Teoria de conmutación de circuitos

1. Algebra de Boole

Un álgebra de Boole es una estructura algebraica con los siguientes elementos:

(B, +, ·, ¬, 0, 1)

Siendo B, el álgebra en estudio; +, · y ¬, las operaciones permitidas en el álgebra; y 0 y 1 los únicos elementos permisibles dentro del álgebra.

Axiomas

1.	$a + a = a$	$a \cdot a = a$	idempotencia
2.	$a + (b + c) = (a + b) + c$ asociatividad	$a \cdot (b \cdot c) = (a \cdot b) \cdot c$	
3.	$a + b = b + a$	$a \cdot b = b \cdot a$	conmutatividad
4.	$a + a \cdot b = a$	$a \cdot a + b = a$	ley de absorción
5.	$a \cdot (b + c) = a \cdot b + a \cdot c$ distributividad	$a + b \cdot c = (a + b) \cdot (a + c)$	
6.	$a + 0 = a$	$a \cdot 1 = a$	elemento neutro
7.	$a \cdot \neg a = 0$	$a + \neg a = 1$	elemento opuesto

(para todo a, b, c que pertenece a B)

Si la ley cumple las propiedades (1) a (4) es un retículo.
Si la ley cumple la propiedad (5) es un retículo distributivo.
Si la ley cumple las propiedades (6) a (7) es un álgebra de Boole.

Observación: es un conjunto redundante de axiomas, ya que algunos de ellos pueden ser demostrados a partir de los otros.

Por ejemplo:

$$
\begin{aligned}
(a + b) \cdot (a + c) &= (a + b) \cdot a + (a + b) \cdot c \\
&= a \cdot (a + b) + c \cdot (a + b) \\
&= a \cdot a + a \cdot b + c \cdot a + c \cdot b \\
&= a + a \cdot b + a \cdot c + b \cdot c \\
&= a + a \cdot c + b \cdot c \\
&= a + b \cdot c
\end{aligned}
$$

Ejemplos de aplicación de los axiomas

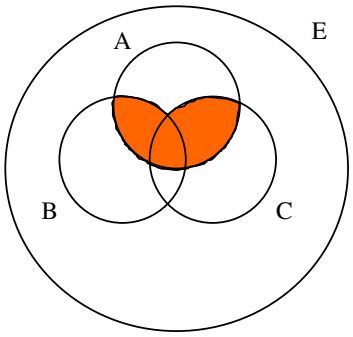
1.1. Comparación con teoría de conjuntos.

Sea: E: un conjunto cualquiera,
B: P(E) = conjunto de los subconjuntos de E,
+: unión,
· : intersección,
¬ : complemento con respecto a E.

Y sea: A: un subconjunto de E y
$\neg A = \{e \in E \mid e \notin A\}$
$0 : \varnothing$
$1 : E$

Véanse algunos ejemplos de axiomas:

$a + 0 = a$ $\quad A \cup \varnothing = A$
$a \cdot 1 = a$ $\quad A \cap E = A$
$a + a \cdot b = a$ $\quad A \cup (A \cap B) = A \quad (A \cap B \subseteq A)$
$a \cdot (b + c) = a \cdot b + a \cdot c$



$A \neg a = 0$ $\quad A \cap \neg A = \varnothing$
$A + \neg a = 1$ $\quad A \cup \neg A = E$
Etc.

1.2 . Algebra de Boole en B_2.

$B_2 = \{0,1\}$

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

OR

x	y	x · y
0	0	0
0	1	0
1	0	0
1	1	1

AND

x	¬ x
0	1
1	0

NOT

En realidad es un caso particular del anterior.

$E = \{e_1, e_2, ..., e_m\}$

a cada subconjunto X de E se asocia un vector $(x_1, x_2, ..., x_m)$ en el cual

$x_i = 1$ si $e_i \in X$
y $x_i = 0$ si $e_i \notin X$

entonces:

$X \cup Y =$ $(x_1 + y_1, x_2 + y_2, ..., x_m + y_m)$ unión
$X \cap Y =$ $(x_1 \cdot y_1, x_2 \cdot y_2, ..., x_m \cdot y_m)$ intersección

$\neg X = (\neg x_1, \neg x_2, ..., \neg x_m)$

complemento

Esta álgebra contiene 2^m elementos (B_2m)

1.3. Funciones booleanas.

$f : (B_2)^n \rightarrow B_2$

$f(x_1, x_2, ..., x_n)$ donde $x_i \in \{0, 1\}$

definición mediante una tabla:

x_1	x_2	x_3	$f(x_1, x_2, x_3)$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

el conjunto de funciones de $(B_2)^n$ en B_2 es también un álgebra de Boole.

Por ejemplo:

Con n = 2 :

x_1	x_2	$g(x_1, x_2)$	$h(x_1, x_2)$	$g \cdot h$	$\neg h$	$g + h$	$\neg g$	$\neg g \cdot \neg h$
0	0	0	1	0	0	1	1	0
0	1	1	1	1	0	1	0	0
1	0	1	1	1	0	1	0	0
1	1	1	0	0	1	1	0	0

número de funciones de $(B_2)^n$ en B_2 : 2^p con $p = 2^n$.

de filas de la tabla

de valores posibles en cada fila

Propiedades importantes

- *propiedad 1:*

  $x \cdot 0 = 0$, $x + 1 = 1$

  demostración:

  $x \cdot 0 = x \cdot (x \cdot \neg x) = (x \cdot x) \cdot \neg x = x \cdot \neg x = 0$
  $x + 1 = x + (x + \neg x) = (x + x) + \neg x = x + \neg x = 1$

- *propiedad 2: (unicidad del complemento)*

  Si $x \cdot y = 0$ y $x + y = 1$, entonces $y = \neg x$

  demostración:

  $$y = y \cdot 1 = y \cdot (x + \neg x) = \underline{y \cdot x} + y \cdot \neg x = \underline{x \cdot \neg x} + y \cdot \neg x = (\underline{x + y}) \cdot \neg x = \neg x$$
  $$= 0 \qquad = 0 \qquad = 1$$

- propiedad 3:

  $\neg \neg x = x \quad , \quad \neg 0 = 1 \quad , \quad \neg 1 = 0$

  demostración:

  $\neg x \cdot x = 0$
  $\neg x + x = 1$

  por unicidad del complemento: $x = \neg \neg x$

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

  por unicidad del complemento: $1 = \neg 0$

- propiedad 4: (leyes de De Morgan)

  (1): $\neg (x + y) = \neg x \cdot \neg y \quad , \quad$ (2): $\neg (x \cdot y) = \neg x + \neg y$

  demostración:
  (1)

  $(x + y) \cdot \neg x \cdot \neg y = x \cdot \neg x \cdot \neg y + y \cdot \neg x \cdot \neg y = 0$

  $x + y + \neg x \cdot \neg y = (x + y + \neg x) \cdot (x + y + \neg y) = 1$
  $= 1 \qquad = 1$

  entonces: $\neg x \cdot \neg y = \neg (x + y)$

  (2): demostración similar.

  <u>observación:</u> mas generalmente: $\neg (x + y + \ldots\ldots + z) = \neg x \cdot \neg y \cdot \ldots\ldots \cdot \neg z$.

Relación de orden

x $\leq y$ ssi $x \cdot \neg y = 0$

Definiciones equivalentes:

x $\leq y$ ssi $\neg x + y = 1$

$\mathbf{x} \leq y$ ssi $x + y = y$

$\mathbf{x} \leq y$ ssi $x \cdot y = x$

se demuestra (por ejemplo) que:

$x + y = y$ es equivalente a $\neg x + y = 1$

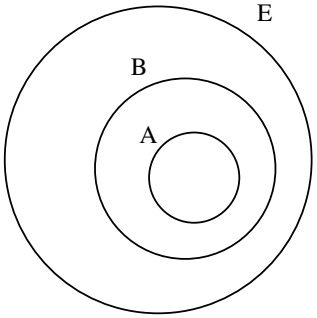
(1) $x + y = y \quad => \quad \neg x + y = \neg x + x + y = 1 + y = 1$

(2) $\neg x + y = 1 \quad => \quad x + y = (x + y) \cdot (\neg x + y) = x \cdot \neg x + y = y$

Ejemplos:

en el caso de P(E) la relación de orden es la inclusión: $\subseteq$

$A \subseteq B$ ssi $A \cap \neg B = \varnothing$



en el caso de B_2: $0 \leq 0$, $0 \leq 1$, $1 \leq 1$
$0 \cdot \neg 0 = 0$ $0 \cdot \neg 1 = 0$ $1 \cdot \neg 1 = 0$

en el caso de funciones $(B_2)^n$ en b_2:

$g \leq h$ ssi $g(x_1, ..., x_n) \leq h(x_1, ..., x_n)$ ssi $g() \cdot \neg h() = 0$ ssi $g \cdot \neg h = 0$

para todo $x_1, ..., x_n$.

ejemplo:

x_1	x_2	g	h	f
0	0	0	0	0
0	1	1	1	0
1	0	1	1	0
1	1	1	0	1

$h \leq g$, $f \leq g$

sin embargo no se cumple que

$h \leq f$ ó $f \leq h$

Operación O Exclusivo en el álgebra de Boole (eXclusive OR – XOR - ⊕)

$x \oplus y = \neg x \cdot y + x \cdot \neg y$

x	y	x ⊕ y
0	0	0
0	1	1
1	0	1
1	1	0

Propiedades:

$x \oplus (y \oplus z) = (x \oplus y) \oplus z$

$x \oplus y = y \oplus x$

$x \cdot (y \oplus z) = x \cdot y \oplus x \cdot z$

$x \oplus 0 = x$

$x \oplus x = 0$

$x \oplus 1 = \neg x$

si $x \oplus y = 0 \Rightarrow x = y$

Expresiones booleanas

Son combinaciones de las variables del álgebra mediante los operadores definidos en la misma.

$f(x_1, x_2, x_3) = x_1 \cdot x_2 + x_1 \cdot x_3 + x_2 \cdot x_3$

x_1	x_2	x_3	$x_1 \cdot x_2$	$x_1 \cdot x_3$	$x_2 \cdot x_3$	f
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	1	1	0	0	1	1
1	0	0	0	0	0	0
1	0	1	0	1	0	1
1	1	0	1	0	0	1
1	1	1	1	1	1	1

Observación: Esta función se llama ‘mayoría’; es igual a uno si por lo menos dos de las tres variables es igual a 1.

2. Biblioteca de Componentes.

Cómo se materializan los sistemas lógicos?

- Circuitos integrados específicos (ASIC – Application Specific Integrated Circuit).

- Circuitos integrados programables por el usuario; por ejemplo: matrices de puertas programables (FPGA – Field Programmable Gate Arrays; PLD – Programmable Logic Devices; PROM – Programmable Read Only Memory).

- Circuitos impresos que permiten ensamblar circuitos integrados (estándar, específicos o programables) y componentes discretos (resistencias, condensadores, conmutadores, etc.).

Sea cual sea la solución elegida, las primeras etapas (diseño lógico) son similares. La diferencia radica en las últimas (materialización física).

El diseño lógico consiste en:

- editar un esquema lógico cuyas primitivas son circuitos predefinidos (puertas, biestables, registros, memorias, etc.),
- depurar el esquema (con un simulador lógico).

Dentro del sistema de desarrollo, las primitivas están almacenadas en bibliotecas (libraries). A cada primitiva corresponde un símbolo (para la edición del esquema) y un modelo de simulación (función, retardos).

Ejemplos de bibliotecas.

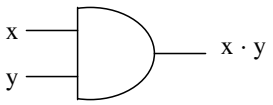
- Biblioteca ECPD07 de ES2: células básicas para diseñar circuitos integrados con la tecnología CMOS de 0,7 micras de ES2 (European Silicon Structures).

- Bilioteca HCMOS de National Semiconductor: Componentes estándar de tecnología HCMos de NS; Por ejemplo el 74HC00 = 4 puertas NAND de 2 entradas.

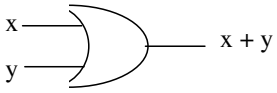
- Biblioteca XC4000 de Xilinx: células básicas para las FPGA de Xilinx.

2.1. Biblioteca Genérica

Puertas lógicas

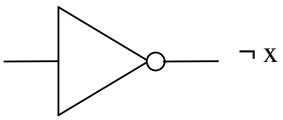


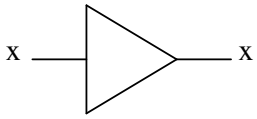
AND2



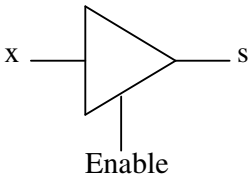
OR2

(mas generalmente: AND3, AND4, OR3, OR4,.....)



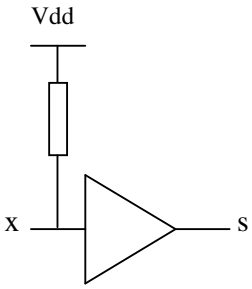


BUF (buffer amplificador)



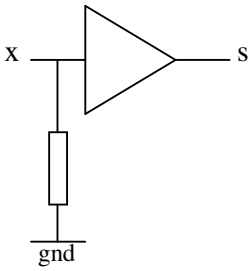
3-STATE BUFFER

Enable	s
0	Z (alta impedancia)
1	x



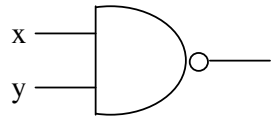
INPUT BUFFER
CON PULL UP

x	s
0	0
1	1
Z	1

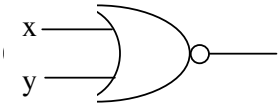


INPUT BUFFER
CON PULL DOWN

x	s
0	0
1	1
Z	0

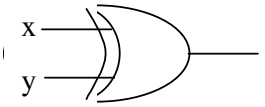


NAND2

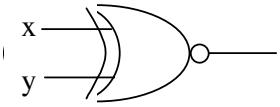


NOR2

(Mas generalmente: NAND3, NAND4, NOR3, NOR4,.....)



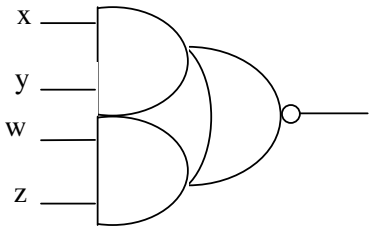
XOR2



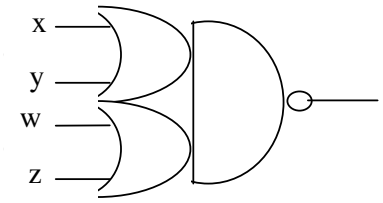
XNOR2

NOTA: ¬ (x ⊕ y) = ¬ x ⊕ y = x ⊕ ¬ y = 1 ⊕ x ⊕ y

(Mas generalmente: XOR3, XOR4, XNOR3, XNOR4,.....)



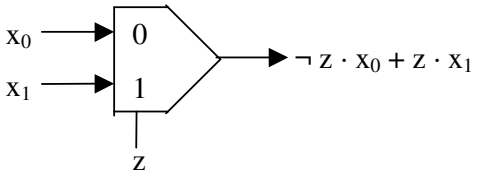
ANDORI



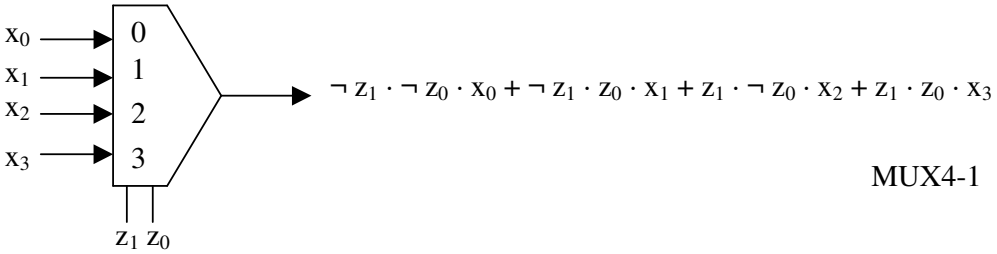
ORANDI

Multiplexores, desmultiplexores, decodificadores

Son dispositivos que poseen dos o mas entradas de señal y una salida. Además tienen una entrada de control que determina, con sus diferentes valores posibles, cual de las entradas se conecta con la salida.

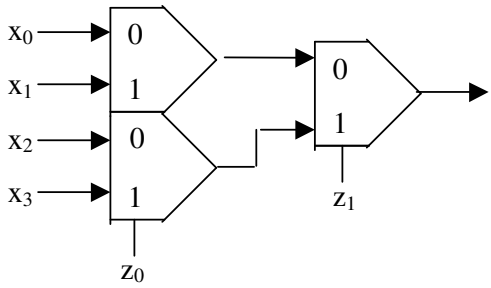


MUX2-1



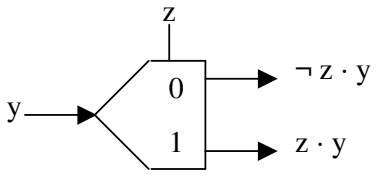
MUX4-1

Es posible diseñar multiplexores de varias entradas utilizando combinaciones de multiplexores de menor tamaño como se ve a continuación:

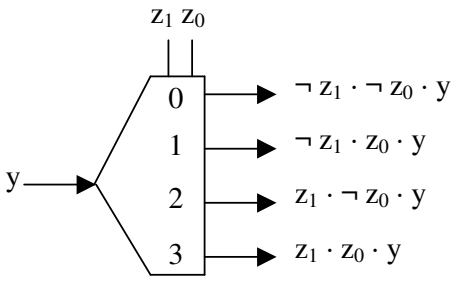


equivalente a MUX4-1

Un desmultiplexor posee una única entrada que es 'conectada' a una de las múltiples salidas dependiendo del valor presente en sus líneas de control.

