ESCUELA DE PREPARACIÓN DE OPOSITORES
E. P. O.
C/. La Merced, 8 – Bajo A Telf.: 968 24 85 54
30001 MURCIA

INF9 – SAI9

Lógica de circuitos. Circuitos combinacionales y secuenciales.

Esquema.

1 Introducción.

En este tema pretendemos dar una visión global del objetivo del diseño lógico y de los elementos constructivos con los que se realiza cualquier sistema digital, entre los que se encuentran los computadores. Para ello, realizamos una introducción al diseño de

sistemas digitales, presentando en primer lugar el álgebra de conmutación, los mapas de Karnaugh y el algoritmo de Quine-McCluskey como herramientas útiles en el proceso de diseño. A continuación, analizamos diferentes bloques combinacionales, como son los codificadores, decodificadores, multiplexores, etc. Posteriormente describimos los diferentes elementos de memoria, así como los registros de desplazamiento y contadores más usuales. Por último, concluimos nuestro estudio con un análisis de los sistemas secuenciales, y una visión general sobre el diseño de los mismos.

2 Sistemas digitales.

Un sistema digital es cualquier dispositivo destinado al tratamiento de información, en el que dicha información está representada por magnitudes físicas limitadas a tomar sólo unos determinados valores discretos. Un sistema digital en el que sólo se utilicen dos valores discretos se conoce como sistema binario; usualmente estos dos valores discretos se representan como cero y uno (0 y l).

Cualquier sistema digital se puede representar como un bloque con unas entradas y unas salidas, de manera que la acción del sistema consiste en asociar una determinada combinación de salidas para cada combinación de entradas. Como ejemplo podemos considerar un sistema con tres entradas y dos salidas; el funcionamiento de este sistema puede especificarse de una forma completa mediante la tabla de la figura.

	$x_1\ x_2\ x_3$	$z_1\ z_2$
0	000	0 0
1	001	0 1
2	010	1 1
3	011	0 0
4	100	1 1
5	101	0 1
6	110	0 0
7	111	0 0

Dado que las entradas sólo pueden tomar los valores 0 y 1, para tres entradas hay ocho combinaciones diferentes de entradas, que aparecen numeradas de 0 a 7. A cada una de las combinaciones de entradas le corresponde una combinación de salidas. A este tipo de tablas se les denomina tablas verdad.

3 Álgebra de Boole.

El álgebra de Boole está formado por variables lógicas, operadores lógicos y un conjunto de leyes que rigen ciertas combinaciones de los elementos anteriores.

3.1 Postulados del álgebra de Boole.

Los axiomas o postulados del álgebra de Boole que vamos a utilizar son los siguientes:

1. Conjunto de elementos. Existe un conjunto de elementos, B, en el que al menos hay dos elementos diferentes: $\exists B \mid x, y \in B,\ \ x \neq y$

2. Leyes de composición interna. En B se definen dos leyes de composición interna, que se representan de la forma y +, y se denominan Y y O respectivamente: $\forall x, y \in B,\ \ x \cdot y \in B,\ \ x + y \in B$

3. Elementos neutros únicos.
   - $\exists 0 \in B \mid \forall x \in B,\ \ x + 0 = 0 + x = x$
   - $\exists 1 \in B \mid \forall x \in B,\ \ x \cdot 1 = 1 \cdot x = x$
   - $0 \neq 1$

4. Conmutatividad de las leyes de composición interna.
   $\forall x, y \in B,\ \ x + y = y + x,\ \ x \cdot y = y \cdot x$

5. Distributividad de las leyes de composición interna.
   $\forall x, y, z \in B,\ \ x + (y \cdot z) = (x + y) \cdot (x + z),\ \ x \cdot (y + z) = (x \cdot y) + (x \cdot z)$

6. Asociatividad de las leyes de composición interna.
   $\forall x, y, z \in B,\ \ x + (y + z) = (x + y) + z = x + y + z,\ \ x \cdot (y \cdot z) = (x \cdot y) \cdot z = x \cdot y \cdot z$

7. Elemento opuesto (complemento) único.
   $\forall x \in B,\ \ \exists \bar{x} \in B \mid x + \bar{x} = 1,\ \ x \cdot \bar{x} = 0$

3.2 Teoremas del álgebra de Boole.

A continuación, vamos a enunciar un conjunto de teoremas, seleccionados por la aplicación que tienen o por los hechos fundamentales que establecen.

- Idempotencia: $\forall a \in B,\ \ a + a = a,\ \ a \cdot a = a$
- Elemento neutro. $\forall a \in B,\ \ a + 1 = 1,\ \ a \cdot 0 = 0$
- Complementación. $\bar{1} = 0,\ \ \bar{0} = 1$
- Ley de Morgan. $\forall a, b \in B,\ \ \overline{a + b} = \bar{a} \cdot \bar{b},\ \ \overline{a \cdot b} = \bar{a} + \bar{b}$
- Absorción. $\forall a, b \in B,\ \ a + a \cdot b = a,\ \ a \cdot (a + b) = a$
- Involución. $\forall a \in B,\ \ \bar{\bar{a}} = a$
- $\forall a, b \in B,\ \ a + \bar{a} \cdot b = a + b,\ \ a \cdot (\bar{a} + b) = a \cdot b$

4 Funciones lógicas.

Definimos como variable lógica a un símbolo que en un instante determinado sólo puede tomar uno de los valores 0 ó 1. Dado el conjunto $C=\{0,1\}$, una función lógica de n variables, $f(x_1,...,x_n)$, es una aplicación del conjunto producto cartesiano $\{0,1\}^n$ en el conjunto $\{0,1\}$.

Una función lógica se puede dar de varias formas; las dos más usuales son las siguientes:

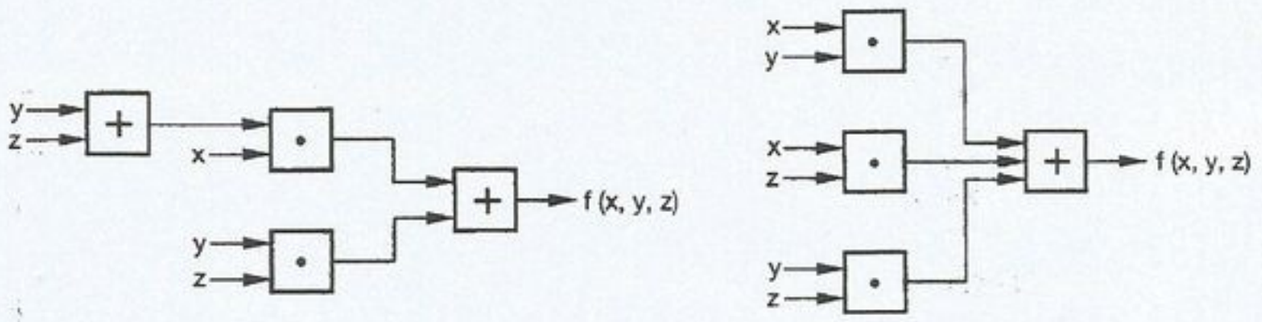
a) En forma de tabla verdad.

b) Mediante una igualdad, como: $f(x, y, z) = x \cdot (y+z) + y \cdot z$

La expresión de la forma b) significa que, para cada combinación de valores de las entradas, el valor 0 ó 1 que $f(x, y, z)$ asigna, se determina evaluando la expresión a la derecha del signo de igualdad. Concretamente esta función es {000, 001, 010, 011, 100, 101, 110, 111} → {0, 0, 0, 1, 0, 1, 1, 1}. En la siguiente figura mostramos la expresión de esta función en forma de tabla verdad.

xyz	*F*
000	0
001	0
010	0
011	1
100	0
101	1
110	1
111	1

Si se dispone de dispositivos que sinteticen las funciones Y y O, un posible circuito para sintetizar $f(x, y, z)$ es el de la siguiente figura. Este circuito se obtiene trasladando directamente la expresión algebraica de partida. Aplicando a esta expresión la propiedad distributiva resulta $f(x, y, z) = x\cdot(y+z)+y\cdot z=x\cdot y+x\cdot z+y\cdot z$. Es decir, una misma función se puede sintetizar con diferentes circuitos.



4.1 Álgebra de conmutación.

El conjunto F de todas las funciones lógicas de n variables, con la relación de igualdad y las operaciones Y, O, y complementación definidas a continuación, forma un álgebra de Boole finita, que se conoce como álgebra de conmutación.

- Relación de igualdad: dos funciones lógicas de n variables son iguales si para todos los posibles valores de las variables se establecen las mismas asignaciones.
- Operación O (+): dadas f, $g\in F$, se define $h=f+g$, $h\in F$, como sigue: "para cada elemento de $\{0,1\}^n$, h asigna el valor 0 si y sólo si f y g asignan el valor 0; en cualquier otro caso h asigna el valor 1". De acuerdo con esta definición, la operación O queda descrita mediante la tabla de la siguiente figura. Además, en la figura se tiene el símbolo para cualquier sistema digital que realice la operación O.

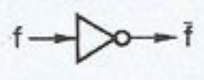
O	0	1
0	0	1
1	1	1

- Operación Y (·): dadas $f, g \in F$, se define $h=f \cdot g$, $h \in F$, como sigue: "para cada elemento de $\{0,1\}^n$, h asigna el valor 1 si y sólo si f y g asignan ambas el valor 1; en cualquier otro caso h asigna el valor 0". De acuerdo con esta definición, la operación Y queda descrita mediante la tabla de la siguiente figura. Además, en la figura se da el símbolo que se va a utilizar para cualquier sistema digital binario (electrónico, por ejemplo) que realice la operación Y.

