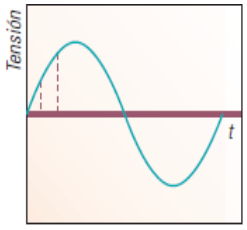
CONTENIDO OBTENIDO DEL LIBRO DIGITAL DE EDEBE 4º DE LA ESO

TEMA 3: ELECTRÓNICA DIGITAL

1. TIPOS DE SEÑALES.

El control sobre un proceso depende del tipo de señal que este proporciona.

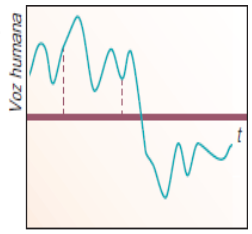
Señal: variación en el tiempo de una magnitud física (temperatura, presión, velocidad…), que permite transmitir información.



Las señales se clasifican en dos grandes grupos: analógicas y digitales.

1.1. SEÑALES ANALÓGICAS.

Las señales analógicas son continuas, la variable estudiada **toma un número infinitos valores en un intervalo de tiempo.** En la naturaleza las señales son analógicas (luz, sonido…)

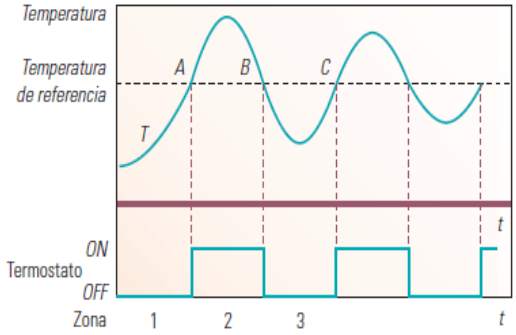


En las gráficas se pueden ver 2 ejemplos de señales analógicas: la evolución del registro de la voz humana y de la tensión de una corriente alterna monofásica en función del tiempo.

1.2. SEÑALES DIGITALES.

Las señales digitales son discretas, la variable estudiada **toma un número finito valores en un intervalo de tiempo**. Se clasifican según el número de valores en binarias (las más utilizadas), ternarias, etc.

Por ejemplo, en un horno en el que se controla su funcionamiento con un termostato, La gráfica que representa la temperatura es una señal analógica. En cambio, el termostato solo puede tomar dos valores (OFF y ON), se trata por tanto de una señal digital binaria.



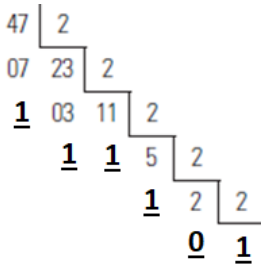
Las señales digitales binarias pueden tomar dos valores, el 0 y el 1, tal como se muestra en la tabla:

VALOR LÓGICO	CIRCUITO	SEÑAL
0	Abierto	*OFF*
1	Cerrado	*ON*

El **código binario natural (**también llamado simplemente *binario*) utiliza los símbolos **0** y **1.** Todos los números naturales del sistema decimal tienen sus equivalentes en el sistema binario.

Para obtener la equivalencia de un número natural en el sistema binario:

- Dividimos el número entre 2 y repetimos la división con el cociente obtenido hasta que el cociente sea igual a 1.
- Escribimos un número formado por el último cociente y los restos obtenidos, en orden inverso a como aparecen.



Ejemplo: la expresión del número 47 en código binario natural es: $47 = 101111_{2)}$

El **código BCD** representa las cifras del sistema decimal (0-9) utilizando **cuatro bits**, por lo que solo utiliza 10 combinaciones de las $2^4 = 16$ posibles. **La asignación de bits coincide con la del código binario natural.**

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

2. ÁLGEBRA DE BOOLE.

En 1854 el matemático George Boole publicó un estudio sobre cómo representar y trabajar la lógica proposicional con elementos y operaciones del álgebra matemática. El contenido de este estudio definió las bases del álgebra de Boole.

La lógica proposicional es un sistema formal que permite representar enunciados y demostrar su validez.

2.1. OPERACIONES DEL ÁLGEBRA DE BOOLE.

El álgebra de Boole define:

- Que **cada variable de entrada** solo puede tomar dos posibles **valores, 0 y 1**.
- Que **cada variable** o expresión **está identificada por una letra** (A, B, C, etc.). Cada una de estas variables puede representar las entradas de una señal en un sistema digital.
- **Tres operaciones básicas:**
  - **Producto lógico:** se usa el operador (·) o **ningún símbolo** entre las dos variables implicadas en la operación.
  - **Suma lógica:** se emplea el operador (+) entre las dos variables implicadas en la operación.
  - **Negación lógica o complementario**: se utiliza el operador (¯) sobre la variable afectada.