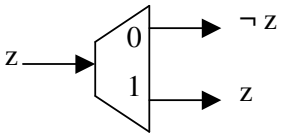


DEMUX1-2



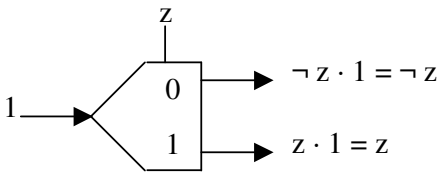
DEMUX1-4

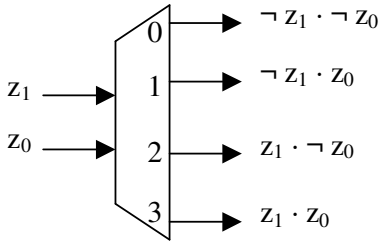
Los decodificadores (también llamados decodificadores de direcciones) ponen una de sus salidas a '1' lógico dependiendo del valor binario presente en su entrada, en tanto que el resto de sus salidas permanecen en '0'.



DECODER1-2

Observar que un decodificador de direcciones es equivalente a un desmultiplexor con su entrada conectada permanentemente a '1'.



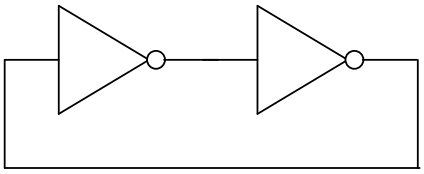


DECODER2-4

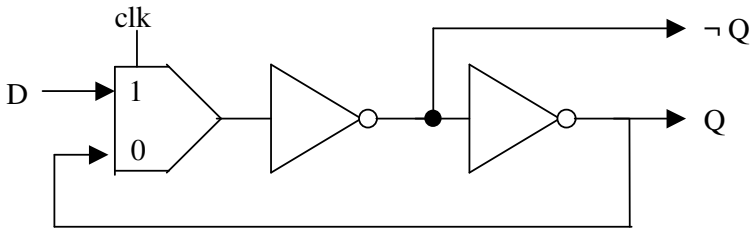
Biestables

Latch D

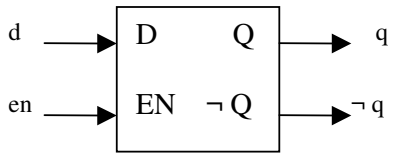
El elemento de memoria más simple es el biestable, que en su implementación más trivial, consta de dos inversores interconectados, como se ve en la figura que sigue:

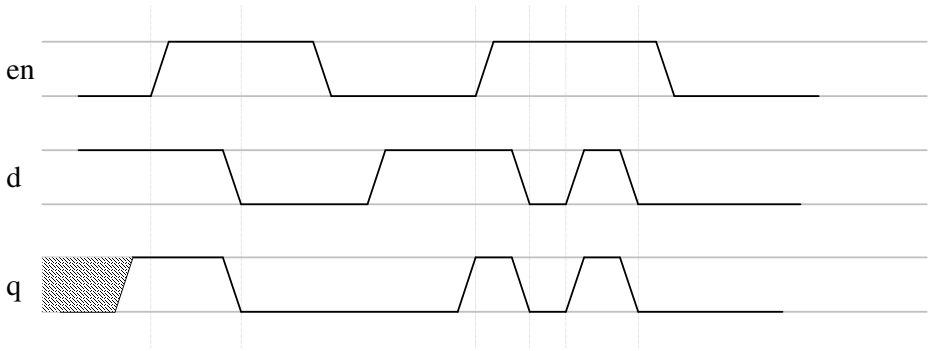


El estado del biestable puede ser controlado a través de puertas de transición; como se ve a continuación se muestra una báscula cerrojo o latch construido de esta forma (se supone que el multiplexor consta de dos puertas de transición).



El biestable explicado anteriormente es el llamado D-Latch cuyo símbolo esquemático y tabla funcional son los siguientes:

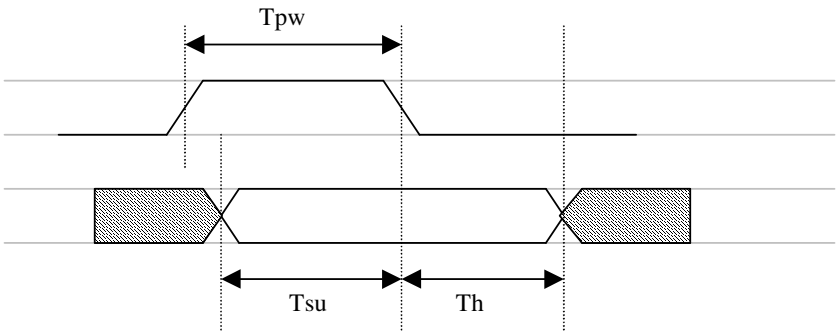




D	EN	Q
0	1	0
1	1	1
-	0	Q

Características importantes de las básculas:

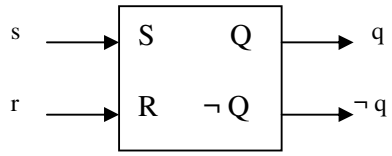
- Tiempo de establecimiento (setup time, Tsu)
- tiempo de retención (hold time, Th)
- anchura de pulso (pulse width Tpw)



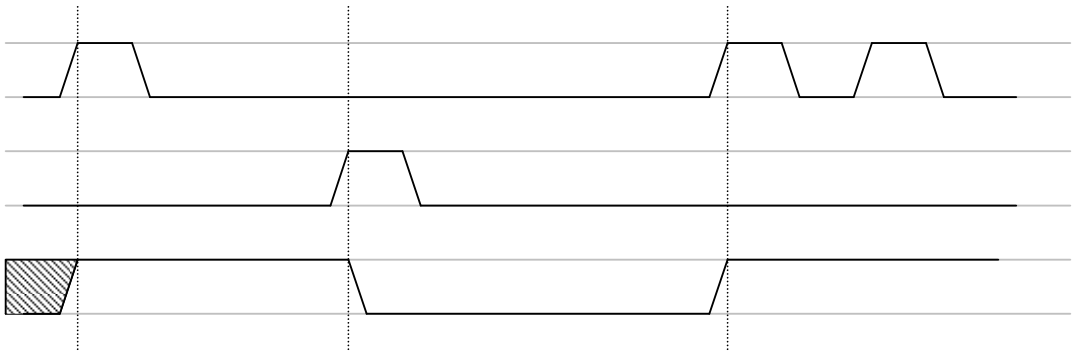
Existen valores mínimos de Tsu, Th y Tpw que deben ser respetados para un óptimo funcionamiento del latch.

Latch SR

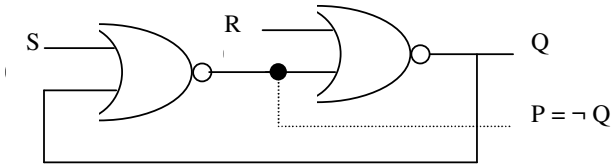
El latch SR es un dispositivo biestable de dos entradas –set y reset- que permiten poner la salida 'Q' del latch explícitamente en '1' o '0' lógico.

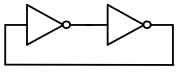


S	R	Q
0	0	Q
0	1	0
1	0	1
1	1	?



El circuito equivalente al latch SR es el siguiente:



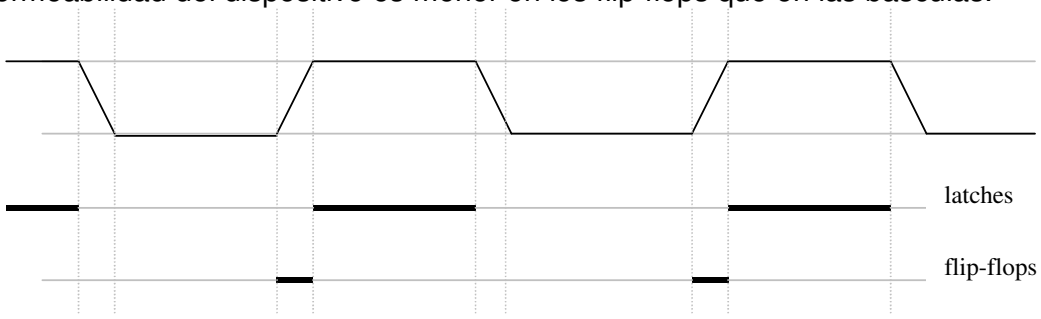
Si R = S = 0 entonces tenemos un biestable básico como el que sigue 

Si S = 1 y R = 0 : P = 0 , Q = 1 y

Si S = 0 y R = 1 : Q = 0 , P = 1.

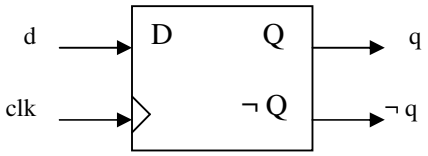
Flip-Flops

Aunque ya se ha visto la primera configuración de un biestable tipo D (el latch), no se hizo énfasis explícitamente en el mecanismo de sincronización.
Particularmente , en el caso de las básculas (latchs) dicha sincronización se materializa a partir de la señal ENABLE. Del estudio de la tabla y del diagrama de señal del latch se ve claramente que sólo cuando la señal ENABLE está en valor lógico '1' los valores presentes en la entrada se transfieren a la salida.
Este tipo de sincronización (o activación) normalmente se denomina por 'estado' ya que durante uno de los estados (para el caso, el '1') la báscula es permeable y permite el paso de todos los valores presentes en la entrada hacia la salida.
Pro el contrario existe otro tipo de biestables que sólo permiten la actualización de la salida (con el valor de la entrada) durante el período de transición de un estado a otro (normalmente la transición de '0' a '1' ó de '1'a '0'). Este tipo de sincronización se denomina por 'flancos' y denominaremos a los biestables activados de este modo 'flip-flop'.
Enseguida se nota que el período de transición entre ambos estados de la señal de sincronización es mas chico que el período de la misma señal, por ende el tiempo de permeabilidad del dispositivo es menor en los flip-flops que en las básculas.

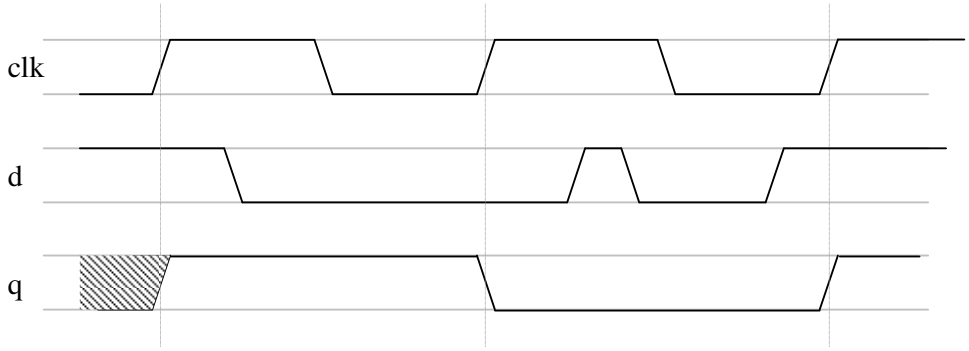


Zona activa de latches y flip-flpos

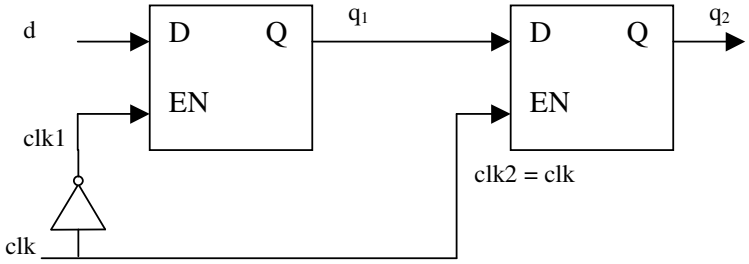
Flip-flop D

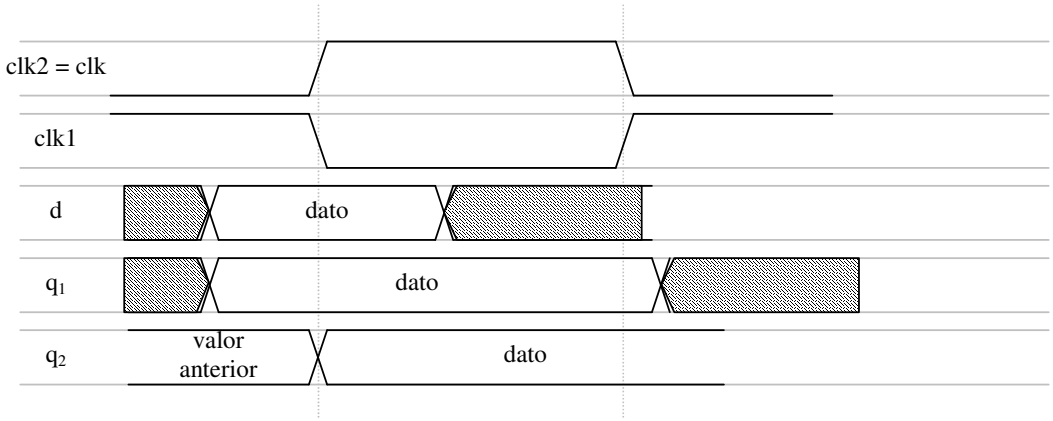


D	clk	Q
0	↑	0
1	↑	1
-	0, 1, ↓	Q



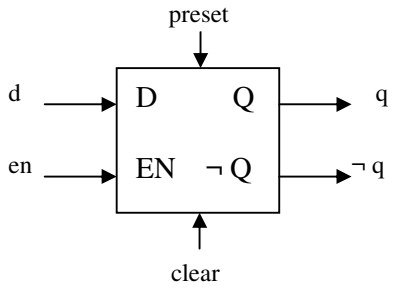
El flip-flop D es equivalente a disponer dos básculas D una a continuación de la otra y sincronizadas de manera que se activen en estados opuestos del reloj.



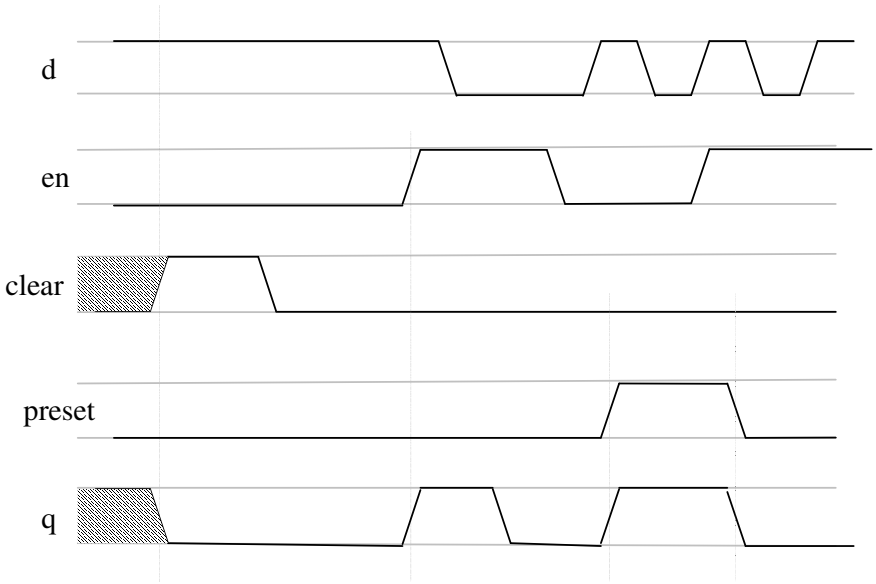


D-Latch with CLEAR · PRESET

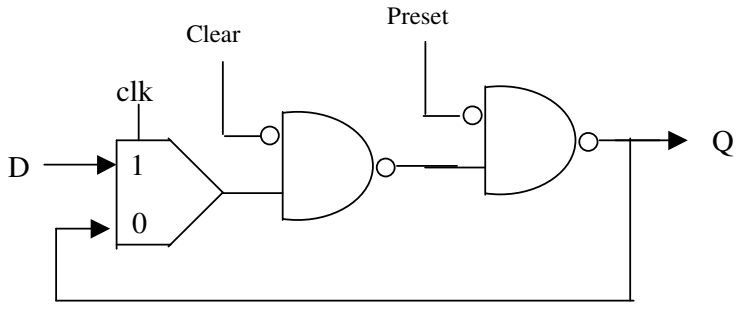
Este dispositivo es funcionalmente igual al latch D pero con dos entradas adicionales de PRESET (puesta a '1') y CLEAR (puesta a '0') asincrónicas respecto de la señal de ENABLE. Esto significa que la activación de cualquiera de estas señales causará la modificación pertinente en la salida aún cuando la señal de ENABLE esta desactivada (a '0'lógico).