Y	0	1
0	0	0
1	0	1

- Complemento de una función: dada $f \in F$, se define $\overline{f} \in F$ como sigue: "para cada elemento de $\{0,1\}^n$, $\overline{f}$ asigna el valor 0 si y sólo si f asigna el valor 1, y viceversa". La operación de complementación queda descrita mediante la tabla de la siguiente figura, donde además se tiene el símbolo de un complementador.

–	
0	1
1	0



- Elemento cero: la función cero (0^n) es la que a cada elemento de $\{0,1\}^n$ le asigna el valor 0.
- Elemento uno: la función uno (1^n) es la que a cada elemento de $\{0,1\}^n$ le asigna el valor 1.

4.2 Minimización de funciones.

Resulta muy interesante minimizar la representación de una función, puesto que así a la hora de la representación física de dicha función se obtiene un circuito más sencillo y barato. Existen varios métodos para simplificar funciones lógicas:

- Método algebraico. Utiliza los postulados, teoremas y propiedades del álgebra de Boole para simplificar las funciones lógicas.
- Método gráfico de Karnaugh. Es un método que se basa en la representación gráfica de la función en lo que se denomina mapa de Karnaugh.
- Método numérico de Quine-McCluskey. Se trata de un método fácilmente programable, que se utiliza para simplificar funciones de un gran número de variables (4 o más).

4.2.1 Formas canónicas o minterms y maxterms.

La utilidad de las funciones minterm y maxterm reside en que cualquier función lógica se puede expresar en términos de estas funciones, conociéndose esta expresión como forma canónica de una función.

Dado un número entero I, $0 \leq I \leq 2^n-1$, y su expresión en base 2 con n bits, $(i_1,\ldots,i_n)$, se define la función minterm $m_I(x_1,\ldots,x_n)$ de la forma:

$m_I(x_1, ..., x_n) = 1$ si $x_1 = i_1, ..., x_n = i_n$
$m_I(x_1, ..., x_n) = 0$ en cualquier otro caso

Esta función asigna el valor 1 para una sola *n*-tupla (de aquí la denominación de término mínimo) de $\{0,1\}^n$ y esta *n*-tupla representa, por tanto, unívocamente al minterm. Para *n* variables hay 2^n minterms.

Un minterm se puede expresar como un producto de la forma $x_1^{i_1} \ldots x_n^{i_n}$ donde

$x_j^{i_j} = x_j$ si $i_j = 1$
$x_j^{i_j} = \bar{x}_j$ si $i_j = 0$

En la siguiente figura se relacionan los ocho minterms de tres variables, expresados como productos Y. En cada fila se da el producto Y que corresponde al minterm con el 1 en esa fila de la tabla verdad. Cada variable aparece en el producto complementada o sin complementar, dependiendo de que esta variable en esa fila tome el valor 0 ó 1, respectivamente.

	$x_1\,x_2\,x_3$	$m_0\,m_3\,m_4\,m_6$	...	$M_0\,M_2\,M_5\,M_7$	$f_1\,f_2$
0	000	1 0 0 0		0 1 1 1	0 0
1	001	0 0 0 0		1 1 1 1	1 0
2	010	0 0 0 0		1 0 1 1	1 0
3	011	0 1 0 0		1 1 1 1	0 1
4	100	0 0 1 0		1 1 1 1	0 1
5	101	0 0 0 0		1 1 0 1	0 1
6	110	0 0 0 1		1 1 1 1	0 1
7	111	0 0 0 0		1 1 1 0	1 0

	xyz	*minterms*	*Maxterms*
0	000	$\bar{x}\,\bar{y}\,\bar{z}$	$x+y+z$
1	001	$\bar{x}\,\bar{y}\,z$	$x+y+\bar{z}$
2	010	$\bar{x}\,y\,\bar{z}$	$x+\bar{y}+z$
3	011	$\bar{x}\,y\,z$	$x+\bar{y}+\bar{z}$
4	100	$x\,\bar{y}\,\bar{z}$	$\bar{x}+y+z$
5	101	$x\,\bar{y}\,z$	$\bar{x}+y+\bar{z}$
6	110	$x\,y\,\bar{z}$	$\bar{x}+\bar{y}+z$
7	111	$x\,y\,z$	$\bar{x}+\bar{y}+\bar{z}$

Dado un número entero *I*, $0 \le I \le 2^n-1$, y su expresión en base 2 con *n* bits, $(i_1,\ldots,i_n)$, se define la función maxterm $M_I(x_1,\ldots,x_n)$ de la forma:

$M_I(x_1, ..., x_n) = 0$ si $x_1 = i_1, ..., x_n = i_n$
$M_I(x_1, ..., x_n) = 1$ en cualquier otro caso

Esta función asigna el valor 0 para una sola *n*-tupla (de aquí la denominación de término máximo) de $\{0,1\}^n$; esta *n*-tupla representa, por tanto, unívocamente al maxterm. Para *n* variables hay 2^n maxterms.

Un maxterm se puede expresar como una suma de la forma $\bar{x}_1^{i_1} \ldots \bar{x}_n^{i_n}$ donde

$$\bar{x}_j^{i_j} = \bar{x}_j \quad \text{si } i_j = 1$$
$$\bar{x}_j^{i_j} = x_j \quad \text{si } i_j = 0$$

En la figura anterior se dan los ocho maxterms de tres variables, expresados en forma de suma O. En cada fila se da la suma O que corresponde al maxterm con el 0 (cero) en esa fila de la tabla verdad. Cada variable aparece en la suma complementada o sin complementar, dependiendo de que esta variable en esa fila tome el valor 1 ó 0, respectivamente.

4.2.2 Teorema de Shannon

Se tienen dos alternativas para realizar cualquier función lógica con estructuras de dos niveles: bien con un nivel de puertas Y que alimentan a una puerta O de salida, bien con un nivel de puertas O que alimentan a una puerta Y de salida. Ninguna de estas dos formas es ventajosa frente a la otra, y los procedimientos que sean aplicables a una de ellas se pueden aplicar de forma paralela a la otra.

Teorema de Shannon (primera forma).

Toda función lógica de n variables, $f(x_1, \ldots x_n)$, puede expresarse como una suma única de minterms. Por ejemplo, las funciones f_1 y f_2 de la figura anterior tienen la siguiente expresión en forma de suma de minterms:

$$f_1 = \bar{x}\bar{y}z + \bar{x}y\bar{z} + xyz$$
$$f_2 = \bar{x}yz + x\bar{y}\bar{z} + x\bar{y}z + xy\bar{z}$$

Dada una función lógica expresada en forma de tabla verdad, para obtener su expresión en forma de suma de minterms hay que escribir un producto Y (un minterm) por cada 1 que aparezca en la columna del valor de la función en la tabla de verdad.

Teorema de Shannon (segunda forma)

Toda función lógica de n variables, $f(x_1, \ldots x_n)$, puede expresarse como un producto único de maxterms. Aplicando esta segunda forma del teorema de Shannon resulta un producto de maxterms, tal que hay un maxterm por cada 0 que aparezca en la tabla verdad de la función. Por ejemplo, las funciones f_1, y f_2 anteriores tienen la siguiente expresión en forma de producto de maxterms:

$$f_1 = (x+y+z)\cdot(x+\bar{y}+\bar{z})\cdot(\bar{x}+y+z)\cdot(\bar{x}+y+\bar{z})\cdot(\bar{x}+\bar{y}+z)$$
$$f_2 = (x+y+z)\cdot(x+y+\bar{z})\cdot(x+\bar{y}+z)\cdot(\bar{x}+\bar{y}+\bar{z})$$

Dada una función lógica expresada en forma de tabla de verdad, para obtener su expresión en forma de producto Y de maxterms hay que escribir una suma O (un maxterm) por cada 0 (cero) que aparezca en la columna del valor de la función en la tabla de verdad.

4.2.3 Método gráfico de Karnaugh.

Adyacencias.

A continuación, vamos a definir de una manera recurrente las adyacencias de cualquier orden.

- Definición 1: un minterm es una adyacencia de orden cero.

- Definición 2: dos adyacencias de orden *i*, siendo *i* cualquier entero mayor o igual que cero, forman una adyacencia de orden *i*+1 si las dos adyacencias de partida se expresan como productos en los que aparezcan las mismas variables, y únicamente se diferencian en que una de esas variables aparece en un producto complementada y en el otro sin complementar. La adyacencia de orden *i*+1 se expresa como un producto en el que aparecen todas las variables idénticas en las dos adyacencias de orden *i* originarias (es decir, no aparece la variable en que difieren).