Las tablas siguientes muestran los resultados de cada operación para cada posible valor de las variables de entrada:

PRODUCTO LÓGICO

a	b	$a \cdot b = P$
0	0	0
0	1	0
1	0	0
1	1	1

SUMA LÓGICA

a	b	$a + b = S$
0	0	0
0	1	1
1	0	1
1	1	1

ELEMENTO SIMÉTRICO

a	$\bar{a}$	$a \cdot \bar{a}$
1	0	0
0	1	0

ELEMENTO SIMÉTRICO

a	$\bar{a}$	$a + \bar{a}$
1	0	1
0	1	1

2.2. PROPIEDADES DEL ÁLGEBRA DE BOOLE.

Al igual que en el álgebra matemática, las operaciones lógicas de suma y producto también cumplen las siguientes propiedades:

PROPIEDADES	SUMA	PRODUCTO
Conmutativa:	**a + b = b + a**	**ab = ba**
Asociativa:	**(a + b) + c = a + (b + c)**	**(ab)c = a(bc)**
Distributiva:	**a(b + c) = (ab) + (ac)**	**a + (bc) = (a + b)(a + c)**
Elemento neutro:	**0 + a = a**	**1a = a**
Complementario:	**a + ā = 1**	**a · ā = 0**

PRINCIPIO DE DUALIDAD

3. FUNCIONES LÓGICAS.

Una **función lógica** es la **expresión algebraica que evalúa el resultado** de una combinación de determinadas **operaciones lógicas** (sumas, productos y negaciones) sobre **variables binarias.**

Una **variable binaria** es la que solo puede tomar dos posibles valores: cierto (1) o falso (0).

Las *expresiones algebraicas* son combinaciones de números, variables y operaciones matemáticas, como la suma, resta, multiplicación y división.

Las siguientes expresiones son **ejemplos de funciones lógicas** donde se combinan **operaciones con operadores lógicos** y **variables binarias** (0, 1): $F = (A + B)C + (BC)$ $F = A(B + C)$

El **resultado de una función lógica también es binario**, o sea, que solo puede tomar el valor 0 o 1. Este valor dependerá del valor actual que tome cada una de las variables de entrada de la función.

4 TABLAS DE VERDAD.

Permiten **representar** todas las **posibles combinaciones de valores** que pueden tomar las variables de entrada y, por cada combinación, **ver el resultado** que evalúa **la función representada** en la tabla.

Para construir la tabla de verdad de una función lógica hay que seguir los siguientes pasos:

1. Se añaden tantas columnas como variables diferentes tenga la función lógica.
2. Se descompone la expresión inicial y se añaden tantas columnas como operaciones simples y operaciones entre ellas.
3. En cada fila se coloca una combinación única de valores de las variables de entrada. El número de filas dependerá del número de variables (n): **filas = 2^n**. (Ejemplo: con 3 variables necesitaremos $2^3 = 8$ filas).
4. Por cada combinación de valores de las variables de entrada se resuelven cada una de las columnas.

EJEMPLO: construye la tabla de verdad de la función: $F = AB + \bar{A}B$

- Preparamos una tabla e incluimos en ella las cuatro combinaciones de estados posibles y todas las operaciones simples en las que se ha descompuesto la función.
- Completamos la columna ***AB*** con el resultado del producto lógico de cada combinación de valores de las variables *A* y *B*.
- Completamos la columna $\bar{A}$ con el resultado del complemento o negación de la variable *A*.
- Completamos la columna $\bar{A}B$ con el resultado del producto lógico de cada combinación de valores de las variables $\bar{A}$ y *B*.
- Completamos la columna $F = AB + \bar{A}B$ con el resultado de la suma lógica de las columnas *AB* y $\bar{A}B$.

A	B	AB	$\bar{A}$	$\bar{A}B$	$F=AB+\bar{A}B$
0	0				
0	1				
1	0				
1	1				

A	B	AB	$\bar{A}$	$\bar{A}B$	$F=AB+\bar{A}B$
0	0	**0**			
0	1	**0**			
1	0	**0**			
1	1	**1**			

A	B	AB	$\bar{A}$	$\bar{A}B$	$F=AB+\bar{A}B$
0	0	0	**1**		
0	1	0	**1**		
1	0	0	**0**		
1	1	1	**0**		

A	B	AB	$\bar{A}$	$\bar{A}B$	$F=AB+\bar{A}B$
0	0	0	1	**0**	
0	1	0	1	**1**	
1	0	0	0	**0**	
1	1	1	0	**0**	

A	B	AB	$\bar{A}$	$\bar{A}B$	$F=AB+\bar{A}B$
0	0	0	1	0	**0**
0	1	0	1	1	**1**
1	0	0	0	0	**0**
1	1	1	0	0	**1**

5. ENCONTRAR LA FUNCIÓN LÓGICA A PARTIR DE SU TABLA DE VERDAD.

También es posible el proceso inverso: encontrar la función lógica a partir de su tabla de verdad. Tomando la tabla de verdad, podemos determinar la función lógica de dos formas posibles: suma de productos o producto de sumas.

5.1. <u>SUMA DE PRODUCTOS</u> <u>CON 2 VARIABLES</u> (Sum of Products: SoP).