D	EN	Preset	Clear	Q
-	-	1	0	1
-	-	0	1	0
0	1	0	0	0
1	1	0	0	1
-	0	0	0	Q



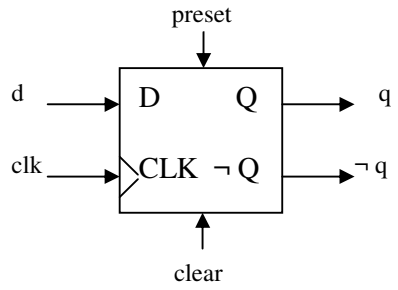
Circuito equivalente:



CLEAR = PRESET = 0 : equivale a un D-LATCH
CLEAR = 1 , PRESET = 0 : Q = 0
Preset = 1 : Q =1

D-flip-flop with CLEAR · PRESET

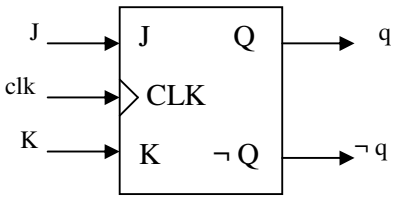
El flip-flop D con preset y clear es funcionalmente equivalente al latch D con la única diferencia que se habilita por flancos. En este tipo de dispositivos las señales de clear y preset son asincrónicas respecto de la señal de reloj.



D	CLK	Preset	Clear	Q
-	-	1	0	1
-	-	0	1	0
0	↑	0	0	0
1	↑	0	0	1
-	0, 1, ↓	0	0	Q

JK Flip-flop

El flip-flop JK podría verse como un tipo particular de flip-flop D en el que las tres señales D, Clear y Preset se reemplazan por el par J y K, de manera que las 4 combinaciones lógicas de estas dos señales las cuatro salidas características del flip-flop: $Q^+ = 0$, $Q^+ = 1$, $Q^+ = Q$, $Q^+ = \neg Q$.
(Q^+ : estado de la salida en el tiempo T (siguiente). Q : estado de la salida en el tiempo T-1 (actual)).



J	K	CLK	Q
1	0	↑	1
0	1	↑	0
1	1	↑	¬ Q
0	0	-	Q
-	-	0, 1, ↓	Q

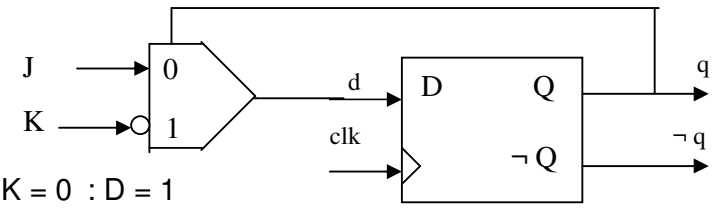
Se puede materializar un flip-flop JK a partir de un flip-flop D y utilizando algo de lógica adicional deducible a partir de la tabla de verdad que define al JK.

$$Q^+ = \neg J \cdot \neg K \cdot Q + J \cdot \neg K \cdot \neg Q + J \cdot \neg K \cdot Q + J \cdot K \cdot \neg Q =$$

$$Q^+ = (\neg J + J) \cdot \neg K \cdot Q + (\neg K + K) \cdot J \cdot \neg Q$$

$$\mathbf{Q^+ = J \cdot \neg Q + \neg K \cdot Q}$$

A partir de la ecuación se ve que el flip-flop D recibe en su entrada un multiplexor cuyas entradas son J y K y la señal de control del mismo es la salida Q del flip-flop D.



J = 1 , K = 0 : D = 1

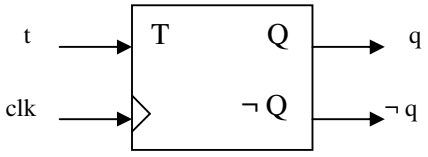
J = 0 , K = 1 : D = 0

J = 0 , K = 0 : si Q = 0 , D = 0
: si Q = 1 , D = 1 => D = Q

J = 1 , K = 1 : si Q = 0 , D = 1
: si Q = 1 , D = 0 => D = ¬ Q

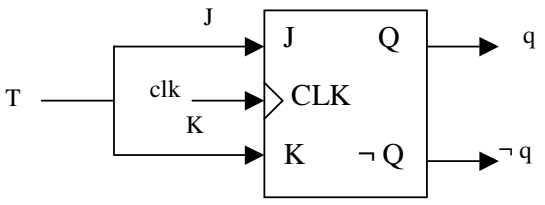
T Flip-flop (Toggle)

El flip-flop T tiene una única entrada (T) que le permite dos alternativas de funcionamiento: Si T está en '0', la salida no varía ($Q^+ = Q$). Si T está en '1' la salida se complementa en casa flanco de habilitación del reloj.



T	clk	Q^+
0	↑	Q
1	↑	¬ Q
-	0, 1, ↓	Q

Es fácil ver que el flip-flop T se puede implementar a partir del un JK uniendo las dos entradas j y k.



2.2. macrocélulas

Al comenzar a hablar de macrocélulas es necesario hacer explícita la diferencia entre los dispositivos que se estudiaron hasta el momento, en las secciones anteriores, y los que se tratarán a continuación.
Normalmente se considera a las puertas (AND , OR, NOT, etc.), multiplexores, decodificadores, flip-flops, etc. como células o componentes predefinidos. Esto es,

los fabricantes provee librerías que ya traen versiones de estos componentes con algunas variantes impuestas por el fabricante. Es normal, entonces, encontrar en una librería una versión de una compuerta NAND de 2, 3 y hasta 4 entradas, pero el fabricante no provee otras versiones de mas entradas.

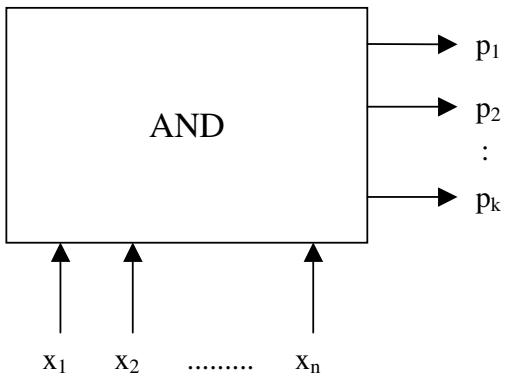
A diferencia de las células predefinidas -que implementan componentes más estándares y de poca complejidad-, las macrocélulas proveen al diseñador de sistemas lógicos de componentes de mayor complejidad, no tan estándares y con parámetros variables que pueden ser parametrizados por el propio diseñador.

De esta manera, por ejemplo, el diseñador puede disponer de un componente que sea un multiplicador de dos entradas de M y N bits respectivamente y una salida de M+N bits, siendo M y N dos parámetros que el mismo diseñador puede configurar a gusto.
Otro ejemplo, igual de interesante, es una macrocélula que materialice una memoria RAM de X celdas de N bits cada una, permitiendo, así, configurar la cantidad y ancho de las palabras a usar.

Planos de puertas

La primera y mas difundida macrocélula es la compuesta por un plano interconectado de compuertas de igual tipo, generalmente AND y OR.

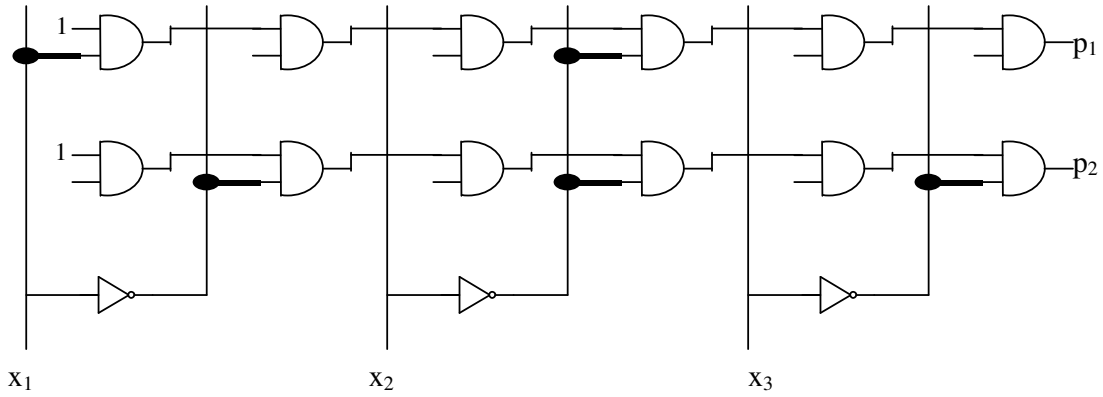
Para el primero de los planos a tratar, el plano AND, la matriz de compuertas puede tener hasta X_n entradas y hasta P_k salidas, donde cada salida p_i es el producto de alguna de las entradas X pudiendo estas estar en forma normal (X_i) o negada ($\neg X_i$).



Veamos un sencillo ejemplo con n = 3 y k = 2:

Queremos que $p_1 = x_1 \cdot \neg x_2$ y

$p_2 = \neg x_1 \cdot \neg x_2 \cdot \neg x_3$



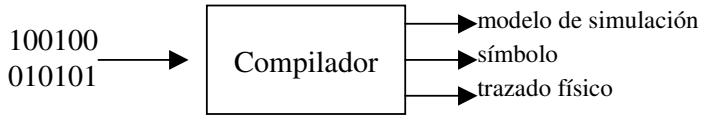
El esqueleto básico del plano AND es una matriz de puertas interconectadas horizontalmente y accedidas por las variables de entrada y sus complementos. De esta manera cada variable (en forma directa y complementada) puede intervenir en la formación de cada término de salida p_i.
La programación del plano AND se realiza a través de una cadena de bits de tamaño 2 * #x *#p.
en donde se representa con un '1' cada posición del plano que debe ser conectada.
Para el caso mostrado anteriormente sería:

1 0 0 1 0 0	->	$x_1 \cdot \neg x_2$
0 1 0 1 0 1	->	$\neg x_1 \cdot \neg x_2 \cdot \neg x_3$

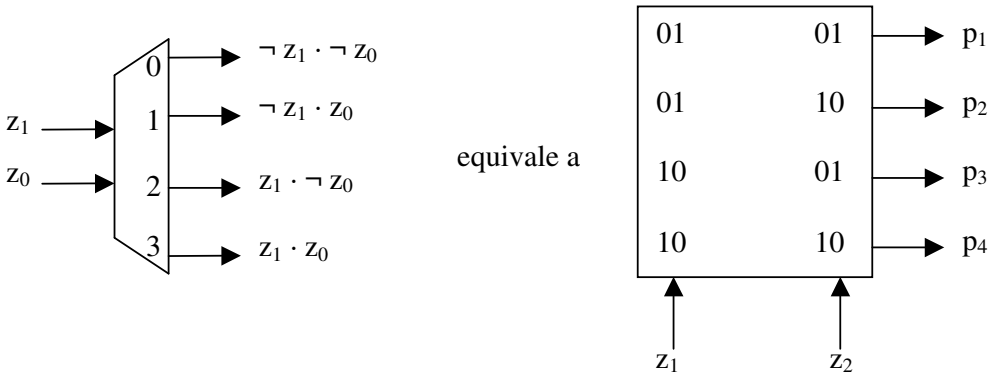
Generalmente los planos AND se materializan de dos formas:

- Como componentes estándar se construyen con fusibles, en el caso de implementación de memorias PROM o con puertas flotantes, en el caso de implementación de EPROM's y E^2PROM's.

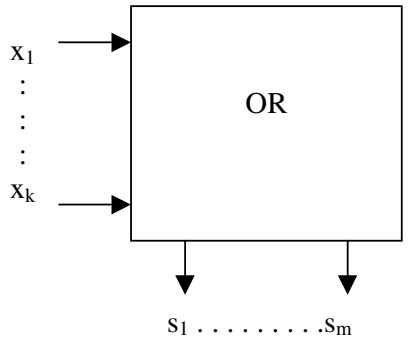
- Como circuitos integrados, cuando se compila el diseño total del sistema (incluyendo el plano) y se 'hornea' en un chip.



Ejemplo Clásico: un decodificador de direcciones se implementaría con un plano AND de la siguiente manera:

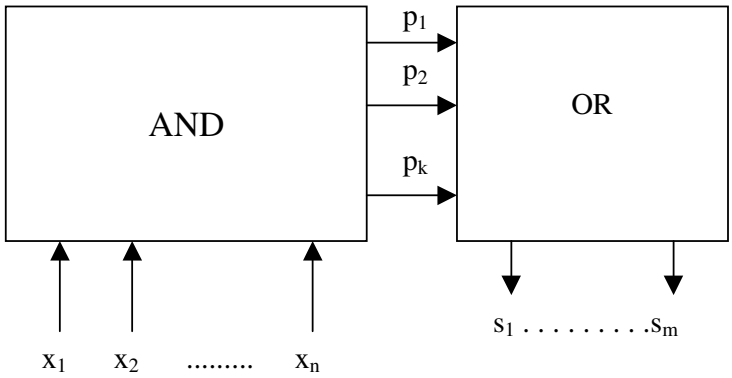


El siguiente plano a tratar es el plano OR, el cual es una matriz de compuertas OR interconectadas con k entradas –x_1, x_2, ..., x_k- y m salidas –s_1, s_2, ..., s_m-, donde cada s_i es la suma (booleana) de algunas variables de entrada, *siempre en forma normal*.



Un plano OR normalmente no tiene mayor utilidad por si solo ya que no representa más que *m* compuertas OR de varias (hasta *k*) entradas, pero si es extremadamente útil en conjunción con un plano AND, ya que juntos permiten sintetizar funciones booleanas completas materializando componentes denominados PLA (Programmable Logic Array's).

La estructura básica de un PLA es la siguiente:



Estos dispositivos permiten sintetizar cualquier conjunto de funciones booleanas expresadas como sumas de productos. Específicamente se pueden realizar hasta *m* funciones, de hasta *k* términos de hasta *n* variables cada uno (directas o negadas).

Veamos un sencillo ejemplo:

Se pretende materializar un sistema que defina las dos funciones y1 e y2 como sigue:

$$y_1 = x_1 \cdot \neg x_2 + x_1 \cdot x_3$$

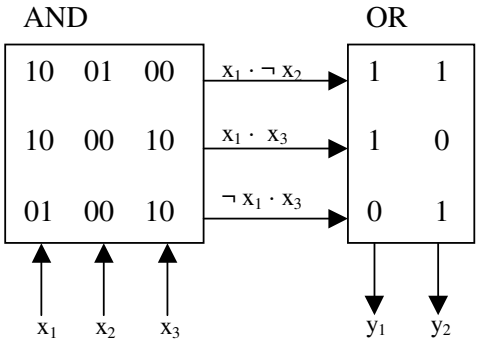
$$y_2 = x_1 \cdot \neg x_2 + \neg x_1 \cdot x_3$$

Solución:

En el plano AND se deben implementar los términos:

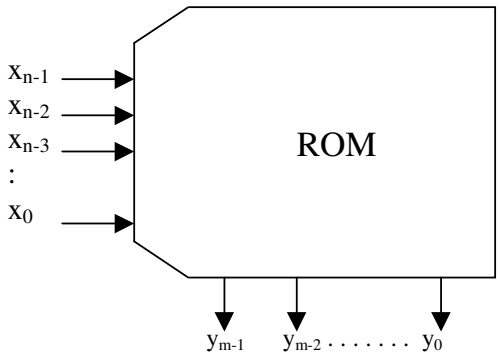
x1 · ¬ x2 ,
x1 · x3 y
¬ x1 · x3

con los cuales se pueden implementar las dos funciones.



Existen programas de síntesis lógica que permiten que a partir de una lista de funciones y1, y2, ..., ym, se generen las secuencias de bits de programación de los correspondientes planos AND y OR.

Memorias pasivas (ROM = Read Only Memory)

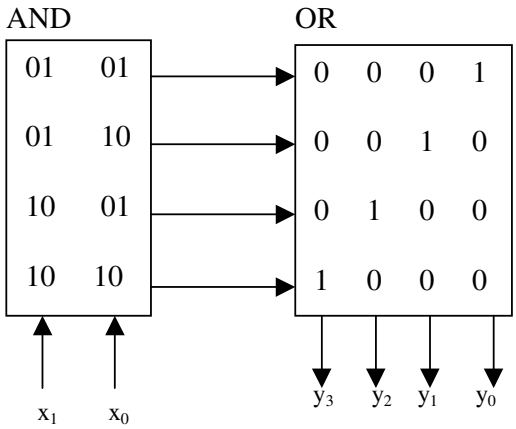


2^n palabras de m bits de ancho

Una memoria pasiva es un dispositivo totalmente cableado que ante cada combinación lineal de valores de los x_i da como resultado un determinado valor en cada salida y_i.

Observar que este tipo de dispositivo es compatible con un plano AND-OR con tantas salidas (y_i) como ancho de la celda de datos, y con *n* señales de entrada (xi).

La figura siguiente muestra un ejemplo de implementación de una memoria ROM pasiva de cuatro celdas (n = 2) de 4 bits cada una (m = 4).