Por ejemplo, refiriéndonos a las variables *x*, *y*, *z*, *u*, dos minterms o adyacencias de orden cero son:

$$x\,\overline{y}\,\overline{z}\,u$$

$$x\,\overline{y}\,\overline{z}\,\overline{u}$$

Estas dos adyacencias de orden cero forman una adyacencia de orden uno, dado que en ambos minterms se tienen las variables x, $\overline{y}$, $\overline{z}$ (todas menos una) idénticas. La adyacencia de orden uno resultante se expresa como un producto de estas tres variables, es decir, $x\overline{yz}$. Concretamente, se ha utilizado la siguiente sustitución:

$$x\overline{y}\overline{z}u + x\overline{y}\overline{z}\overline{u} = x\overline{y}\overline{z}\,(u + \overline{u}) = x\overline{y}\overline{z} \cdot 1 = x\overline{y}\overline{z}$$

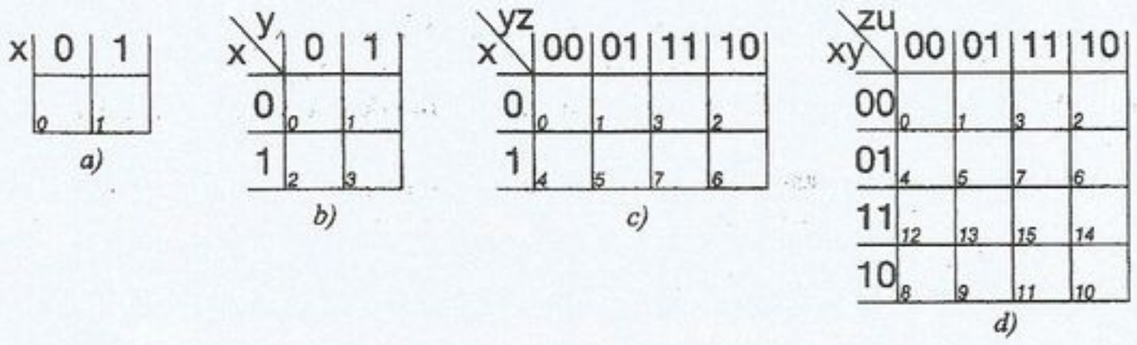
Dos adyacencias de orden uno son $x\overline{yz}$, $x\overline{y}z$. Estas dos adyacencias de primer orden forman una adyacencia de segundo orden expresada mediante el producto $x\overline{y}$. Dada una adyacencia de un orden cualquiera referida a unas determinadas variables, es inmediato obtener las adyacencias de orden cero de las que procede. Por ejemplo, sean las variables *x*, *y*, *z*, *u* y sea la adyacencia de orden dos $x\overline{z}$. Las adyacencias de orden cero de las que procede (serán cuatro) se obtienen como sigue:

$$x\overline{z} = x(y + \overline{y})\,\overline{z}\,(u + \overline{u}) = xy\overline{z}u + xy\overline{z}\,\overline{u} + x\overline{y}\overline{z}u + x\overline{y}\overline{z}\,\overline{u}$$

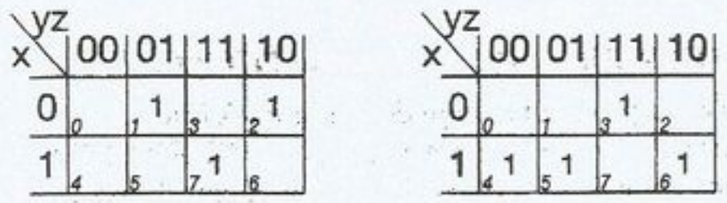
El concepto de adyacencia es muy útil con vistas a la realización de las funciones lógicas.

Mapas de Karnaugh.

Los mapas de Karnaugh constituyen una representación gráfica de las funciones lógicas. Cualquier función de una variable se puede representar utilizando el mapa de Karnaugh correspondiente. Consta de dos casillas, correspondientes a los dos posibles valores de la variable. Poniendo en cada casilla el valor 0 ó 1 que la correspondiente función asigne, se tiene un medio de representar las cuatro funciones de una variable. En la siguiente figura se tienen los mapas de Karnaugh para dos, tres y cuatro variables, respectivamente. Cada casilla corresponde a un minterm; en el ángulo inferior izquierdo de cada casilla se ha escrito el valor decimal del minterm correspondiente.



En el caso de tres variables, las columnas se han asignado de forma que cada columna sea adyacente a sus dos vecinas, entendiendo que la primera y la última son vecinas. Por ejemplo, la columna 11 es adyacente a 01 y 10, la 10 es adyacente a 11 y 00, y así sucesivamente. Esto mismo se ha hecho para las filas y columnas del mapa de cuatro variables. De forma similar se construyen mapas de Karnaugh para más variables. En la siguiente figura se tienen las representaciones en forma de mapa de Karnaugh de las funciones f_1 y f_2 anteriores (únicamente se han especificado los valores 1; las casillas en blanco corresponden a los ceros).



El principal interés de los mapas de Karnaugh reside en su posible utilización en la minimización de funciones booleanas. Cada celda de un mapa de Karnaugh de n variables puede caracterizarse con n coordenadas. Por ejemplo, la celda 5 del mapa de Karnaugh de tres variables tiene como coordenadas 101 (x=1, y=0, z=l),y corresponde al minterm $x\bar{y}z$. Dos celdas (o cubos de orden cero o adyacencias de orden cero) de un mapa de Karnaugh son adyacentes y forman un cubo de orden uno, si las coordenadas de ambas celdas son idénticas salvo una, que en un cubo valdrá cero y en otro valdrá uno. Un cubo de orden uno de n variables puede darse con (n-1) coordenadas binarias y una coordenada vacua, para la que se utilizará el símbolo -. Por ejemplo, los cubos de orden cero para cuatro variables 0011 y 0001 son adyacentes, y forman un cubo de orden uno de coordenadas 00-1. Un cubo de orden uno (o adyacencia de primer orden) corresponde a un producto de (n-1) variables, faltando la variable correspondiente a la coordenada vacua. Por ejemplo, al cubo 00-1 le corresponde el producto $\overline{xyu}$.

En general, dos cubos de orden i son adyacentes y forman un cubo de orden i+1 si ambos tienen coordenadas vacuas en las mismas posiciones, y las coordenadas no vacuas son todas iguales salvo una; el cubo de orden i+1 resultante tendrá i+1 coordenadas vacuas: las i de los cubos adyacentes y otra en la posición en la que difieren las coordenadas de los cubos adyacentes. En este caso, se dice que el cubo de mayor orden cubre a aquellos de los que procede.

Dada una función de n variables, se dice que una adyacencia de cualquier orden de esas variables es un implicante de la función, si para aquellos valores de las variables para los que la adyacencia toma el valor uno, también la función toma el valor uno.

Se dice que un implicante de una función es un implicante primo si las adyacencias que resultan del implicante eliminando una cualquiera de las variables ya no son implicantes de la función.

Una vez introducidos todos estos conceptos, es inmediato presentar un algoritmo para simplificar funciones lógicas utilizando el mapa de Karnaugh. Una vez representada la función a simplificar en un mapa de Karnaugh, el algoritmo consta de los pasos siguientes:

a) Escribir todos los cubos (y las adyacencias correspondientes) de cualquier orden, tales que cada uno cumpla las dos condiciones siguientes:
   - Todas las celdas del cubo corresponden a valores 1 de la función.
   - No está cubierto por ningún otro cubo que cumpla la condición anterior.

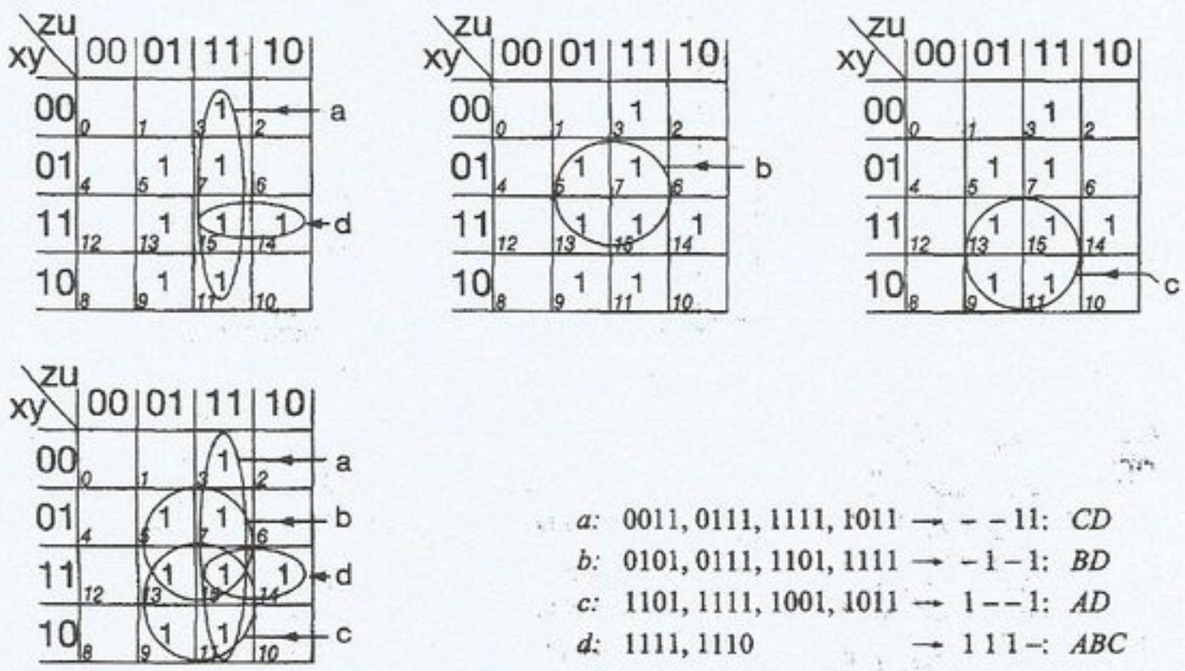
b) Con las adyacencias anteriores, construir una cobertura mínima para todos los unos de la función. La forma de obtener esa cobertura mínima va a ser intuitiva, aunque es fácil sistematizarla.

Las adyacencias obtenidas en a) son los implicantes primos de la función de que se trate, como se puede comprobar en cada caso.