Esta forma de obtener la función lógica también se conoce como **suma de minterms, primera forma canónica** o **forma canónica disyuntiva (OR)**. Debemos seguir los siguientes pasos:

A	B	F	
0	0	**0**	
0	1	**1**	→ $(\bar{A}B)$
1	0	**0**	
1	1	**1**	→ (AB)

1. **Localizar cada fila** donde el resultado de **la función devuelve 1 (F = 1). En el ejemplo hay 2 filas en las que F = 1.**
2. Para obtener los minitérminos: en cada fila en la que **F = 1**, aquellas variables que tomen **valor 1** se representan de **forma natural** en el producto canónico, mientras que aquellas variables que tomen **valor 0** se representan de **forma complementada. En el ejemplo obtenemos los siguientes minitérminos: *$(\bar{A}B)$ y (AB)*.**
3. La **primera forma canónica** se obtiene <u>**sumando**</u> **todos los minitérminos** en los que **la función vale 1:**

$$\mathbf{F = (\bar{A}B) + (AB)}$$

5.2. PRODUCTO DE SUMAS CON 2 VARIABLES (Products of Sums: PoS).

Esta forma también se conoce **producto de maxterms**, **segunda forma canónica** o **forma canónica conjuntiva (AND)**. Debemos seguir los siguientes pasos:

A	B	F	
0	0	0	→ (A + B)
0	1	1	
1	0	0	→ ($\bar{A}$ + B)
1	1	1	

1. **Localizar cada fila** donde el resultado de **la función devuelve 0 (F = 0). En el ejemplo hay 2 filas en las que F = 0.**
2. Para obtener los maxitérminos: en cada fila en la que **F = 0**, aquellas variables que tomen **valor 0** se representan de **forma natural** en el producto canónico, mientras que aquellas variables que tomen **valor 1** se representan de **forma complementada. En el ejemplo obtenemos los siguientes maxitérminos: *(A + B) y ($\bar{A}$ + B).***
3. La **segunda forma canónica** se obtiene **multiplicando todos los maxitérminos** en los que **la función vale 0:**

$$F = (A + B) \cdot (\bar{A} + B)$$

$$F = (A+B)(A'+B) = AA'+AB+BA'+BB = 0+AB+BA'+B = B(A+A')+B = B+B = B$$

(0 = AA′; B = BB; 1 = A+A′)

Se pueden obtener los mismos resultados con diferentes funciones lógicas, es decir, serían funciones lógicas equivalentes. **Las funciones se pueden simplificar** usando las propiedades y los teoremas del álgebra de Boole.

Otro ejemplo, con tres variables, de obtención de la función a partir de la tabla de la verdad sería:

5.3. SUMA DE PRODUCTOS CON 3 VARIABLES (SoP).

	A	B	C	F(A,B,C)	
0	0	0	0	1	→ ($\bar{A}\,\bar{B}\,\bar{C}$)
1	0	0	1	0	
2	0	1	0	0	
3	0	1	1	1	→ ($\bar{A}BC$)
4	1	0	0	0	
5	1	0	1	1	→ ($A\bar{B}C$)

1. En el ejemplo hay **4 filas** en las que **F = 1**.
2. Los **minitérminos** serían: ***($\bar{A}\,\bar{B}\,\bar{C}$), ($\bar{A}BC$), ($A\bar{B}C$) y (ABC).***
3. La **primera forma canónica** sería: **F = ($\bar{A}\,\bar{B}\,\bar{C}$) + ($\bar{A}BC$) + ($A\bar{B}C$) + (ABC)**

$$S = \sum_{i=0}^{2^n-1} m_i \, cuando \, F(i) = 1$$

S=m0+m
$S = (\bar{A}\cdot B\cdot \bar{C}) + (A\cdot B\cdot \bar{C}) + (A\cdot B\cdot C) + (A\cdot \bar{B}\cdot \bar{C})$

$$F = (A+B)(A'+B) = AA'+AB+BA'+BB = 0+AB+BA'+B = B(A+A')+B = B+B = B$$

(0 = AA′; B = BB; 1 = A+A′)

5.4. PRODUCTO DE SUMAS CON 3 VARIABLES (PoS)

	A	B	C	F(A,B,C)	
0	0	0	0	1	
1	0	0	1	0	→ (A + B + $\bar{C}$)
2	0	1	0	0	→ ($A + \bar{B} + C$)
3	0	1	1	1	
4	1	0	0	0	→ ($\bar{A} + B + C$)
5	1	0	1	1	
6	1	1	0	0	→ ($\bar{A} + \bar{B} + C$)
7	1	1	1	1	

1. En el ejemplo hay **4 filas** en las que **F = 0**.
2. Los maxitérminos serían: ***(A + B + $\bar{C}$), (A + $\bar{B}$ + C), ($\bar{A}$ + B+ C), ($\bar{A}$ + $\bar{B}$ + C)***
3. La **segunda forma canónica sería:**

$$F = (A + B + \bar{C}) \cdot (A + \bar{B} + C) \cdot (\bar{A} + B + C) \cdot (\bar{A} + \bar{B} + C)$$

$$S = \prod_{i=0}^{2^n-1} M_i \, cuando \, F(i) = 0$$

Segunda forma canónica

S=M1·M2·M4·M6

$S = (A + B + \bar{C}) \cdot (A + \bar{B} + C) \cdot (\bar{A} + B + C) \cdot (\bar{A} + \bar{B} + C)$

Las formas canónicas usan minterms o maxterms.

