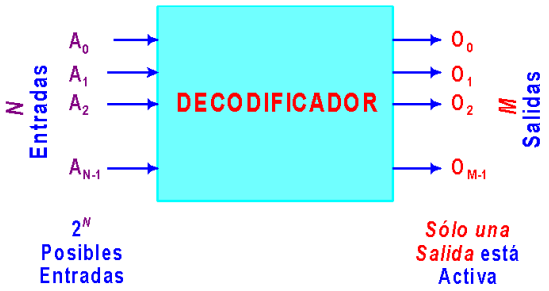
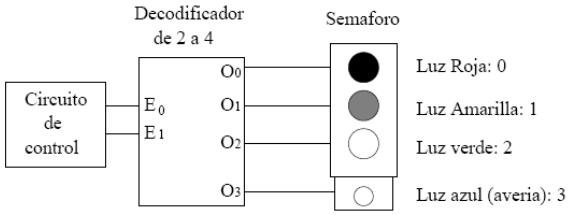


FIGURA 1. DIAGRAMA GENERAL DE UN *DECODIFICADOR*

Pueden ser de dos tipos:

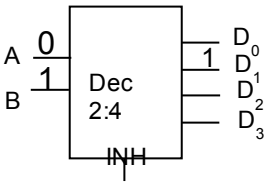
1. De **nivel activo alto**: La línea esta activa cuando están en 1



Si utilizamos un decodificador de 2 a 4, conseguiremos controlar el semáforo asegurándonos que sólo estará activa una luz en cada momento. Además, el circuito de control que diseñemos sólo tienen que tener 2 salidas.
Si el circuito de control envía el número 2, se encenderá la luz verde (que tiene asociado el número 2) y sólo la luz verde!!!. Un decodificador activa sólo una de las salidas, la salida que tiene un número igual al que se ha introducido por la entrada. En el ejemplo del semáforo, si el circuito de control envía el número 3, se activa la salida O_3 y se encenderá la luz azul (y sólo esa!!).

Ejemplo: Q_1 está tiene salida 1 si AB= 01, las demás salidas están a cero.

Entradas		Salidas			
A	**B**	**D_1**	**D_2**	**D_3**	**D_4**
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0



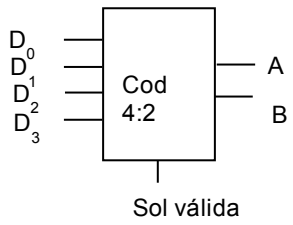
n entradas

2^n salidas

2. **Nivel activo bajo**: La línea se encuentra activa cuando está a cero. Siguiendo el ejemplo anterior.

Entradas		Salidas			
A	**B**	**D_1**	**D_2**	**D_3**	**D_4**
0	0	1	1	1	0
0	1	1	1	0	1
1	0	1	0	1	1
1	1	0	1	1	1

b) Codificadores



Es un circuito integrado que presenta 2^n líneas de entras y n líneas de salida, en las que aparece el código binario correspondiente a la línea de entrada activa en ese momento. Realiza la función opuesta al decodificador. No puede activarse más de una entrada al mismo tiempo.

2^n entradas
n salidas

Tabla de verdad:

D_0	D_1	D_2	D_3	A	B	Sol válida
1	0	0	0	0	0	1
0	1	0	0	0	1	1
0	0	1	0	1	0	1
0	0	0	1	1	1	1
0	0	0	0	0	0	0

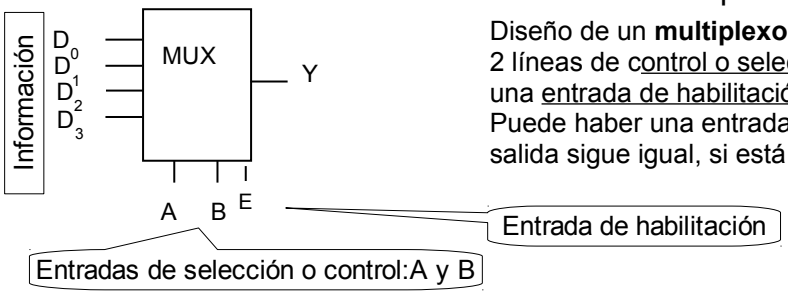
La solución válida será “1” siempre que haya un “1” en las señales de entrada.

c) Multiplexores y demultiplexores

El **multiplexor** es un circuito integrado en el que las entradas de control seleccionan una entrada entre varias para llevar la información de ésta entrada a una única salida.

Multiplexor: n líneas de **control o selección**, 2^n entradas, una línea de habilitación y una salida

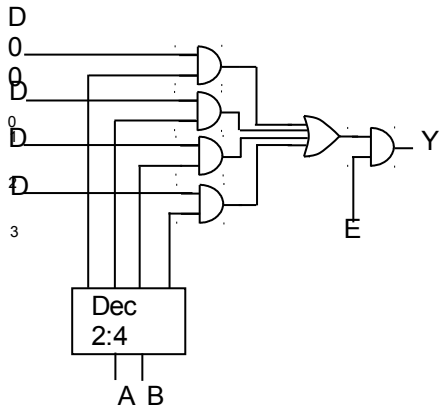
El funcionamiento de un multiplexor es el siguiente: cuando una combinación binaria aparece en las entradas de selección, la información que está presente en las entradas aparecerá en **una** salida. La línea de habilitación da paso a la salida si está a 1.