Ejemplo: Simplificar la función *F* siguiente:

$$F = \bar{A}\bar{B}CD + \bar{A}B\bar{C}D + \bar{A}BCD + A\bar{B}\bar{C}D + A\bar{B}CD + AB\bar{C}D + ABC\bar{D} + ABCD$$

En la siguiente figura se representa *F* en un mapa de Karnaugh. En esta misma figura se indican las agrupaciones de celdas correspondientes a la aplicación del algoritmo.



Cada uno de *F* se puede utilizar múltiples veces para construir adyacencias de mayor orden. Por ejemplo, la celda 1111 se utiliza en cuatro ocasiones. Para realizar *F* hay que cubrir todos sus unos. Teniendo en cuenta esto último, es inmediato obtener una cobertura mínima para *F* como suma de implicantes primos; basta fijarnos en que cada una de las cuatro adyacencias, *a*, *b*, *c*, y *d*, incluye un uno no cubierto por ninguna otra adyacencia, con lo que es imprescindible incluir las cuatro adyacencias en la realización de *F*, y con las cuatro queda cubierta *F*; es decir:

$$F = a + b + c + d = CD + BD + AD + ABC$$

El mapa de Karnaugh se puede utilizar de forma similar para la realización en forma de producto de sumas.

4.2.4 Algoritmo de Quine–McCluskey.

Cuando la función a minimizar contiene más de cinco o seis variables empleamos el algoritmo de Quine–McCluskey, el cual vamos a presentar mediante la minimización de la función: A'·B'·C'·D + A'·B'·C·D'+ A'·B'·C·D + A'·B·C·D'+ A'·B·C·D + A·B'·C'·D' + A·B'·C'·D + A·B'·C·D'+ A·B·C·D.

1. Enumerar en una tabla todos los minterm en forma binaria, organizados según el número de unos que contenga.

Minterms	A	B	C	D	Grupo
1	0	0	0	1	Grupo 1

2	0	0	1	0	
8	1	0	0	0	
3	0	0	1	1	Grupo 2
6	0	1	1	0	
9	1	0	0	1	
10	1	0	1	0	
7	0	1	1	1	Grupo 3
15	1	1	1	1	Grupo 4

2. Entre los grupos adyacentes buscar los minterm que sólo difieren en un bit en la misma posición, para hallar los primeros implicantes primos, comparando para ello cada minterm con el resto de los términos del siguiente grupo. Los minterm utilizados se marcan para ir diferenciando los términos utilizados, y la variable apareada en el proceso se reemplaza con un guión para denotar la eliminación de la variable. Los términos no marcados en la tabla son los primeros implicantes primos (PI_X).

Minter	A	B	C	D	Minterm	A	B	C	D	PIx	Minterm	A	B	C	D	PIx
1 √	0	0	0	1	1–3	0	0	-	1	PI2	2–6 - 3-7	0	-	1	-	PI1
2 √	0	0	1	0	1–9	-	0	0	1	PI3	2-3 - 6-7	0	-	1	-	
8 √	1	0	0	0	2–3 √	0	0	1	-							
3 √	0	0	1	1	2–6 √	0	-	1	0							
6 √	0	1	1	0	2–10	-	0	1	0	PI4						
9 √	1	0	0	1	8–9	1	0	0	-	PI5						
10 √	1	0	1	0	8-10	1	0	-	0	PI6						
7 √	0	1	1	1	3–7 √	0	-	1	1							
15 √	1	1	1	1	6–7 √	0	1	1	-							
					7-15	-	1	1	1	PI7						

3. Construir una tabla que enumere los implicantes primos y los minterm contenidos por cada implicante primo. La letra X en la tabla indica el minterm contenido en cada implicado por fila.

Implicante Primo	1	2	3	6	7	8	9	10	15
* PI1		X	X	X	X				
PI2	X		X						
PI3	X						X		
PI4		X						X	
PI5						X	X		
PI6						X		X	
* PI7					X				X

4. En la tabla se seleccionan las columnas de los minterms que contengan solamente una cruz (6 y 15 en nuestro ejemplo). Es decir, la selección del primer implicado PI1 (A'·C) garantiza que el término mínimo 6 está incluido en la función. De la misma forma, el término mínimo 7 está cubierto por el primer implicado PI7 (A'·B·C·D). Los primeros implicados que cubren los mintérminos con una sola cruz, se llaman primeros implicados esenciales (marcados con un asterisco) y son indispensables en la construcción de la función.
5. Seleccionar en cada columna los minterms que estén cubiertos por los primeros implicados esenciales. Por ejemplo, el primer implicado esencial * PI1 (A'·C)

cubre los minterms 2, 3, 6 y 7, y *PI7 (A'·B·C·D) cubre 7 y 15. Los minterms 1, 8, 9 y 10 deben ser seleccionados por medio de otros primeros implicados esenciales.

Implicante Primo	1	8	9	10
PI2	X			
*PI3	X		X	
PI4				X
PI5		X	X	
*PI6		X		X

La función simplificada se obtiene de la suma de los primeros implicados hallados:

$$F= PI_1 + PI_3 + PI_6 + PI_7$$

$$F= (0\text{-}1\text{-}) + (\text{-}001) + (10\text{-}0) + (\text{-}111)$$

$$F = A'\cdot C + B'\cdot C'\cdot D + A\cdot B'\cdot D' + B\cdot C\cdot D$$

5 Sistemas combinacionales.

Un circuito o dispositivo digital es combinacional si las salidas en un instante cualquiera están determinadas exclusivamente por las entradas en ese instante, es decir

$$z_i(t) = f_i(x_1(t), ..., x_n(t)), \qquad i = 1, ..., m$$

Normalmente no se explicitará la variable *t*, escribiéndose

$$z_i = f_i(x_1, ..., x_n), \qquad i = 1, ..., m$$

En esta definición se está suponiendo que el tiempo de propagación de las señales a través del circuito o dispositivo es suficientemente pequeño, de manera que cualquier variación en el valor de las entradas afecta instantáneamente a las salidas.

5.1 Puertas combinacionales combinadas.

La tecnología de fabricación de circuitos lógicos permite integrar muchos circuitos en un mismo chip. Por esta razón, y dado que las puertas fundamentales se utilizan en gran número incluso en los sistemas digitales más sencillos, no se construyen por separado, sino que se incluyen varias puertas dentro de un mismo chip.

Las puertas NO-Y (o NAND en inglés) y las NO-O (o NOR) son importantes como elementos de diseño, pues tanto la función NAND como la función NOR constituyen un conjunto suficiente de conectores. Esto quiere decir que para sintetizar cualquier función combinacional basta utilizar sólo puertas NAND (o sólo puertas NOR), en vez de utilizar tres tipos de puertas (Y, O y NO).

5.2 Sumador binario.

Supongamos que tenemos, utilizando notación posicional, los números binarios de *n* bits $X = x_{n-1}x_{n-2}...x_1x_0$ e $Y = y_{n-1}y_{n-2}...y_1y_0$. Nuestro objetivo es diseñar un sistema combinacional, al que llamaremos sumador binario, que nos permita sumar los bits *x*, e *y*, del mismo orden, de manera que asociando adecuadamente *n* sumadores binarios se pueda realizar la suma *X*+*Y*. Este bloque combinacional para sumar cada bit tendrá tres entradas (los bits *x*, e *y*, más el acarreo precedente, *C*) y dos salidas (la suma *S*, y el acarreo para la etapa siguiente, C_{i+1}).

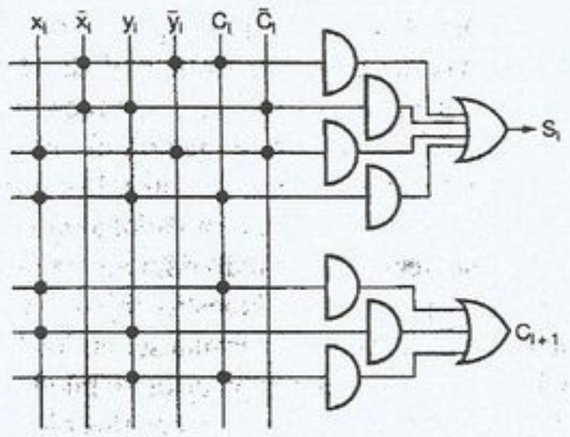
La tabla verdad correspondiente a este sistema combinacional es:

x_i	y_i	C_i	S_i	C_{i+1}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

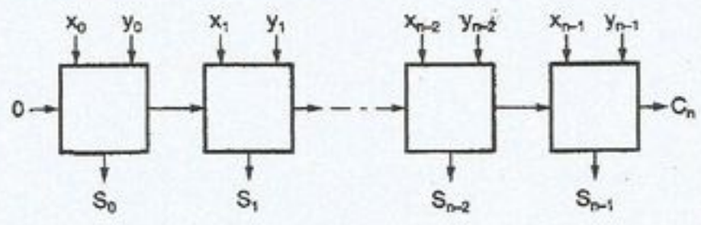
Representando S_i y C_{i+1} en forma de mapa de Karnaugh se concluye que las expresiones de ambas en forma de suma de productos es

$$S_i = \bar{x}_i \bar{y}_i C_i + \bar{x}_i y_i \bar{C}_i + x_i \bar{y}_i \bar{C}_i + x_i y_i C_i$$
$$C_{i+1} = x_i C_i + x_i y_i + y_i C_i$$

El circuito correspondiente es el de la siguiente figura, en la que, para simplificar el dibujo, las diferentes entradas a cada puerta Y se han representado mediante una sola línea con múltiples conexiones (una por cada entrada).