Para que sean **minterms** o **maxterms** tienen que **aparecer todas las variables** (negadas o no) en el término.

Si tengo 3 variables (A, B, C), un ejemplo de minterm sería ***A·B·$\bar{C}$***, pero no sería ***A·B***.
Un ejemplo de maxterm sería ***$\bar{A}$+B+C***, pero no sería ***$\bar{A}$+C***.

6. PRINCIPIOS Y LEYES DEL ALGEBRA DE BOOLE.

PRINCIPIO DE DUALIDAD	1 ⇨ 0 0 ⇨ 1	+ ⇨ · · ⇨ +

6.1. PRINCIPIO DE DUALIDAD.

Cualquier teorema puede transformarse en un segundo teorema sin más que **intercambiar (+) por (·), (·) por (+), 1 por 0 y 0 por 1.** Ejemplo: $A + \overline{A} = 1 \rightarrow A \cdot \overline{A} = 0$

6.2. LEYES DE MORGAN.

LEYES DE MORGAN

1ª: $\overline{(AB \dots Z)} = \overline{A}+\overline{B}+\dots+\overline{Z}$

2ª: $\overline{(A+B+\cdots+Z)} = \overline{A}\cdot\overline{B}\cdot\dots\cdot\overline{Z}$

Primera ley de Morgan: el **complemento de un producto de *n variables* es igual** a la **suma de los complementos de esas *n variables***. $\overline{(AB \dots Z)} = \overline{A} + \overline{B} + \dots + \overline{Z}$

Segunda ley de Morgan: el **complemento de una suma de *n variables* es igual** al **producto de los complementos de esas *n variables***. $\overline{(A + B + \cdots + Z)} = \overline{A} \cdot \overline{B} \cdot \dots \cdot \overline{Z}$

7. SIMPLIFICACIÓN DE FUNCIONES LÓGICAS.

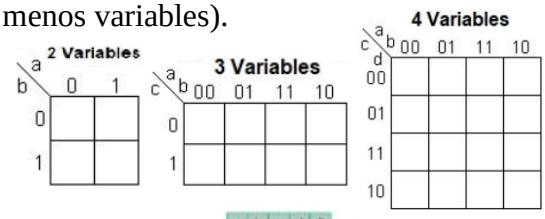
7.1. SIMPLIFICACIÓN CON TEOREMAS.

Es importante simplificar una función lógica porque permite que su realización sobre un circuito digital sea más simple. Seguidamente se muestra una lista de teoremas y su versión dual que se utilizan para la simplificación de expresiones booleanas:

	FUNCIÓN	FUNCIÓN DUAL
1	A + 0 = A	A · 1 = A
2	A + 1 = 1	A · 0 = 0
3	$\overline{\overline{A}} = A$	
4	A + A = A	A · A = A
5	$A + \overline{A} = 1$	$A \cdot \overline{A} = 0$
6	A + AB = A	A(A + B) = A
7	A + B = B + A	A · B = B · A
8	(A + B) + C = A + (B + C)	(A · B)C = A(B · C)
9	A(B + C) = AB + AC	A + BC = (A + B)·(A + C)
10	$(A + B)\cdot(A + \overline{B}) = A$	$AB + A\overline{B} = A$ $\boxed{A(B + \overline{B}) = A}$
11	$\overline{A + B} = \overline{A} \cdot \overline{B}$ (L. MORGAN)	$\overline{A \cdot B} = \overline{A} + \overline{B}$ (L. MORGAN)

7.2. SIMPLIFICACIÓN GRÁFICA (MAPAS DE KARNAUGH).

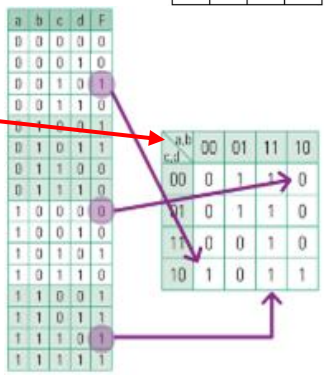
Un mapa de Karnaugh es un método gráfico para simplificar funciones lógicas. Permiten **reducir una función lógica a su forma más simple** (con menos términos y con menos variables).