El proceso de síntesis consiste en compilar (en un compilador de circuitos digitales) la secuencia de bits del plano OR, ya que la secuencia de bits del plano AND es deducible puesto que es la secuencia de valores de las señales xi desde todos '0' hasta todos '1'

Por ejemplo, para una ROM de 4 palabras de 8 bits cada una con los valores:

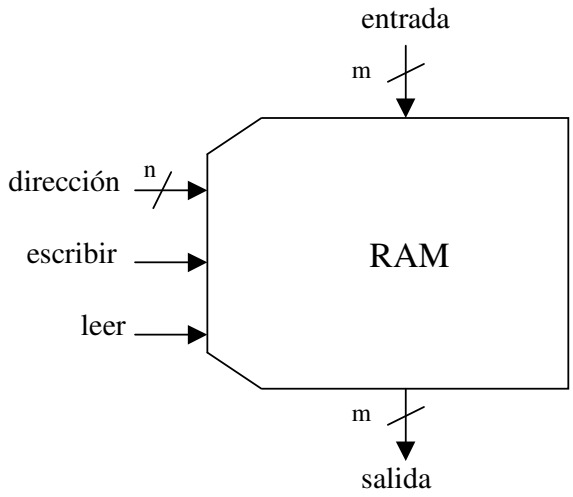
Dirección	Valor
0	AAh
1	FFh
2	03h
3	54h

Plano AND : 01 01
01 10
10 01
10 10

Plano OR : 10101010
11111111
00000011
01010100

Memoria Activa (RAM = Random Access Memory)

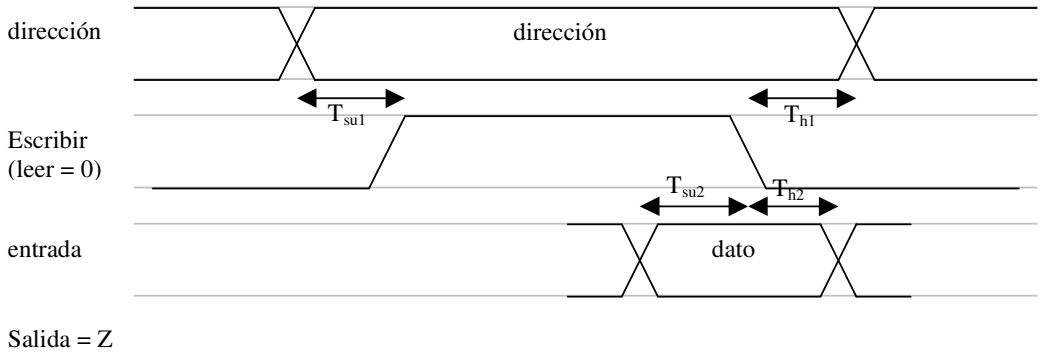
Las memorias activas deben tener la capacidad no sólo de guardar un dato, como las pasivas, sino también la capacidad de poder recibir un dato nuevo para memorizar. Esta nueva capacidad implica que los dispositivos básicos de memorización deben guardar el dato escrito hasta tanto se cambie el mismo o se suspenda el suministro eléctrico del circuito.



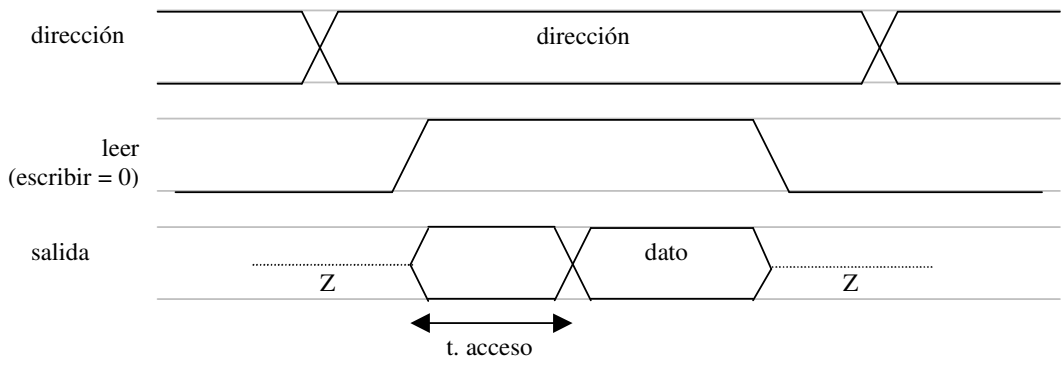
2^n palabras de m bits de ancho

La capacidad de leer o escribir la memoria a voluntad implica la necesidad de un par de señales de habilitación de los elementos internos para recibir nuevos datos o para entregar por la salida los datos almacenados.

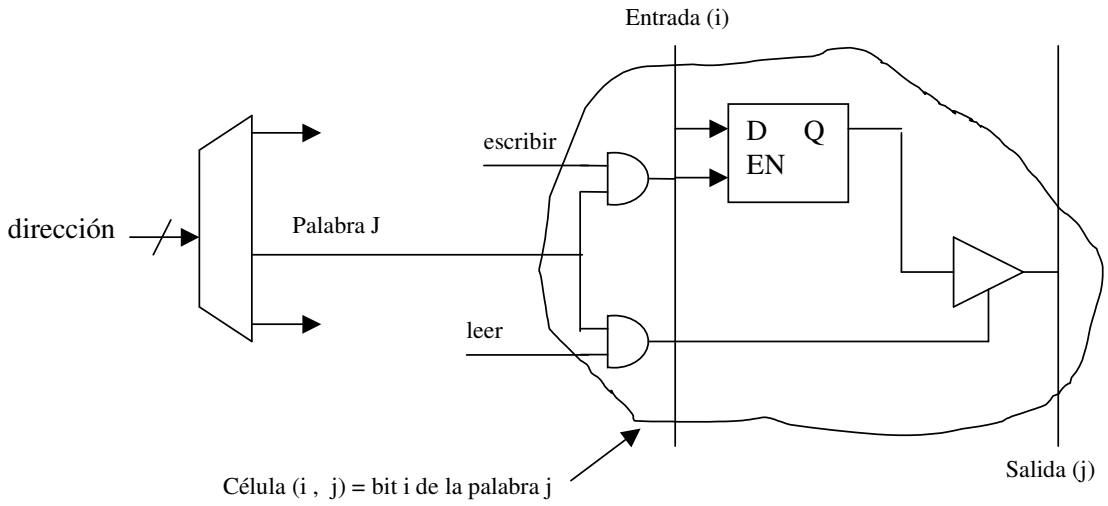
El diagrama de señales del proceso de escritura se muestra a continuación:



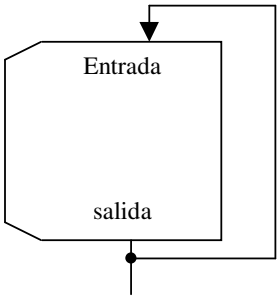
El diagrama de señales del proceso de lectura es como sigue:



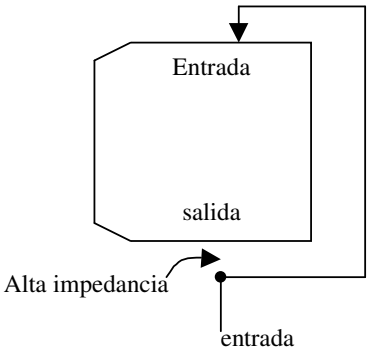
El circuito equivalente a cada unidad de memoria (un bit) es el siguiente:



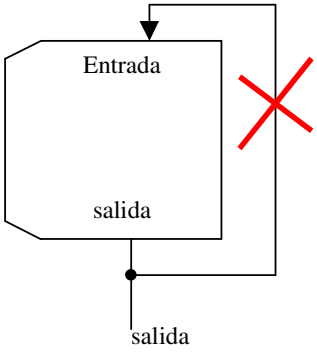
Una modificación interesante de circuito es la de conectar la entrada y la salida de manera que a través de un único canal de datos se puede leer y escribir la memoria.



- Cuando se realiza la escritura la salida está automáticamente en alta impedancia (Z)

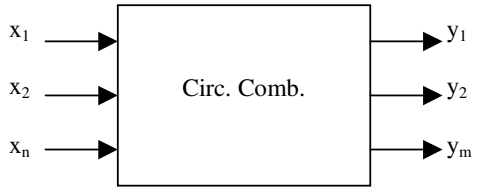


- En el proceso de lectura se supone que el dispositivo externo (por ejemplo la CPU) está en modo recepción



3. Síntesis de Circuitos Combinacionales: Métodos Generales.

En líneas generales, un circuito combinacional es aquel en el cual sus salidas dependen únicamente del valor de las entradas. Dicho de otro modo, el valor de salida de un circuito combinacional en un tiempo dado (t_i) no depende del valor de salida del ciclo anterior de reloj (t_{i-1}), sino (y solamente) del valor de las entradas en el ciclo actual (t_i).



$$Y_i = f_i(x_1, x_2, ..., x_n)$$

Normalmente, este tipo de circuitos se denominan *dispositivos sin memoria*.

Formas de especificar el funcionamiento de los circuitos combinacionales

3.1 Circuito descrito por una tabla.

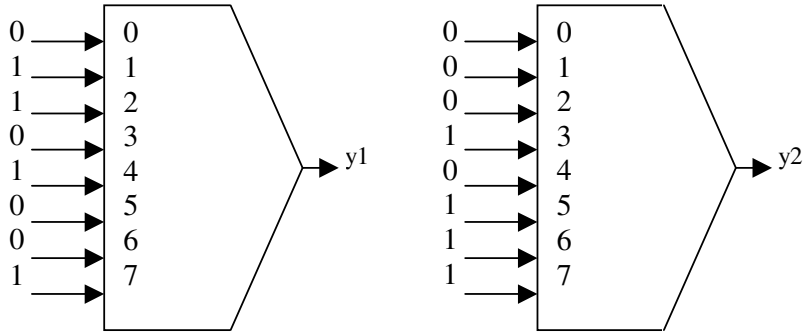
Se utiliza una tabla que describe todas las posibles combinaciones de los valores de entrada y para cada combinación de entradas el valor que adopta cada una de las salidas.

x2	x1	x0	y1	y2
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

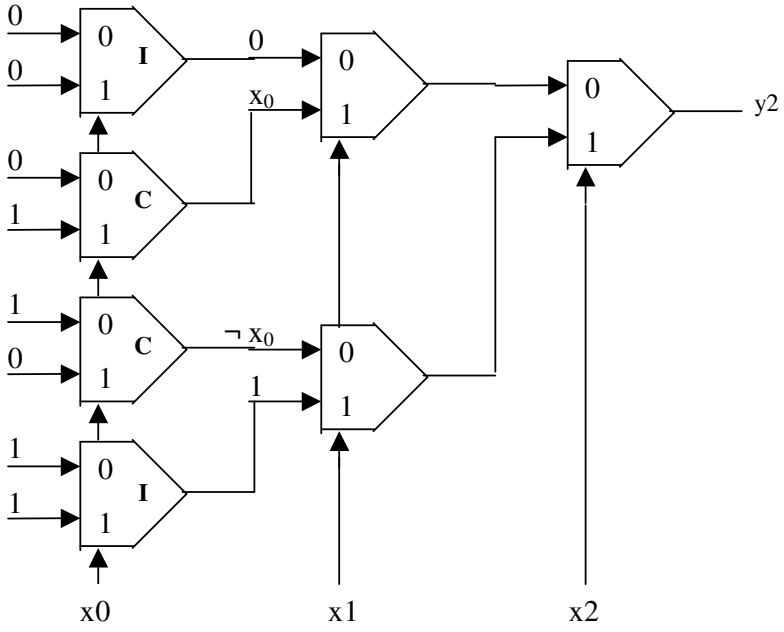
Dada una descripción de un circuito combinacional mediante una tabla el mismo se puede sintetizar en diversas alternativas: mediante multiplexores, mediante memorias ROM, mediante ROM's y multiplexores.

Síntesis con multiplexores

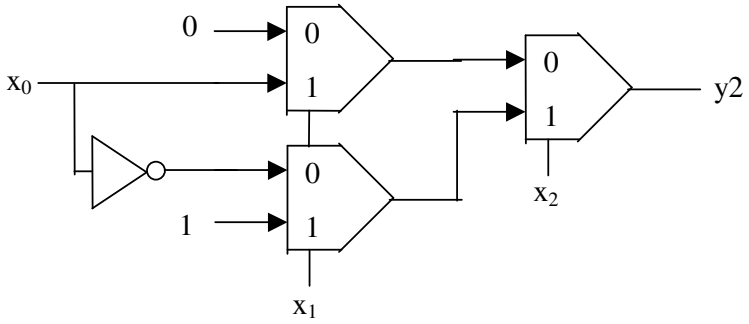
Se emplea un multiplexor con tantas entradas de control como variables de entrada tenga el circuito bajo diseño. Cada una de las entradas del MUX (2^{xn}) se conectan a '0' ó a '1' según corresponda al valor de la salida.



En la práctica, en lugar de usar un MUX grande (no disponibles comercialmente) se implementa el circuito usando varios MUX más chicos.
El ejemplo siguiente muestra como sustituir el MUX8-1 de la función y2 anterior por varios MUX2-1:

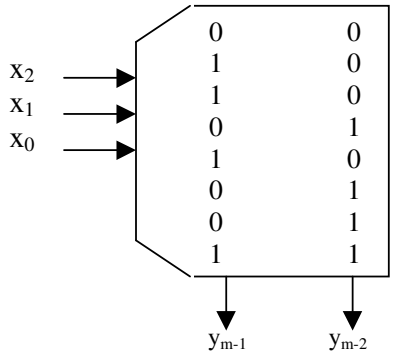


El circuito anterior es representativo de la función que implementa, aunque se observa que existen multiplexores inoperantes, cuyas entradas tienen los mismos valores lógicos (marcados en la figura anterior con una **I**) o pares de multiplexores complementarios (**C**).



Síntesis con una memoria ROM

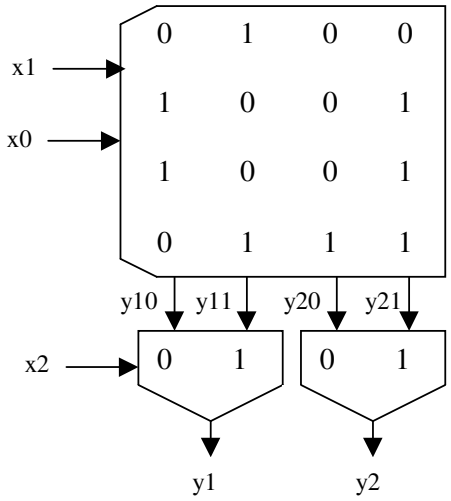
Otra posibilidad es utilizar una memoria ROM con 2^{xn} celdas de y_m bits de longitud cada una.
La ventaja de esta implementación es que cambiando la ROM es posible implementar funciones diferentes.



Síntesis con ROM y multiplexores (doble direccionamiento)

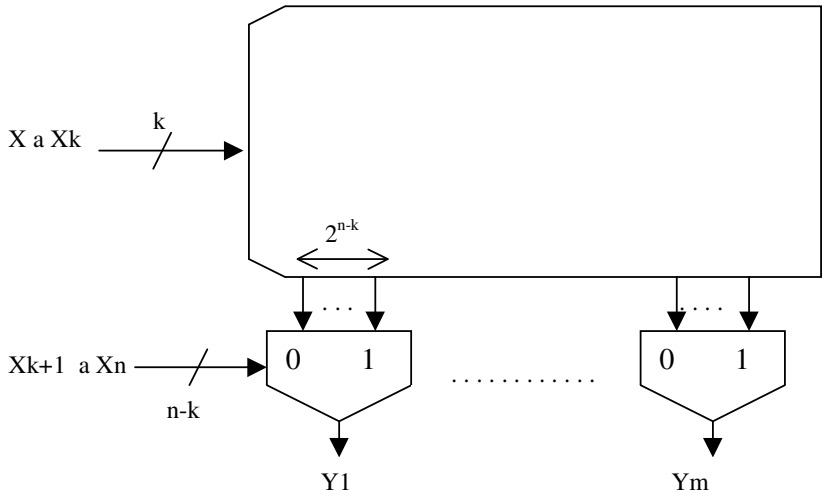
Ésta alternativa híbrida permite implementar circuitos en los que las funciones de salida se dividen en grupos funcionales, de manera que la ROM activa la salida correspondiente de cada función y los multiplexores seleccionan las salidas adecuadas de todas las disponibles.

Para empezar consideremos un ejemplo simple con cuatro funciones de salida divididas en dos grupos de dos funciones cada uno –y_{10}, y_{11}, y_{20}, y_{21}-; y 3 señales de entrada –x_0, x_1, x_2-.



y11 = y1(x2 = 1)
y10 = y1(x2 = 0)
y21 = y2(x2 = 1)
y20 = y2(x2 = 0)

En general:



Tamaño de la Memoria ROM:

2^k filas * m * 2^{n-k} columnas = m * 2^n.