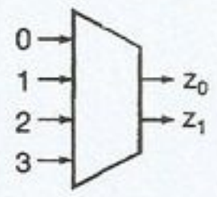
Para sumar las palabras X e Y de n bits basta conectar n sumadores binarios en la forma representada en la siguiente figura; en este caso los acarreos se propagan en cascada a través de los n sumadores.



5.3 Codificadores.

Un codificador es un dispositivo combinacional con n entradas y m salidas, tal que en un instante cualquiera, sólo una de las entradas puede tomar el valor 1. El codificador genera como salida una combinación de m bits que es única para cada entrada al valor 1, conociéndose esta combinación como código de ese carácter de entrada. Los codificadores más usuales son los binarios, que tienen 2^n entradas y n salidas. En cada instante generan como salida el código binario correspondiente a la entrada activa. Por ejemplo, un codificador binario con 4 entradas y 2 salidas realiza la codificación dada en la siguiente figura:

	z_1	z_0
0	0	0
1	0	1
2	1	0
3	1	1



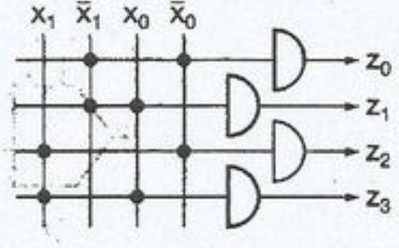
5.4 Decodificadores/demultiplexores.

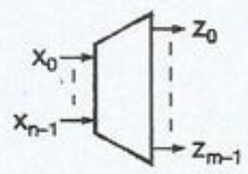
Un decodificador es un dispositivo combinacional con *n* entradas y *m* salidas y funciona de manera que cada combinación de valores en las entradas pone a 1 una y sólo una de las salidas, permaneciendo las restantes salidas al valor cero (0). Es decir, un decodificador realiza la operación inversa de un codificador. En realidad, lo importante en la actuación de un decodificador es que diferencia una y sólo una de sus salidas de entre todas las demás, poniéndola al valor 1 mientras las restantes permanecen al valor 0, o también poniéndola al valor 0 mientras las otras salidas están al valor 1.

Dado que todos los decodificadores se diseñan utilizando el mismo procedimiento, vamos a considerar únicamente los decodificadores binarios. Un decodificador binario tiene *n* entradas y 2^n salidas, que podemos considerar numeradas desde 0 hasta 2^n-1, de manera que cada combinación de entradas excita (pone a 1 si las demás están a 0, o pone a 0 si las demás están al valor 1) la salida correspondiente a su valor en binario. El decodificador binario de 1 entre 4 tiene dos entradas y cuatro salidas; su funcionamiento queda descrito mediante la tabla verdad de la siguiente figura.

x_1	x_0	z_3	z_2	z_1	z_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

Como se ve en esta tabla verdad, las cuatro funciones de salida son los cuatro minterms de dos variables, por lo que este decodificador se puede construir con cuatro puertas Y (una por cada salida), tal como se representa en la siguiente figura.





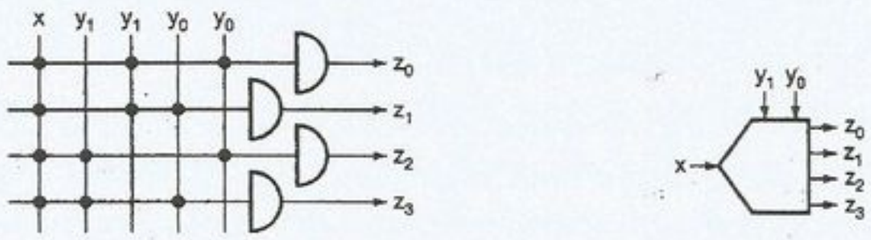
Cualquier decodificador binario con *n* entradas y 2^n salidas se puede construir con 2^n puertas Y.

Un demultiplexor es un dispositivo combinacional con una entrada de datos, *n* entradas de control, y 2^n salidas de datos. Funciona de manera que en un instante cualquiera, la entrada se conecta a una y sólo una de las salidas, determinada dicha salida por la combinación de señales de control: concretamente se selecciona la salida z_i tal que las entradas de control tienen el valor *i* en binario. Los demultiplexores, según esta descripción, son dispositivos para la conducción de la información, es decir, la

entrada no sufre modificación alguna y con el demultiplexor únicamente se controla el camino que en cada caso ha de seguir la señal aplicada a la entrada. Por ejemplo, un demultiplexor de 1 a 4 tiene una entrada de datos, dos entradas de control, y cuatro salidas de datos; su funcionamiento queda descrito mediante la siguiente tabla.

y_1	y_0	z_3	z_2	z_1	z_0
0	0	0	0	0	x
0	1	0	0	x	0
1	1	0	x	0	0
1	1	x	0	0	0

Es inmediato que si se le añade una nueva entrada x al decodificador anterior, de manera que esta entrada adicional lo es de todas las puertas Y, y las entradas x_1 y x_0 pasan a ser las entradas de control y_1 e y_0, tal como se representa en la siguiente figura, el decodificador se convierte en un demultiplexor, es decir, el mismo circuito mostrado en la figura puede actuar como un demultiplexor de 1 a 4, o como un decodificador con cuatro salidas (para esto último basta hacer x=l).

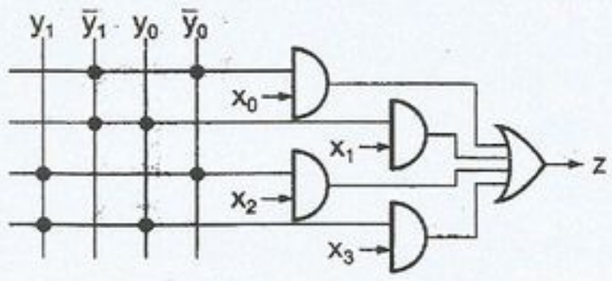


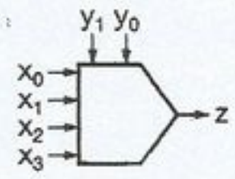
5.5 Multiplexores.

Un multiplexor es un sistema combinacional para la conducción de la información, con n entradas de control, 2^n entradas de datos, y una salida. Funciona de manera que en un instante cualquiera, una y sólo una de las entradas de datos se conecta a la salida, determinándose mediante las entradas de control qué entrada de datos es la que en cada instante se conecta a la salida. Es decir, un multiplexor realiza la operación inversa a la de un demultiplexor. Por ejemplo, un multiplexor de 4 a 1 tiene cuatro entradas de datos, tiene dos entradas de control y una salida; su funcionamiento queda descrito mediante la siguiente tabla.

y_1	y_0	z_0
0	0	x_0
0	1	x_1
1	0	x_2
1	1	x_3

Es fácil comprobar que el siguiente circuito es un multiplexor de 4 a 1.



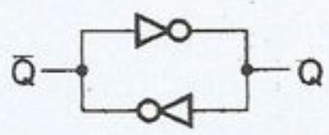


6 Sistemas secuenciales.

Un sistema es secuencial cuando las salidas en un instante cualquiera están determinadas por las entradas en ese instante y por la "historia" del sistema. Es decir, un sistema secuencial puede producir salidas diferentes para las mismas combinaciones de entradas, si estas se aplican en instantes diferentes. La "historia" del sistema está determinada por el estado inicial en el que empieza a funcionar el sistema y por las entradas que ha recibido; esta historia se cuantifica, se concreta, en diferentes estados, de manera que, en términos generales, el sistema evoluciona pasando de un estado a otro, según sean las entradas en cada instante. En este caso, se dice que la salida en cualquier instante es función de las entradas en ese instante y del estado en el que se encuentre el sistema. Para representar el estado se utilizan variables binarias de estado.

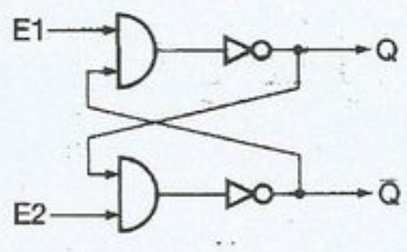
6.1 Elementos de memoria.

El elemento de memoria básico, o almacén de un bit, se obtiene acoplando dos circuitos inversores de manera que la salida de uno sea la entrada del otro, y viceversa.

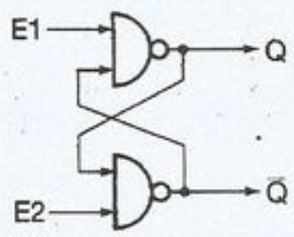


Esta asociación de dos inversores la llamaremos biestable o flip-flop. Actuando a través de los terminales Q o $\overline{Q}$ se puede escribir en el biestable cualquier información que se desee, entendiendo por escribir el situarlo en el estado 1 ó 0; la información escrita en el biestable permanece en él mientras esté adecuadamente conectado a la tensión de alimentación. Esta información almacenada por el biestable puede leerse en los puntos Q y $\overline{Q}$ cuantas veces se desee.