Para realizar un mapa de Karnaugh se siguen los siguientes pasos:

1. **Dibujar la tabla**, hay que asignar a cada fila y a cada columna una combinación de valores de las variables de tal manera que entre cada fila o columna consecutiva **no haya más de un cambio de valor de las variables.** Truco: poner todos los **1** seguidos (00, 0**1**, **11**, **1**0).
2. **Se rellena la tabla** con los valores de cada combinación de las variables de entrada.
3. **Se hacen grupos de unos (1).** A la hora de agrupar los unos:
   - **A.** Debemos utilizar **todos los unos**.
   - **B.** Es mejor crear el **menor número de grupos**.
   - **C.** Los **unos pueden estar en varios grupos**.
   - **D.** En los grupos el **número de unos** debe ser **potencia de 2,** es decir, (1, 2. 4, 8, 16…). No valdría 3, 5,…

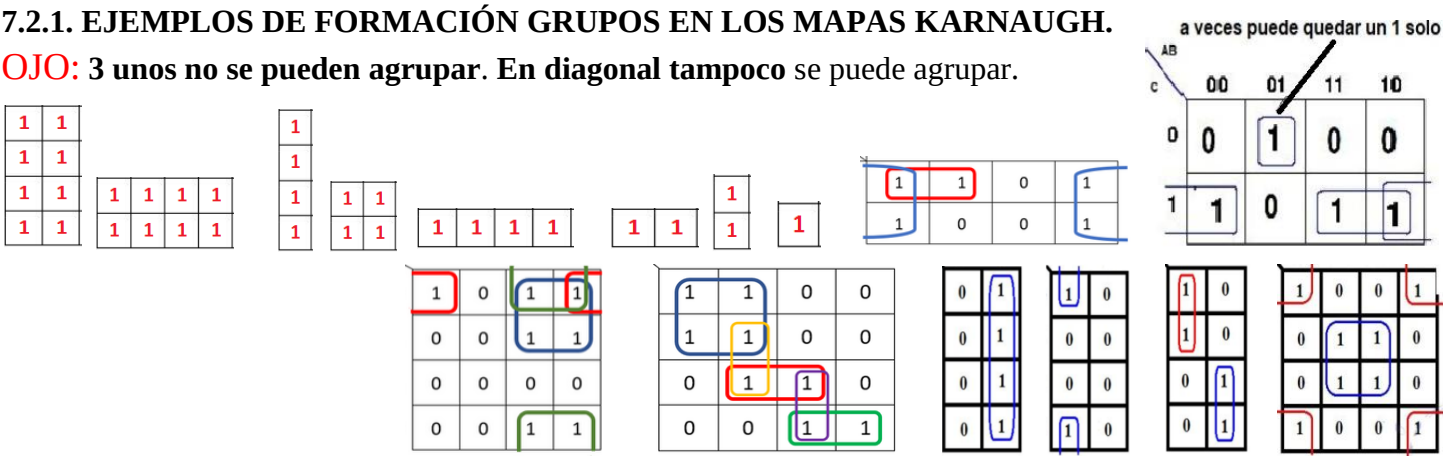
     Es **importante** tener en cuenta que en la tabla la **parte de arriba es como si estuviese unida a la parte de abajo y la parte de la izquierda, a la parte de la derecha.**
   - **E. Cuanto más grande sea un grupo mejor**, la simplificación de la función será mejor.
4. **Cada grupo se traduce** como el **producto de las variables que se mantienen constantes** con el mismo valor dentro del grupo. **Las variables que tomen valor 1** se representan de **forma natural** y **las variables que tomen valor 0** se representan de **forma complementada.**
5. **Se suman todas estas expresiones** correspondientes a cada grupo.

