Circuitos combinacionales. Funciones integradas

Salvador Marcos González
salvador.marcos@uah.es

Funciones integradas

Introducción

La introducción en el diseño de sistemas digitales de circuitos MSI (media escala de integración) da como resultado la sustitución de los anteriores métodos de diseño. La obtención de una función lógica como solución a un problema particular, la búsqueda de la minimización de la misma y la implementación con funciones SSI, era el proceso habitual. Sin embargo, al implementar funciones lógicas con circuitos MSI, la función booleana se expresa en su forma canónica, es decir cada término de la función booleana contiene todas las variables, bien en su forma directa o negada, y se implementa directamente sin minimización.

El coste de un sistema digital es proporcional al número de circuitos integrados del sistema, de forma que el objetivo es reducir el número de pastillas.

En este capítulo vamos a estudiar diversas funciones MSI empleadas frecuentemente en circuitos digitales y que deben conocerse para el análisis y diseño de circuitos.

Multiplexores

El multiplexor o selector de datos, selecciona 1 de entre n líneas de entrada de datos, donde n es una potencia de 2. Para realizar esta función dispone de un número de líneas de entrada de selección, de tal forma que puedan referenciarse las n líneas de entrada de datos, además suele tener una o varias líneas de habilitación, de tal forma que cuando la línea está activa queda habilitada la función multiplexora y cuando no lo está se inhibe el funcionamiento del mismo.

La siguiente implementación, figura 5.2, corresponde a la de un multiplexor a nivel de puertas. También se muestra un diagrama de bloque representativo en la figura 5.1.