Para facilitar los procesos de lectura y escritura y dar mayor versatilidad a los elementos de memoria, resulta adecuado añadirle dos entradas al biestable de la figura que, junto con las entradas de realimentación, pasen a través de sendas puertas Y antes de actuar como entradas de los inversores. El circuito resultante es:



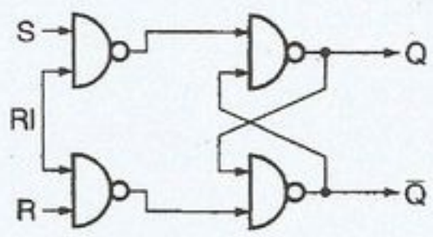
Cada puerta Y seguida de un inversor es una puerta NO-Y o NAND, y el circuito de se representa como:



Cuando $E1$=0 y $E2$=0, resultan Q=1 y $\overline{Q}$=1; en este caso no se cumple que las dos salidas sean complementarias; por esta razón, se considera que esta combinación de entradas debe evitarse. Cuando $E1$=0 y $E2$=1, es Q=1 y $\overline{Q}$=0; esta combinación de entradas, por consiguiente, sirve para poner a 1 el elemento de memoria. Cuando $E1$=1

y *E2*=0, resulta *Q*=0 y $\overline{Q}$=1; con esta combinación de entradas se pone a 0 el elemento de memoria. Cuando se aplican *E1*=1 y *E2*=1, las salidas *Q* y $\overline{Q}$ permanecen al valor que tuviesen previamente (esta combinación de entradas es la combinación de reposo y sirve para mantener inalterada la información que previamente se hubiese almacenado). Por tanto, con *E1* y *E2* se puede escribir cualquier información (un 0 o un 1) y se puede controlar que esa información se mantenga indefinidamente.

Es frecuente que el funcionamiento de los sistemas secuenciales esté controlado por una señal de reloj, *RI*, de manera que las transiciones en las variables de estado tengan lugar únicamente cuando la señal de reloj valga *RI*=1, y cuando sea *RI*=0 el sistema permanezca en el mismo estado. La señal de reloj es una señal periódica que cada *T* segundos toma el valor 1, permaneciendo en este valor durante Δt segundos. Concretando a los elementos de memoria, cuando sea *RI*=0, debe ocurrir que *E1*=*E2*=1, para que el biestable permanezca con su valor anterior. Cuando sea *RI*=1, con las señales externas (que se denominarán *S* y *R*) se situará al elemento de memoria en el estado que se desee. En la siguiente figura se muestra el funcionamiento síncrono que se desea.

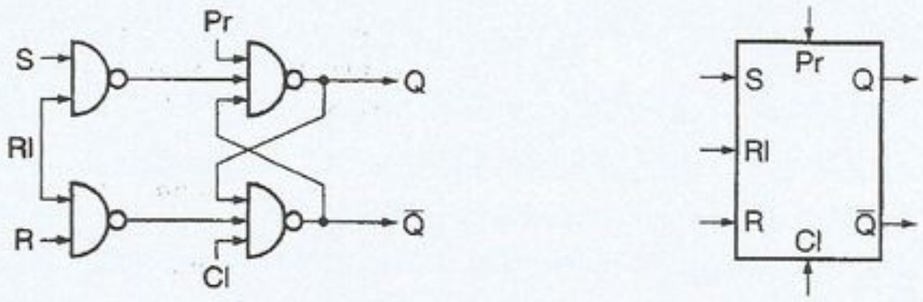


Este circuito presenta dos entradas de control externas, *S* (poner a uno) y *R* (poner a cero), y una entrada de reloj, *RI*. Sólo cuando *RI*=1 se puede modificar el estado de este biestable, que se conoce como biestable SR; cuando *RI*=0, la salida no varía.

En la siguiente tabla se tiene descrito el funcionamiento del biestable SR. En ella se representan dos situaciones de indeterminación mediante un símbolo *.

Q_n	*S*	*R*	Q_{n+1}
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	*
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	*

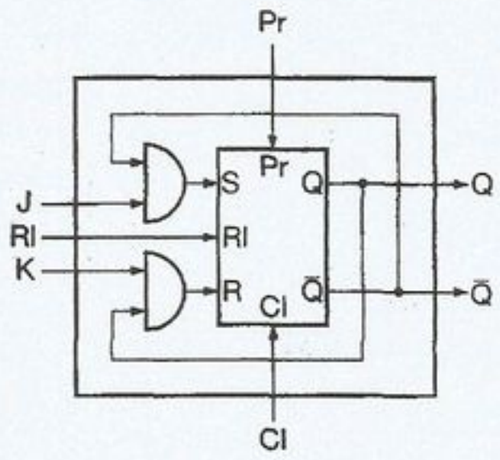
En ocasiones los biestables SR disponen de dos entradas más, conocidas como entradas asíncronas, denotadas respectivamente por *Pr* (fijar a uno) y *Cl* (borrar o poner a cero). Con estas entradas asíncronas se puede poner el biestable al valor cero o al uno con independencia de la señal de reloj. En el circuito de la siguiente figura se ve cómo incluir estas dos entradas asíncronas.

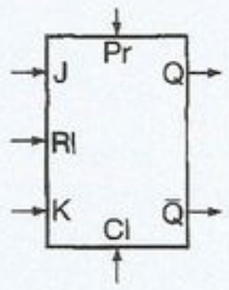


Un claro inconveniente del biestable SR es la indeterminación en las salidas cuando *SR*=11 Este inconveniente se soluciona con el biestable JK, que funciona en todo igual al SR salvo que cuando *JK*=11, el biestable cambia de estado. Es decir, la tabla del biestable JK es:

Q_n	J	K	Q_{n+1}
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

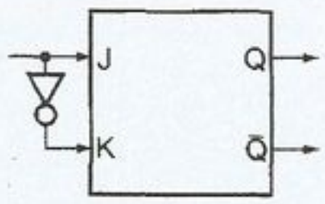
A partir de un biestable SR se puede construir un biestable JK añadiéndole dos puertas Y, tal como se representa en la siguiente figura.





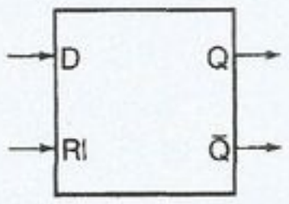
Otro biestable muy utilizado es el tipo D, que tiene una sola entrada, *D*, y funciona de manera que la salida Q_{n+1} en cualquier instante es la entrada D_n en el instante anterior; es decir, el biestable D es el que con más claridad actúa como elemento de almacenamiento, pues cualquier información que se aplique a la entrada se escribe en él. También es claro que el biestable D es un elemento de retardo de un período de reloj.

A partir de un biestable JK (o de un SR) se puede construir un biestable D imponiendo que $K=\overline{J}$ (o $R=\overline{S}$), tal como se representa en la siguiente figura:



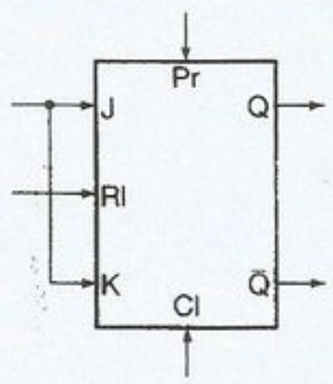
A continuación mostramos la tabla del biestable D y su representación.

D	Q_{n+1}
0	0
1	1

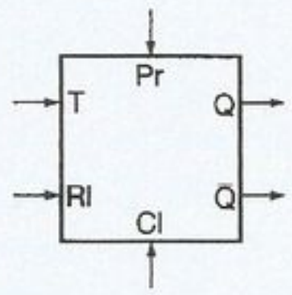


El último elemento de memoria que vamos a considerar es el biestable tipo T. Tiene una sola entrada síncrona, T, y funciona de manera que cuando T=1, el biestable cambia de estado con cada pulso de reloj, y cuando T=0 no cambia de estado.

Un biestable JK se puede transformar en un biestable T si se hace J=K. En la siguiente figura representamos su realización, la tabla del biestable y su representación gráfica.



Q_n	T	Q_{n+1}
0	0	0
0	1	1
1	0	1
1	1	0

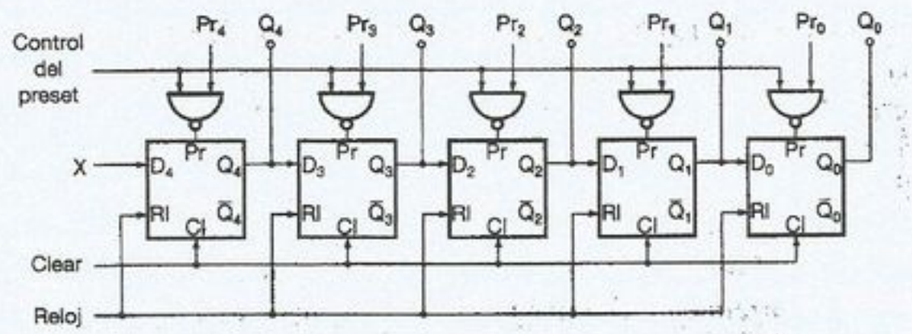


6.2 Registros de desplazamiento.

Un registro de desplazamiento de p bits es un sistema secuencial síncrono con una entrada externa, formado por p biestables conectados en cascada y cuyas salidas en el instante n+1, $Q_{j,n+1}$, cumplen la siguiente relación:

$$Q_{j+1,n+1} = Q_{j,n}$$
$$Q_{1,n+1} = x_n$$

Hay diferentes tipos de registros de desplazamiento, con varios grados de complejidad. En lo que sigue vamos a referirnos a un registro de desplazamiento de complejidad media, que está representado en la siguiente figura.



