Tema 9.- Lógica de circuitos. Circuitos combinacionales y secuenciales.

Índice

1. Introducción. 3

2. Lógica de circuitos. 3

2.1. Álgebra de Boole. 3
2.2. Teoremas de Boole. 5
2.3. Leyes de Morgan. 5
2.4. Definiciones sobre el Álgebra de Boole. 6
2.4.1. Expresiones algebraicas. 7
2.4.2. Tablas de verdad. 8
2.4.3. Expresiones algebraicas y tablas de verdad. 9
2.5. Otras funciones comunes. 10
2.6. Simplificación de funciones booleanas. 12
2.6.1. Simplificación de expresiones. 12
2.6.2. Método de Karnaugh. 14

3. Sistemas digitales. 19

3.1. Puertas lógicas. Síntesis y análisis. 19

4. Circuitos combinacionales. 22

4.1. Sumadores. 23
4.2. Codificadores y decodificadores. 23
4.3. Multiplexores y demultiplexores. 25
4.4. Comparadores. 28
4.5. Bloques AND, OR y NOT. 29

5. Circuitos secuenciales. 29

5.1. Biestables. 31
5.1.1. Biestables R-S con puertas NOR. 31
5.1.2. Biestables R-S con puertas NAND. 32
5.1.3. Biestables R-S con entrada de habilitación. 33
5.1.4. Biestables R-S disparados por flanco. 33
5.1.5. Biestables D. 34
5.1.6. Biestables J-K. 35
5.1.7. Biestables T. 36
5.1.8. Biestables Maestro/esclavo. 37
5.2. Registros de desplazamiento. 39
5.3. Contadores. 41
5.3.1. Contadores asíncronos. 41
5.3.2. Contador binario ascendente. 41
5.3.3. Contador binario descendente. 41
5.4. Autómatas programables. 42
5.5. Modelo de Moore. 43
5.5.1. Estado. Transiciones. 43
5.5.2. Representación gráfica. 45
6. Conclusión. 47
7. Bibliografía. 48

uni10.es

1. Introducción.

En el siglo XIX, el matemático **George Boole** desarrolló unas reglas de cálculo para aplicar a la lógica deductiva. En los años 30, **Claude Shanon** estableció la relación entre el álgebra de Boole y los circuitos de conmutación. Consiguió elaborar circuitos que ejecutaban operaciones de Boole elementales, y demostró que la combinación de circuitos podía representar operaciones aritméticas y lógicas complejas. Desde entonces, se ha utilizado el álgebra de Boole para la elaboración de circuitos digitales.

Como veremos más adelante, los circuitos digitales pueden ser **combinacionales**, en los que los valores de salida están determinados por los valores de entrada, o **secuenciales**, en los que los valores de salida están determinados por los valores de entrada y por los valores previos, almacenados en memoria. En este tema veremos con todo detalle la lógica de Boole y su aplicación a los circuitos lógicos, partiendo desde las puertas lógicas simples hasta los circuitos combinacionales y secuenciales.

Conoceremos a fondo los dispositivos electrónicos básicos. Los dispositivos electrónicos más elementales son las puertas lógicas y los bloques lógicos, que forman los circuitos lógicos. Estos últimos se pueden ver como un conjunto de dispositivos que manipulan de una manera determinada las señales electrónicas que les llegan (las señales de entrada), y generan como resultado otro conjunto de señales (las señales de salida).

2. Lógica de circuitos.

2.1. Álgebra de Boole.

El **Álgebra de Boole** o álgebra booleana es un sistema algebraico cerrado formado por un conjunto K de dos o más elementos (en lógica de circuitos, K = {0,1}) en el que se han definido dos operaciones binarias (la suma lógica o OR y el producto lógico o AND) y una operación unitaria (la negación o NOT).

Así pues, los dos elementos de un álgebra de Boole se pueden llamar falso y cierto o, más usualmente, 0 y 1. De este modo, una variable booleana o variable lógica puede tomar los valores 0 y 1.

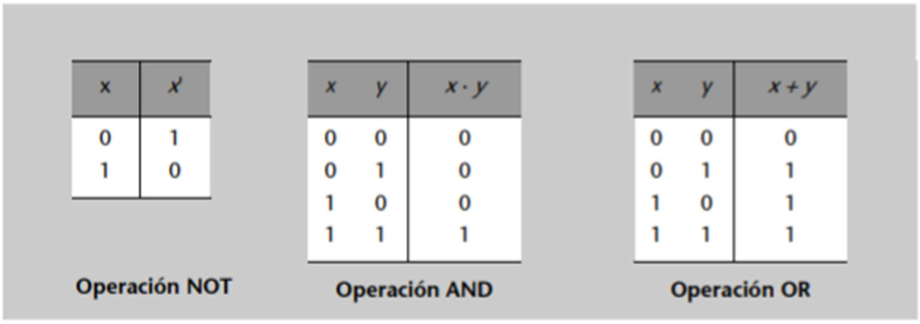
Las operaciones booleanas básicas son las siguientes:

- La negación o complementación o NOT, que corresponde a la partícula no y se representa con una comilla simple (’). Así, la expresión x’ denota la negación de la variable x, y se lee “no x”.

- El producto lógico o AND, que corresponde a la conjunción y de la lógica y se representa con el símbolo “·”. Así, si x e y son variables lógicas, la expresión $x \cdot y$ denota su producto lógico y se lee “x e y”.

- La suma lógica u OR, que corresponde a la conjunción o y se representa con el símbolo “+”. Así, la expresión “x + y” denota la suma lógica de las variables x e y, y se lee “x o y”.

Estas operaciones booleanas básicas se pueden definir escribiendo el resultado que dan para cada posible combinación de valores de las variables de entrada, tal como se muestra en la siguiente figura. En las tablas de esta figura aparecen a la izquierda de la línea vertical todas las combinaciones posibles de las variables de entrada, y a la derecha, el resultado de la operación para cada combinación.



x	x′
0	1
1	0

Operación NOT

x	y	x · y
0	0	0
0	1	0
1	0	0
1	1	1

Operación AND

x	y	x + y
0	0	0
0	1	1
1	0	1
1	1	1

Operación OR

Los axiomas que describen el comportamiento de las operaciones booleanas son los siguientes:

Nombre	Definición	Lógica
Cerrado	El conjunto K es cerrado para las dos operaciones AND y OR	∀ a, b ∈ K: • a + b ∈ K • a · b ∈ K
Conmutativa	La suma o producto de sus valores no altera el resultado	∀ a, b ∈ K: • a + b = b + a • a · b = b · a
Asociativa	Todo elemento de K tiene que cumplir la propiedad asociativa	∀ a, b, c ∈ K • (a + b) + c = a + (b + c) • (a · b) c = a (b · c)
Distributiva	Cada una de las operaciones es distributiva en relación con la otra.	∀ a, b, c ∈ K • a · (b + c) = a · b + a · c • a + (b · c) = (a + b) · (a + c)
Identidad	Existe un elemento identidad para las dos operaciones. Para la suma es el 0 y para el producto el 1	∀ a ∈ K • a + 0 = 0 + a = a • a · 1 = 1 · a = a
Complementario	Todo elemento de K tiene su complementario que verifica	∀ a ∈ K, ∃ā • a + ā = 1 ∀ a ∈ K, ∃ā • a · ā = 0

2.2. Teoremas de Boole.

Boole estableció una serie de teoremas para completar su álgebra, relacionadas con la interacción con el resto de elementos de su sistema algebraico.

Como notación, en las expresiones algebraicas utilizamos los paréntesis de la misma forma que estamos acostumbrados a hacerlo en aritmética entera; por ejemplo, la expresión x + y · z es lo mismo que la x + (y · z) y es diferente que (x + y) · z. Con todo esto, los teoremas de Boole son los siguientes:

Nombre	Definición	Lógica
Teorema de la dualidad	Dada una expresión booleana podemos pasar a su dual simplemente intercambiando entre sí el (+) por (·) y el (0) por el (1)	∀ a ϵ K: • a + 0 = a · 1
Teorema de la idempotencia	Dada una expresión booleana, la suma de su elemento consigo mismo o su multiplicación consigo mismo da como resultado el mismo elemento	∀ a ϵ K: • a + a = a • a · a = a
Teorema de los elementos dominantes	Cualquier elemento multiplicado por 0 dará como resultado 0, y sumado a 1 dará como resultado 1	∀ a ∈ K • a · 0 = 0 • a + 1 = 1
Teorema de Unicidad	Para cada elemento perteneciente a K, su complementario es único	∀ a ∈ K ∃ único ā • a + ā = 1 • a · ā = 0 • Si a + b = 1 → b = ā • Si a · b = 0 → b = ā
Teorema de absorción	Dada una suma o una multiplicación con dos valores, el segundo valor quedará absorbido por el primero.	∀ a, b ∈ K • a + a · b = a • a · (a + b) = a
Teorema de la involución	Cualquier cosa doblemente negada es esa misma cosa	∀ a ∈ K • ¬¬a = a

2.3. Leyes de Morgan.

Las **leyes de Morgan** son dos propiedades lógicas que sirven para romper las negaciones que afectan a varios términos y nos permiten relacionar la operación lógica de la suma con el producto, teniendo en cuenta el conjunto binario K = {0,1}:

$$\overline{(a+b)} = \overline{a} \cdot \overline{b}$$

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

2.4. Definiciones sobre el Álgebra de Boole.

Además de los teoremas y leyes, debemos tener claros algunos conceptos que son necesarios para desarrollar la lógica de circuitos. Estos conceptos son:

- Algebra de Boole de conmutación: Cumple con todas las propiedades del álgebra de Boole, pero cambiando los operadores de nombre para adecuarse a los circuitos lógicos. Así, la suma (+) se sustituye por OR, la multiplicación (·) se sustituye por AND y el complemento se sustituye por NOT.
- Variable booleana: Un símbolo x es una variable booleana si representa un elemento de un álgebra de Boole, es decir, puede tomar solo dos valores.
- Función booleana: Es una correspondencia unívoca entre dos conjuntos booleanos, de manera que cuando las variables tomen valores, la función tenga un único valor de salida.

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

- Término canónico: El término de una función booleana es todo producto o suma en el que aparecen todas las variables o sus complementos. Ejemplo:

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

- Forma canónica: Es una función expresada como suma de productos (forma canónica disyuntiva) o producto de sumas (forma canónica conjuntiva).

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

  - Maxterms: Son los términos canónicos que se expresan como producto de variables. Determinan la forma canónica conjuntiva, solo aparecen los términos en los que la función vale '0'.
  - Minterms: Son los términos canónicos que se expresan como suma de variables. Determinan la forma canónica disyuntiva, solo aparecen los términos en los que la función vale '1'.

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

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

Forma canónica disyuntiva f(a,b,c) = m1 + m3 = ∑ m(0,3)
Forma canónica conjuntiva f(a,b,c)=M0 · M2 · M4 · M5 · M6 · M7 = ∏ M(0,2,4,5,6,7)

2.4.1. Expresiones algebraicas.

Las expresiones algebraicas están formadas por variables lógicas, los elementos 0 y 1, los operadores producto (·), suma (+) y negación (’), y los símbolos (,) y =. Mediante estos elementos se puede expresar cualquier función lógica. Por ejemplo:

$$g_4(x,y) = x' \cdot y$$

La expresión vale 1 (es cierta) sólo si valen 1 (son ciertas) las subexpresiones x’ e y simultáneamente. La expresión x’ vale 1 sólo si x vale 0.

Una misma función lógica se puede expresar mediante infinitas expresiones algebraicas equivalentes. Para saber si dos expresiones algebraicas son equivalentes (es decir, expresan la misma función) podemos analizar si una se puede derivar de la otra usando los axiomas y leyes del álgebra de Boole. Por ejemplo, la primera ley de De Morgan nos dice que las funciones lógicas f y g son la misma:

$$f(x, y) = (x + y)',$$
$$g(x, y) = x' \cdot y'.$$

A la inversa, a partir de la expresión algebraica de una función podemos encontrar otras expresiones equivalentes aplicándole los axiomas y leyes del álgebra de Boole. Por ejemplo, si tenemos la función siguiente:

$$f(x, y, z) = x \cdot y' \cdot z + x' \cdot y \cdot z + x \cdot y \cdot z,$$

y aplicamos la propiedad distributiva, obtenemos la siguiente expresión equivalente:

$$f(x, y, z) = x \cdot y' \cdot z + (x' + x) \cdot y \cdot z.$$

Si aplicamos el complemento y después el teorema de los elementos neutros, obtenemos lo siguiente:

$$f(x, y, z) = x \cdot y' \cdot z + 1 \cdot y \cdot z = x \cdot y' \cdot z + y \cdot z.$$

Por la propiedad distributiva…

$$f(x, y, z) = (x \cdot y' + y) \cdot z.$$

Todas ellas sin expresiones equivalentes.

Al igual que cuando trabajamos con ecuaciones, podemos omitir el signo "·" en los productos lógicos para simplificar la escritura de las expresiones algebraicas: si escribimos seguidos los nombres de diferentes variables, sobreentenderemos que el producto lógico se obtiene de éstas. Así, las dos expresiones siguientes son igualmente válidas:

$$x_1' \cdot x_0 + x_2 \cdot x_1' \cdot x_0', \text{ o bien}$$
$$x_1' x_0 + x_2 x_1' x_0'.$$

2.4.2. Tablas de verdad.

Una tabla de verdad expresa una función lógica especificando el valor que tiene la función para cada posible combinación de valores de las variables de entrada.

Concretamente, a la izquierda de la tabla hay una lista con todas las combinaciones de valores posibles de las variables de entrada y, a la derecha, el valor de la función para cada una de las combinaciones.

Si una función tiene n variables de entrada, y dado que una variable puede tomar sólo dos valores, 0 ó 1, las entradas pueden adoptar 2n combinaciones de valores diferentes. Por tanto, su tabla de verdad tendrá a la izquierda n columnas (una para cada variable) y 2n filas (una para cada combinación posible). A la derecha habrá una columna con los valores de la función.

Las filas se escriben siempre en orden lexicográfico, es decir, si interpretamos las diferentes combinaciones como números naturales, escribiremos primero la combinación correspondiente al 0 (formada sólo por ceros), después la correspondiente al 1 y así sucesivamente, en orden creciente hasta la correspondiente al 2n – 1 (formada sólo por unos). En la siguiente figura muestra la estructura de las tablas de verdad para funciones de una, dos y tres variables de entrada.

1 variable

a	f
0	
1	

2 variables

a	b	f
0	0	
0	1	
1	0	
1	1	

3 variables

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

Es bastante usual denominar las variables de una función con una misma letra y diferentes subíndices; por ejemplo, x2, x1, x0. Por convención, pondremos en la columna situada más a la izquierda la variable de subíndice más alto y, en la que se encuentra más a la derecha, la de subíndice más bajo.

Diremos que la variable que ponemos en la columna más a la izquierda en una tabla de verdad es la variable de más peso, y la que ponemos en la columna más a la derecha es la de menos peso. Una vez fijado el orden de las columnas y las filas, la tabla de verdad de una función es única.