Hipótesis:

- Costo (área) del decodificador de dirección k → 2^k = C1 * (2^k - 1)
  (1 decodificador k→2^k es equivalente a 2^k – 1 decodificadores 1→2).
- Costo de m multiplexores 2^{n-k}→1 = m * C2 * (2^{n-k} - 1)
  (1 multiplexor 2^{n-k} → 1 es equivalente a 2^{n-k} – 1 multiplexores 2→1)

Conclusión:

$$Costo\ total = C1 * (2^k - 1) + m * C2 * (2^{n-k} - 1) + C3 * m * 2^n$$

El problema es hallar el valor óptimo de k que minimiza el espacio y el costo del dispositivo.
Para ello hay que minimizar la función obtenida:

Minimización de $f(k) = C1 * 2^k + m * C2 * 2^{n-k}$

Matemáticamente:

$$d\,2^x / d\,x = d\,e^{x * \ln 2} / d\,x = \ln 2 * e^{x * \ln 2} = \ln 2 * 2^x.$$

Reemplazando en f(k) quedaría:

$$\boldsymbol{d\,f / d\,k = C1 * \ln 2 * 2^k - m * C2 * \ln 2 * 2^{n-k}.}$$

Y se puede ver que $d\,f / d\,k = 0$ si

$$\mathbf{2^k / m * 2^{n-k} = C2 / C1}$$

Es decir:

C2 / C1 = H / W (altura / ancho)

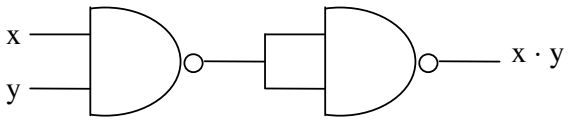
En particular, si C2 (costo de un multiplexor 2-1) = C1 (costo de un decodificador 1-2), el valor óptimo de k corresponde a una memoria cuadrada.

3.2 Circuito descrito por expresiones.

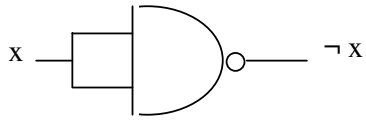
Las expresiones booleanas para generación de circuitos suelen generarse a partir de los operadores booleanos estándares: AND, OR, NOT y XOR, aunque a la hora de materializar los circuitos correspondientes las operaciones se modifiquen (usando los axiomas y propiedades del álgebra de Boole) para lograr circuitos mas rápidos, mas baratos, mas pequeños y de menos consumo.

Por lo general la compuerta que mejores resultados prácticos brinda a la hora de diseñar un circuito es la NAND, así que cualquier alternativa de síntesis abarca algún paso intermedio de conversión del circuito a un equivalente en compuertas NAND.
Otra compuerta de utilidad a la hora de sintetizar es la compuerta NOR.

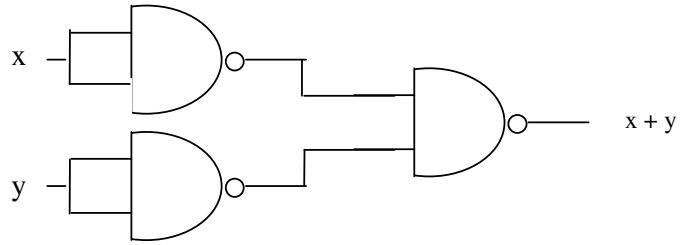
Representación del AND con NAND



Representación del NOT con NAND



Representación del OR con NAND



3.3 generación de una expresión a partir de una tabla.

Hasta ahora se trabajó con expresiones genéricas, esto es, expresiones con variables booleanas vinculadas a través de operadores lógicos, de manera que la interpretación de la expresión se realiza a partir de ciertas reglas de precedencia predeterminada por el álgebra. Así, la operación OR separa términos, siendo cada término una conjunción (AND) de una o más variables (directas o negadas) de las que intervienen en la expresión.

$$X1 + X1 \cdot \neg X2 + \neg X2 \cdot X3 + X4$$

Una expresión se dice canónica si cada uno de los términos hace referencia a todas las variables que intervienen en la expresión, ya sea en forma directa o negada.
Las expresiones canónicas pueden ser *conjuntivas* o *disyuntivas*. Las primeras son sumas de términos conjuntivos (también llamados minterms), mientras que las segundas son productos de términos disyuntivos (maxterms).

Expresiones canónicas conjuntivas

X2	X1	X0	Y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

$$Y = \neg \underbrace{X2 \cdot \neg X1 \cdot X0}_{\text{minterm}} + \neg X2 \cdot X1 \cdot \neg X0 + X2 \cdot \neg X1 \cdot \neg X0$$

Notar que una expresión canónica conjuntiva (o suma de minterms) se genera sumando las filas en las que la función de salida (Y) es 1, y en cada término sumado las variables que en esa fila tienen valor 1 aparecen en forma directa mientras que las que tienen valor 0 aparecen negadas.

Expresiones canónicas disyuntivas

X2	X1	X0	Y
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

$$Y = \underbrace{(X2 + X1 + \neg X0)}_{\text{maxterm}} \cdot (X2 + \neg X1 + X0) \cdot (\neg X2 + \neg X1 + X0)$$

Notar que una expresión canónica disyuntiva (o producto de maxterms) se genera multiplicando las filas en las que la función de salida (Y) es 0, y en cada término las variables que en esa fila tienen valor 0 aparecen en forma directa mientras que las que tienen valor 1 aparecen negadas.

3.4. Simplificación de funciones

Las expresiones surgidas de un proceso deductivos, y en especial las expresiones canónicas contienen a menudo términos y variables redundantes que entorpecen el entendimiento del circuito (y su tamaño y costo) y que pueden ser eliminadas utilizando algún método apropiado de simplificación.
Existen varios métodos más o menos complejos y efectivos, pero nos concentraremos particularmente en dos de ellos. El primero muy práctico por su sencillez y visualización gráfica, y el segundo más apropiado para obtener optimizaciones en expresiones con gran (mas de 5) cantidad de variables distintas.

3.4.1. Método de Karnaugh

Los diagramas de Karnaugh dan lugar a una técnica de tipo gráfico usada para la simplificación de las ecuaciones lógicas, que se basa en disponer las combinaciones posibles de una forma apta para su simplificación.
Las simplificaciones que permiten los diagramas de Karnaugh se basan en la siguiente identidad:

$$A \cdot B \cdot C + A \cdot B \cdot \neg C = A \cdot B \cdot (C + \neg C) = A \cdot B$$

La ecuación anterior indica que si una variable (la C) aparece negada en un término y no negada en otro que tiene el resto de las variables iguales, puede eliminarse por completo. Los diagramas de Karnaugh ayudan mucho a la localización de estas variables que se pueden suprimir.

Disposición de las variables en el diagrama

Por razones obvias relacionadas con el carácter gráfico de este método, el mismo suele ser útil para cálculos que contengan hasta 5 variables, luego de lo cual se torna en complejo y potencialmente erróneo.

Cuando se trabaja con dos variables , estas se pueden disponer en una cuadrícula como se muestra a continuación:

X	Y	X · Y
0	0	0
0	1	0
1	0	0
1	1	1

		¬ X	X
		0	**1**
¬ Y	**0**	0	0
Y	**1**	0	1

Con tres variables, el número de cuadrículas en el diagramas es de $2^3 = 8$, por lo cual, para obtenerlas, se 'desdobla el diagrama realizado para dos variables hacia la derecha y se coloca en las columnas nuevas la nueva variable con valor '1', como se ve en la figura:

Z	X	Y	f
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Diagrama de karnaugh para tres variables:

		¬ Z ¬ X	¬ Z X	Z X	Z ¬ X
		0 0	**0 1**	**1 1**	**1 0**
¬ Y	**0**	0	1	1	0
Y	**1**	1	0	1	1

Para cuatro variables el diagrama correspondiente se obtiene 'desdoblando' hacia abajo el diagrama correspondiente a tres variables.

¬ Z ¬ X	¬ Z X	Z X	Z ¬ X
0 0	**0 1**	**1 1**	**1 0**

¬ W	¬ Y	**00**				
¬ W	Y	**01**				
W	Y	**11**				
W	¬ Y	**10**				

Método de trabajo con diagramas de Karnaugh.

Se trata de encontrar parejas de términos iguales a excepción de una variable, que en uno esté negada y en el otro no. Obsérvese que en todos los diagramas de Karnaugh, al pasar de una cuadrícula a la adyacente siguiendo una fila o una columna (no en diagonal) siempre cambia el estado de una de las variables. Cambia incluso entre la primera y última cuadrícula de cada fila o columna.
Para la simplificación se trata de agrupar cuadrículas adyacentes en las que se cumpla la ecuación, para ir eliminando variables. Las agrupaciones de cuadrículas con valor '1' se denominan lazos y alrededor de ellas se dibuja una línea que los contiene. Cada lazo formará un término en la versión simplificada de la ecuación.

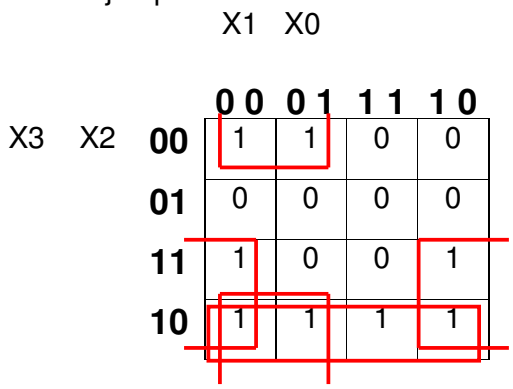
Reglas para la formación de lazos.

1. Cada lazo debe contener el mayor número de '1' posible, siendo ese número siempre una potencia de 2: 1, 2, 4, 8,...
2. Los lazos pueden quedar superpuestos y no importa que haya cuadrículas de valor '1' que correspondan a la vez a dos lazos diferentes.
3. No se pueden formar lazos entre parejas de '1' situados en diagonal.
4. Debe tratarse de conseguir el mínimo número de lazos, cada uno de ellos con el mayor número de '1'.
5. La columna más a la derecha se considera adyacente a la de más a la izquierda, y la primera fila se considera adyacente a la última.

Cada lazo representa un minterm de la ecuación, la cual reúne a todos los minterms mediante sumas lógicas u operaciones OR.
Si en un lazo hay una variable que está en estado '1' en una cuadrícula y en estado '0'en otra, se elimina. Si una variable está con el mismo estado en todas las cuadrículas de un lazo, debe ser incluida en la expresión simplificada.

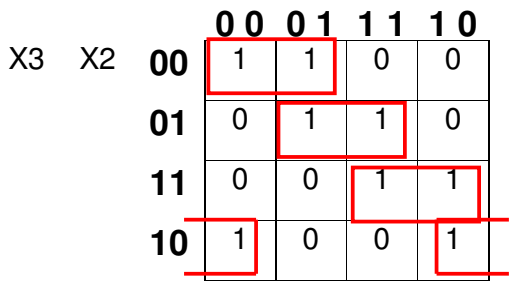
Ejemplos:

ejemplo 1.



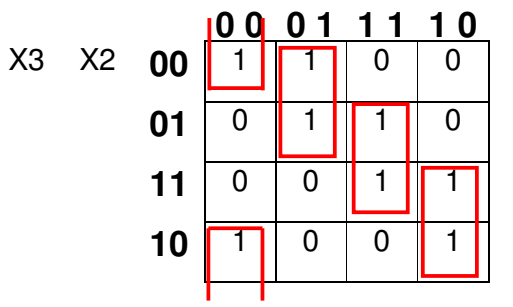
$\neg X2 \cdot \neg X1 + X3 \cdot \neg X0 + X3 \cdot \neg X2$

ejemplo 2.



$\neg X3 \cdot \neg X2 \cdot \neg X1 + \neg X3 \cdot X2 \cdot X0 + X3 \cdot X2 \cdot X1 + X3 \cdot \neg X2 \cdot \neg X0$

otra forma:



$\neg X2 \cdot \neg X1 \cdot \neg X0 + \neg X3 \cdot \neg X1 \cdot X0 + X2 \cdot X1 \cdot X0 + X3 \cdot X1 \cdot \neg X0$

otra forma:

X3 X2 \ X1 X0	0 0	0 1	1 1	1 0
00	1	1	0	0
01	0	1	1	0
11	0	0	1	1
10	1	0	0	1

$\neg X2 \cdot \neg X1 \cdot \neg X0 + \neg X3 \cdot \neg X2 \cdot \neg X1 + \neg X3 \cdot X2 \cdot X0 + X2 \cdot X1 \cdot X0 + X3 \cdot X1 \cdot \neg X0$

¡5 TÉRMINOS! – No redundante.

otra forma:

X3 X2 \ X1 X0	0 0	0 1	1 1	1 0
00	1	1	0	0
01	0	1	1	0
11	0	0	1	1
10	1	0	0	1

$\neg X3 \cdot \neg X2 \cdot \neg X1 + \underline{\neg X3 \cdot \neg X1 \cdot X0} + \neg X3 \cdot X2 \cdot X0 + X3 \cdot X2 \cdot X1 + X3 \cdot \neg X2 \cdot \neg X0$

¡5 TÉRMINOS! – Redundante.

3.4.2. Simplificación de funciones incompletas

Se pueden encontrar funciones en las que alguna de las combinaciones de valores de entrada no tienen definida una salida en particular. Este tipo de funciones se denominan incompletas.
A la hora de aplicar algún método de minimización es necesario asumir un valor determinado para esas salidas, a condición de todas las salidas indefinidas tengan el mismo valor.
Para el método de Karnaugh aconsejable disponer este tipo de salidas a valor '1', ya que esto permite hacer mayor cantidad de lazos con menos variables cada uno de ellos.

X3 X2 \ X1 X0	0 0	0 1	1 1	1 0
00	0	0	0	-
01	-	-	1	1
11	0	1	1	-
10	0	0	1	1

X3 X2 \ X1 X0	0 0	0 1	1 1	1 0
00	0	0	0	**1**
01	**1**	**1**	1	1
11	0	1	1	**1**
10	0	0	1	1

3.4.3. Método tabular de Quine-Mc Cluskey

Cuando las ecuaciones tienen 5 o más variables es complicado utilizar los diagramas de Karnaugh, siendo el método de Quine-Mc cluskey el más apropiado. El mismo consiste en ordenar según el número de '1'que tengan las combinaciones de variables que cumplen la ecuación (que la hacen '1'). A continuación se buscan las combinaciones que comparadas con los grupos adyacentes, con un 1 más o menos, difieren sólo en una variable que en una combinación estará negada y en la otra sin negar, eliminándose la misma.
Como se ve , el principio de simplificación de este método es el mismo que el del método de Karnaugh:

$$m \cdot C + m \cdot \neg C = m \cdot (C + \neg C) = m$$

El proceso de simplificación consiste en la aplicación de una serie de fases o pasos que se explicarán a continuación con el siguiente ejemplo:

$$X = \neg A \cdot B \cdot \neg C + \neg A \cdot C \cdot D + \neg A \cdot B \cdot C \cdot \neg D + A \cdot C \cdot D$$

Fase 1
Todos los términos de la ecuación lógica deben contener todas las variables. Para los términos que carezcan de alguna variable, ésta se incluye realizando la operación AND del término por la variable mas la variable negada, teniendo en cuenta que $(V + \neg V) = 1$.

En el caso de nuestro ejemplo, la aplicación de esta regla transforma la ecuación de la siguiente manera:

$$X = \neg A \cdot B \cdot \neg C \cdot (D + \neg D) + \neg A \cdot C \cdot D \cdot (B + \neg B) + \neg A \cdot B \cdot C \cdot \neg D + A \cdot C \cdot D \cdot (B + \neg B)$$

$$X = \neg A \cdot B \cdot \neg C \cdot D + \neg A \cdot B \cdot \neg C \cdot \neg D + \neg A \cdot B \cdot C \cdot D + \neg A \cdot \neg B \cdot C \cdot D + \neg A \cdot B \cdot C \cdot \neg D + A \cdot B \cdot C \cdot D + A \cdot \neg B \cdot C \cdot D$$

fase 2
Se determina el "índice" de cada término, siendo dicho índice el número de variables de valor 1, que contenga el mismo. Así por ejemplo el término $\neg A \cdot B \cdot \neg C \cdot D$ (0101), tiene como índice 2.

También y para distinguir entre sí los diferentes términos aparte de por su índice, se asigna a cada uno el valor decimal que su código binario, correspondiente al estado de las variables, representa. Por ejemplo, $\neg A \cdot B \cdot \neg C \cdot D$, tiene índice 2 y le corresponde el valor decimal 5.

Teniendo en cuenta los dos valores que se acaban de definir los términos de la ecuación del ejemplo quedan de la siguiente manera:

Término	Estado de las variables	Índice	Valor decimal
$\neg A \cdot B \cdot \neg C \cdot D$	0101	2	5
$\neg A \cdot B \cdot \neg C \cdot \neg D$	0100	1	4
$\neg A \cdot B \cdot C \cdot D$	0111	3	7
$\neg A \cdot \neg B \cdot C \cdot D$	0011	2	3
$\neg A \cdot B \cdot C \cdot \neg D$	0110	2	6

A · B · C · D	1111	4	15
A · ¬ B · C · D	1011	3	11

fase 3

Se hace una primera lista de los términos de la ecuación, clasificándolos por su índice. En el caso del ejemplo dicha lista sería la siguiente:

Índice	Estado de variables	Valor decimal
1	0100	4
2	0101	5
2	0011	3
2	0110	6
3	0111	7
3	1011	11
4	1111	15

fase 4

Se forma una segunda lista combinando los términos expresados en la lista anterior, siguiendo la regla que se indica:

> *"los términos a combinar no deben diferir entre sí, más que en el estado de una de las variables, la cual será sustituida por un guión"*

Aplicando esta regla al ejemplo, se obtiene la siguiente lista:

Términos combinados (valor decimal)	Combinación	índice combinación
4, 5	010-	1
4, 6	01-0	1
5, 7	01-1	2
3, 7	0-11	2
6, 7	011-	2
3, 11	-011	2
7, 15	-111	3
11, 15	1-11	3

fase 5

Se forma una nueva lista, combinando parejas de términos de acuerdo con la misma regla de la fase anterior. Las nuevas combinaciones dispondrán, por lo tanto, de dos guiones, uno correspondiente a la lista anterior más el de la nueva variable que cambia de estado en la nueva lista.