Este registro de desplazamiento está constituido por cinco biestables D conectados en cascada y con entradas asíncronas. Las entradas *Cl* se activan conjuntamente mientras que las *Pr* pueden actuar por separado, con una señal común de control. Hay una entrada serie, x, y todas las salidas Q están disponibles externamente. Vamos a suponer que por la línea x entra la información serie 01011, que se quiere escribir en el registro. (El convenio que se sigue usualmente en la transmisión serie es que el primer bit en llegar es el menos significativo.) Aplicando a la entrada de reloj cinco pulsos al mismo tiempo que por la entrada x se introducen los cinco bits 01011, en el orden 1, 1, 0, 1, 0, se escribe la información en el registro. En efecto, cuando llegue el primer pulso será x=1, de manera que ésta información se escribirá en el biestable 4, resultando $Q_{4,1}$=1. En t=2 es de nuevo x=1 (segundo bit a escribir), y la entrada del

biestable 3 es también 1, resultando $Q_{4,2}$=11 y $Q_{3,2}$=1. Y así sucesivamente, hasta el quinto pulso de reloj, en el que quedarán $Q_{4,5}$=0, $Q_{3,5}$=1, $Q_{2,5}$=0, $Q_{1,5}$=1, $Q_{0,5}$=1. Es decir, cada vez que llega un pulso, la información almacenada en el registro se desplaza una posición hacia la derecha; por ello este dispositivo se denomina registro de desplazamiento hacia la derecha. Una vez escrita esta información serie, si se quiere que permanezca en el registro, hay que detener la llegada de pulsos de reloj.

La información contenida en el registro es totalmente accesible en cualquier instante, pues todas las salidas son accesibles externamente; esto quiere decir que la información contenida en el registro, que se había escrito en serie, se puede leer en paralelo. En otras palabras, este registro de desplazamiento transforma una información serie en información en paralelo: pasa de un código temporal (cada bit aparece en un instante diferente) a un código espacial (cada bit ocupa una posición diferente).

La información almacenada en el registro también se puede leer en serie, a través de la salida Q_0. Para esto, basta introducir cinco pulsos por la entrada de reloj. Esta operación de lectura en serie se podría simultanear con la escritura en serie. Si la señal de reloj no se detiene, el registro de desplazamiento con entrada serie y salida serie actúa como un retardo de n pulsos de reloj.

Utilizando las entradas asíncronas, también se puede escribir en paralelo cualquier información que se desee. Esta operación de escritura se realiza en dos fases: en la primera, con la entrada de *Cl*, se ponen todos los biestables a cero; en la segunda, con las entradas de *Pr*, que se activan individualmente, se ponen a 1 aquellos biestables que deban almacenar este valor. Esta información escrita en paralelo puede, a continuación, leerse en serie o en paralelo.

En resumen, hay cuatro modos de utilización del registro de desplazamiento: entrada serie-salida serie; entrada serie-salida paralelo (conversor serie-paralelo); entrada paralelo-salida serie (conversor paralelo-serie); y entrada paralelo-salida paralelo. En este último modo, el registro actúa como un simple dispositivo de almacenamiento formado por cinco biestables independientes'

6.3 Contadores.

Un contador es un sistema secuencial con una entrada de reloj y unas salidas, tal que las salidas presentan una serie de configuraciones preestablecidas y en un determinado orden, pasando de una configuración a la siguiente cada vez que llega un pulso de reloj.

Un contador binario módulo 2^n presenta a la salida, de forma cíclica, todas las configuraciones de n variables, en orden creciente si se trata de un contador ascendente, o en orden decreciente si se trata de un contador descendente. Este dispositivo cuenta en binario, módulo n, el número de pulsos que llegan; de ahí la denominación. Los pulsos de entrada no han de ser periódicos necesariamente, aunque actúan en el contador a través de la entrada de reloj.

A continuación vamos a diseñar un contador binario ascendente módulo 8. Dado que hay ocho configuraciones diferentes de salidas, el sistema tiene ocho estados, A, B, C, D, E, F, G, y H, de manera que en el estado A la salida debe ser 000, en B, 001, y así sucesivamente. Para codificar estos ocho estados son necesarias tres variables de estado. Se necesitan, por tanto, tres biestables para guardar las variables de estado. Para simplificar el diseño, vamos a considerar que las variables de estado, y_2, y_1 e y_0, son al mismo tiempo las salidas, de manera que el estado A se representa por $y_2y_1y_0$=000, el B

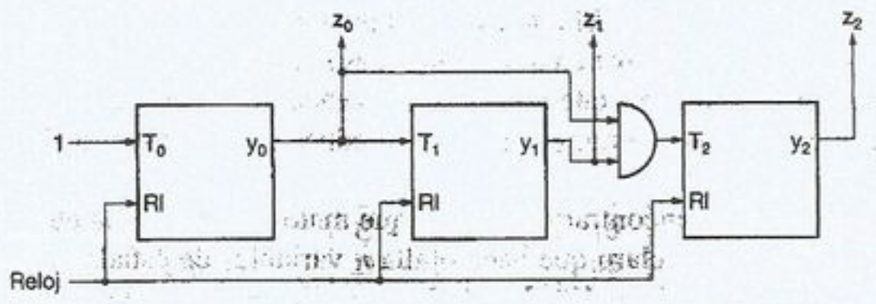
por $y_2y_1y_0$=001, y así sucesivamente. Para diseñar este contador vamos a utilizar biestables tipo T (se puede utilizar cualquier tipo de biestable).

Si el contador está en el estado 000 (es decir, sus salidas son 000), cuando llegue un pulso por la entrada de reloj, el sistema tiene que pasar al estado 001. Es decir, y_2 e y_1 no han de cambiar, e y_0 ha de pasar de 0 a 1 (ha de cambiar). Repitiendo este mismo razonamiento para las diferentes filas obtenemos las funciones T_2, T_1 y T_0 dadas en la siguiente figura.

E_n			E_{n+1}					
y_2	y_1	y_0	y_2	y_1	y_0	T_2	T_1	T_0
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	1
0	1	0	0	1	1	0	0	1
0	1	1	1	0	0	1	1	1
1	0	0	1	0	1	0	0	1
1	0	1	1	1	0	0	1	1
1	1	0	1	1	1	0	0	1
1	1	1	0	0	0	1	1	1

$T_2 = y_1 y_0$
$T_1 = y_0$
$T_0 = 1$

Utilizando este resultado obtenemos el siguiente circuito:



6.4 Diseño de sistemas secuenciales.

Partiendo de una especificación en la que se indica el comportamiento deseado para el sistema y se fijan los elementos a utilizar en el diseño, la primera etapa del mismo consiste en obtener un diagrama de estados que responda a esa especificación. Consiste en una representación gráfica que indica la secuencia de los estados que se presentan en un circuito secuencial, teniendo en cuenta las entradas y salidas. Se forma con círculos y líneas. Los círculos representan los estados del circuito secuencial y cada uno de ellos contiene un número que identifica su estado. Las líneas indican las transiciones entre estados y se marcan con dos números separados por un (/), estos dos números corresponden a la entrada y salida presentes antes de la transición.

A continuación se construye la tabla de estados que contiene la secuencia de los estados de entradas, estados internos y salidas del sistema, considerando todas las posibles combinaciones de estados actuales y entradas. Las tablas de estado por lo general se dividen en tres partes: estados actuales, estados siguientes y salidas.

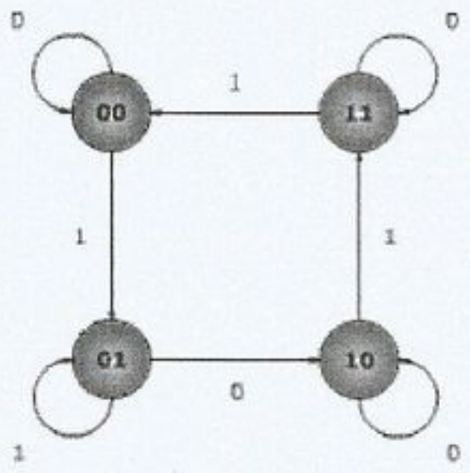
Una vez construida la tabla de estados, la siguiente etapa es la minimización del número de estados. El interés en minimizar reside en que lo esperable es que una tabla de estados más simple se traduzca en un circuito más simple. Hay procedimientos sistemáticos para cubrir esta tarea, procedimientos que no vamos a considerar en esta breve incursión en los sistemas digitales. Como resultado de la minimización se produce otra tabla de estados, usualmente con menos estados que la original.

A continuación, se trata de encontrar el circuito que materializa la tabla de estados. Si en ésta hay m estados, siendo $m \leq 2^n$, hacen falta n variables de estado para codificarlos. La siguiente etapa, conocida como asignación de estados, consiste en asignar a cada estado una combinación específica de las variables de estado. Diferentes asignaciones pueden llevar a circuitos finales diferentes, particularmente con costes diferentes. Hay procedimientos orientativos de cómo hacer la asignación de estados, que no siempre generan indicaciones suficientes como para decidir la asignación; en cualquier caso, estos procedimientos tampoco vamos a considerarlos.

Una vez asignados los estados y decidido el tipo de biestables a utilizar, lo siguiente es obtener las excitaciones de los biestables y generar las salidas, con lo que se tienen todos los datos necesarios para sintetizar el circuito.