Podemos expresar el comportamiento de diferentes funciones que tienen las mismas variables de entrada en una única tabla de verdad. En este caso, a la derecha de la línea vertical habrá tantas columnas como funciones. No estableceremos ninguna convención para ordenar las columnas correspondientes a las funciones (se pueden poner en cualquier orden). A continuación se muestra un ejemplo.

x_2	y_1	x_0	f_0	f_1	f_2
0	0	0	1	0	0
0	0	1	1	0	1
0	1	0	0	1	1
0	1	1	0	0	0
1	0	0	0	1	1
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	0	1	1

2.4.3. Expresiones algebraicas y tablas de verdad.

Es importante saber pasar con agilidad de una expresión algebraica de una función a su tabla de verdad, y viceversa. Para obtener la tabla de verdad a partir de una expresión algebraica podemos actuar de dos maneras:

a) Evaluar la expresión para cada combinación de las variables y escribir en el lugar correspondiente de la tabla el resultado de la evaluación.

Por ejemplo, con la función siguiente:

$$f(a, b, c) = a + a'\,b'\,c + ac'.$$

Si evaluamos la expresión para el caso de la combinación $[a\ b\ c] = [0\ 0\ 0]$, obtenemos el siguiente resultado:

$$0 + 1 \cdot 1 \cdot 0 + 0 \cdot 1 = 0 + 0 + 0 = 0.$$

Por tanto, en la primera fila de la tabla irá un 0.

Para la combinación $[a\ b\ c] = [0\ 0\ 1]$ obtenemos el resultado que presentamos a continuación:

$$0 + 1 \cdot 1 \cdot 1 + 0 \cdot 0 = 0 + 1 + 0 = 1.$$

Por tanto, en la segunda fila irá un 1. Y así sucesivamente hasta haber evaluado la función para todas las combinaciones.

b) Analizar la expresión a vista, deducir para qué valores de las variables de la función vale 1 ó 0, y rellenar la tabla.

Tomamos la misma función del ejemplo anterior. Vemos que siempre que $a = 1$, la función vale 1 (por la ley 4, de dominancia). Por tanto, podemos poner un 1 en todas las filas en las que $a = 1$ (las cuatro últimas).

A continuación estudiamos los casos en que $a = 0$ (los que nos faltan por rellenar). La expresión de la función nos queda de la siguiente manera:

$$f(0, b, c) = 0 + 0' \cdot b' \cdot c + 0 \cdot c' = b' \cdot c.$$

Esta expresión vale 1 sólo en el caso $[b\ c] = [0\ 1]$. Por tanto, pondremos un 1 en la fila correspondiente a la combinación [0 0 1].

Para las combinaciones que aún nos quedan por rellenar, la función vale 0.

Observamos que el paso de expresión a tabla de verdad nos puede ser muy útil a la hora de determinar si dos expresiones algebraicas son equivalentes o no. Podemos obtener la tabla de verdad a partir de cada una de las expresiones, de modo que si las tablas resultantes son iguales, podemos concluir que las dos expresiones corresponden a una misma función.

2.5. Otras funciones comunes.

De entre las funciones de dos variables, existen otras que también reciben un nombre propio, ya que se utilizan de manera común. Son las siguientes:

Función O-exclusiva o XOR

Esta función vale 1 cuando alguna de las dos variables de entrada vale 1, pero no cuando valen 1 las dos a la vez; por eso recibe el nombre de O-exclusiva. Es la función g6 de la tabla anterior.

Se representa con símbolo “+”, y presenta esta tabla de verdad:

a	b	$a \oplus b$
0	0	0
0	1	1
1	0	1
1	1	0

$a \oplus b = a'b + ab'$

La función XOR de más de dos variables se calcula de este modo:

$$a \oplus b \oplus c \oplus d = (((a \oplus b) \oplus c) \oplus d).$$

Por tanto, da 0 si un número par de las variables vale 1, y da 1 si un número impar de las variables vale 1. Fijémonos que en el caso de dos variables, este hecho se traduce en lo siguiente:

$$a \oplus b = 0 \Rightarrow a \text{ y } b \text{ tienen el mismo valor,}$$
$$a \oplus b = 1 \Rightarrow a \text{ y } b \text{ tienen valores diferentes.}$$

Por tanto, la función XOR nos permite saber si dos variables son iguales o no lo son.

Otras propiedades de la XOR:

$$0 \oplus a = a,$$
$$1 \oplus a = a'.$$

Función NAND

Es la negación de la AND, es decir:

$$a \text{ NAND } b = (a \cdot b)'.$$

Por tanto, vale 1 siempre que no se cumpla que las dos variables de entrada valgan 1. Ésta es su tabla de verdad:

a	b	$(a+b)'$
0	0	1
0	1	1
1	0	1
1	1	0

Función NOR

Es la negación de la OR, es decir:

$$a \text{ NOR } b = (a + b)'.$$

Por tanto, vale 1 sólo cuando ninguna de las dos variables de entrada vale 1. Ésta es su tabla de verdad:

a	*b*	(*a*+*b*)'
0	0	1
0	1	0
1	0	0
1	1	0

2.6. Simplificación de funciones booleanas.

Ya hemos visto que uno de los objetivos que se tiene que conseguir cuando construimos un circuito es que sea tan rápido como sea posible. La síntesis a dos niveles nos permite conseguir de manera sistemática un grado de rapidez aceptable

También es deseable que un circuito sea pequeño y barato. Estas dos características están relacionadas con el número de puertas del circuito: cuantas menos haya, más pequeño y barato será.

Se puede **simplificar** una función aplicando todas las leyes, propiedades y teoremas que hemos visto hasta ahora. También se puede aplicar una herramienta llamada mapas de Karnaugh que se utiliza para la simplificación de funciones de hasta seis variables. A continuación se verán ambos métodos.

2.6.1. Simplificación de expresiones.

Se ha visto cómo se puede obtener la expresión de una función en suma de mintérminos. A veces, estas expresiones se pueden simplificar, lo que nos permitirá implementar la función con un circuito de menores dimensiones. En concreto, las simplificaciones que nos serán útiles corresponden a los casos en que dos o más mintérminos se pueden reducir a un único término producto, en el que aparecerán menos variables. Por ejemplo, sea la siguiente función:

$$f(x, y, z) = x'yz' + x'yz + \ldots$$

Estos dos mintérminos nos dicen que la función vale 1 si x = 0, y = 1 y z = 0 o bien si x = 0, y = 1 y z = 1. Sin embargo, esta información se puede resumir diciendo que la función vale 1 siempre que x = 0 e y = 1, independientemente del valor de z. Esto se puede expresar algebraicamente sacando el factor común en los dos mintérminos (es decir, aplicando las propiedades distributiva, de complementación y de elemento neutro):

$$f(x,y,z) = x'yz' + x'yz + \ldots =$$
$$= x'y(z'+z) + \ldots = x'y \cdot 1 + \ldots = x'y + \ldots$$

Tomamos ahora una función que vale 1 para las siguientes combinaciones [x y z] (entre otras):

[1 0 0]
[1 0 1]
[1 1 0]
[1 1 1]

Esta función vale 1 siempre que x = 1, independientemente del valor de y y z. Aplicando el mismo razonamiento que antes, lo podemos expresar algebraicamente de este modo:

$$f(x,y,z) = xy'z' + xy'z + xyz' + xyz + \ldots =$$
$$= xy'(z'+z) + xy\,(z'+z) + \ldots =$$
$$= xy' + xy + \ldots = x\,(y'+y) + \ldots = x + \ldots$$

Así pues, vemos que podemos obtener un solo término producto en los siguientes casos:

1. Si en dos mintérminos todas las variables se mantienen constantes menos una, entonces podemos sacar factor común eliminando la variable que cambia. En el término producto resultante aparecerán n – 1 variables, siendo n el número de variables de la función.

2. Si en cuatro mintérminos todas las variables se mantienen constantes menos dos, entonces podemos sacar factor común dos veces y eliminar las dos variables que cambian. En el término producto resultante aparecerán n – 2 variables.

3. En general, si en 2m mintérminos todas las variables se mantienen constantes menos m, entonces podemos sacar factor común m veces y eliminar las m variables que cambian. En el término producto resultante aparecerán n – m variables.

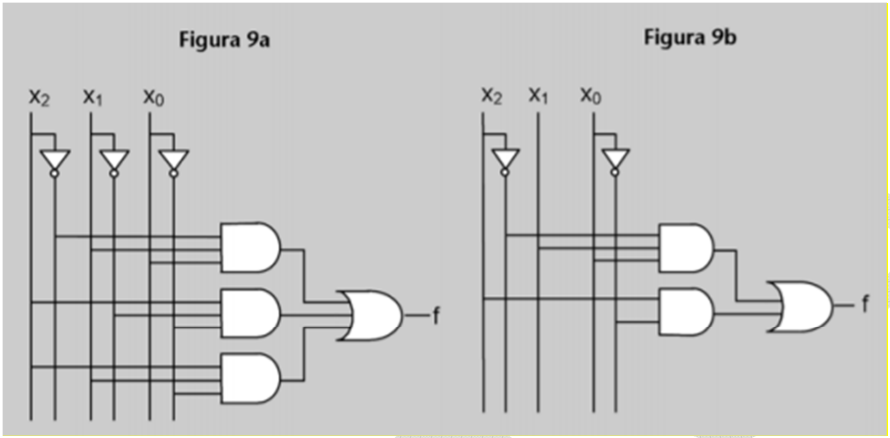
El hecho de obtener la expresión en suma de productos más simplificada posible de una función, sacando factor común en los casos que se acaban de describir, se denomina minimizar la función.

A partir de la expresión minimizada de una función, podremos construir circuitos de menores dimensiones y más baratos que los que obteníamos de la expresión en suma de mintérminos: quizá se necesitarán menos puertas NOT, algunas de las puertas AND serán de menos entradas, tendrán menos puertas AND y la puerta OR será de menos entradas.

En la siguiente figura se muestra los circuitos correspondientes a la expresión en suma de mintérminos (figura 9a) y a la expresión minimizada (figura 9b) de esta función.

$$f(x_2,x_1,x_0) = x_2'x_1x_0 + x_2x_1'x_0' + x_2x_1x_0' =$$
$$= x_2'x_1x_0 + x_2x_0' (x_1'+x_1) = x_2'x_1x_0 + x_2x_0'.$$

Se puede observar que el circuito de la figura 9a tiene tres puertas NOT, tres AND de tres entradas y una OR de tres entradas. El circuito simplificado, en cambio, sólo tiene dos puertas NOT, dos AND (una de dos entradas y la otra de tres) y una puerta OR de dos entradas.



2.6.2. Método de Karnaugh.

Podemos **simplificar** una función aplicando todas las leyes, propiedades y teoremas que hemos visto hasta ahora. No obstante es un proceso muy tedioso. Existe una herramienta llamada mapas de Karnaugh que se utiliza para la simplificación de funciones de hasta seis variables.

Utilizaremos una matriz de celdas donde representaremos todas las combinaciones posibles de las variables de una función, de manera que una vez representado, agrupamos los valores allí representados en potencias de dos. El número de casillas de un mapa de Karnaugh es 2^n, siendo n el número de variables. La función debe estar en forma canónica para poder ser representada.

El método de Karnaugh proporciona una mecánica sencilla para detectar visualmente los casos en que se puede minimizar una expresión en suma de mintérminos. Por tanto, entre todos los circuitos a dos niveles que implementan una función, permite obtener fácilmente el menor de todos.

El método de Karnaugh consta de cuatro pasos:

1) Construcción del mapa de Karnaugh: Trasladar la tabla de verdad de la función a una estructura que se llama mapa de Karnaugh.

2) Detectar visualmente los casos en que se puede sacar factor común.

3) Deducir los términos producto más simples posible.

4) Obtener la expresión mínima de la función haciendo la suma lógica de los términos producto.

Veamos detalladamente cómo se lleva a cabo cada uno de estos procesos.

1) Construcción del mapa de Karnaugh.

Rellenar las **tablas de valores** no es un proceso trivial. Se han de seguir una serie de normas para hacerlo correctamente. La numeración de las celdas se realiza pues, utilizando la codificación Gray, que consiste en tan solo cambiar un valor de una variable por cada celda adyacente. Se dice que dos casillas del mapa son adyacentes si corresponden a combinaciones en las que sólo cambia el valor de una variable.

El mapa de Karnaugh es una transcripción de la tabla de verdad de una función a una estructura formada por casillas en la que cada una de estas corresponde a una combinación de las variables (y, por tanto, a una fila de la tabla de verdad).

La siguiente figura muestra la estructura del mapa de Karnaugh para funciones de 2, 3 y 4 variables . Por ejemplo, la casilla que está sombreada con gris claro corresponde a la combinación [a b c d] = [1 1 0 1].

Estructuras de los mapas de Karnaugh para funciones de

2 variables

b \ a	0	1
0		
1		

3 variables

c \ a b	00	01	11	10
0				
1				

4 variables

c d \ a b	00	01	11	10
00				
01				
11				
10				

Fijémonos en que en las cabeceras de las filas y las columnas de los mapas de Karnaugh las combinaciones no están en orden lexicográfico.

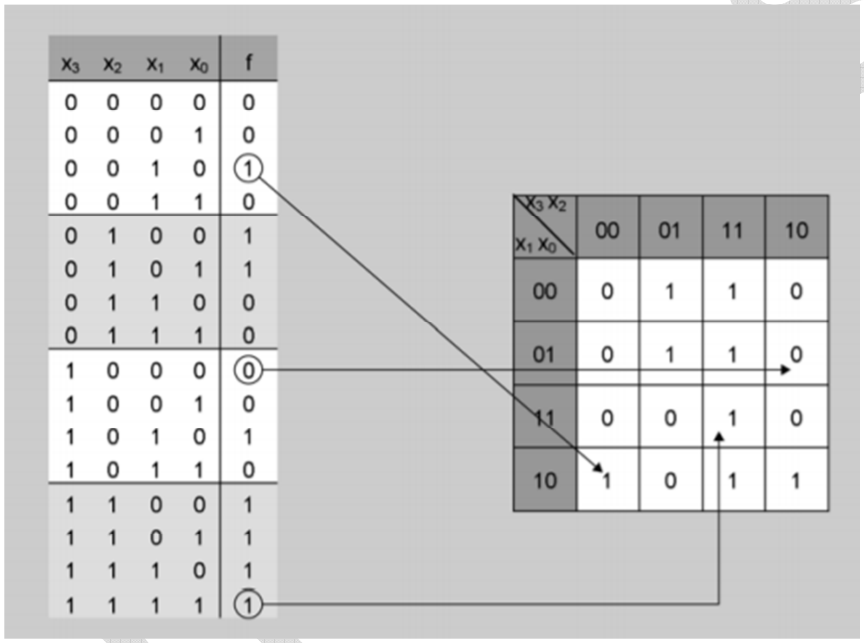
De esta manera se cumple que las casillas adyacentes quedan dispuestas en el mapa de la siguiente manera:

1. Dos casillas donde únicamente cambia el valor de una variable son adyacentes.

2. En los mapas de tres y cuatro variables, las casillas de la columna más a la derecha también son adyacentes con las de la columna más a la izquierda.

3. En los mapas de cuatro variables, las casillas de la fila superior también son adyacentes con las de la fila inferior.

Una vez dibujado el mapa, pondremos dentro de cada casilla el valor de la función para la combinación correspondiente de variables, a partir de la tabla de verdad. En la siguiente figura se puede ver un ejemplo de ello donde se muestra explícitamente la posición de algún mintérmino en la tabla.



x_3	x_2	x_1	x_0	f
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

$x_1 x_0$ \ $x_3 x_2$	00	01	11	10
00	0	1	1	0
01	0	1	1	0
11	0	0	1	0
10	1	0	1	1

De hecho, es suficiente con rellenar las casillas para las que la función vale 1.