Los términos que se repiten en la nueva lista se eliminan.

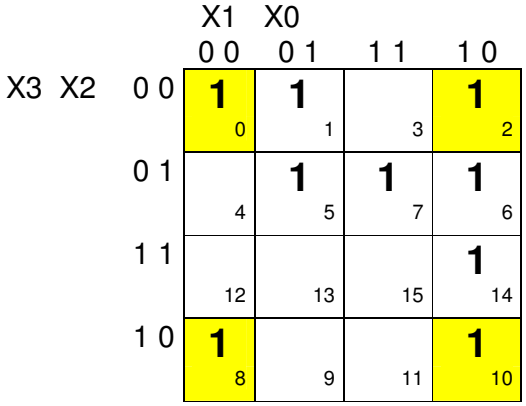
Pareja de términos combinados	Combinación	Índice
4, 5, 6, 7	01--	1
4, 5, 6, 7	01--	1 (eliminada)
3, 7, 11, 15	--11	2
3, 7, 11, 15	--11	2 (eliminada)

fase 6

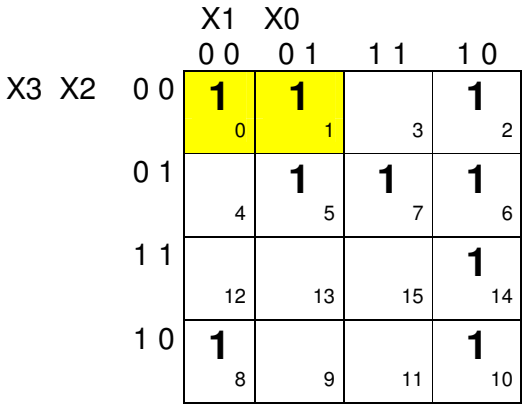
La ecuación simplificada se obtiene mediante la suma lógica de los términos no eliminados. En nuestro ejemplo, teniendo en cuenta las combinaciones no eliminadas de la última lista (01—y --11), será:

$$X = \neg A \cdot B + C \cdot D$$

3.4.4. Método de los consensos



$$a = \overline{x2} \cdot \overline{x0}$$



$$b = \overline{x3} \cdot \overline{x2} \cdot \overline{x1}$$

X3 X2 \ X1 X0	0 0	0 1	1 1	1 0
0 0	**1** 0	**1** 1	3	**1** 2
0 1	4	**1** 5	**1** 7	**1** 6
1 1	12	13	15	**1** 14
1 0	**1** 8	9	11	**1** 10

$$c = \overline{x3} \cdot \overline{x1} \cdot x0$$

X3 X2 \ X1 X0	0 0	0 1	1 1	1 0
0 0	**1** 0	**1** 1	3	**1** 2
0 1	4	**1** 5	**1** 7	**1** 6
1 1	12	13	15	**1** 14
1 0	**1** 8	9	11	**1** 10

$$d = \overline{x3} \cdot x2 \cdot x0$$

X3 X2 \ X1 X0	0 0	0 1	1 1	1 0
0 0	**1** 0	**1** 1	3	**1** 2
0 1	4	**1** 5	**1** 7	**1** 6
1 1	12	13	15	**1** 14
1 0	**1** 8	9	11	**1** 10

$$e = \overline{x3} \cdot x2 \cdot x1$$

X3 X2 \ X1 X0	0 0	0 1	1 1	1 0
0 0	**1** 0	**1** 1	3	**1** 2
0 1	4	**1** 5	**1** 7	**1** 6
1 1	12	13	15	**1** 14
1 0	**1** 8	9	11	**1** 10

$$g = x1 \cdot \overline{x0}$$

a	0, 2, 8, 10
b	0, 1
c	1, 5
d	5, 7
e	6, 7
g	2, 6, 10, 14

0	a + b
1	b + c
2	a + g
5	c + d
6	e + g
7	d + e
8	a
10	a + g
14	g

(a + b) . (b + c) . (a + g) . (c + d) . (e + g) . (d + e) . a . (a + g) . g = 1

(b + c) . (c + d) . (d + e) . a . g = 1

(c + b.d) . (d + e) . a . g = 1

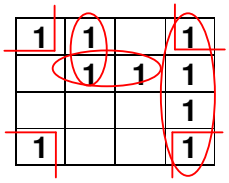
a.g.c.d + a.g.c.e + a.g.b.d + a.g.b.d.e = 1

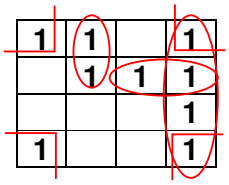
Entonces quedan 3 soluciones:

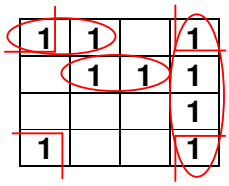
$$f = \overline{x2} \cdot \overline{x0} + x1 \cdot \overline{x0} + \overline{x3} \cdot \overline{x1} \cdot x0 + \overline{x3} \cdot x2 \cdot x0$$

$$f = \overline{x2} \cdot \overline{x0} + x1 \cdot \overline{x0} + \overline{x3} \cdot \overline{x1} \cdot x0 + \overline{x3} \cdot x2 \cdot x1$$

$$f = \overline{x2} \cdot \overline{x0} + x1 \cdot \overline{x0} + \overline{x3} \cdot \overline{x2} \cdot \overline{x1} + \overline{x3} \cdot x2 \cdot x0$$







3.4.5. Minimización de funciones de múltiples salidas

El método de Quine-Mc Kluskey se puede generalizar a funciones de múltiples salidas. Supongamos que tenemos una función múltiple $f=\{f_1, .., f_m\}$. Cada término en f tiene una parte de entrada y una parte de salida. La parte de salida indica cual de los conjuntos de mintérminos de la parte de entrada están a '1'.
Por ejemplo, la función doble $f=\{f_1, f_2\}$ con $f_1=\text{and}(x_1, x_2)$ y $f_2=\text{and}(x_2, x_3)$ puede ser representada como:

11- 10
-11 01

Podemos representar los términos-0 de cada salida como:

110 10
111 10
011 01
111 01

Los términos-0 para salidas múltiples serán:

011 01
110 10
111 11

La generación de los términos primos a partir de los términos-0 es similar a la usada para una única salida. 2 términos-0 son combinados en un término-1 si sus partes de entrada difieren en exactamente un bit y sus partes de salida se intersectan, o sea, hay al menos una salida en común. El término-1 formado tendrá un "-" en el bit de entrada que resultó diferente entre los términos-0 y tendrá una parte de salida igual a la unión (bitwise AND) de las partes de salida de los términos-0 en cuestión.

Ejemplo: Supongamos una función de 2 salidas:

$f=\{f_1, f_2\}$ con $f_1=(1, 3, 4, 5)$ y $f_2=(1, 3, 6, 7)$

Los términos-0 se colocan en columna de forma separada para cada salida y los números a la izquierda de cada término-0 corresponden al valor decimal del término-0 incluyendo su entrada y su salida (PE: 001 10 tiene valor decimal 6).

6	001	10	5,6	001	11	**5,6,13,14**	**0-1**	**11**	**[A]**
14	011	10	13,14	011	11	**5,6,22**	**-01**	**10**	**[B]**
18	100	10	18	100	10	**13,14,29**	**-11**	**01**	**[C]**
22	101	10	22	101	10	**18,22**	**10-**	**10**	**[D]**
			25	110	01	**25,29**	**11-**	**01**	**[E]**
5	001	01	29	111	01				
13	011	01							
25	110	01							
29	111	01							
Término-0			Término-0 para funciones múltiples			Término-1			

Si se da el caso de términos-0 (primera columna) cuyas entradas son las mismas y sus salidas distintas (el mintermino activa múltiples salidas) se genera un término-0 compuesto con un “1” en cada salida que se activa.

Para este ejemplo, los términos-1 no pueden ser reducidos más, de manera que se obtienen los términos primos A, B, C, D y E.

La tabla de términos primos para este caso se muestra a continuación. Aquí las columnas corresponden a los 5 términos primos y las filas los términos-0 de cada una de las funciones considerados por separado. A fin de obtener una implementación para la función múltiple debemos seleccionar un conjunto de términos primos tal que haya una X en cada fila.

		A	B	C	D	E
001	10	X	X	.	.	.
011	10	X	.	.	.	.
001	01	X	.	.	.	.
011	01	X	.	X	.	.
100	10	.	.	.	X	.
101	10	.	X	.	X	.
110	01	.	.	.	.	X
111	01	.	.	X	.	X

Vemos que: 011 10 es cubierto solo por A, 100 10 por D y 110 01 por E. Por lo tanto, A, D y E son esenciales. Por otro lado, seleccionando A, D y E resulta en una X en cada fila, por lo que obtenemos como resultado una implementación mínima consistente de 3 términos

$$x_2\overline{x_0} \ , \ x_2\overline{x_1} \ \text{ y } \ x_2x_1$$

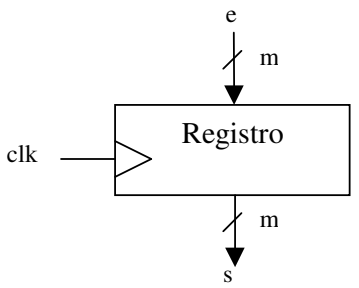
4. Síntesis de Circuitos Secuenciales.

4.1.- Biestables

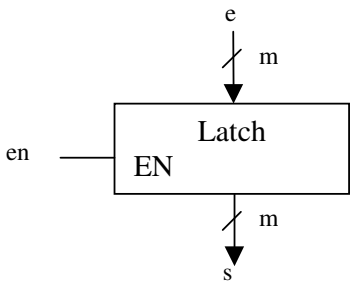
(Referirse a la biblioteca de componentes)

4.2.- Registros

Desde un punto de vista práctico, un registro es una concatenación (conjunto) de biestables interconectados de manera de poder realizar determinadas funciones como almacenar valores de m bits (siendo m el número de biestables), desplazar los bits a izquierda o derecha, conversiones de paralelo a serie y viceversa, etc..
Funcionalmente responden a los siguientes símbolos:



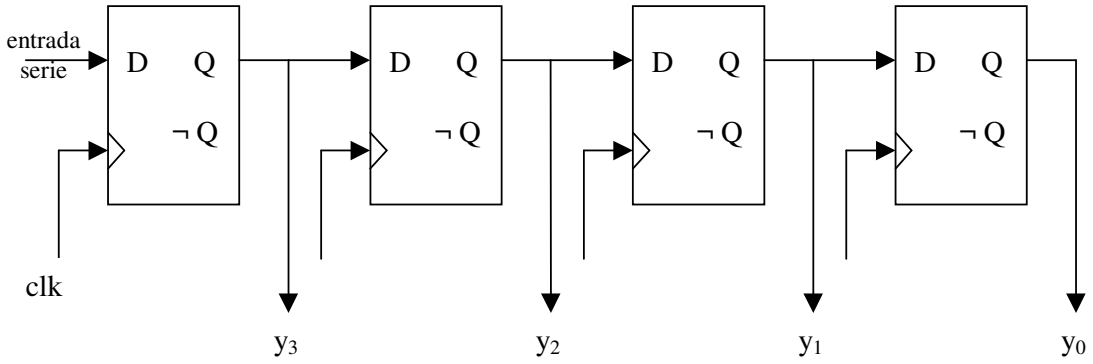
m biestables D en paralelo



m latches D en paralelo

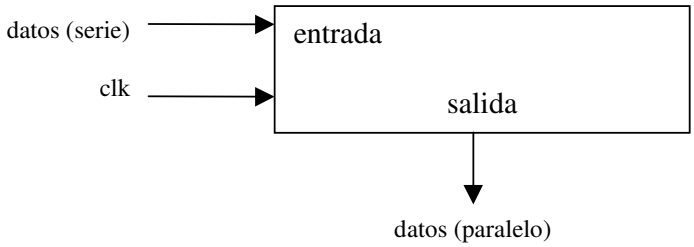
Registros de desplazamiento

El primer tipo especial de registro es el llamado registro de desplazamiento. En él los datos son cargados bit a bit a través del primer biestable D y circulan de cada biestable al siguiente en cada pulso de reloj.
Es fácil comprobar que este tipo de registro sólo se puede implementar con biestables (activados por flanco) y no con latches (activados por pulso o estado), ya que en estos últimos, al activarse todos los latches (por un estado de activación válido) la cadena de bits presente en la entrada circularía por todos los latches hasta la salida.

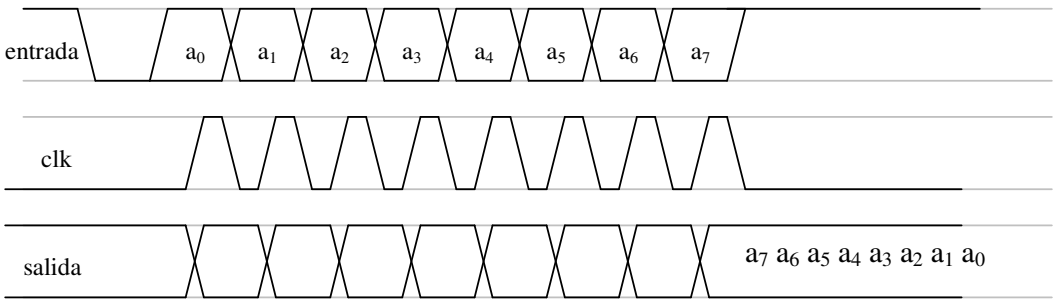


salida paralela

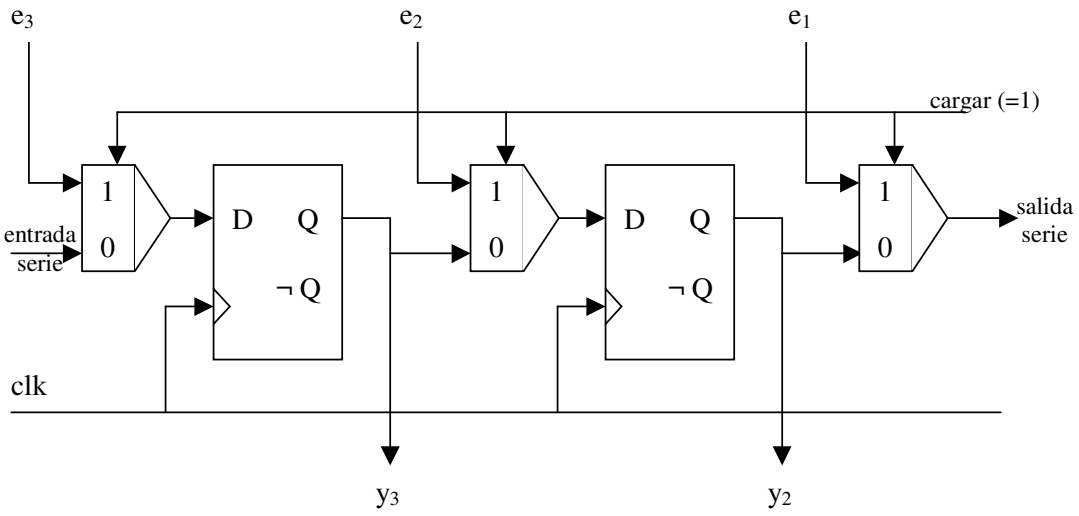
La aplicación más directa de este tipo de registros es la recepción de valores multibytes por su entrada serie en m ciclos de reloj (tantos como bits se reciben) y su posterior salida en paralelo por las lineas y_i.



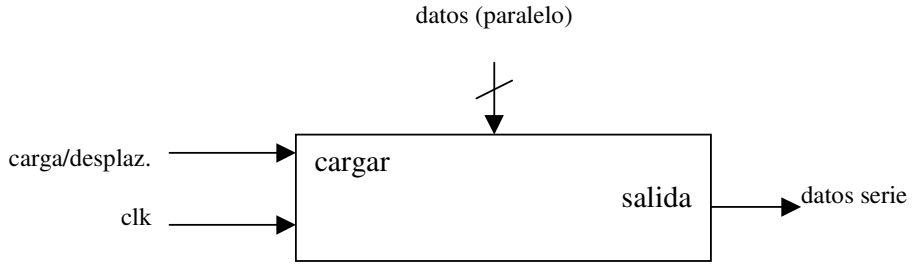
En el siguiente esquema temporal se puede apreciar como entran los datos uno a uno al conjunto de biestables hasta que, luego de m pulsos de reloj, cada biestable tiene cargado un valor de bit que puede enviar a la salida paralela del dispositivo.