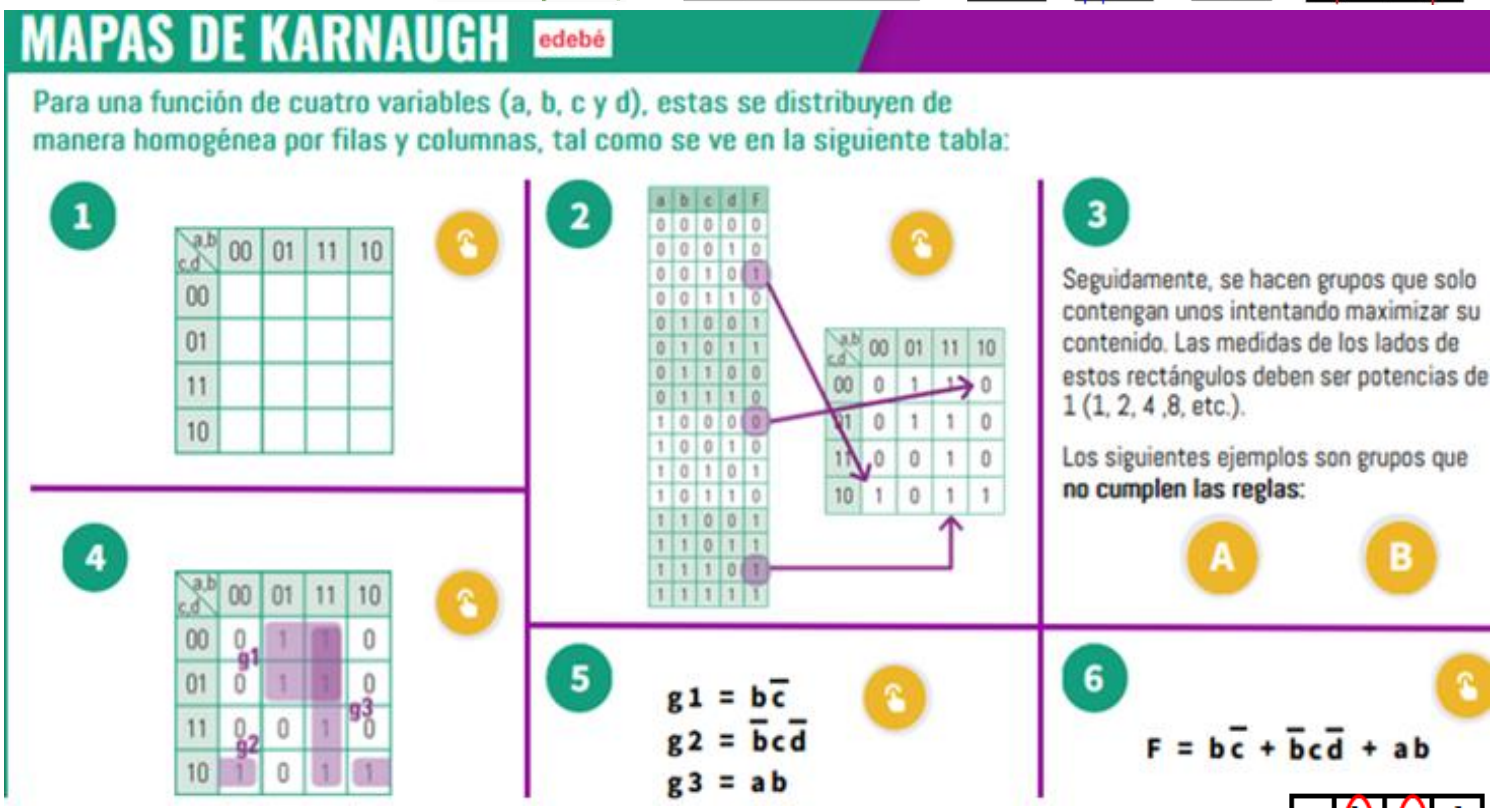


Karnaught se suele usar como máximo con 4 variables, porque si no se complica mucho.

7.2.1. EJEMPLOS DE FORMACIÓN GRUPOS EN LOS MAPAS KARNAUGH.

OJO: **3 unos no se pueden agrupar**. **En diagonal tampoco** se puede agrupar.



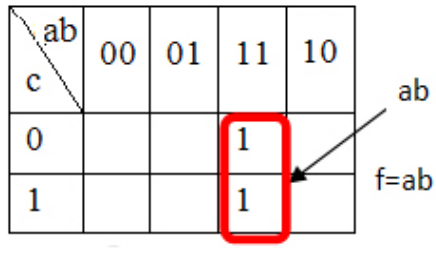


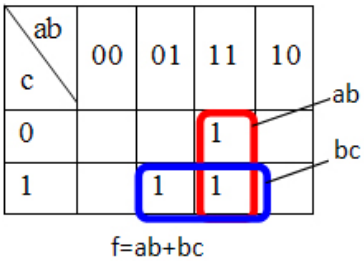
Por ejemplo, el **grupo 1 (g1)** se tradujo como **g1 = bc̄** porque las únicas variables que no cambian en las cuatro combinaciones (filas) son la **b** y la **c. La b** se pone **de forma natural,** porque siempre vale 1**, y la c** se pone **de forma complementada,** porque siempre vale 0.

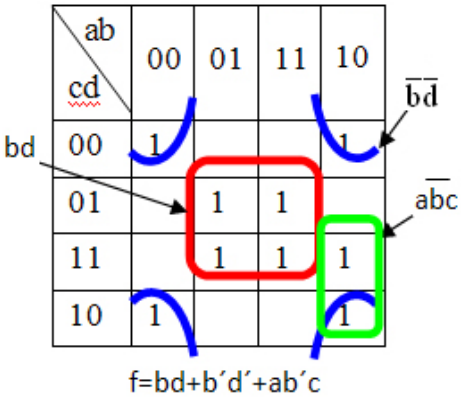
a	b	c	d
0	1	0	0
0	1	0	1
1	1	0	0
1	1	0	1

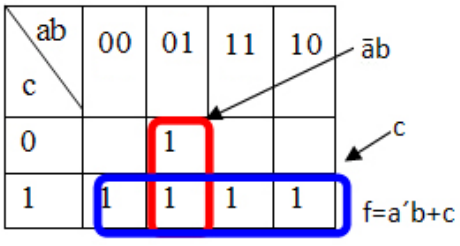
b c̄

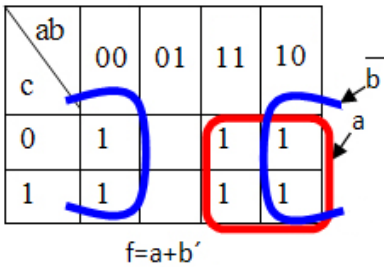
EJEMPLOS KARNAUGH:











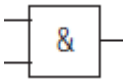
8. PUERTAS LÓGICAS.