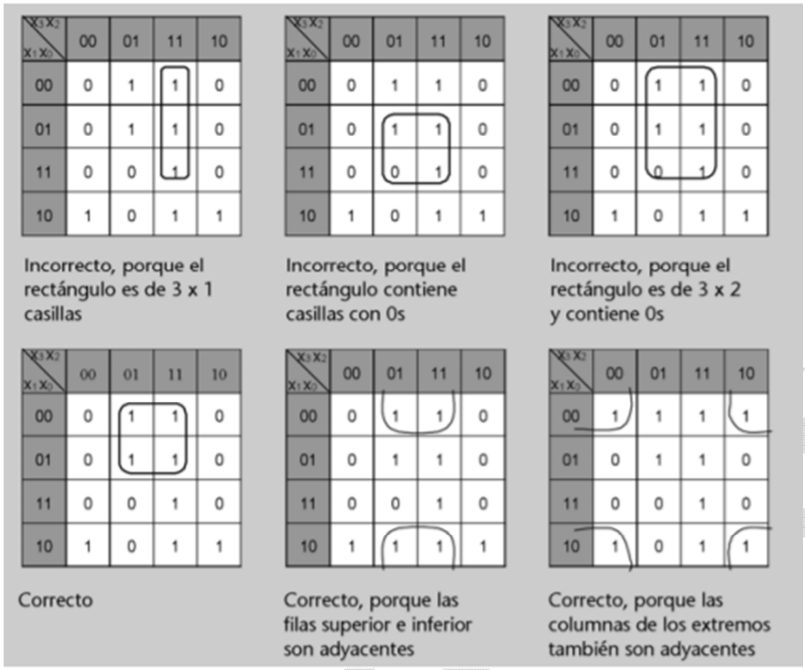
2) Detección de los casos en que se puede sacar factor común.

Supongamos que dos casillas adyacentes contienen unos; corresponden, pues, a dos mintérminos de la función. Sin embargo, entre éstas sólo cambia el valor de una variable (por la definición de adyacencia) y, por tanto, corresponden a un caso en el que se puede sacar factor común de los dos mintérminos.

Supongamos ahora que cuatro casillas adyacentes contienen unos. Entre éstas sólo cambia el valor de dos variables y, por tanto, corresponden a un caso en el que se puede sacar factor común dos veces.

Así, el segundo paso del método Karnaugh consiste en agrupar con rectángulos los unos que estén en casillas adyacentes, formando grupos de 1, 2, 4, 8 ó 16 unos. Los lados de estos rectángulos deben ser de un número de

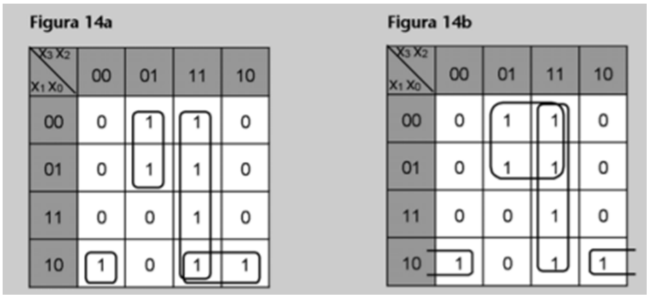
casillas potencia de 2, y en su interior sólo puede haber unos. En la siguiente figura se muestran varios ejemplos de rectángulos correctos e incorrectos.



La manera de agrupar los unos de un mapa no es única. Al hacer las agrupaciones, se debe tener en cuenta lo siguiente:

- Todos los unos deben formar parte de algún grupo.
- Los grupos tienen que ser lo más grandes posible (para que en cada término aparezca el menor número posible de variables). Por este motivo, empezaremos buscando los mayores grupos que podamos hacer (recordando que dentro de un grupo sólo puede haber unos).
- Cuantos menos grupos haya, mejor (para obtener el menor número posible de términos producto). El hecho de buscar los grupos más grandes reduce el número total de grupos que se harán.
- Un mismo 1 puede formar parte de más de un grupo si eso ayuda a satisfacer los dos objetivos anteriores. Si después de hacer los grupos grandes que veíamos a primera vista, queda algún 1 disperso, vemos si lo podemos agrupar con otros unos que ya formen parte de algún grupo.

En la siguiente figura se muestran dos formas de agrupar los unos del mapa del ejemplo anterior. La de la figura 14a no es incorrecta, pero la de la figura 14b es mejor. Siempre se debe procurar encontrar la agrupación óptima.



3) Deducción de los términos producto.

Tomamos el mapa de la figura 14b. El grupo de los cuatro unos de la tercera columna corresponde a las combinaciones [x3 x2 x1 x0] =

[1 1 0 0]
[1 1 0 1]
[1 1 1 0]
[1 1 1 1]

La expresión en suma de mintérminos que se obtendría de estos cuatro unos es x3x2x1'x0' + x3x2x1'x0 + x3x2x1x0' + x3x2x1x0. Si sacamos factor común, obtenemos x3x2, porque el valor de la función es independiente de x1 y x0. Si observamos el mapa, podemos ver que las variables x3 y x2 no cambian de valor en el rectángulo, mientras que x1 y x0 adoptan todas las combinaciones posibles.

Así pues, obtendremos un término producto de cada grupo de la siguiente manera:

1. Sólo aparecen las variables cuyo valor es constante para todas las casillas que forman el grupo.
2. Si en todas las casillas del grupo una variable vale 1, la variable aparece en el término producto sin negar.
3. Si en todas las casillas del grupo una variable vale 0, ésta aparece negada en el término producto.

Así, en el mapa de la figura 14b, del otro grupo de cuatro unos obtenemos el término.

$$x_2 x_1'.$$

Del grupo de dos unos obtenemos el término.

$$x_2' x_1 x_0'.$$

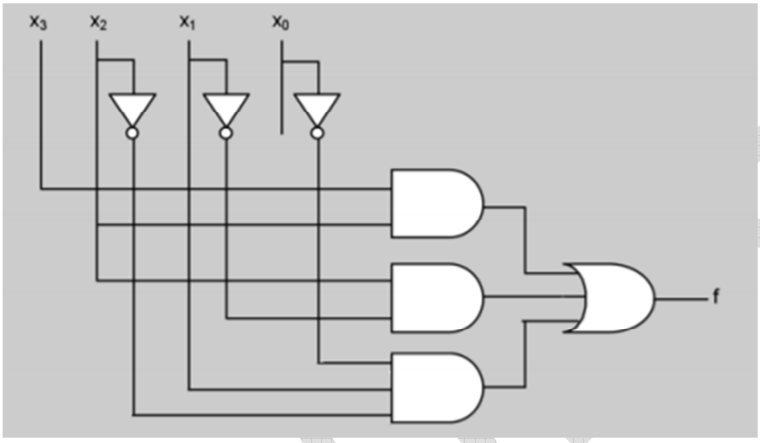
4) Obtención de la expresión mínima de la función.

Ya sabemos que una función se puede expresar haciendo la suma lógica de todos sus mintérminos. Los términos producto obtenidos en el paso anterior "resumen" los mintérminos de una función. Por tanto, podemos expresar la función realizando la suma lógica de estos términos producto.

En el ejemplo de la figura 14b, la expresión minimizada de la función es la siguiente:

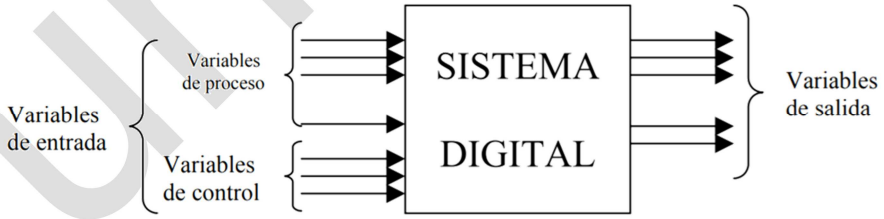
$$f(x_3,x_2,x_1,x_0) = x_3x_2 + x_2x_1' + x_2'x_1x_0'.$$

En la siguiente figura se muestra el circuito a dos niveles que implementa esta función a partir de la expresión minimizada. Se puede ver que es menor y más barato que el que obtendríamos a partir de la expresión en suma de mintérminos original, ya que éste tendría cuatro puertas NOT, ocho AND de tres entradas cada una y una OR de ocho entradas. De todos los circuitos a dos niveles que implementan esta función, el de la figura 15 es el de dimensiones más reducidas y el más barato; se dice que es el circuito mínimo a dos niveles.



3. Sistemas digitales.

Un sistema digital es un conjunto de elementos binarios relacionados entre sí de alguna manera. Los sistemas digitales reciben variables de entrada y producen variables de salida. Un circuito lógico es un sistema digital construido con una serie de puertas lógicas que trabajan con señales binarias.



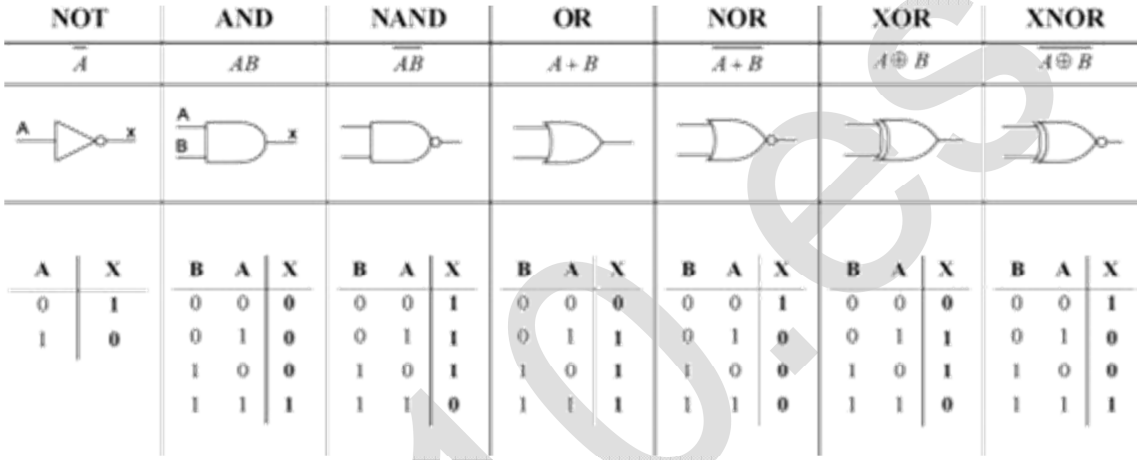
3.1. Puertas lógicas. Síntesis y análisis.

Una **puerta lógica** es un pequeño sistema electrónico digital que realiza una función booleana simple. Las operaciones de álgebra de Boole se implementan mediante puertas lógicas.

Hemos visto que se pueden construir dispositivos electrónicos diferentes para funciones de una y dos variables de entrada. Ahora bien, en la práctica sólo se construyen los que calculan las funciones NOT, AND, OR, XOR, NAND y NOR.

Los dispositivos electrónicos que calculan funciones lógicas se llaman puertas lógicas. Son dispositivos que están conectados a un cierto número de señales de entrada y a una señal de salida.

A continuación presentamos el símbolo gráfico con el que se representa cada puerta lógica. En todas las figuras que aparecen a continuación, las señales de entrada quedan a la izquierda de las puertas y la señal de salida queda a la derecha.



A	X
0	1
1	0

B	A	X
0	0	0
0	1	0
1	0	0
1	1	1

B	A	X
0	0	1
0	1	1
1	0	1
1	1	0

B	A	X
0	0	0
0	1	1
1	0	1
1	1	1

B	A	X
0	0	1
0	1	0
1	0	0
1	1	0

B	A	X
0	0	0
0	1	1
1	0	1
1	1	0

B	A	X
0	0	1
0	1	0
1	0	0
1	1	1

Puerta NOT o inversor.

El funcionamiento es el siguiente: si en la entrada e hay un 0 (tensión baja), entonces en la salida s habrá un 1 (tensión alta). Y a la inversa, si hay un 1 en el punto e, entonces habrá un 0 en el punto s. Por tanto, la puerta NOT implementa físicamente la función de negación: s = e'.

Puertas AND y NAND.

La puerta AND implementa la función lógica AND, es decir, la salida s1 vale 1 sólo si las dos entradas a y b valen 1. Por tanto, s1 = a · b.

La puerta NAND implementa la función lógica NAND. En el punto s2 hay un 1 siempre que en alguna de las dos entradas a o b haya un 0. Es decir, s2 = (a · b)'.

Puertas OR y NOR.

La puerta OR implementa la función lógica OR, es decir, en la salida s1 hay un 1 si cualquiera de las dos entradas está en 1. Por tanto, s1 = a + b.

La puerta NOR implementa la función lógica NOR. En el punto s2 encontraremos un 1 sólo cuando en las dos entradas haya un 0, es decir: s2 = (a + b)'.

Puerta XOR.
Implementa la función lógica XOR, es decir, la salida s vale 1 si alguna de las dos entradas vale 1, pero no si valen 1 las dos a la vez.

Todas las puertas (menos la NOT) pueden tener más de dos señales de entrada. Su funcionamiento es el siguiente:

- En una puerta AND de n entradas, la salida vale 1 sólo cuando todas las n entradas valen 1.

- En una puerta OR de n entradas, la salida vale 1 cuando una o más de las n entradas valen 1.

- En una puerta XOR de n entradas, la salida vale 1 cuando hay un número impar de entradas en 1. El 0 se considera un número par y, por tanto, si todas las entradas valen 0, la salida también vale 0.

Síntesis y análisis.

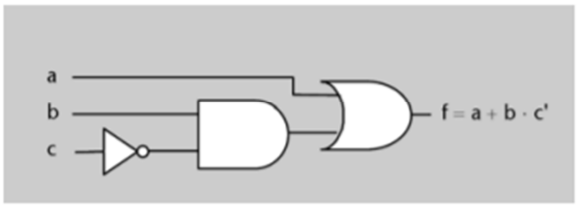
Cualquier función lógica se puede implementar usando estas puertas, es decir, se puede construir un circuito que se comporte como la función.

El proceso de obtener el circuito que implementa una función a partir de una expresión algebraica se denomina síntesis.

El proceso de obtener una expresión de una función a partir del circuito que la implementa se denomina análisis.

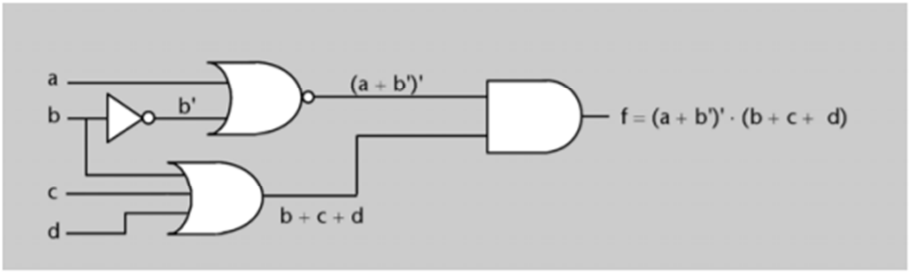
Para sintetizar (o implementar) una función a partir de su expresión algebraica, es suficiente con sustituir cada operador de la función por la puerta lógica adecuada. Por ejemplo, un término producto de tres variables se implementará con una puerta AND de tres entradas. Las puertas deben estar interconectadas entre sí y con las entradas tal como indica la expresión.