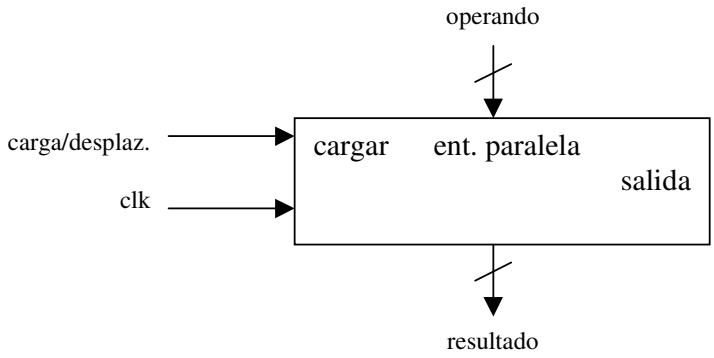
Registro de desplazamiento con entrada paralela

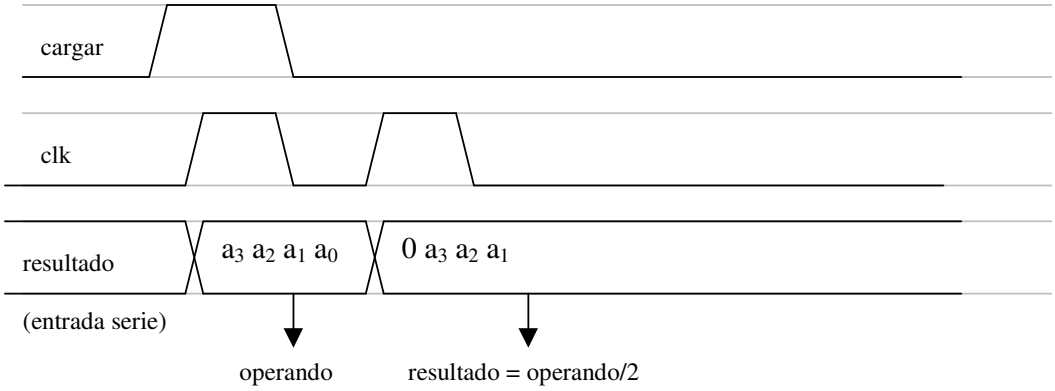


La aplicación más difundida se da en dispositivos serializadores más conocidos como UsART's (transmisores/receptores sincrónicos/asincrónicos universales) usados ampliamente para comunicaciones seriales.



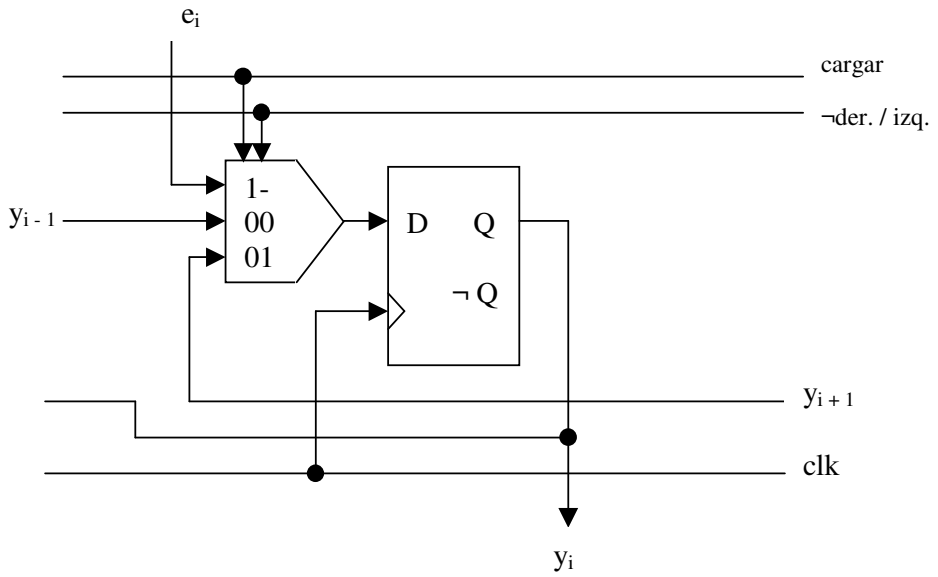
Otra aplicación que utiliza el dispositivo recién visto es el circuito divisor (multiplicador) por potencias de 2





Para la multiplicación se debe implementar el dispositivo de manera que los bits se desplacen de derecha a izquierda.

La configuración completa de un elemento de almacenamiento que implementa un registro de desplazamiento bidireccional y con entrada paralela se muestra en la figura siguiente.
Las tres funciones principales del dispositivo mixto son *carga*, *desplazamiento a derecha* y *desplazamiento a izquierda*. Para implementarlas es necesario utilizar un multiplexor con dos señales de control, una para habilitación de carga y la otra para indicar el sentido del desplazamiento a izquierda o derecha.



4.3.- Contadores

Los circuitos contadores binarios difieren de los registros de desplazamiento en que sus flip-flops están conectados de una manera diferente. El objeto de un circuito contador es dar salida a la información de una forma específica, o bien aumentar al máximo el número de distintos estados que pueden obtenerse con un determinado número de flip flop.

La mayoría de los contadores dan salida a la información codificada en 8421, 2421, exceso a 3 o algún otro código binario corriente, pero diseñando una lógica de interconexión apropiada puede obtenerse cualquier configuración arbitraria a la salida.

Los contadores se utilizan normalmente como circuitos básicos en otros circuitos lógicos. Se emplean en cómputo, como secuenciadores de equipos u operaciones de proceso, en medición y división de frecuencia, manipulación aritmética, medición de intervalo de tiempo, etc.

Existen muchas variantes de contadores. Todos se fabrican mediante flip flops de los tipos JK, T, D ó biestable del tipo RS y se pueden clasificar en dos tipos fundamentales:

- Asíncronos, conocidos también como contadores serie.
- Síncronos, a los que se llama contadores paralelo.

En los contadores síncronos todos los flip flop cambian de estado simultáneamente, en tanto que en los asíncronos, cambia el estado de un flip flop y este cambio activa un segundo flip flop, el cual puede después activar a un tercero, luego a un cuarto, y así sucesivamente.

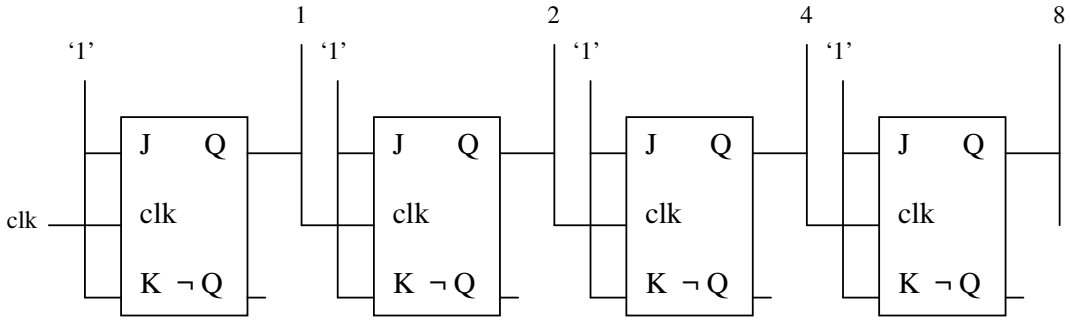
Dentro de cada una de las categorías básicas, puede diseñarse un contador que cuente hasta cualquier número binario deseado antes de repetir la secuencia del cómputo. El número de estados sucesivos a través de los cuales un determinado contador realiza una secuencia antes de que se repita se denomina “módulo”. Los contadores de módulo 2, 4, 8, 16 o cualquier otra potencia de 2, son los más fáciles de construir. Sin embargo, son también comunes los de módulo 6 ó 10.

Los contadores pueden clasificarse de acuerdo con el código ponderado en que cuentan (por ejemplo, 8421 ó exceso 3). Además un contador puede contar hacia arriba o hacia abajo o hacer ambas cosas, según la lógica adicional de que disponga en una entrada de control.

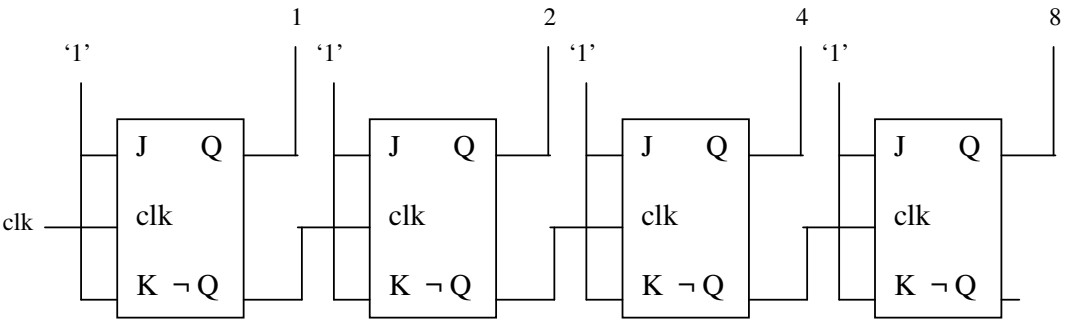
Por lo tanto, de forma general, un contador es un circuito que realiza una secuencia a través de M estados diferentes en un orden especial, siendo M el módulo del contador. El contador cambia de un estado a otro a través de una línea para la señal de reloj.

Contadores asíncronos

El contador de propagación binaria (asíncrono) es el tipo más básico de todos. Una serie de flip flop JK conectados según se muestra en la figura siguiente contará hacia arriba (o incrementando) en código 8421.



Notar (como se ve en la figura siguiente) que un leve cambio en la interconexión del circuito hará cambiar el sentido de la cuanta (decremento) con cada señal de reloj (aunque ahora los flip flop se activan con el pulso de bajada del reloj).



De la inspección ocular de las dos últimas figuras se deduce rápidamente la característica de los contadores asíncronos: la salida de cualquier flip flop (a excepción del último) dispara al siguiente flip flop por su entrada de reloj.

Una característica importante de cualquier contador es la velocidad con la que puede funcionar. Si cada flip flop de las figuras anteriores tiene un retardo de propagación de tr_f ns. el tiempo total, desde que se aplica al primer flip flop el flanco de bajada del reloj hasta que el último flip flop actualiza su estado será de $tr_f * n$, con n igual al número de flip flop en serie. esto es bastante importante, ya que indica que ese es el tiempo mínimo del periodo del reloj maestro que dispara el contador. Un simple cálculo muestra que la frecuencia máxima de salida será entonces:

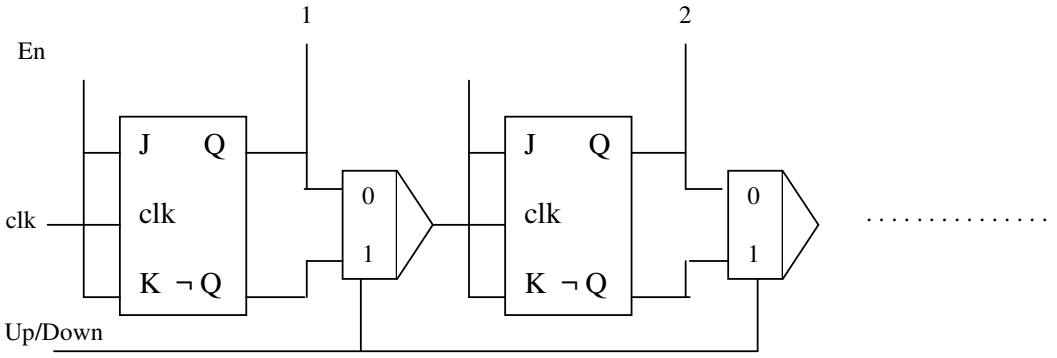
$$\text{Frem}_{max} = 1 / tr_f * n$$

Esta limitación de la frecuencia máxima de los impulsos de reloj es el principal inconveniente del contador asíncrono.

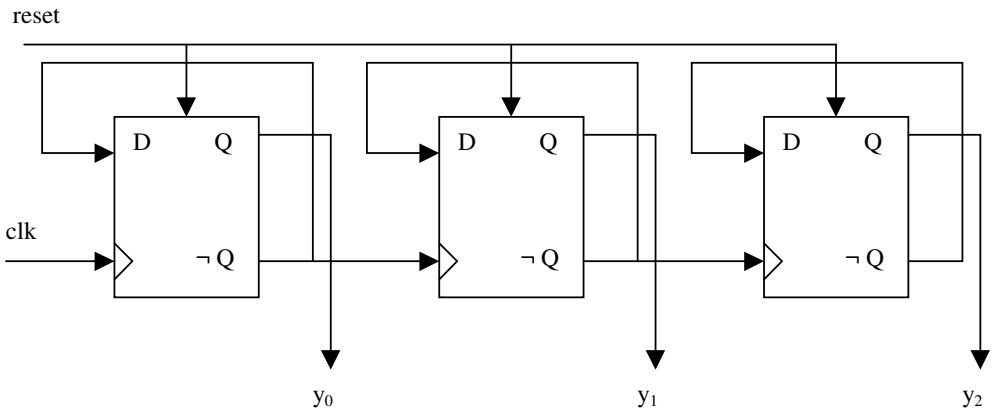
Contador asíncrono ascendente-descendente (UP/DOWN)

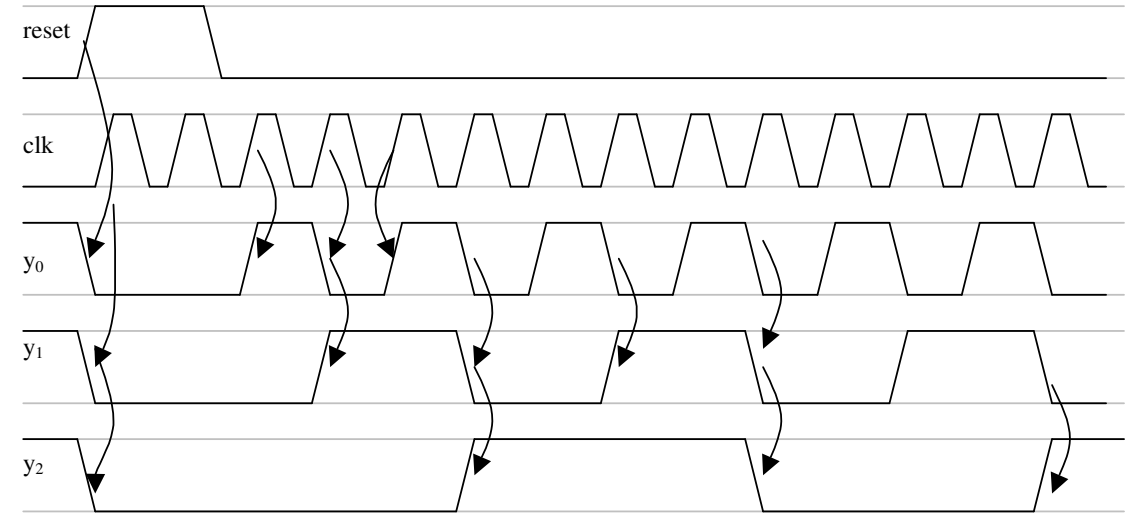
Los dos circuitos mostrados anteriormente pueden combinarse en un contador ascendente-descendente, cuya modalidad de trabajo se selecciona con una señal

de control distribuida por alguna lógica adicional. Además se puede agregar una señal de habilitación de cómputo (Enable)



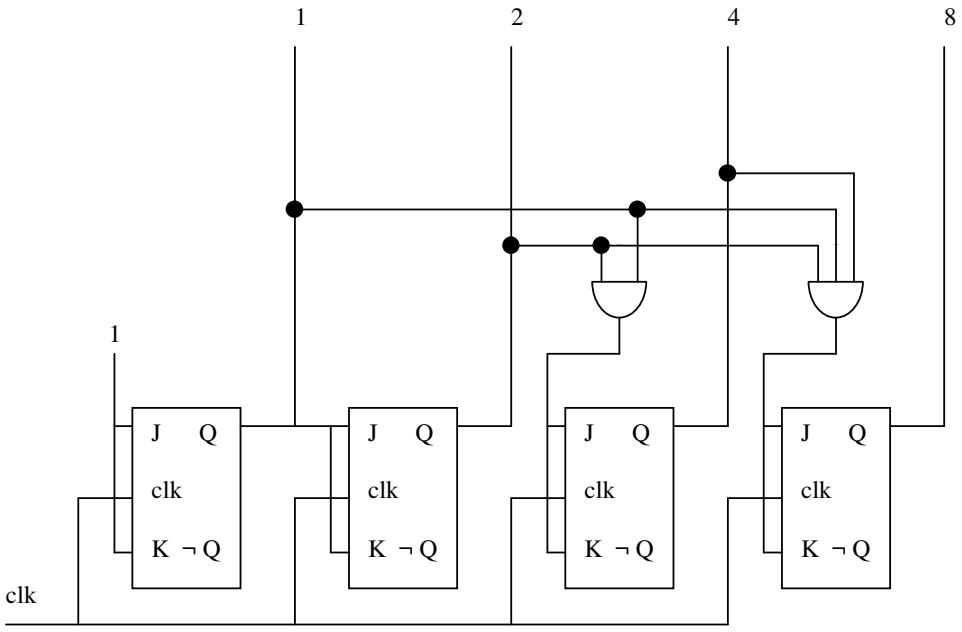
También se puede implementar un contador binario asíncrono a partir de flip flop tipo D. En este caso basta con realimentar la salida negada de cada flip flop a su propia entrada (D) para hacerlo oscilar en cada pulso de reloj recibido por su entrada clk. La figura siguiente muestra la configuración tratada.





Contadores síncronos

Los contadores síncronos se basan en el mismo circuito flip flop JK (o tipo T) que los contadores asíncronos, con la diferencia que todos los flip flop son activados mediante una señal de reloj común y, por lo tanto, todos cambian de estado sincrónicamente (al mismo tiempo). Las entradas J y K de cualquier flip flop están conectadas a las salidas Q de todos los flip flop anteriores que hay en la cadena del contador a través de una puerta AND. Por lo tanto, cualquier flip flop se activará cuando la puerta AND que se aplica a las entradas J y K, tengan una lógica 1 y esto se produce únicamente cuando todos los flip flop anteriores de la cadena están en estado 1.