Las **puertas lógicas** son los componentes básicos de los sistemas digitales. Generan un resultado o salida a partir de los datos de entrada que se le suministran.

SIMBOLOGÍA.

Las puertas lógicas se pueden representar de dos formas:

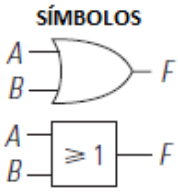
- La **norma DIN** emplea cuadrados para representar cada una de las funciones y, para distinguirlas, incorporan un símbolo en su interior: &, ≥ 1, = 1...
- La **norma ASA** usa símbolos que adoptan diferente forma para cada una de las funciones.



8.1. Función OR (O).

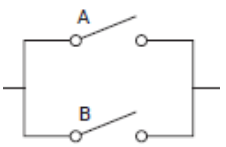
Se identifica con la función **suma lógica (+):** suma las señales que recibe. Genera un 1 en la salida cuando alguna de sus variables o entradas es un 1.

ECUACIÓN LÓGICA $F = A + B$



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

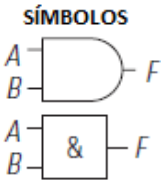
TABLA DE LA VERDAD



8.2. Función AND (Y).

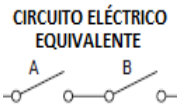
Se identifica con la función **producto lógico (* o nada):** multiplica las señales que recibe. Proporciona un 1 en salida cuando todas las variables o entradas toman el valor 1.

ECUACIÓN LÓGICA $F = A * B = AB$



A	B	F=AB
0	0	0
0	1	0
1	0	0
1	1	1

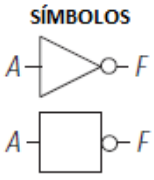
TABLA DE LA VERDAD



8.3. Función NOT (NO).

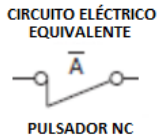
Corresponde a la función **complementación o negación**(¯): invierte el valor de la entrada.

ECUACIÓN LÓGICA $F = \overline{A}$



A	F=Ā
0	1
1	0

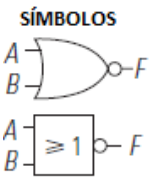
TABLA DE LA VERDAD



8.4. Función NOR (NO O).

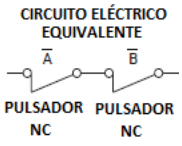
Es la **función complementaria o negación de la función OR,** es decir, justamente opuesta a ella. Produce señal de salida cuando no hay ninguna señal de entrada.

ECUACIÓN LÓGICA $F = \overline{A + B}$



A	B	F = Ā + B̄
0	0	1
0	1	0
1	0	0
1	1	0

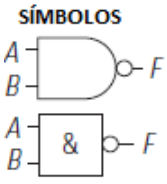
TABLA DE LA VERDAD



8.5. Función NAND (NO Y).

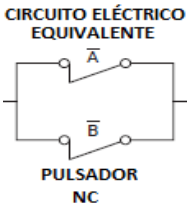
Es la **función complementaria o negación de la función AND.** Proporciona un 1 en la salida, excepto cuando todas las variables de entrada son 1.

ECUACIÓN LÓGICA $F = \overline{A * B} = \overline{AB}$



A	B	F = ĀB̄
0	0	1
0	1	1
1	0	1
1	1	0

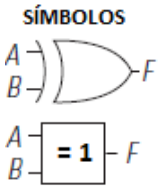
TABLA DE LA VERDAD



8.6. Función XOR (O exclusiva).

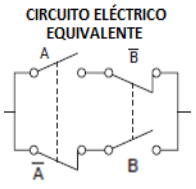
Se denomina **función dilema (⊕)**. Deriva de una combinación de funciones AND, OR y NOT, aunque se considera una función elemental. No produce señal de salida cuando las dos entradas son iguales.