Por ejemplo, el circuito que implementa la función f(a, b, c) = a + bc' es el siguiente:



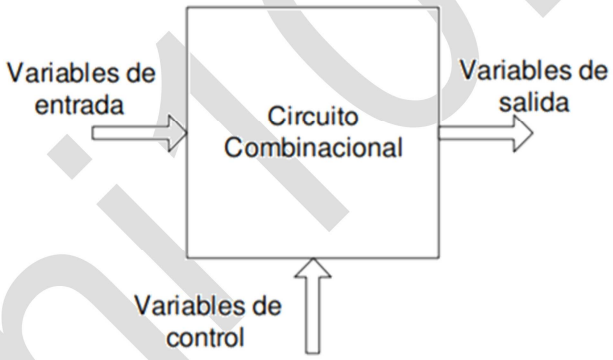
Para analizar un circuito, es necesario escribir las expresiones que corresponden a la salida de cada puerta, empezando desde las entradas del circuito hasta obtener la expresión correspondiente a la línea de salida del mismo.

Por ejemplo, el circuito siguiente implementa la función f(a, b, c, d) = (a + b')'(b + c + d).



4. Circuitos combinacionales.

Un **circuito combinacional** es un circuito lógico cuyos valores de salida están completamente determinados por sus valores de entrada, de forma que dependen de las entradas y del propio circuito. Para una misma combinación de valores en sus entradas, se obtendrán los mismos valores de salida. Para n variables hay 2^n combinaciones posibles, de las que para cada combinación de entrada solo hay una combinación de salida.



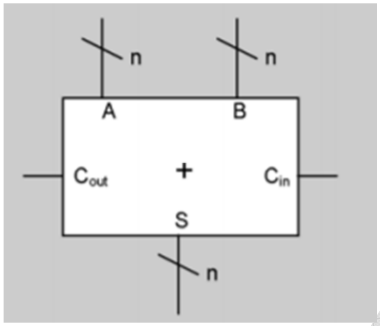
Los **elementos** que actúan en un circuito combinacional son:

- Variables de entrada: Son las entradas al circuito combinacional, y pueden estar en dos estados, activo e inactivo (1 y 0 lógicos).
- Variables de salida: Son las salidas del circuito combinacional con el resultado de la operación lógica que realiza el circuito sobre las entradas
- Variables de control: Modifican el funcionamiento de un circuito combinacional. Pueden ser de habilitación (si no están activas el circuito no realiza su función) o de selección (determinan el modo de operación de un circuito)

Para **diseñar** circuitos combinacionales, partimos de la definición de un problema lógico y tras obtener las funciones simplificadas, se crea el diagrama lógico con puertas lógicas. A continuación veremos los circuitos combinacionales más importantes.

4.1. Sumadores.

El sumador es un bloque combinacional que realiza la suma de dos números codificados en binario o bien en complemento a 2. En la siguiente figura se muestra la representación gráfica de un sumador de n bits. Las señales que tiene son las siguientes:



1. Dos entradas de datos de n bits, que llamaremos A y B, por donde llegarán los números que se tienen que sumar.

2. Una salida de n bits, llamada S, que tomará el valor de la suma de los números A y B.

3. Una salida de 1 bit, Cout, que valdrá 1 si cuando se haga la suma se produce acarreo en el bit de más peso.

4. Una entrada de 1 bit, Cin, por donde llega un acarreo de entrada.

4.2. Codificadores y decodificadores.

La función de un codificador es generar la codificación binaria de un número. Los codificadores tienen las siguientes señales:

1. Una entrada de validación, VAL, que funciona igual que en el caso de los multiplexores: si vale 0, todas las salidas valen 0 (cuando no dibujemos la entrada de validación, asumiremos que vale 1).

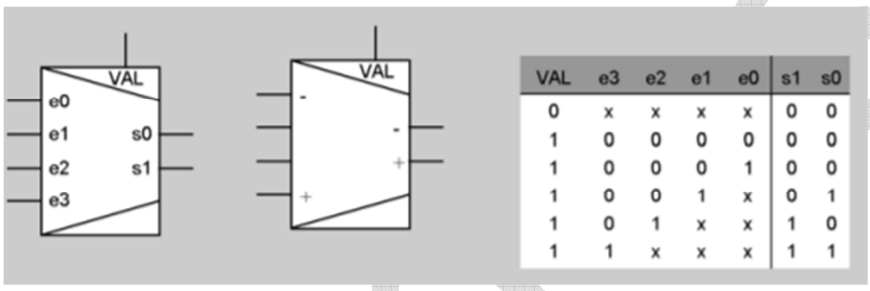
2. 2^m entradas de datos (de un bit), identificadas por la letra e y numeradas de 0 a $2^m - 1$ (la de número más alto es la de más peso).

3. m salidas de datos (de un bit), identificadas por la letra s y numeradas de 0 a m – 1, que se interpretan como si formaran un número codificado en binario con m bits (la salida de más peso corresponde al bit de mayor peso).

El funcionamiento de un codificador es el siguiente:

Cuando la entrada de validación vale 1, si la entrada de mayor peso de entre las que están en 1 es la numerada con el número i, entonces las salidas codifican en binario el número i. O, dicho de otro modo: para que un codificador genere la representación binaria del número i, es preciso que la entrada activa de más peso sea la numerada con el número i.

En la siguiente figura se muestra la representación gráfica de un codificador 4-2 y la tabla de verdad que explica su funcionamiento (observamos que, como en el caso de los multiplexores, podemos utilizar los símbolos "+" y "–" en lugar de los nombres de las entradas y salidas). Vemos que, por ejemplo, cuando e3 = 0 y e2 = 1, las salidas codifican el número 2, independientemente del valor de e1 y e0, porque la entrada de más peso que está activa es e2. Se dice que, cuanto más peso tiene una entrada, más prioridad tiene.



VAL	e3	e2	e1	e0	s1	s0
0	x	x	x	x	0	0
1	0	0	0	0	0	0
1	0	0	0	1	0	0
1	0	0	1	x	0	1
1	0	1	x	x	1	0
1	1	x	x	x	1	1

Los decodificadores cumplen la función inversa a los codificadores: dada una combinación binaria presente en la entrada, indican a qué número decimal corresponde.

Los decodificadores tienen las siguientes señales:

1. Una entrada de validación.

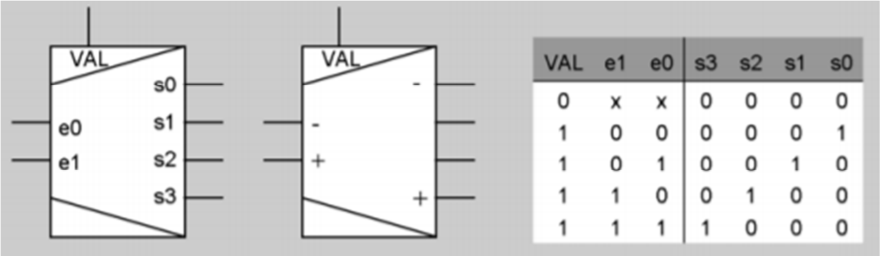
2. m entradas de datos, identificadas por la letra e y numeradas de 0 hasta m – 1, que se interpretan como si formaran un número codificado en binario (la entrada de más peso corresponde al bit de mayor peso).

3. 2^m salidas, identificadas por la letra s y numeradas de 0 a 2^m – 1, de las cuales sólo una vale 1 en cada momento (si la entrada de validación está en 0, entonces todas las salidas valen 0).

El funcionamiento de un decodificador es el siguiente:

Cuando la entrada de validación vale 1, si las entradas codifican en binario el número i, entonces se pone en 1 la salida numerada como i.

En la siguiente figura se muestra la representación gráfica de un descodificador 2-4 y la tabla de verdad que describe su funcionamiento.



VAL	e1	e0	s3	s2	s1	s0
0	x	x	0	0	0	0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0

4.3. Multiplexores y demultiplexores.

Imaginemos que en una ciudad hay tres calles que confluyen en otra calle de un solo carril. Será necesario un urbano o algún tipo de señalización para controlar que en cada momento circulen hacia la calle de salida los coches provenientes de una única calle confluente.

Un multiplexor es un bloque que cumple la función de guardia urbano en circuitos electrónicos. Tiene un determinado número de señales de entrada que "compiten" para conectarse a una señal única de salida, y unas señales de control que sirven para determinar qué señal de entrada se conecta en cada momento con la salida.

Más concretamente, las entradas y salida de un multiplexor son las siguientes:

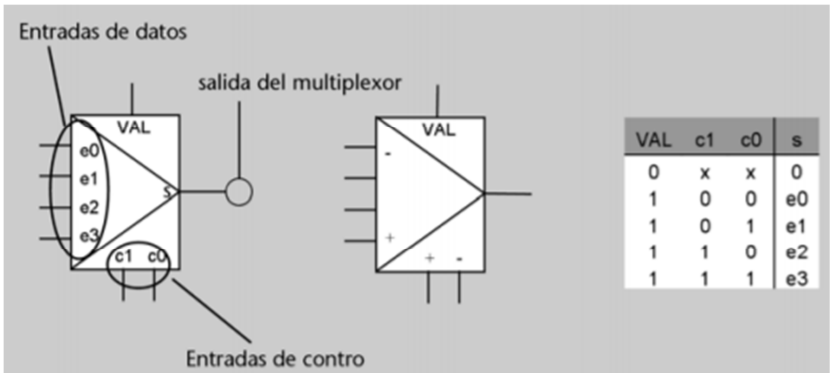
1. 2^m entradas de datos, identificadas por la letra e y numeradas desde 0 hasta $2^m - 1$. Diremos que la entrada de datos numerada con el 0 es la de menos peso y, la numerada con el $2^m - 1$, la de más peso.

2. Una salida de datos, s.

3. m entradas de control o de selección, identificadas por la letra c y numeradas desde 0 hasta m – 1. Diremos que la entrada de control numerada con el 0 es la de menos peso, y la numerada con m – 1, la de más peso.

4. Una entrada de validación, que denominamos VAL.

En la siguiente figura se muestran dos posibles representaciones gráficas de un multiplexor 4-1, es decir, con m = 2 (se puede ver el primer recuadro al margen). La diferencia entre éstas es que en la primera ponemos los nombres de las entradas, mientras que en la segunda sólo indicamos con los signos "+" y "–" cuáles son las de más y las de menos peso (se asume que el resto está ordenado en orden de peso). Las dos representaciones son igualmente válidas.



VAL	c1	c0	s
0	x	x	0
1	0	0	e0
1	0	1	e1
1	1	0	e2
1	1	1	e3

En general, no escribiremos el nombre de la salida (tal y como se hace en la segunda representación) porque la podemos identificar de manera inequívoca por el hecho de que está dibujada en el punto del multiplexor donde se juntan las dos líneas oblicuas interiores. También, en el caso de los multiplexores 2-1, que sólo tienen una entrada de control, podemos identificarla por la letra “c” (sin número) o bien no ponerle el nombre, ya que la podemos identificar inequívocamente por el hecho de que estará dibujada en un lado corto del multiplexor.

En la figura anterior también muestra la tabla de verdad que describe el funcionamiento del multiplexor (en este caso un multiplexor 4-1), que es el siguiente:

- Cuando la entrada de validación vale 0, la salida del multiplexor se pone en 0 independientemente del valor de las entradas de datos. En la tabla, esto lo reflejamos poniendo x en las columnas correspondientes a las entradas de datos, en la fila en la que VAL = 0.

- Cuando la entrada de validación está activa (vale 1), entonces las entradas de control determinan cuál de las entradas de datos se conecta con la salida de la forma siguiente: se interpretan las variables conectadas en las entradas de control (c1 y c0 en el ejemplo) como un número codificado en binario con m bits (la entrada de más peso corresponde al bit de más peso); si el número codificado es i, la entrada de datos que se conecta con la salida es la numerada con el número i.

Multiplexor de buses.

Una palabra de n bits es una agrupación de n bits, usualmente con un significado semántico conjunto (por ejemplo, un número codificado en binario con n bits). Un bus es una agrupación de un cierto número de cables, por cada uno de los cuales circula un bit. Así, una palabra de n bits circulará por un bus de n bits. Diremos que es su ancho o tamaño del bus. Los buses se representan gráficamente tal como se muestra en la figura 19a. En la línea que representa la señal ponemos una línea transversal acompañada de un número que indica la cantidad de bits del bus.

A veces interesa disgregar los bits que forman un bus o agregarle bits adicionales. Las figuras 19b, 19c y 19d muestran unos cuantos ejemplos de cómo lo representaremos gráficamente.

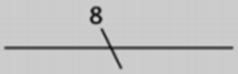


Figura 19a: bus de 8 bits

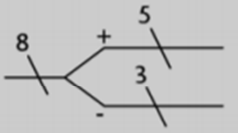


Figura 19b: el bus de 8 bits se disgrega en dos buses. uno formado por los 5 bits de mayor peso y otro formado por los 3 de menor peso

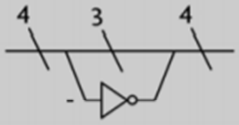


Figura 19c: la línea inferior corresponde al bit de menos peso del bus de 4 bits. Este bit se niega y después se agrega de nuevo al bus original

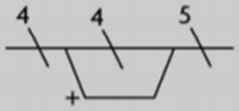
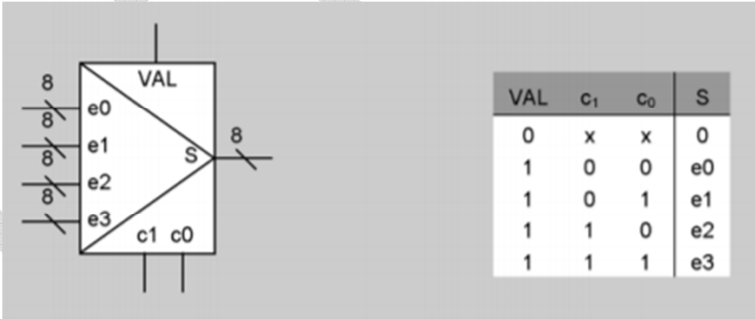


Figura 19d: la línea inferior corresponde a una replicación del bit de más peso del bus de 4 bits, porque el bus original continúa siendo de 4 bits. Este nuevo bit se agrega posteriormente al bus original, de modo que finalmente queda un bus de 5 bits

Un multiplexor de buses funciona igual que un multiplexor de bits, pero en este caso las entradas de datos y la salida son buses de un determinado número de bits. En la siguiente figura se muestra la representación gráfica de un multiplexor 4-1 de buses de ocho bits.



VAL	c_1	c_0	S
0	x	x	0
1	0	0	e0
1	0	1	e1
1	1	0	e2
1	1	1	e3

Demultiplexor.

Un demultiplexor hace la función inversa de un multiplexor: dada una señal de entrada, determina con qué señal de salida se debe conectar.

Los multiplexores tienen las señales siguientes:

1. Una entrada de datos de n bits, e (n puede ser igual a 1).

2. 2^m salidas de datos de n bits, identificadas por la letra s y numeradas desde 0 hasta $2^m - 1$. Diremos que la salida de datos numerada con el 0 es la de menos peso, y la numerada con el $2^m - 1$, la de más peso.