6.4.1 Ejemplo 1. Diseño de circuitos secuenciales con biestables JK.

Supongamos, a modo de ejemplo, que queremos diseñar el circuito secuencial del proceso que se cumple de acuerdo al diagrama de estados de la figura:



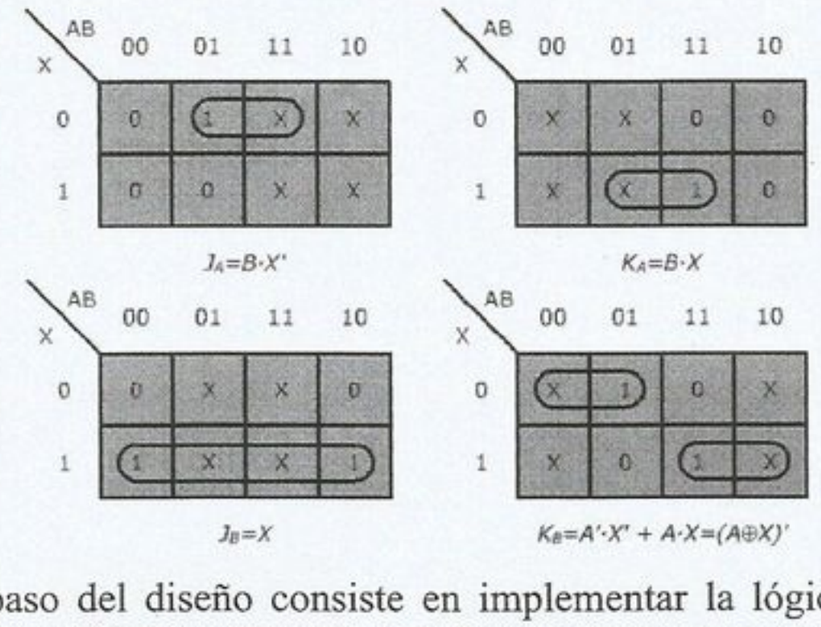
Este proceso tiene cuatro estados, una entrada y no tiene salidas (se pueden considerar como salidas las de los biestables). Para representar los cuatro estados se usarán dos biestables JK identificados como *A* y *B*, y la entrada se identificará como *X*.

La figura corresponde al diagrama de transición. Analizando este diagrama se observa que el estado *10* se mantiene mientras *X=0* y en el momento que *X=1* pasa al estado *11*, después al estado *00* y finalmente al estado *01*, hasta el momento que nuevamente *X=0*, volviendo de esta forma al estado *AB=10*. Adicionalmente observe que los estados *00*, *10* y *11*, se mantienen cuando *X=0* y el estado *01* se mantiene cuando *X=1*.

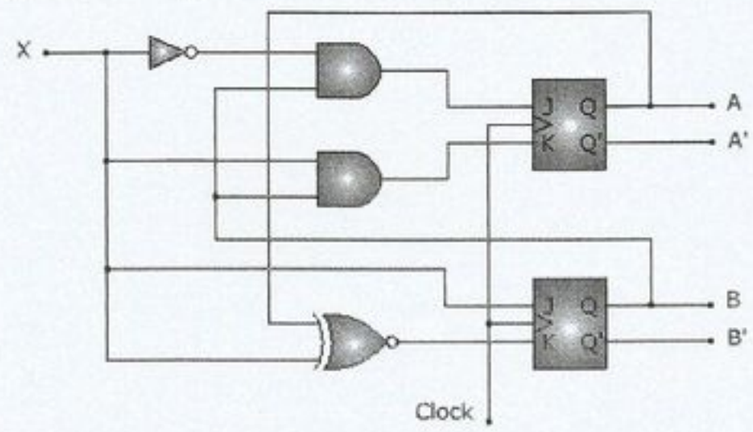
A partir del diagrama de estados y de la tabla de transición del biestable JK, la tabla de estados es:

Entrada	Estado Actual		Estado Siguiente		Excitaciones			
X	A	B	A	B	JA	KA	JB	KB
0	0	0	0	0	0	X	0	X
1	0	0	0	1	0	X	1	X
0	0	1	1	0	1	X	X	1
1	0	1	0	1	0	X	X	0
0	1	0	1	0	X	0	0	X
1	1	0	1	1	X	0	1	X
0	1	1	1	1	X	0	X	0
1	1	1	0	0	X	1	X	1

A continuación se obtienen las funciones lógicas para las entradas de los biestables *(J_A, K_A, J_B* y *K_B)* y el objetivo es deducir la lógica combinatoria de estado siguiente, mediante el uso de Mapas de Karnaugh.

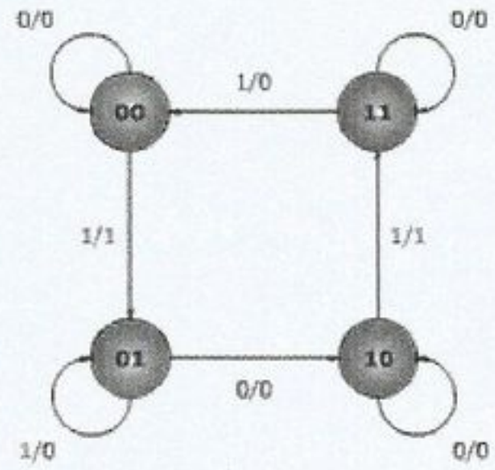


El último paso del diseño consiste en implementar la lógica combinacional a partir de las ecuaciones lógicas obtenidas en el paso anterior para las entradas *J* y *K* de los biestables. En la siguiente figura se muestra el diseño final del circuito lógico.



6.4.2 Ejemplo 2. Diseño de circuitos secuenciales con biestables D.

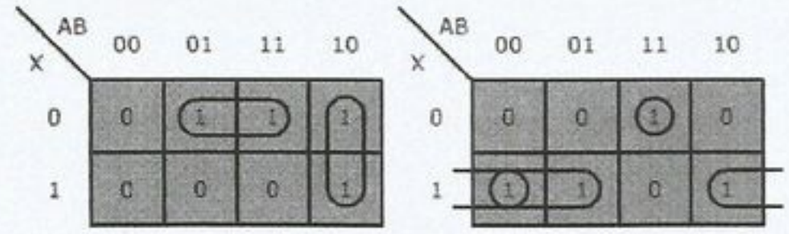
En esta ocasión queremos diseñar el circuito secuencial del proceso que se cumple de acuerdo al diagrama de estados de la figura (es la misma del ejemplo anterior, pero con una salida):

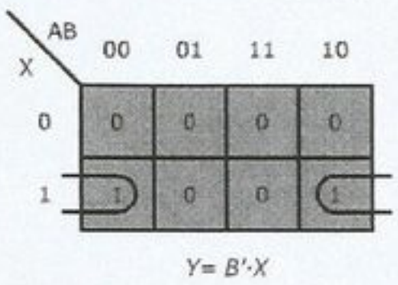


Este proceso tiene cuatro estados de dos bits (*AB*), una entrada (*X*) y una salida (*Y*). Para representar los cuatro estados se usarán dos biestables D identificados como *A* y *B*. En cuanto a la tabla de estados, esta queda como:

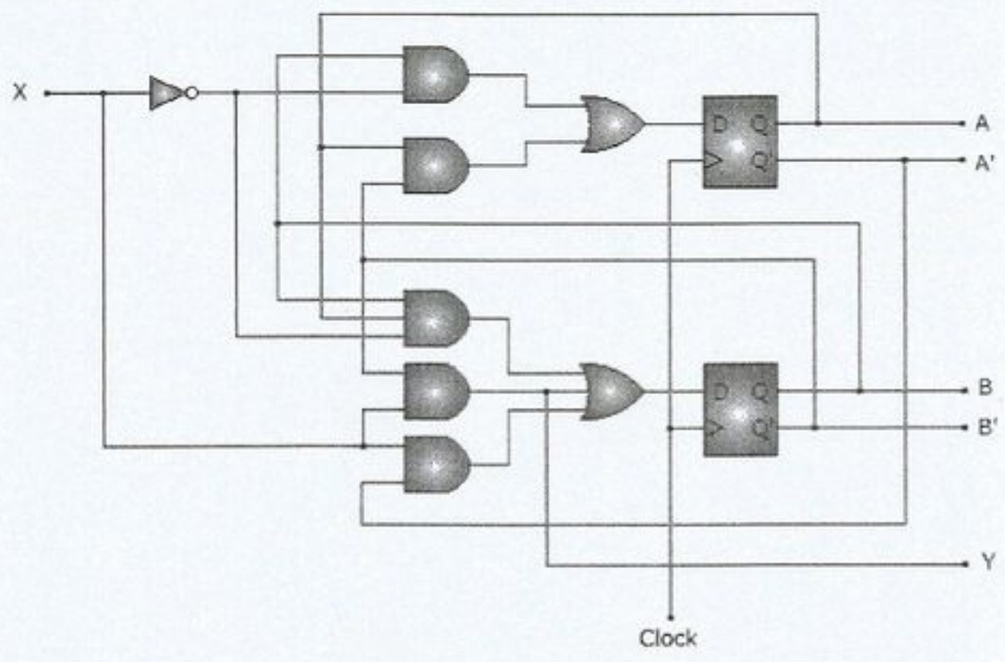
Entrada	Estado actual		Estado siguiente		Salida
X	A	B	A(DA)	B(DB)	Y
0	0	0	0	0	0
1	0	0	0	1	1
0	0	1	1	0	0
1	0	1	0	1	0
0	1	0	1	0	0
1	1	0	1	1	1
0	1	1	1	1	0
1	1	1	0	0	0

A continuación obtenemos las funciones lógicas para las entradas de los biestables (*DA*, *DB*) y la salida (*Y*).





Finalmente, el diseño del circuito es:



6.4.3 Estados no usados.

Durante el diseño de los circuitos secuenciales es conveniente emplear los estados no usados como variables que pueden valer 0 ó 1 indistintamente (estos estados se identifican con una *X* en los mapas de Karnaugh). Esto implica una reducción en las expresiones lógicas y por consiguiente en el tamaño del circuito, que en otros términos representará obviamente un menor tiempo de desarrollo y costo de implementación.

7 Conclusiones.

En este tema hemos dado una visión general, pero al mismo tiempo suficientemente rigurosa, del diseño de sistemas digitales a nivel de puertas lógicas, enseñando a diseñar sistemas sencillos, tanto combinacionales como secuenciales.