ECUACIÓN LÓGICA $F = A\overline{B} + \overline{A}B = A \oplus B$



A	B	Ā	B̄	AB̄	ĀB	F = A ⊕ B
0	0	1	1	0	0	0
0	1	1	0	0	1	1
1	0	0	1	1	0	1
1	1	0	0	0	0	0

TABLA DE LA VERDAD

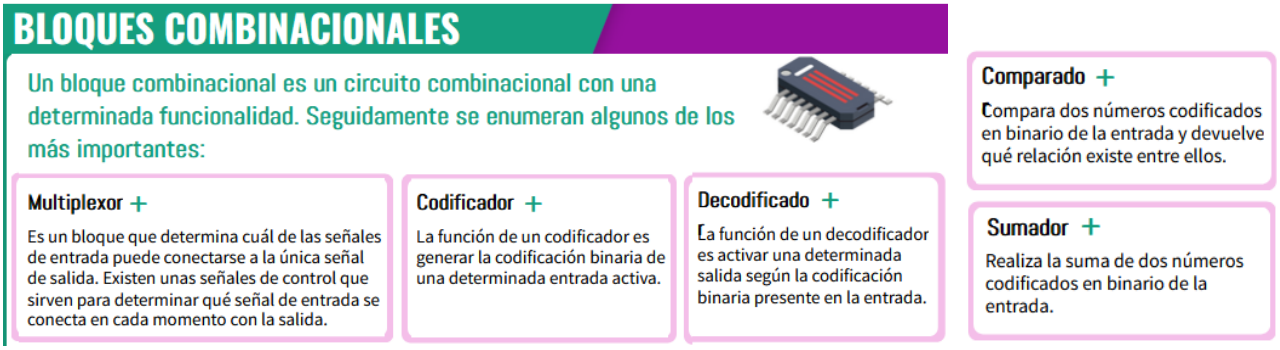


9. CIRCUITOS CON PUERTAS LÓGICAS.

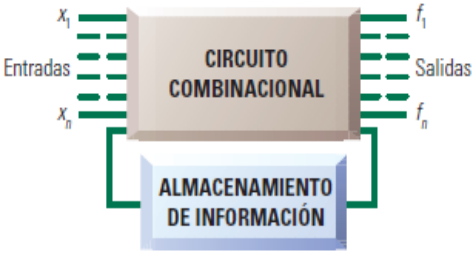
Los circuitos lógicos, atendiendo a su naturaleza, pueden ser de dos tipos: combinacionales o secuenciales.

9.1. CIRCUITOS COMBINACIONALES.

La característica principal es que **los estados de las salidas dependen únicamente de la combinación de los estados de las entradas** en cada instante de tiempo. Así, una combinación determinada de los estados de las variables de entrada, ***A*** y ***B,*** siempre da lugar a los mismos estados de salida, ***S*** y ***C***.



9.2. CIRCUITOS SECUENCIALES.

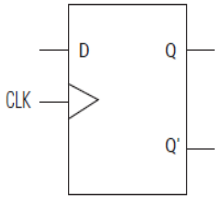


Se caracterizan por que en ellos **los estados de las variables de salida dependen no solo del estado actual de las entradas**, **sino también del estado anterior** en el que se encontraban; es decir, evolucionan con las entradas pero **«recordando» su estado anterior.**

Son capaces de almacenar secuencias de datos de forma indefinida, dando lugar, de este modo, a las variables internas del sistema.

9.3. BIESTABLES.

Son circuitos secuenciales cuya salida presenta **dos estados estables,** es decir, el circuito puede permanecer en cada uno de estos estados de forma indefinida, aunque desaparezcan las señales que los produjeron. También llamados *básculas*, *flip-flops* o **células elementales de memoria.**



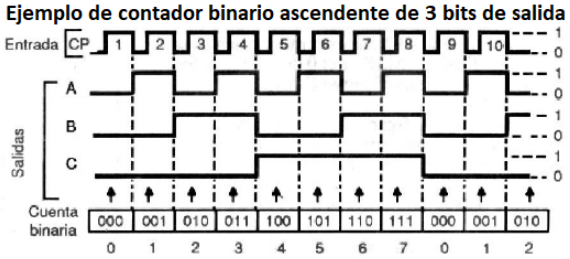
El acoplamiento adecuado de estos dispositivos permite obtener subsistemas más complejos, como los **registros** de desplazamiento o los **contadores**.

El esquema de un biestable tiene una salida Q que toma como valor la entrada D en cada pulso de reloj (CLK). Mientras la señal de entrada no cambie, la salida tampoco lo hace.

Un **bloque secuencial** es un circuito secuencial con una determinada funcionalidad. Algunos de los más importantes son:

- **Registro:** es un bloque formado por *n* biestables que **permite guardar el valor de *n* bits.**
- **Contadores:** circuito secuencial en el que sus salidas **siguen una secuencia de activación fija** que, cuando termina, vuelve a empezar.

C	B	A	Valor
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7



9.3.1. BIESTABLE RS (Set Reset) ASÍNCRONO.

Se compone internamente de dos puertas lógicas NAND o NOR. Es asíncrono porque **no lleva una entrada de sincronismo** (que va alterando 0 y 1).

En el estado de memoria (q) conserva el valor que tenga en ese momento, es decir, puede conservar 2 valores distintos (1 o 0) de ahí el nombre de biestable.

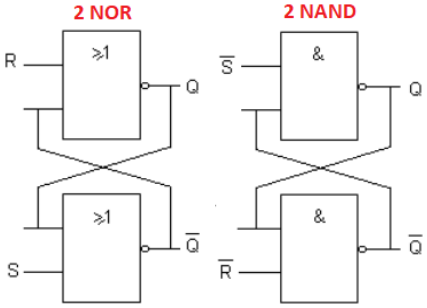


Tabla de verdad biestable RS

R	S	Q (NOR)	Q (NAND)
0	0	q	N. D.
0	1	1	0
1	0	0	1
1	1	N. D.	q

N. D.= Estado no deseado
q= Estado de memoria