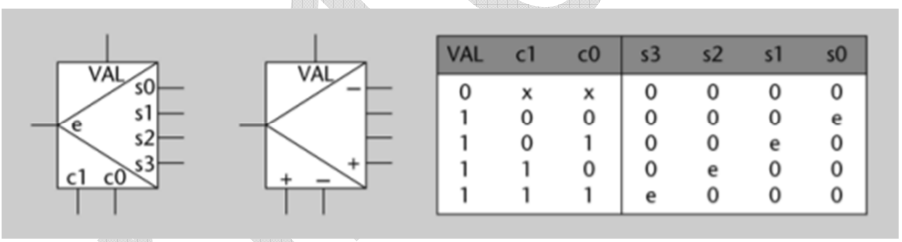
3. m entradas de control o de selección, de un bit, identificadas por la letra c y numeradas desde 0 hasta m – 1. Diremos que la entrada de control numerada con el 0 es la de menos peso, y la numerada con m – 1, la de más peso.

4. Una entrada de validación de un bit, que denominamos VAL.

El funcionamiento del demultiplexor es el siguiente:

- Cuando la entrada de validación vale 0, todas las salidas valen 0.
- Cuando la entrada de validación vale 1, entonces las entradas de control determinan cuál de las salidas de datos se conecta con la entrada, de la misma manera que en el caso del multiplexor (si el número que codifican las entradas de control es i, la entrada se conecta con la salida numerada con el número i). Las salidas de datos no seleccionados se ponen a 0.

La figura siguiente muestra las dos posibilidades para la representación gráfica de un demultiplexor 1-4 y la tabla que describe su funcionamiento. Al igual que en el caso de los multiplexores, podemos usar los signos "+" y "–" para indicar el peso de las entradas de control y de las salidas de datos. También, al igual que en el caso de los multiplexores, en general no escribiremos el nombre de la entrada e, y si no dibujamos la entrada VAL, asumiremos que está activa.



VAL	c1	c0	s3	s2	s1	s0
0	x	x	0	0	0	0
1	0	0	0	0	0	e
1	0	1	0	0	e	0
1	1	0	0	e	0	0
1	1	1	e	0	0	0

4.4. Comparadores.

El comparador es un bloque combinacional que compara dos números codificados en binario e indica qué relación existe entre éstos. Un comparador tiene las siguientes señales:

1. Dos entradas de datos de n bits, que reciben los nombres A y B. Se interpretan como números naturales codificados en binario.

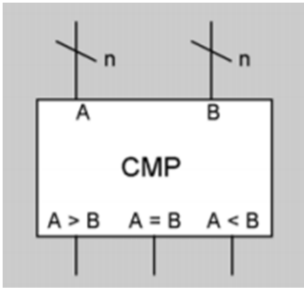
2. Tres salidas de 1 bit, de las que sólo una vale 1 en cada momento:

– La salida A > B vale 1 si el número que llega por la entrada A es mayor que el que llega por la entrada B.

– La salida A = B vale 1 si los dos números de entrada son iguales.

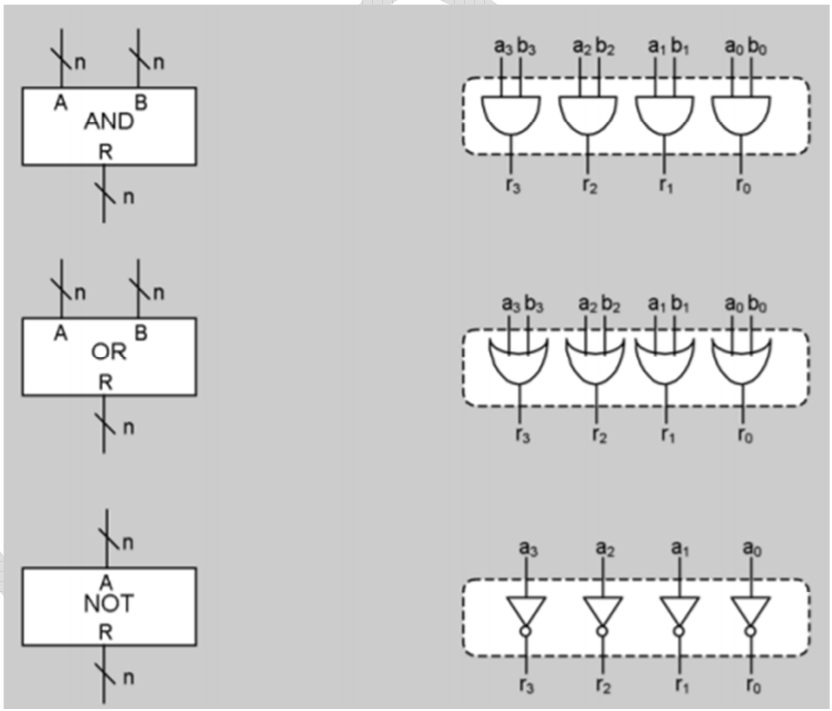
– La salida A < B vale 1 si el número que llega por la entrada A es menor que el que llega por la entrada B.

En la siguiente figura se muestra la representación gráfica de un comparador.



4.5. Bloques AND, OR y NOT.

Estos bloques hacen las operaciones AND, OR y NOT bit a bit sobre entradas de n bits. Tienen dos entradas de datos (sólo una en el caso del bloque NOT) y una salida, todas de n bits. En la siguiente figura se muestra su representación gráfica y la implementación interna para el caso n = 4.

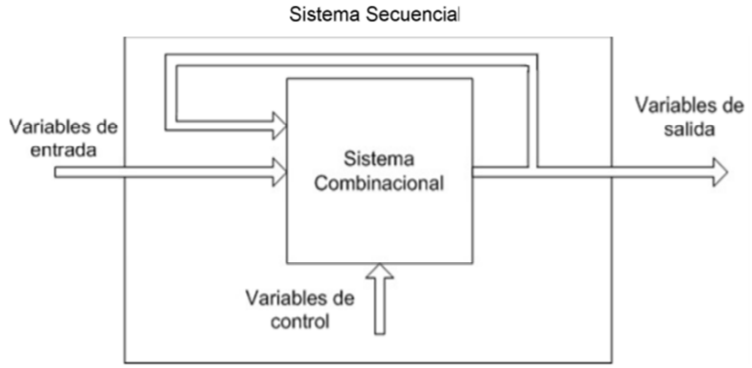


Podemos construir bloques análogos a estos, por ejemplo un bloque XOR sería aquel que hace la XOR bit a bit de las entradas.

5. Circuitos secuenciales.

Un **circuito secuencial** es un circuito lógico, en el que el estado de las variables de salida depende del estado de las variables de entrada y de los valores previos, se dice que el circuito secuencial tiene “memoria”. Un sistema

secuencial se puede construir a partir de un sistema combinacional añadiéndole realimentación. Los circuitos secuenciales se utilizan principalmente en la unidad de control y en la unidad de memoria de cualquier ordenador Gráficamente se puede representar:



Los sistemas secuenciales **pueden ser**:

- Asíncronos: el cambio de estado se puede producir en cualquier instante de tiempo, solo tiene en cuenta los cambios de estado de la entrada
- Síncronos: este cambio solo puede ocurrir en unos instantes de tiempo determinados por un reloj que emite una señal de sincronismo. Al ser más fiables y fáciles de construir, son los que utilizan los ordenadores actuales.

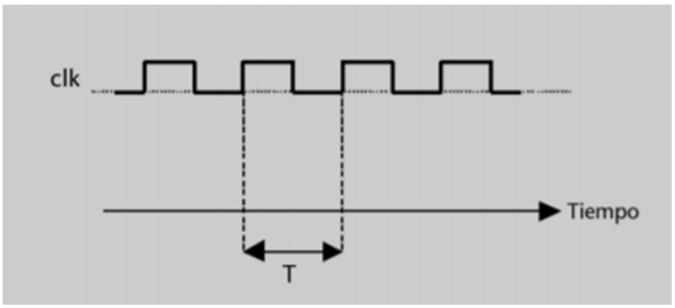
En los ordenadores los circuitos secuenciales que se construyen son síncronos, ya que son más fiables y fáciles de construir que los asíncronos.

En los circuitos combinacionales, la única noción temporal que interviene es el presente. En cambio, en los circuitos secuenciales se tiene en cuenta la evolución temporal de las señales (y aparece, como se verá más adelante, la noción de futuro). En los circuitos secuenciales que estudiaremos en este módulo se utiliza una señal de reloj como forma de sincronización.

El reloj es una señal que sirve para determinar los instantes en que un circuito secuencial “ve” el valor de las señales, o “es sensible”, y responde en consecuencia. Esta tarea que lleva a cabo la señal de reloj se llama sincronización de los circuitos.

Concretamente, la señal del reloj toma los valores 0 y 1 de forma cíclica y continua desde la puesta en marcha de un circuito hasta que éste se detiene. De forma habitual, se utiliza la noción clk para hacer referencia a la señal del reloj (deriva del inglés clock).

En la siguiente figura se muestra el cronograma de la señal de reloj. El ciclo que forma la secuencia de valores 0 y 1 tiene una duración determinada y constante, T, que se llama periodo. Se puede medir en segundos o, más habitualmente, en nanosegundos (mil millonésimas de segundo).



Los instantes en que la señal de reloj pasa de 0 a 1 se llaman flancos ascendentes. El intervalo que hay entre un flanco ascendente y el siguiente se llama ciclo o ciclo de reloj. Por tanto, la duración de un ciclo es un periodo de T segundos.

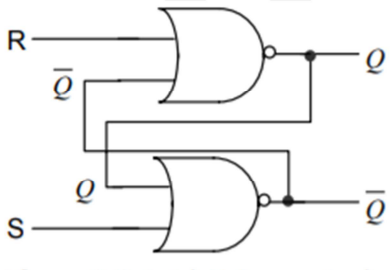
Los circuitos secuenciales más importantes son los biestables, los registros y los contadores.

5.1. Biestables.

Un biestable o latch es el elemento básico de la memoria que puede estar en dos estados que corresponden con el "0" y el "1" lógico. Después de adquirir un estado, es capaz de mantenerlo de manera indefinida.

5.1.1. Biestables R-S con puertas NOR.

El latch R-S (Reset-Set) con entrada activa a nivel alto es un tipo de dispositivo lógico biestable con dos salidas Q y ¬Q (una la complementaria de la otra), compuesto de dos puertas NOR acopladas tal y como muestra en la siguiente figura. Se puede observar que la salida de cada puerta NOR se conecta a la entrada de la puerta opuesta.



NOR

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

S	R	Q_n	$\overline{Q_n}$	
0	0	Q_{n-1}	$\overline{Q_{n-1}}$	Sin cambio
0	1	0	1	RESET
1	0	1	0	SET
1	1	0	0	Ambiguo

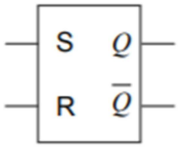
Q_{n-1} estado anterior
Q_n salida

El funcionamiento del latch R-S con entrada activa a nivel alto se deriva del comportamiento de la puerta NOR (ver tabla de verdad de la anterior figura). Si una de las entradas de una puerta NOR (de dos entradas) se mantiene a '0' la salida será la inversa de la otra entrada. En consecuencia si en el latch R y S son '0', la salida del circuito se mantendrá en el estado en el estuviera (ver tabla de verdad del latch en la siguiente figura). Si la entrada R del latch se

pone a ‘1’ mientras que la entrada S permanece a ‘0’, la salida Q se pondrá a ‘0’ sin importar su estado previo (en una puerta NOR en cuanto hay una entrada a ‘1’ la salida es ‘0‘) y a su vez la salida negada, Q se pondrá a ‘1‘, el latch pasará al estado de Reset. Si ahora R vuelve a ‘0’ el circuito entrará de nuevo en su modo de memoria. De manera similar si S se lleva a ‘1’ mientras R permanece a ‘0’, entonces la salida negada Q se pondrá a ‘0’, con lo que la salida Q se colocará a ‘1’. En resumen el funcionamiento del latch es el siguiente:

- La entrada R activa (‘1’) realiza un RESET del latch (pone la salida a ‘0’).
- La entrada S activa (‘1’) realiza un SET del latch (pone la salida a ‘1’).
- Si las entradas están desactivadas (R=0 y S=0) la salida del latch no cambia (Qn=Qn-1).
- Si se activan las dos entradas (R=1 y S=1) el circuito no funciona correctamente (Q=0 y Q =0).

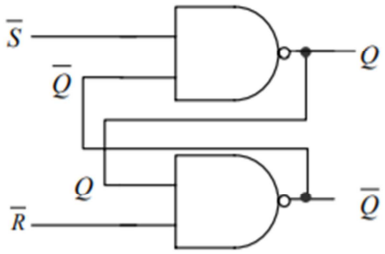
El símbolo lógico del latch R-S con entrada activa y su tabla de verdad se muestra en la siguiente figura.



R	S	Q
0	0	Q
0	1	1
1	0	0
1	1	X

5.1.2. Biestables R-S con puertas NAND.

El latch R-S (Reset-Set) con entrada activa a nivel bajo es un tipo de dispositivo lógico biestable compuesto de dos puertas NAND acopladas tal y como muestra la siguiente figura.



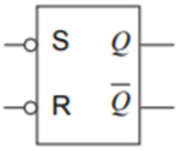
NAND

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

$\overline{S}$	$\overline{R}$	Q_n	$\overline{Q_n}$	
0	0	1	1	Ambiguo
0	1	1	0	SET
1	0	0	1	RESET
1	1	Q_{n-1}	$\overline{Q_{n-1}}$	Sin cambio

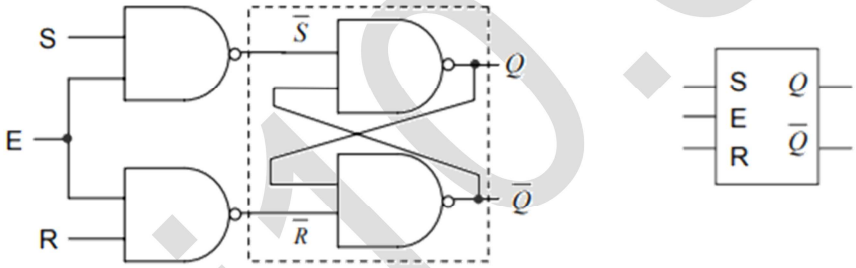
Al comparar el funcionamiento de una puerta NAND con otra NOR podemos ver que si bien una puerta NOR se asemeja a un inversor cuando una de sus entradas está conectada a ‘0’, la puerta NAND se asemeja a un inversor cuando una de sus entradas está conectada a ‘1’ (ver tabla de verdad de una puerta NAND en la figura anterior). Por tanto el modo de memoria del latch (Qn=Qn-1) corresponde en este caso con las dos entradas a ‘1’. Si la entrada S se lleva a nivel bajo ‘0’ la salida Q se pone a ‘1’ (SET) y si la entrada R se lleva a nivel bajo ‘0’ la salida Q será ‘0’ (RESET). De ahí que a este latch se le dé el nombre de latch con entrada activa a nivel bajo.

En la Figura anterior podemos apreciar la tabla de verdad correspondiente a este lacth y la siguiente figura su símbolo lógico.



5.1.3. Biestables R-S con entrada de habilitación.

A menudo resulta de utilidad poder controlar el funcionamiento del latch de manera que las entradas se puedan activar en unos instantes determinados. El diagrama y el símbolo lógico de un latch con entrada de habilitación se muestra en la siguiente figura. Las entradas S y R controlan el estado al que va a cambiar el latch cuando se aplica un '1' en la entrada de habitación (E, enable). El latch no cambiará de estado hasta que la entrada E esté a nivel alto. Esta tercera entrada (E) permite habilitar o inhibir las acciones del resto de entradas.