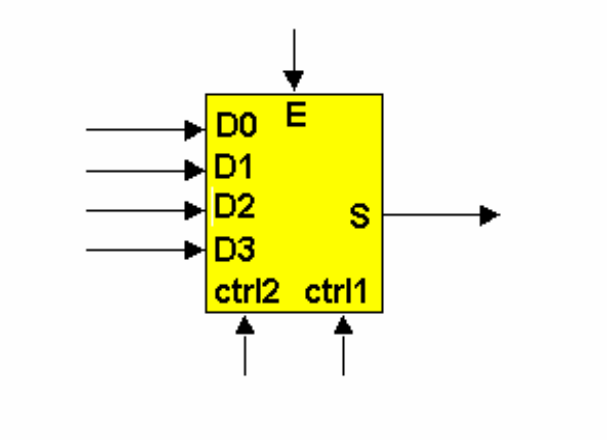


Figura 5.1. Bloque representativo de un multiplexor de 4 canales

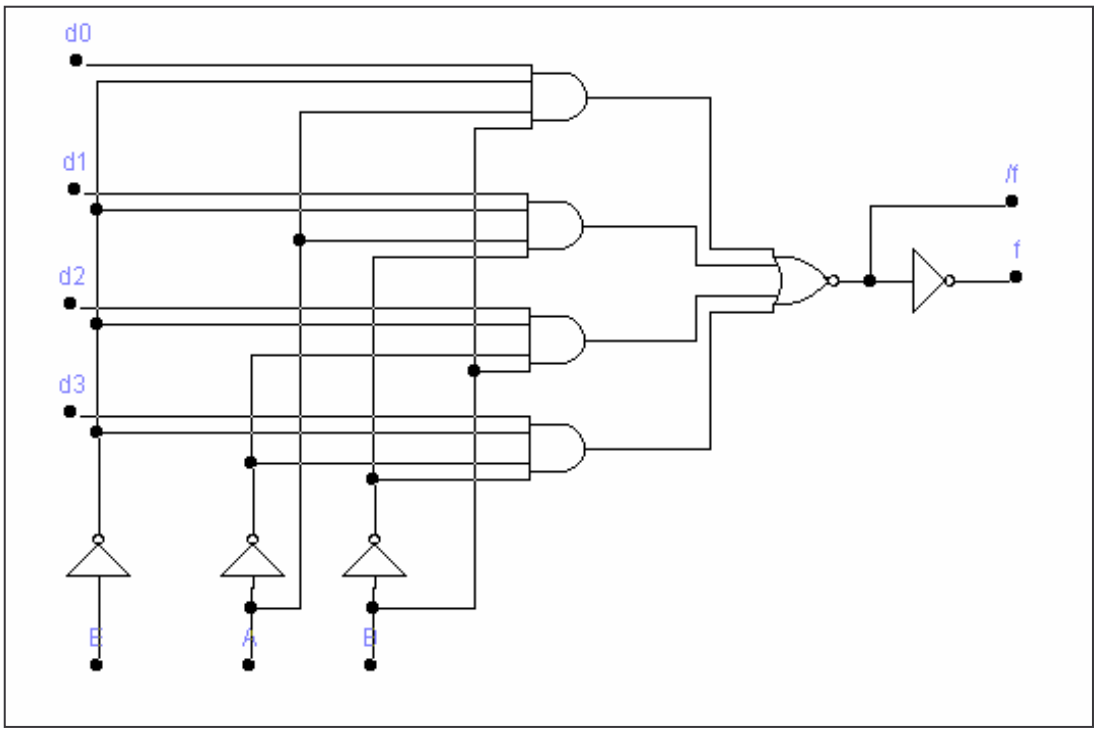


Figura 5.2. Circuito de puertas de un multiplexor de 4 canales

La función Booleana que representa la salida f del circuito viene dada por la expresión:

$$f = ABD_1 + ABD_2 + ABD_3 + ABD_4$$

Pueden seleccionarse líneas de datos mediante la aplicación de la combinación binaria apropiada aplicada a las líneas de control A y B. Cuando las líneas de control son AB = 00, la salida del circuito es D_0

El número de líneas de datos a seleccionar puede incrementarse mediante la elección de multiplexores con un mayor número de ellas, o mediante una combinación de multiplexores. La figura siguiente (figura 5.3) muestra la ampliación de una función multiplexora de 4 entradas de datos a una de 8 entradas, mediante la utilización de la entrada de habilitación como una entrada de selección adicional.

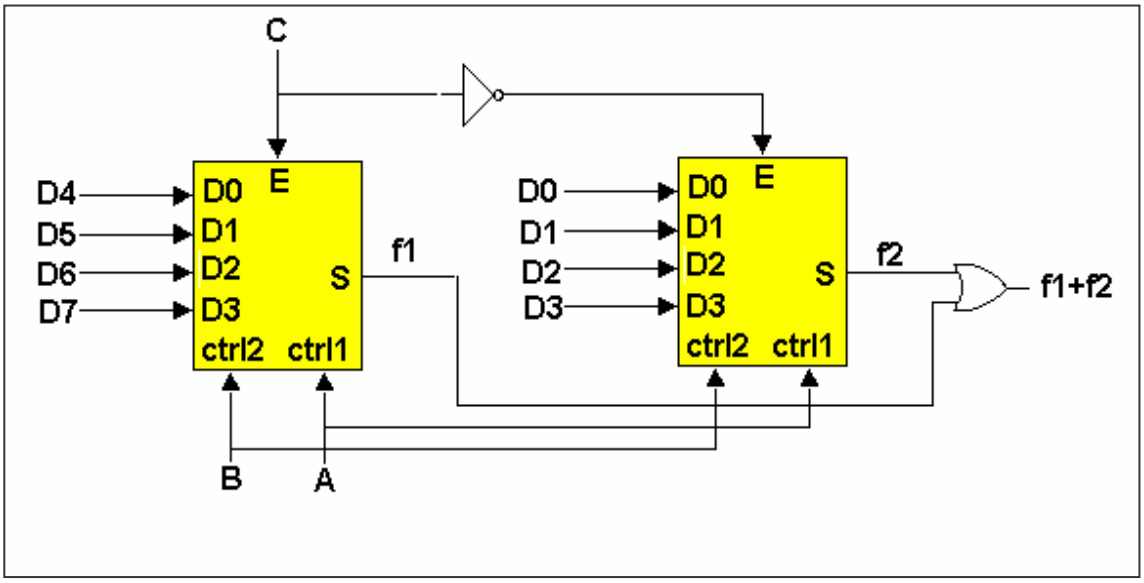


Figura 5.3. Ampliación de las entradas de datos

Otra forma de expandir el número de entradas de datos es mediante la multiplexación a dos o más niveles. La siguiente figura (figura 5.4) muestra la implementación de un multiplexor 1 de entre 64 mediante funciones 1 de entre 8 y con dos niveles de multiplexado.