Diseño de un **multiplexor** 4:1
2 líneas de control o selección (A,B), 4 entradas (D_0,D_1,D_2,D_3) una entrada de habilitación (E) y una salida (Y).
Puede haber una entrada de habilitación (E): si está a “1” la salida sigue igual, si está a “0”, la salida es “0”.

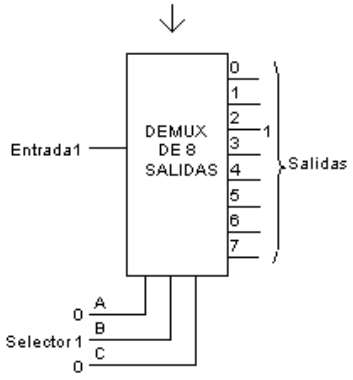
Tabla de verdad:

E. control		**Entradas**				**Salida**
A	**B**	**D_0**	**D_1**	**D_2**	**D_3**	**Y**
0	0	0				0
0	0	1				1
0	1		0			0
0	1		1			1
1	0			0		0
1	0			1		1
1	1				0	0
1	1				1	1



Los **demultiplexores** realizan una función contraria a los anteriores dirigiendo la información a la salida seleccionada mediante las entradas de control.

Es un circuito integrado que presenta una entrada de información, n líneas de control o selección y 2n salidas.



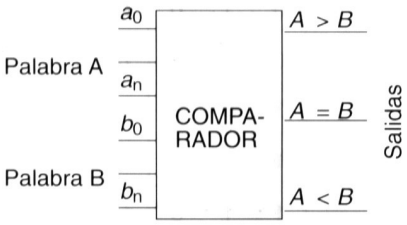
Ejemplo: Veamos un demultiplexor de nivel activo alto de tres líneas de control (A, B,C). Luego, tendrá 8 salidas y una sola entrada.

La tabla de verdad correspondiente sería

Entradas de control			**Salidas**							
A	**B**	**C**	**D_0**	**D_1**	**D_2**	**D_3**	**D_4**	**D_5**	**D_6**	**D_7**
0	0	0	E	0	0	0	0	0	0	0
0	0	1	0	E	0	0	0	0		0
0	1	0	0	0	E	0	0	0	0	0
0	1	1	0	0	0	E	0	0	0	0
1	0	0	0	0	0	0	E	0	0	0
1	0	1	0	0	0	0	0	E	0	0
1	1	0	0	0	0	0	0	0	E	0
1	1	1	0	0	0	0	0	0	0	E

En este caso, por ejemplo, si la entrada es 1 (E=1) y (A,B) = (1,0) en la salida D_2 aparece el valor 1, y si la entrada es 0 (E=0) y (A,B) = (1,0) en la salida D_2 aparece el valor 0.

d) Comparadores



Son circuitos que detectan las relaciones mayor (M >), menor (m <) e igual (I =).
Presentan dos grupos de n lineas de entrada (A y B) que son la expresión en binario de los números que queremos comparar y tres lineas de salida (M, I , m).

e) Operaciones en sistema binario (sumadores y restadores)

-SUMA

Se realiza de manera análoga a la decimal. Se comienza sumando individualmente los dígitos binarios que corresponden al bit menos significativo (el de la derecha) y que ocupan la misma posición, teniendo en cuenta el acarreo (lo que nos llevamos) resultante de la suma correspondiente a la posición anterior.

Ejemplo: Sumar 543 y 226 en binario
Primero pasamos ambos números a binario

$546_{(10)}=1000011111_{(2)}$
$226_{(10)}= \ 11100010_{(2)}$

Procedemos a efectuar la suma

Sistema decimal	**Sistema binario**	
543	11111111	acarreo
+226	10000111111	Sumando 1
769	+11100010	Sumando 2
	1100000001	Suma

En la tabla siguiente se describen las posibles combinaciones de sumas y los acarreos correspondientes

a	0	0	0	0	1	1	1	1
b	0	0	1	1	0	0	1	1
C_n	0	1	0	1	0	1	0	1
S	0	1	1	0	1	0	0	1
C_{n+1}	0	0	0	1	0	1	1	1

donde C_n es el acarreo de la suma anterior (de bits sumados antes de esta operación), S es la suma y C_{n+1} es el acarreo de la suma actual.