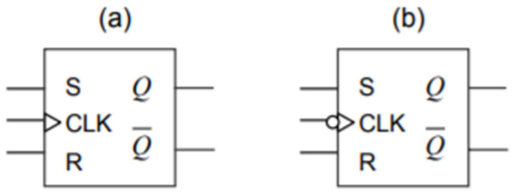
E	S	R	Q_n	$\overline{Q_n}$	
1	0	0	Q_{n-1}	$\overline{Q_{n-1}}$	Sin cambio
1	0	1	0	1	RESET
1	1	0	1	0	SET
0	X	X	Q_{n-1}	$\overline{Q_{n-1}}$	Sin cambio

Cuando la señal de habilitación E esté a nivel bajo, las señales S y R estarán a nivel alto sin importar el valor de las entradas R y S. Esto coloca al biestable en su modo de memoria, evitando que la salida cambie de estado. Cuando se activa la entrada de habilitación, las señales R y S se invierten y se aplican al latch S - R, es decir, el circuito actúa como un latch R-S con entrada activa a nivel alto. La tabla de verdad se puede apreciar en la figura anterior.

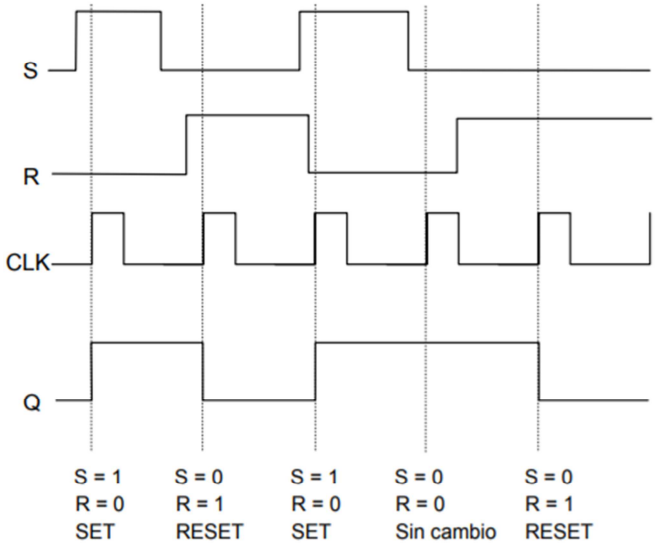
5.1.4. Biestables R-S disparados por flanco.

Se asemeja al latch R-S excepto en que el circuito sólo responde a sus entradas en el flanco ascendente o descendente de la señal de reloj. Los símbolos gráficos (la siguiente figura) se asemejan a los de los latches con

entrada de habilitación, excepto en que esta última entrada se reemplaza por una entrada de reloj.



En ausencia de la transición de reloj el flip-flop permanece en su modo de memoria, como se aprecia en el diagrama de la siguiente figura, correspondiente a un flip-flop disparado con flanco de subida.

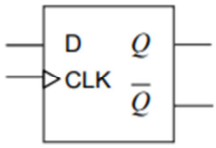


S	R	CLK	Q_n	$\overline{Q_n}$	
0	0	↑	Q_{n-1}	$\overline{Q_{n-1}}$	Sin cambio
0	1	↑	0	1	RESET
1	0	↑	1	0	SET
1	1	↑	0	0	Ambiguo
X	X	X	Q_{n-1}	$\overline{Q_{n-1}}$	Sin cambio

El funcionamiento de un flip-flop R-S activado por flanco descendente es, por supuesto, idéntico, excepto que el disparo tiene lugar en el flanco de bajada de la señal de reloj (cuando cambia de '1' a '0').

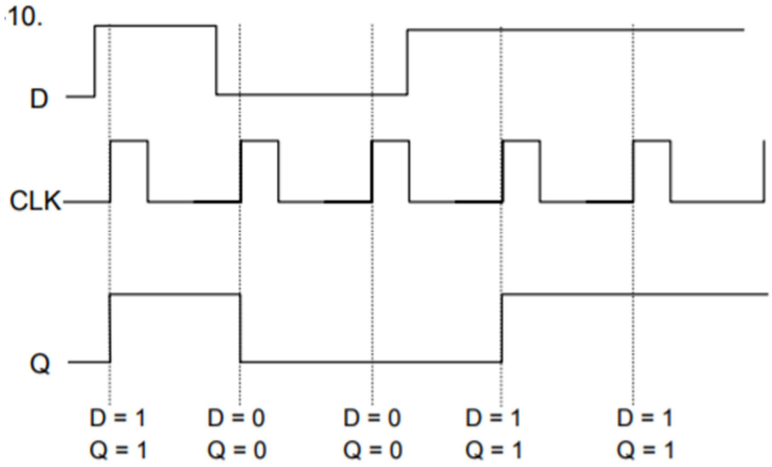
5.1.5. Biestables D.

Tiene una entrada de datos y una entrada de reloj. Si hay un impulso de reloj, la salida tendrá el valor de la entrada D. Este biestable es capaz de almacenar un bit de información un tiempo igual al ciclo de reloj. Su comportamiento se basa en que la salida del flipflop tipo D se igualará a la entrada en el instante en el que se produzca el flanco ascendente o descendente (según el tipo de flip-flop) de la señal de reloj (CLK). En la siguiente figura se observa el símbolo lógico y la tabla de verdad de un flip-flop tipo D disparado por flanco ascendente.



D	CLK	Q_n	$\overline{Q_n}$	
0	↑	0	1	RESET
1	↑	1	0	SET
X	X	Q_{n-1}	$\overline{Q_{n-1}}$	Sin cambio

El funcionamiento de un flip-flop D disparado por flanco ascendente se resume en la siguiente figura.



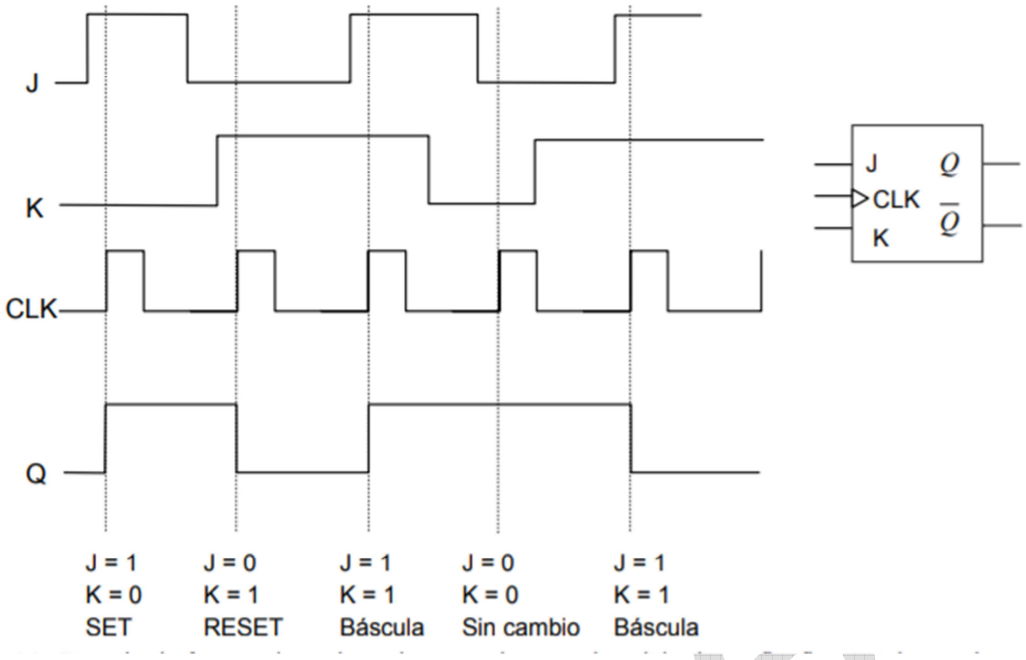
5.1.6. Biestables J-K.

El flip-flop J-K se comporta como el flip-flop R-S a excepción de que resuelve el problema de tener una salida indeterminada cuando las entradas se encuentran activas a la vez. La entrada J es la equivalente a la entrada S de un flip-flop R-S y la entrada K, al equivalente a la entrada R. En este dispositivo cuando las dos entradas se colocan a nivel alto la salida cambia al estado opuesto al que se encontraba. A este modo de funcionamiento se le denomina modo de basculación.

La tabla de transición muestra las características de un flip-flop J-K disparado por flanco ascendente.

J	K	CLK	Q_n	$\overline{Q_n}$	
0	0	↑	Q_{n-1}	$\overline{Q_{n-1}}$	Sin cambio
0	1	↑	0	1	RESET
1	0	↑	1	0	SET
1	1	↑	$\overline{Q_{n-1}}$	Q_{n-1}	Báscula

La siguiente figura ilustra el funcionamiento del flip-flop J-K disparado por flanco ascendente y su símbolo lógico.

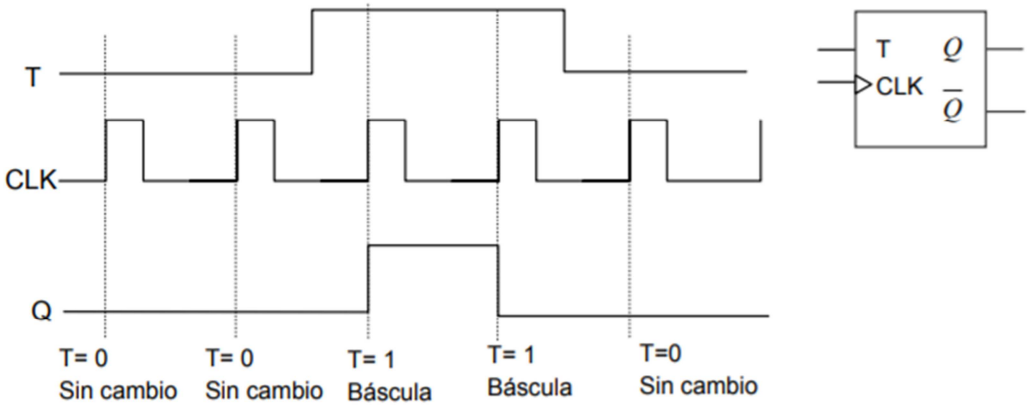


El flip-flop J-K es uno de los flip-flops más ampliamente utilizados.

5.1.7. Biestables T.

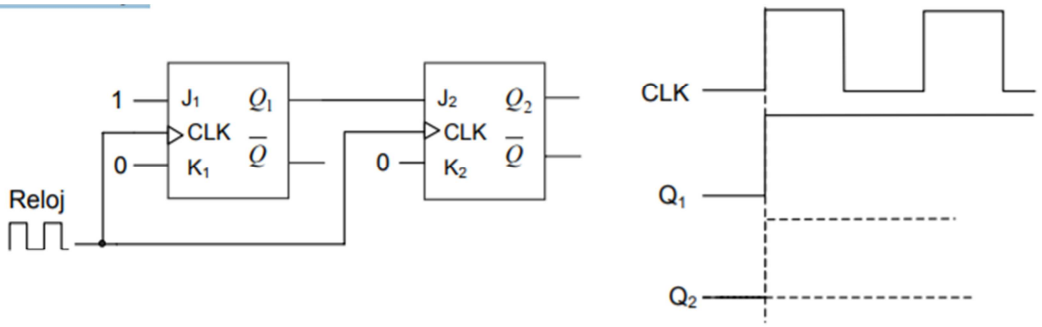
Existe otro tipo de flip-flop con una única entrada (T). El comportamiento de un flip-flop tipo T es equivalente al de un flip-flop tipo J-K con sus entradas J y K unidas. De este modo, si la entrada T presenta un nivel bajo '0' el dispositivo está en su modo de memoria, y si al entrada T se encuentra a nivel alto '1' el dispositivo cambia de estado, es decir la salida bascula. En la siguiente figura se aprecia este comportamiento y el símbolo lógico.

T	CLK	Q_n	$\overline{Q_n}$	
0	↑	Q_{n-1}	$\overline{Q_{n-1}}$	Sin cambio
1	↑	$\overline{Q_{n-1}}$	Q_{n-1}	Báscula



5.1.8. Biestables Maestro/esclavo.

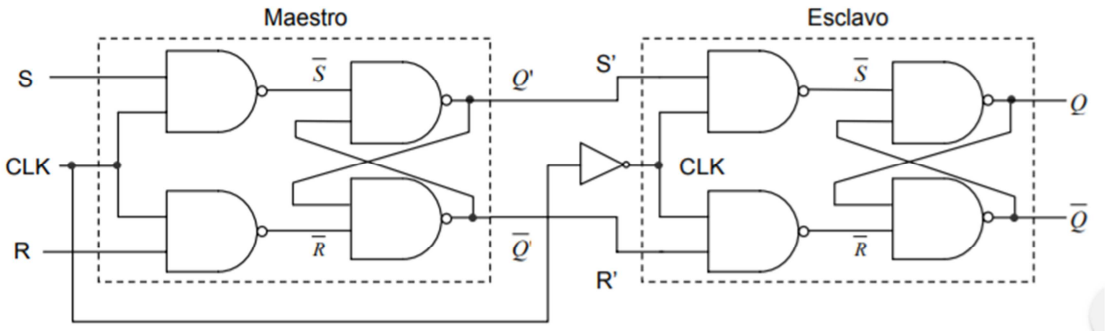
En muchos sistemas digitales es necesario sincronizar el funcionamiento de un gran número de circuitos con una sola señal de reloj. En la siguiente figura se muestra un ejemplo en el que la salida de un flip-flop se une a la entrada de otro y se sincronizan ambos con la misma señal de reloj.



Para analizar el comportamiento del circuito supondremos que inicialmente la salida de los dos flip-flops están a '0'. Si aplicamos al primer flip-flop un nivel alto en la entrada J y un nivel bajo en la entrada K y al segundo flip-flop la salida de primero y un nivel bajo en su entrada K observamos en las formas de onda que el funcionamiento del circuito es incierto. Si el segundo flip-flop responde con rapidez a la señal de reloj, quizá responda antes de que cambie el primer dispositivo, en este caso verá un '0' a la entrada y la salida no cambiará. Sin embargo, si la respuesta del segundo flip-flop es lenta, el primer dispositivo habrá tenido tiempo de cambiar y el segundo flip-flop verá un '1' en su entrada, con lo que pondrá a '1' su salida.

Este problema se puede resolver mediante el uso de flip-flops maestro/esclavo, que no son más que biestables conectados en serie. Este tipo de flip-flop lo podemos encontrar para los modelos R-S, D y J-K. A continuación se describe el funcionamiento de un flipo-flop R-S maestro/esclavo.

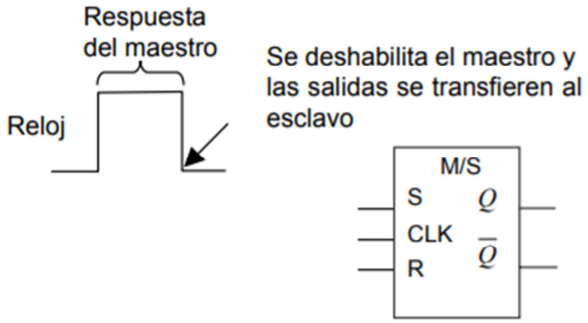
El flip-flop S-R maestro/esclavo básico se muestra en la siguiente figura. El circuito representa dos biestables S-R con entrada de habilitación conectados en serie, en los que la entrada de reloj se usa para habilitar cada uno.