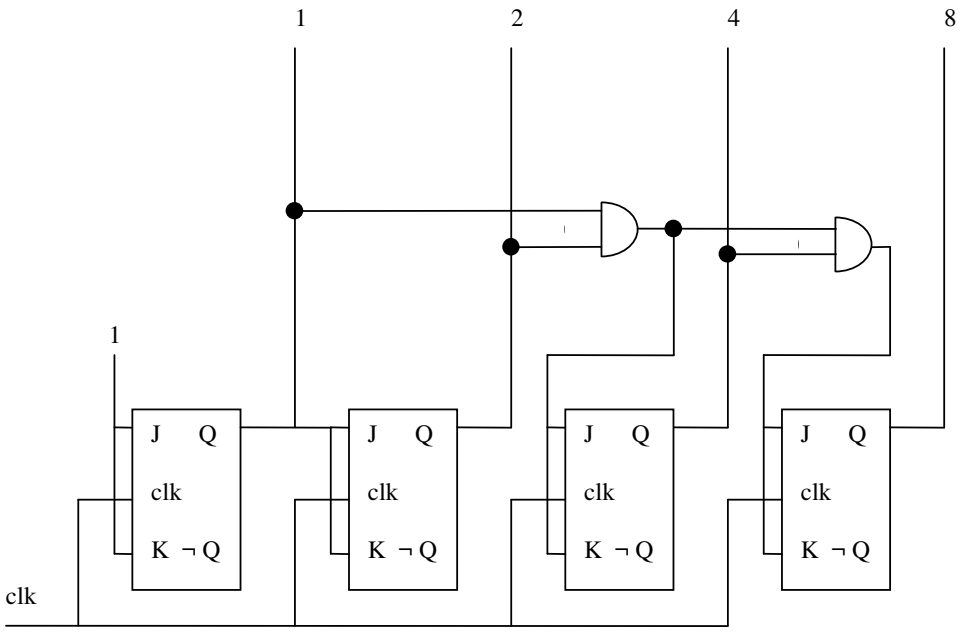
Se debe observar que, debido a que todos los flip flop reciben un mismo impulso de reloj y cambian de estado al mismo tiempo, el retardo total (independientemente del número de flip flop que haya) es equivalente al de un flip flop. Si el tiempo total de retardo de propagación JK y de la puerta AND asociada que conecta su salida con otro flip flop es de 35 ns (25 + 10 ns.), los impulsos de reloj pueden producirse con una frecuencia máxima de 30 Mhz. en un contador síncrono. Compárese este dato con la frecuencia máxima de 10 Mhz del contador asíncrono de 4 bits, usando el mismo retardo de propagación del flip flop.

Otra característica útil del contador síncrono es que todas sus líneas de salida cambian simultáneamente. Por lo tanto no hay estados incorrectos con salidas del contador incorrectas, ya que el mismo avanza de un estado a otro.

También, y como es natural, el contador síncrono tiene limitaciones. En primer lugar precisa más puertas lógicas para ser activado y, por lo tanto, es más costoso y complejo que un asíncrono comparable. En segundo lugar, las puertas AND asociadas a los bits de mayor peso tienen cada vez mayor número de entradas, lo que convierte la construcción de este tipo de compuertas en una limitación de tipo práctico.

Contador síncrono con acarreo.

El contador síncrono con acarreo es una versión simplificada del normal. Continua siendo síncrono en el sentido en que todos los flip flop cambian de estado al mismo tiempo, pero la conexión entre las entradas J y K de los flip flop y las salidas Q de todos los anteriores a él se hacen con puertas AND en serie en lugar de en paralelo. Como consecuencia de ello, el retardo de acumulación de las puertas AND es acumulativo y la frecuencia de funcionamiento se ve reducida algo, en comparación con los contadores síncronos. Cuando el contador se amplía a más de cuatro flip flop el retardo se hace proporcionalmente mayor y disminuye la ventaja de su velocidad con respecto a contadores asíncronos. Aunque este dispositivo es más lento que uno puramente síncrono, el que las salidas cambien de estado simultáneamente y que el circuito sea mucho más simple, hacen de él una solución intermedia entre los contadores síncronos y los asíncronos.



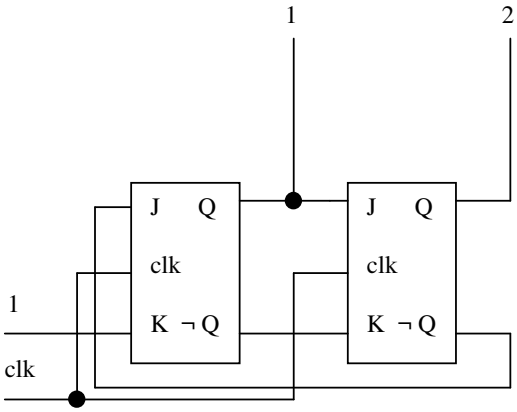
Contadores módulo N

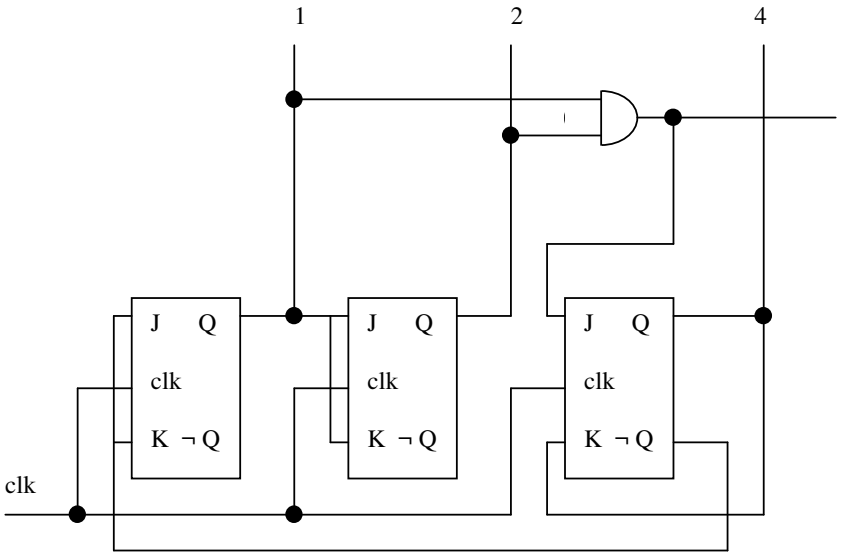
Un contador módulo N es uno que tiene N estados diferentes. Por lo tanto, un contador módulo N puede ser cualquiera, síncrono o asíncrono, que contenga los circuitos precisos para controlar el número de estados que puede tener. Por ejemplo, un contador BCD es el que puede contar hasta 16, pero cuyo módulo está limitado a 10 mediante puertas especiales.

La definición precedente de un contador módulo N es muy general y comprende todos los contadores con módulos grandes o pequeños. En la práctica, una forma de construir un contador de cualquier módulo consiste en conectas en serie varios contadores. Para determinar el módulo de la combinación de contadores en serie se multiplican los módulos particulares de cada uno. Por ejemplo, un contador de

módulo 105 puede construirse interconectando en serie tres contadores de módulos 3, 5 y 7, ya que 3 * 5 * 7 = 105.

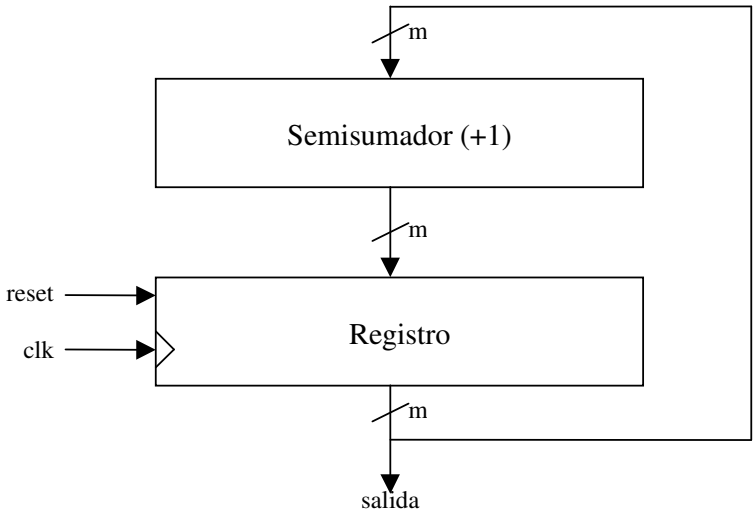
Las figuras siguientes muestran dos contadores de módulos 3 (0 hasta 2) y 5 (0 hasta 4) respectivamente.





Contadores síncronos con sumadores y registros

Una forma ortogonal de armar un contador síncrono es disponer un sumador y un registro de manera que el resultado de la suma se almacene en el registro y en el siguiente ciclo de reloj se use como un nuevo operando del sumador, al que se le suma 1.



En cada ciclo del reloj se actualiza la ecuación:

$$\text{salida} = (\text{salida} + 1) \bmod 2^m$$

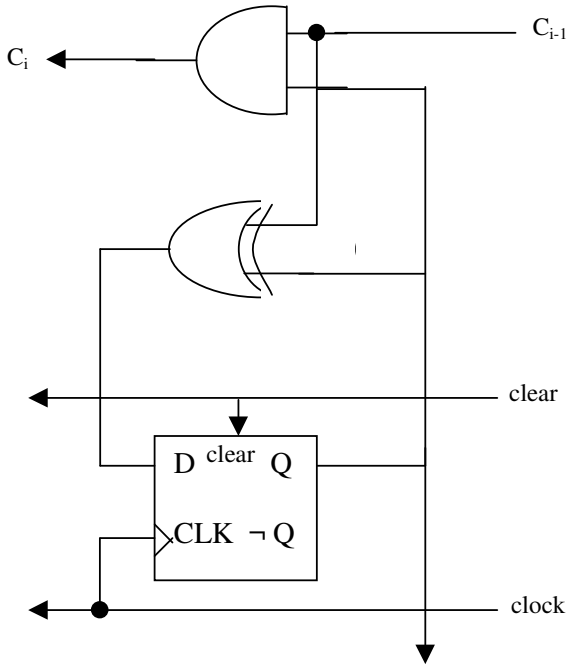
Observación: las funciones de un semisumador son:

$$y_i \oplus C_{i-1}$$

$$y_i \cdot C_{i-1}$$

Además $C_{-1} = 1$ (para sumar 1)

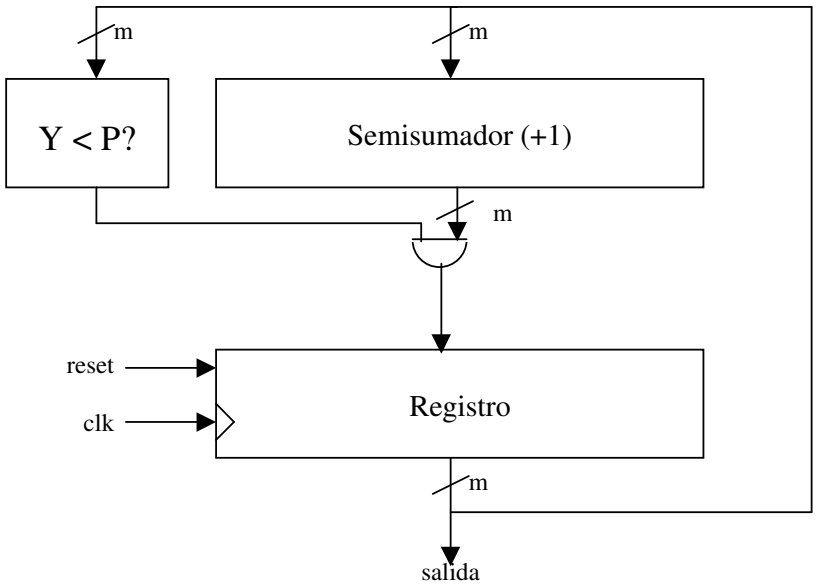
Se puede ver fácilmente que un contador de m bits se compone de m etapas iguales formadas por un semisumador y un biestable tipo D (memoria de almacenamiento temporal) como se muestra en la siguiente figura:



Contador no binario (módulo P) síncrono

Son contadores que responden a a ecuación:

$$y := (y + 1) \bmod P \text{ , con } P <> 2^i$$

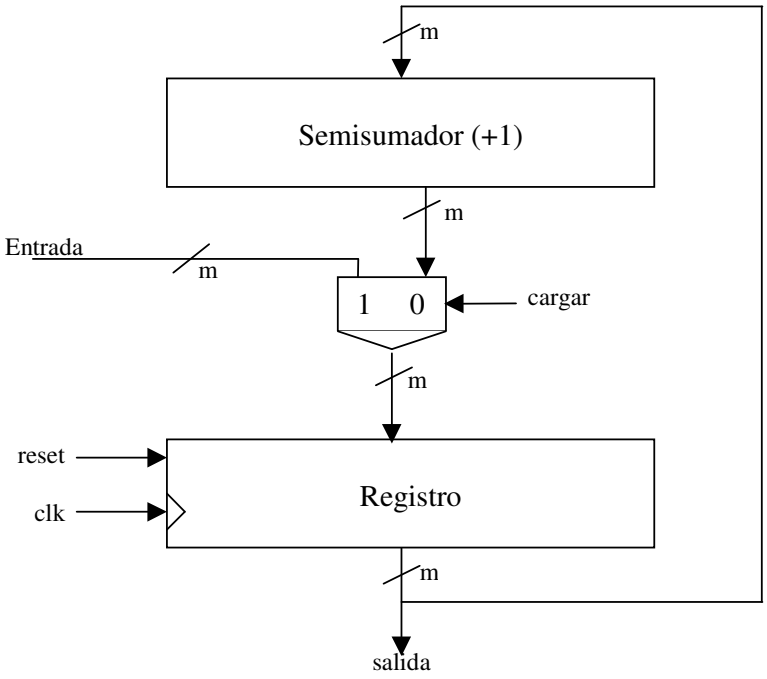


Contador binario programable

Los contadores programables poseen una entrada múltiple de datos (preset del contador) y una señal de carga que habilita al contador a contar o a cargarse con el valor binario presente en la entrada.

La ecuación general de funcionamiento es:

$$y := \neg \text{carga} \cdot (y + 1 \bmod 2^m) + \text{carga} \cdot \text{entrada}$$



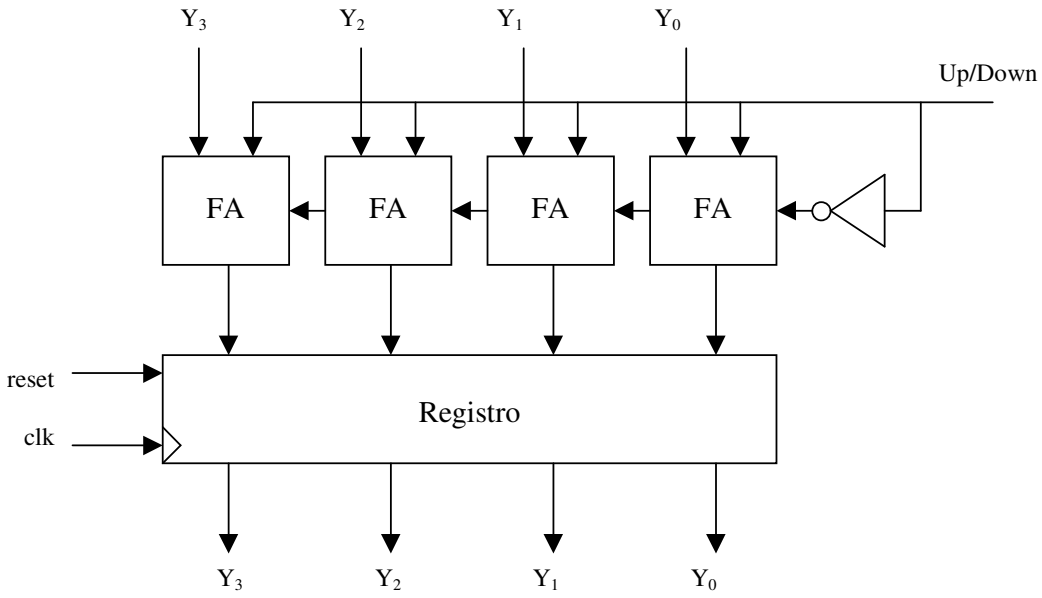
Un ejemplo práctico de un contador binario programable es el contador de programa de los procesadores actuales.

Contador binario bidireccional.

Uno de los esquemas de direccionamiento bastante usado en procesadores son los registros índices. Estos tienen como peculiaridad la capacidad de autoincrementarse o autodecrementarse. Esta posibilidad se puede materializar muy fácilmente modificando las etapas semisumadoras por etapas sumadoras/restadores completas.

La ecuación booleana que modela estos contadores sería:

$$y := \neg \text{Up/Down} \cdot (y + 1) + \text{Up/Down} \cdot (y - 1)$$



Up/Down	y
0	$y + 1 \bmod 2^m$
1	$y + (2^m-1) \bmod 2^m$