- DIFERENCIA

Se realiza de manera similar, utilizando acarreos

Ejemplo:restar 543 y 226 en binario

Primero pasamos ambos números a binario

$546_{(10)}=1000011111_{(2)}$
$226_{(10)}= \ 11100010_{(2)}$

Procedemos a efectuar la resta

Sistema decimal	**Sistema binario**	
543	1111	acarreo
- 226	10000111111	Minuendo
317	- 11100010	Sustraendo
	100111101	Resta

En circuitería electrónica la diferencia se hace con los complementos que permiten realizar la resta con un circuito sumador, para integrar en el hardware únicamente la suma y no la suma y la resta lo que complicaría el circuito. Para ello se utilizan dos métodos, el método de complemento a uno y el de complemento a dos.

Método de complemento a dos:

El complemento a dos de un número binario se obtiene cambiando los 1 por 0y los 0 por 1, sumándolo al resultado el valor 1. Si uno de los números tiene menos bits que el otro

previamente se le añaden ceros delante.
Además se debe añadir delante el bit de signo. Si el número es positivo se pone un 0 delante, si es negativo el 1, es decir, pondremos delante 1 en el minuendo si es negativo y pondremos delante 1 en el sustraendo si es positivo (porque lo estamos restando).
Si el resultado de la operación es positivo, éste aparecerá en binario normal, si el resultado es negativo aparecerá en complemento de a dos.
Si lo que se hace es la resta del tipo -a-b, a y b deben ambos complementarse y ambos deberán llevar un 1 como bit de signo.

Ejemplo

1. Restar 26-13

26 → 11010
13 → 1101; le añado 0 para que tenga los mismos bits que 26 13 → 01101
complemento cambiando 1 por 0 y 0 por 1 y le añado 1: 13_c → 10010+1 → 13_c → 10011

	Bit de signo	
+26	0	11010
- 13	1	10011
13	1 0	01101

Se desprecia Como me ha dado 0, el resultado se lee directamente y es positivo 01101 → 13

2. Restar 13-26

26 → 1101; le añado 0 para que tenga los mismos bits que 26 13 → 01101
13 → 11010; complemento cambiando 1 por 0 y 0 por 1; 26_c → 00101+1 → 26_c → 00110

	Bit de signo	
+13	0	01101
- 26	1	00110
-13	1	10011

Como me ha dado 1, el resultado no se lee directamente, sino que está complementado de a dos, y además sé que es negativo. Para deshacer el complemento primero resto 1: 10011-1=10010. A continuación cambio los 1 por 0 y viceversa, obteniendo el resultado -01101 → 13

Método de complemento a uno:

El complemento a uno de un número binario se obtiene cambiando los 1 por los 0 y los 0 por los 1. Si uno de los números tiene menos bits que el otro previamente se le añaden cero delante.
Además, se debe añadir delante el bit de signo. Si el número es positivo se pone un 0 delante, si es negativo el 1, es decir, pondremos delante 1 en el minuendo si es negativo y pondremos delante 1 en el sustraendo si es positivo (porque lo estamos restando).
Si el resultado de la operación es positivo, éste aparecerá en binario normal, si el resultado es negativo aparecerá en complemento de a uno.
Si lo que se hace es la resta del tipo -a-b, a y b deben ambos complementarse y ambos deberán llevar un 1 como bit de signo.

Ejemplo:

1. Restar 26-13

26 → 11010
13 → 1101; le añado 0 para que tenga los mismos bits que 26 13 → 01101 complemento cambiando 1 por 0 y 0 por 1 y le añado 1: 13_c → 10010

	Bit de signo	
+26	0	11010
- 13	1	10010
13	[1] 0	01100
	0	+1
	[0]	01101

Como me ha dado 0, el resultado se lee directamente y es positivo 01101 → 13

2. Restar 13-26

13 → 1101; le añado 0 para que tenga los mismos bits que 26 13 → 01101
26 → 11010; complemento cambiando 1 por 0 y 0 por 1; 26_c → 00101

	Bit de signo	
+13	0	01101
- 26	1	00101
-13	[1]	10011

Como me ha dado 1, el resultado no se lee directamente, sino que está complementado de a uno, y además sé que es negativo. Para deshacer el complemento cambio los 1 por 0 y viceversa, obteniendo el resultado -01101 → 13

12. Circuitos secuenciales

A los circuitos vistos hasta ahora se les estudia dentro de la llamada “lógica combinacional”, en todos ellos, dependiendo de los valores de las variables de entrada se obtiene una única salida.

Los circuitos secuenciales son aquellos cuya salida en cualquier momento no depende sólo de la entrada al circuito sino también de la secuencia de entradas a las que estuvo sometido anteriormente. Destacan, pues, por su capacidad de “memorizar” información

Pueden clasificarse en dos grandes grupos:

Asíncronos: Los cambios de estado se producen cuando están presentes las entradas adecuadas.

Síncronos: Los cambios de estado se producen cuando además de estar presentes las entradas adecuadas se produce la transición de una señal compartida por los elementos del sistema y que sincroniza su funcionamiento. A esta señal se le llama señal de reloj o de clock

Los dispositivos secuenciales más simples y elementales son los **biestables**. La unión de biestables da lugar a otros circuitos secuenciales más complejos, como son los **contadores** y los **registros de desplazamiento.**