Cuando la señal de entrada de reloj pasa a nivel alto, se habilita el maestro y se deshabilita el esclavo. El maestro se comporta como un latch con entrada de habilitación R-S , y el esclavo como no está habilitado continua en su estado

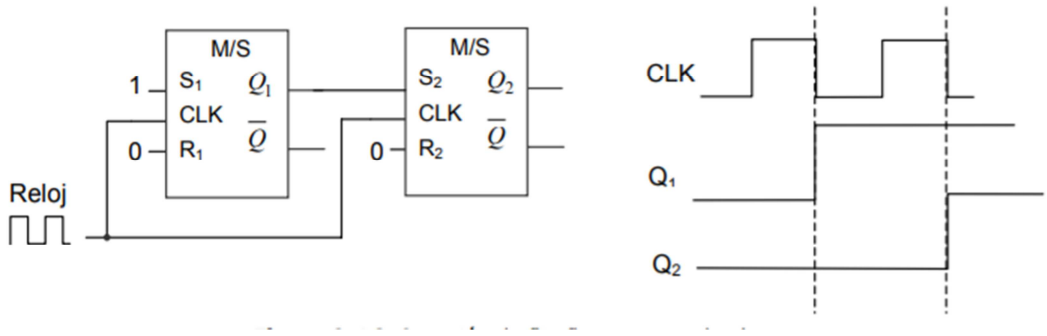
previo, las salidas no cambian. Cuando el reloj se hace ‘0’, el maestro se deshabilita y mantiene su estado previo. El esclavo está ahora habilitado y responde a sus entradas. Como la salida Q’ del maestro está conectada a la entrada S’ del esclavo y la salida Q' del maestro está conectada a la entrada R’ del esclavo, este siempre verá un ‘1’ en una entrada y un ‘0’ en la otra. Si la salida Q del maestro es ‘1’, el esclavo estará en el estado SET y si es ‘0’, estará en el estado RESET. Por tanto, cuando el esclavo está habilitado toma el estado de salida del maestro.

La tabla de verdad de un flip-flop R-S maestro esclavo es la misma que la de los flipflops R-S disparado por flanco, excepto en la manera en que se sincroniza con la señal de reloj. El dispositivo responde a sus entradas mientras el reloj está alto, pero las salidas no se actualizan hasta que el reloj se hace bajo (Figura 3-15), es decir, el dispositivo maestro/esclavo responde al final del pulso de entrada de reloj, en lugar de hacerlo en el flanco ascendente o descendente . La tabla de verdad y el símbolo lógico se muestran en la siguiente figura.



S	R	CLK	Q_n	$\overline{Q_n}$	
0	0	⊓	Q_{n-1}	$\overline{Q_{n-1}}$	Sin cambio
0	1	⊓	0	1	RESET
1	0	⊓	1	0	SET

Si volvemos a analizar el problema anterior en la conexión de dos biestables, sincronizados con la misma entrada de reloj, el circuito sería el de la siguiente figura.



Al igual que antes el primer flip-flop responde al pulso de reloj que entra, pero esta vez la salida cambia en el flanco descendente del pulso. La entrada S2 del segundo flip-flop es ‘0’ mientras el reloj está a nivel alto, y por lo tanto el circuito permanece en su estado de memoria. Después del primer pulso de reloj la salida Q1 se pone a nivel alto ‘1’ y con ella la entrada S2. Entonces cuando termine el segundo pulso de reloj la salida del segundo flip-flop se pondrá a nivel alto.

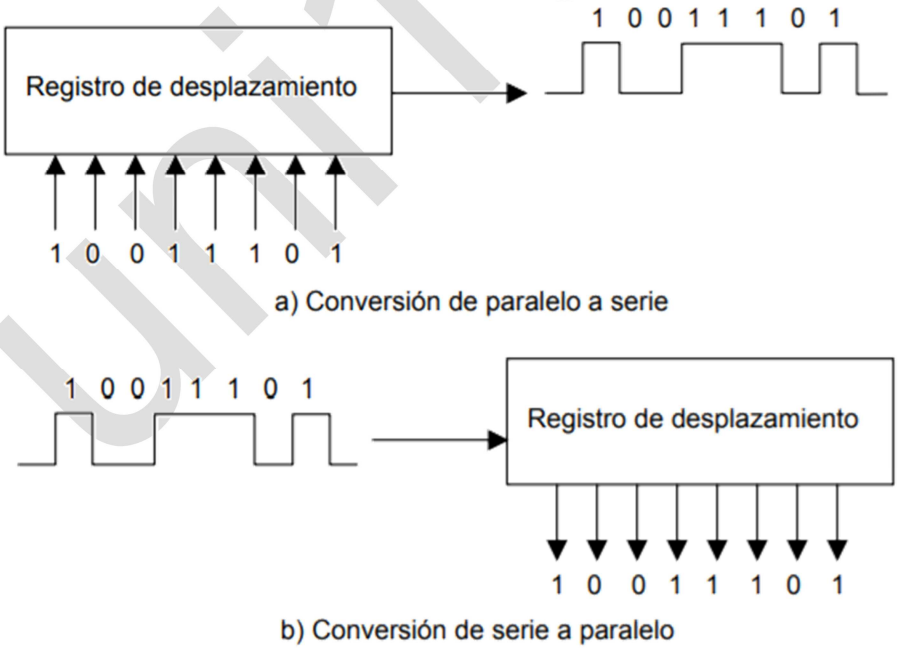
5.2. Registros de desplazamiento.

Los registros de desplazamiento están formados por un conjunto de flip-flops, y son muy importantes en las aplicaciones que precisan almacenar y transferir datos dentro de un sistema digital.

Un registro es un circuito digital con dos funciones básicas, almacenamiento y movimiento de datos. Se diferencian de las memorias, en que el tiempo de almacenamiento es menor. La capacidad de desplazamiento de un registro es la que permite el movimiento de los datos de una etapa a otra dentro del registro.

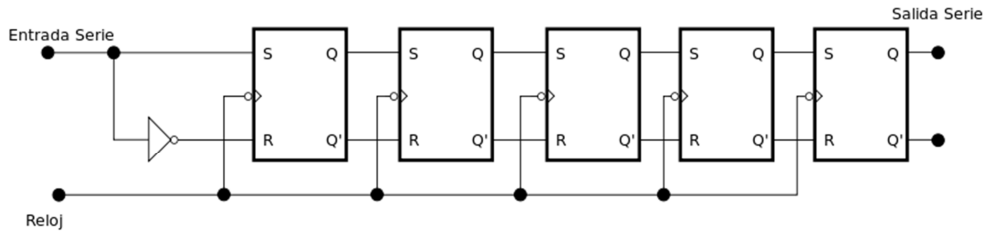
Los dos tipos de registros de desplazamiento son serie y paralelo. En un registro paralelo los bits se almacenan simultáneamente a partir de líneas paralelas, mientras que en un registro de desplazamiento serie, los bits se almacenan de uno a uno.

Estos dos tipos se emplean, entre otras cosas, para transformar (La siguiente figura) palabras de información en paralelo a una sucesión de bits sobre una línea, es decir, datos serie (registro de desplazamiento paralelo-serie), o una sucesión de datos en serie en una palabra de datos en paralelo (registro de desplazamiento serie- paralelo).

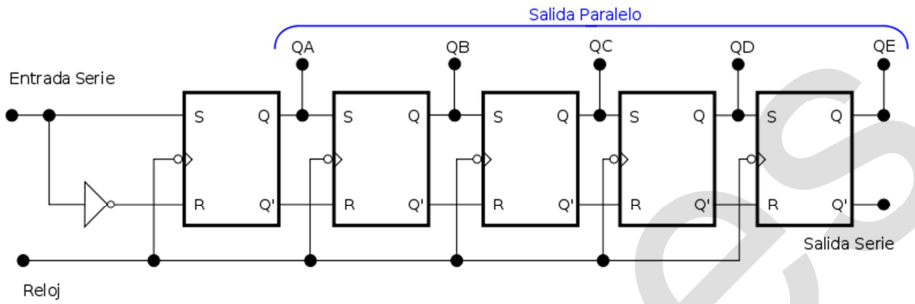


Por tanto, registros pueden ser:

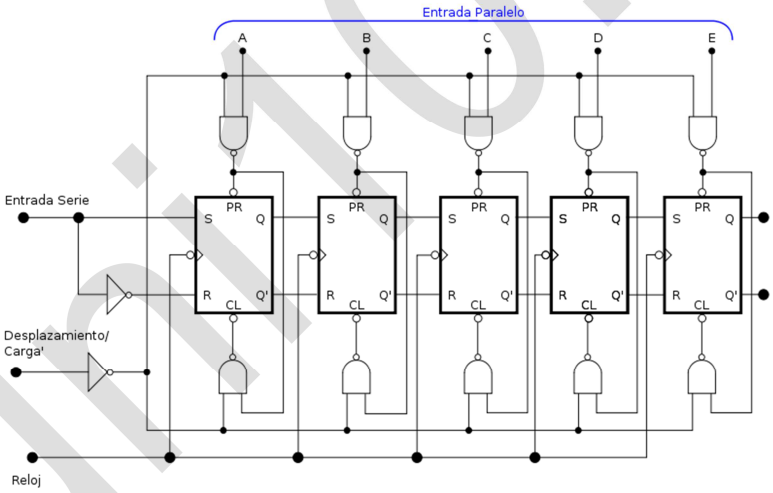
- De desplazamiento Serie-Serie: La entrada y salida de datos se produce uno a uno con cada impulso del reloj



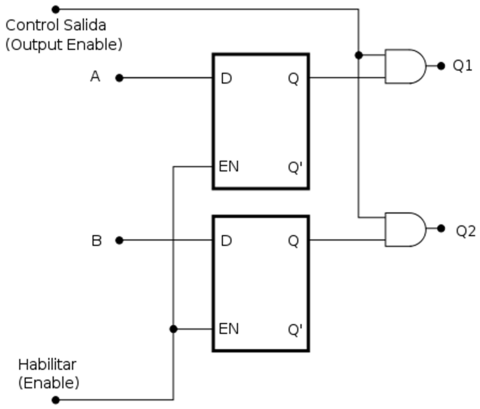
- De desplazamiento Serie-Paralelo: La entrada se realiza uno a uno con cada impulso del reloj, pero en la salida se disponen de todos los bits



- De desplazamiento Paralelo-Serie: Todos los bits se cargan en los biestables con un único impulso, pero se recuperan uno a uno



- De desplazamiento Paralelo-Paralelo: Tanto la entrada como la salida de todos los bits se producen en un único impulso de reloj.



5.3. Contadores.

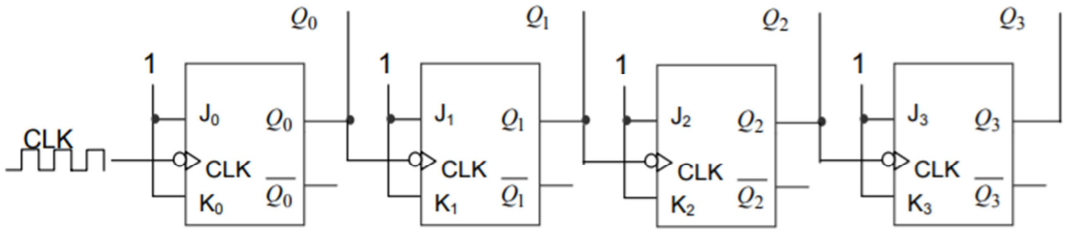
La función de recuento es muy importante en los sistemas digitales. Existen muchos tipos de contadores digitales, pero su objetivo básico es contar sucesos representados por cambios de nivel o impulsos, o generar una secuencia de códigos particular. Para contar, el contador debe recordar el número actual, con el fin de poder pasar correctamente al siguiente valor de la secuencia. Por tanto la capacidad de almacenamiento es una característica importante en todos los contadores, por lo que generalmente se utilizan flip-flops para su implementación.

5.3.1. Contadores asíncronos.

Un contador asíncrono es aquel en el que los flip-flops del contador no cambian de estado exactamente al mismo tiempo, dado que no comparten la misma señal de reloj.

5.3.2. Contador binario ascendente.

En la siguiente figura presenta un contador binario de cuatro bits conectado para que funcione de modo asíncrono (la entrada de reloj sólo está conectada al primer flip-flop). El segundo flipflop se dispara mediante la señal Q0 (salida del primer flip-flop), el tercer flip-flop mediante la señal Q1 (salida del anterior) y así sucesivamente. Cada flip-flop cambia durante el flanco descendente de la señal que se aplica a su entrada de reloj.

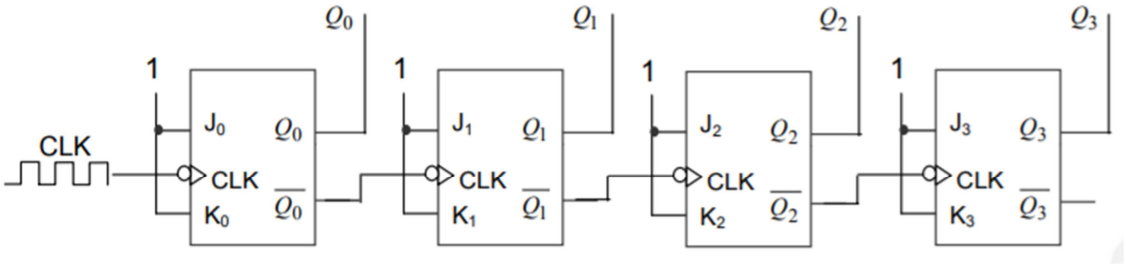


El contador de cuatro bits dispone de dieciséis estados diferentes (24 =16).Hay que tener en cuenta que Q0 representa el bit menos significativo (LSB) y Q3 el bit más significativo (MSB).

5.3.3. Contador binario descendente.

En la siguiente figura se observa un contador asíncrono binario descendente de cuatro bits. El circuito es similar al anterior, con la excepción de que las entradas de cada flip-flop se conectan a la salida complementaria del flip-flop anterior. Cada flip-flop bascula al recibir un flanco descendente en su señal de

reloj, que en este caso se produce cuando la salida anterior cambia de ‘0’ a ‘1’, o lo que es lo mismo la salida complementaria cambia de ‘1’ a ‘0’.



5.4. Autómatas programables.

La mayor aplicación de los circuitos secuenciales que se encuentra hoy en día es en el ámbito de la robótica, a través de los llamados ‘autómatas programables’. Un autómata programable es un sistema secuencial diseñado para controlar en tiempo real los sistemas secuenciales. Un autómata programable es un sistema programable diseñado para ser utilizado en un entorno industrial, que utiliza una memoria programable para el almacenamiento interno de instrucciones orientadas al usuario, para implantar unas soluciones específicas tales como funciones lógicas, secuencia, temporización, recuento y funciones aritméticas con el fin de controlar mediante entradas y salidas, digitales y analógicas diversos tipos de máquinas o procesos

Posee CPU, memoria y entradas y salidas, controladas como decíamos por cualquier tipo de actuador o sensor, como pulsadores, interruptores, termostatos, detectores de proximidad, relés, válvulas…

En general, las entradas y salidas (E/S) de un autómata pueden ser discretas, analógicas, numéricas o especiales. Las E/S discretas se caracterizan por presentar dos estados diferenciados: presencia o ausencia de tensión, relé abierto o cerrado, etc. Las E/S analógicas tienen como función la conversión de una magnitud analógica (tensión o corriente) equivalente a una magnitud física. Las E/S numéricas permiten la adquisición o generación de información a nivel numérico. Por último, las E/S especiales se utilizan en procesos en los que con las anteriores E/S vistas son poco efectivas, bien porque es necesario un gran número de elementos adicionales, bien porque el programa necesita de muchas instrucciones.

Para controlar un determinado proceso, el autómata realiza sus tareas de acuerdo con una serie de sentencias o instrucciones establecidas en un programa. Dichas instrucciones deberán haber sido escritas con anterioridad por el usuario en un lenguaje comprensible para el autómata. En general, las instrucciones pueden ser de funciones lógicas, de tiempo, de cuenta, aritméticas, de espera, de salto, de comparación, de comunicación y auxiliares.

5.5. Modelo de Moore

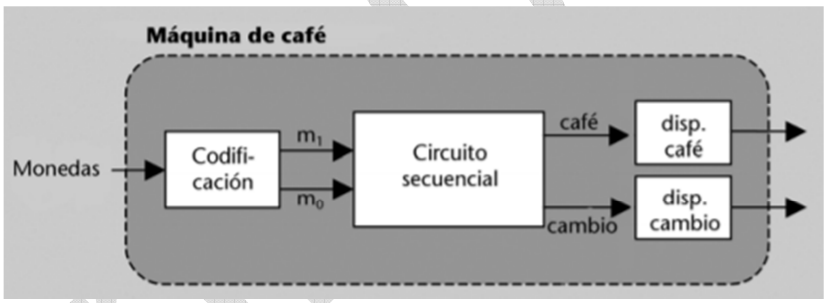
5.5.1. Estado. Transiciones.

El modelo de Moore es una forma de expresar o modelizar el funcionamiento de un circuito lógico secuencial. Es fundamental en los conceptos de estado y transiciones entre estados.

Para introducir estos conceptos utilizaremos un ejemplo:

Imaginemos un circuito que controla el funcionamiento de una máquina expendedora de café. Para simplificar, asumiremos que la máquina sólo sirve un único tipo de café, y que sólo admite monedas de 0,5 euros y de 1 euro. La información sobre las monedas introducidas se codifica mediante dos señales lógicas m1 y m0, tal como se muestra en la tabla al margen. Las señales m1 y m0 serán las entradas del circuito.

El precio del café es de 1,5 euros. La máquina tiene dos dispositivos de salida, uno para servir el café y otro para dar el cambio. Estos dos dispositivos están controlados respectivamente por las señales lógicas café y cambio, de forma que la máquina dará café o cambio cuando la señal correspondiente esté a 1. La siguiente figura muestra el esquema del funcionamiento de la máquina.



Para determinar en cada momento si tiene que servir café o dar cambio, la máquina necesita saber cuánto dinero se le ha introducido hasta el momento. Se pueden dar las siguientes situaciones:

- Se han introducido 0 euros.
- Se han introducido 0,5 euros.
- Se ha introducido 1 euro.
- Se han introducido 1,5 euros.
- Se han introducido 2 euros.

Se llama estado a cada situación diferente en la que se puede encontrar un circuito.

En nuestro ejemplo, la máquina de café puede encontrarse en cinco estados diferentes, los descritos por las cinco posibilidades anteriores. Las señales de salida del circuito tomarán un valor u otro según en qué estado se encuentre el circuito. En el caso de la máquina de café, la señal café estará a 1 cuando se

hayan introducido al menos 1,5 euros, y la señal cambio estará a 1 si se han introducido 2.

Una tabla de salida se expresa mediante el valor que toman las señales de salida en cada estado.

La tabla de salidas tiene a la izquierda los diferentes estados y a la derecha, el valor que toman las diferentes señales de salida en cada estado. La tabla de salidas del circuito de la máquina de café es la siguiente:

Estado	café	cambio
Se han introducido 0 euros.	0	0
Se han introducido 0,5 euros.	0	0
Se ha introducido 1 euro.	0	0
Se han introducido 1,5 euros.	1	0
Se han introducido 2 euros.	1	1

A medida que el tiempo avanza, el circuito cambia de estado en función de los valores que vayan llegando por las entradas. Mientras no se haya introducido ninguna moneda, la máquina de café se encontrará en el estado “se han introducido 0 euros”. Cuando se introduzca una moneda de 0,5 euros, pasará al estado “se han introducido 0,5 euros”; si la moneda es de 1 euro pasará al estado “se ha introducido 1 euro”.

El circuito no tiene que recordar necesariamente todos los valores que han llegado por las entradas, sino que los tiene que resumir en informaciones que sean relevantes pasa su funcionamiento. En el ejemplo de la máquina de café, se puede haber introducido 1 euro mediante una sola moneda de 1 o dos monedas de 0,5 euros. A la máquina le es indiferente cómo se haya hecho, ya que ambas acciones llevan a la misma situación: “se ha introducido 1 euro”. Por eso no están los estados “se han introducido dos monedas de 0,5 euros” y “se ha introducido una moneda de 1 euro”, sino que ambas informaciones se resumen en el estado “se ha introducido 1 euro”.

Llamamos estado actual al estado en que se encuentra la máquina en un instante dado. El nuevo valor (moneda o ausencia de monedas) que llega por la entrada determinará cuál será el próximo estado en que se encontrará el circuito: el estado futuro.

Se llama transición al paso del estado actual a un estado futuro.

Una señal de reloj sincroniza los circuitos secuenciales. En cada flanco ascendente se produce una transición hacia un estado futuro u otro en función del valor de las entradas. Las consideraciones con respecto a la evolución temporal de los circuitos se tratan en otro subapartado.

Todas las transiciones que se pueden dar en un circuito secuencial se especifican mediante la tabla de transiciones.

La tabla de transiciones tiene a la izquierda todas las combinaciones posibles de estados actuales y valores de las entradas, y a la derecha, el estado futuro al que lleva cada combinación. A continuación se muestra la tabla de transiciones del ejemplo de la máquina de café (para hacer la tabla más legible, en lugar de escribir “se han introducido 0 euros” escribimos “cero”, y de forma análoga para todos los estados).

Estado actual	Entrada	Estado futuro
cero	Ninguna moneda	cero
cero	Moneda de 0,5 euros	medio
cero	Moneda de 1 euro	uno
medio	Ninguna moneda	medio
medio	Moneda de 0,5 euros	uno
medio	Moneda de 1 euro	uno y medio
uno	Ninguna moneda	uno
uno	Moneda de 0,5 euros	uno y medio
uno	Moneda de 1 euro	dos
uno y medio	Ninguna moneda	cero
uno y medio	Moneda de 0,5 euros	medio
uno y medio	Moneda de 1 euro	uno
dos	Ninguna moneda	cero
dos	Moneda de 0,5 euros	medio
dos	Moneda de 1 euro	uno

Un circuito secuencial siempre tiene un estado que refleja la situación “aún no ha pasado nada”, es decir, “no es necesario recordar ninguno de los valores que ha llegado por la entrada”. Este estado se denomina estado inicial.

Cuando un circuito se pone en funcionamiento, está en el estado inicial. Ahora bien, también puede estar en otros momentos, si la funcionalidad del circuito así lo requiere. En el caso de la máquina de café, el estado inicial es “se han introducido 0 euros”. Cuando la máquina haya servido un café y mientras no se introduzcan más monedas, volverá al estado inicial.

Una vez especificado cuál es el estado inicial, la tabla de transiciones y la tabla de salidas describen completamente el comportamiento de un circuito lógico secuencial de acuerdo con el modelo de Moore.

5.5.2. Representación gráfica.

La especificación del funcionamiento de un sistema secuencial mediante el modelo de Moore se representa gráficamente con un grafo de estados, de la forma siguiente:

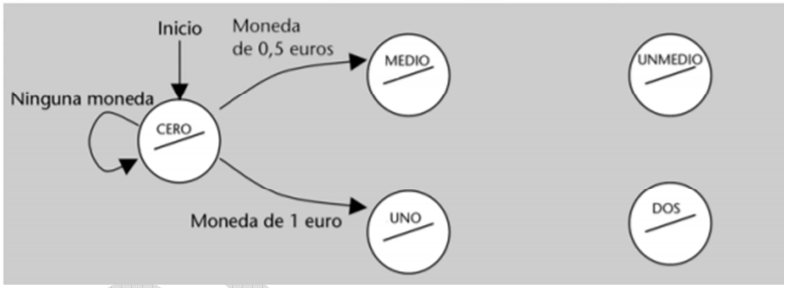
1) Para cada estado se dibuja un círculo, con el nombre del estado en la parte superior. El círculo correspondiente al estado inicial se señala con una flecha y la palabra Inicio.

2) En la parte inferior de cada círculo se escribe el valor que toman las señales de salida cuando el circuito se encuentra en el estado correspondiente a este círculo.

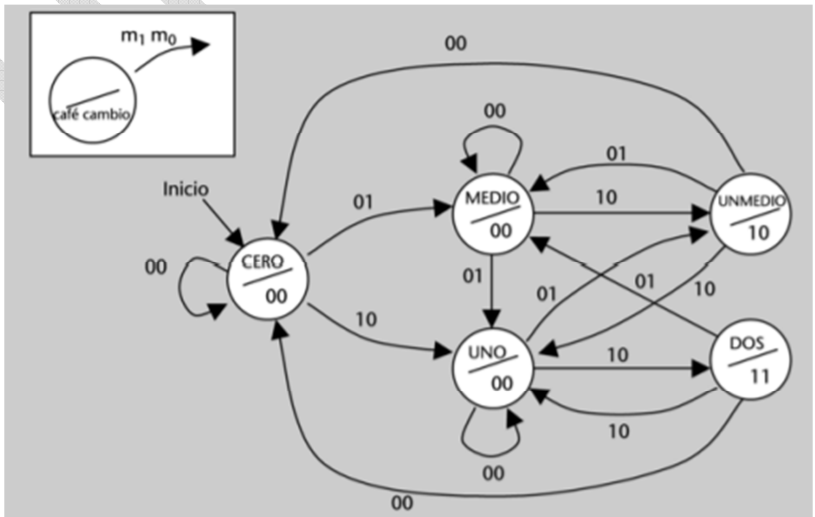
3) Las transiciones se representan mediante flechas o arcos, que tienen su origen en el círculo correspondiente al estado actual y la punta en el círculo correspondiente al estado futuro. El valor de las señales de entrada asociado con la transición se escribe al lado del arco.

La siguiente figura muestra la representación gráfica de los estados de la máquina de café, y las transiciones que parten del estado "se han introducido 0 euros". En esta figura hemos dado a los diferentes estados estos nombres:

Estado	Nombre
Se han introducido 0 euros.	CERO
Se han introducido 0,5 euros.	MEDIO
Se ha introducido 1 euro.	UNO
Se han introducido 1,5 euros.	UNMEDIO
Se han introducido 2 euros.	DOS



El grafo completo se muestra en la siguiente figura, con las entradas codificadas en binario. En el recuadro de la parte superior izquierda encontramos la leyenda del grafo.



Gracias a la leyenda sabemos que, en las parejas de valores de las etiquetas de la anterior figura, el de la izquierda corresponde a m1 y el de la derecha a m0 (y no al revés). La leyenda nos indica también que, de los valores de salida, el de la izquierda corresponde a la señal café y el de la derecha, a la señal cambio.

6. Conclusión.

Un álgebra de Boole es una entidad matemática formada por un conjunto que contiene dos elementos, unas operaciones básicas sobre estos elementos y una lista de axiomas que definen las propiedades que cumplen las operaciones.

Los dos elementos de un álgebra de Boole se pueden llamar falso y cierto o, más usualmente, 0 y 1. De este modo, una variable booleana o variable lógica puede tomar los valores 0 y 1. Para tratar estos valores se utiliza operaciones booleanas básicas, que son las siguientes: la negación o complementación o NOT, el producto lógico o AND y la suma lógica u OR.

Las funciones lógicas se pueden expresar de varias maneras. Las que usaremos nosotros son las expresiones algebraicas y las tablas de verdad. También es deseable que un circuito sea pequeño y barato. Estas dos características están relacionadas con el número de puertas del circuito: cuantas menos haya, más pequeño y barato será. Las simplificaciones las realizaremos mediante el uso de las reglas del álgebra de Boole o mediante el método de Karnaugh.

Hemos visto que se pueden construir sistemas digitales diferentes para funciones de una y dos variables de entrada. Ahora bien, en la práctica sólo se construyen los que calculan las funciones NOT, AND, OR, XOR, NAND y NOR. Los dispositivos electrónicos que calculan funciones lógicas se llaman puertas lógicas. Son dispositivos que están conectados a un cierto número de señales de entrada y a una señal de salida.

Un bloque combinacional es un circuito lógico combinacional con una funcionalidad determinada, que recibe señales de entrada, las procesa y emite señales de salida. Está construido a partir de puertas.

El sumador es un bloque combinacional que realiza la suma de dos números codificados en binario o bien en complemento a 2. Un multiplexor es un bloque que cumple la función de guardia urbano en circuitos electrónicos. Tiene un determinado número de señales de entrada que “compiten” para conectarse a una señal única de salida, y unas señales de control que sirven para determinar qué señal de entrada se conecta en cada momento con la salida. El comparador es un bloque combinacional que compara dos números codificados en binario e indica qué relación existe entre éstos.

El reloj o sincronización en los circuitos combinacionales, la única noción temporal que interviene es el presente. En cambio, en los circuitos secuenciales se tiene en cuenta la evolución temporal de las señales.

El biestable es un tipo de dispositivo de almacenamiento temporal de dos estados. Tenemos que tener en cuenta varios tipos: Biestables R-S con puertas NOR, Biestables R-S con puertas NAND y biestable R-S con entrada de habilitación. Además tenemos biestables síncronos, que utilizan la señal de reloj para cambiar de estado. Dentro de estos, encontramos los biestables R-S síncronos, biestables D, biestables J-K y biestables T.

Los registros de desplazamiento están formados por un conjunto de biestables, y son muy importantes en las aplicaciones que precisan almacenar y transferir datos dentro de un sistema digital.

La función de recuento es muy importante en los sistemas digitales. Existen muchos tipos de contadores digitales, pero su objetivo básico es contar sucesos representados por cambios de nivel o impulsos, o generar una secuencia de códigos particular. Para contar, el contador debe recordar el número actual, con el fin de poder pasar correctamente al siguiente valor de la secuencia. Por tanto la capacidad de almacenamiento es una característica importante en todos los contadores, por lo que generalmente se utilizan biestables para su implementación.

La aplicación de los sistemas secuenciales en la actualidad es la robótica y sus ramas, como la domótica. Se basan en el uso de autómatas programables que mediante una serie de sensores y actuadores, realizan funciones y guardan estados en memoria.

El modelo de Moore es una forma de expresar o modelizar el funcionamiento de un circuito lógico secuencial. Es fundamental en los conceptos de estado y transiciones entre estados.

7. Bibliografía.

- Gajsky, D.D. (1997). “Principios de diseño Digital”. Prentice-Hall.
- Hermida, R.; Corral, A. del; Pastor, E.; Sánchez, F. (1998). “Fundamentos de Computadores”. Madrid: Síntesis.
- Floyd, T.: “Fundamentos de sistemas digitales”. Ed. Prentice Hall
- Nelson V.P. (1996): “Análisis y Diseño de Circuitos Lógicos Digitales”. Ed. Prentice Hall
- Alcalde, Ormaechea (1991): “Arquitectura de ordenadores”. Ed McGraw-Hill