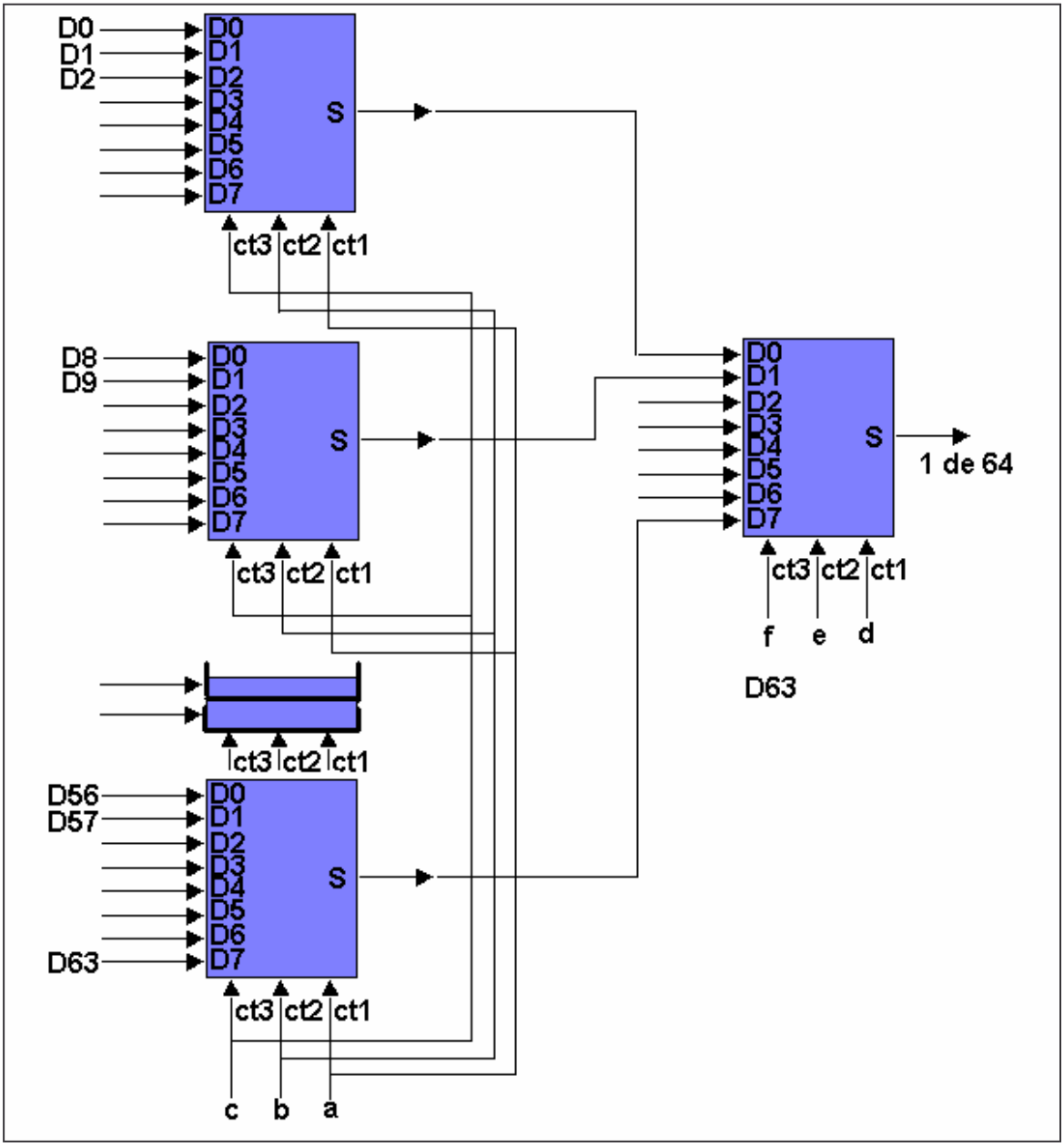
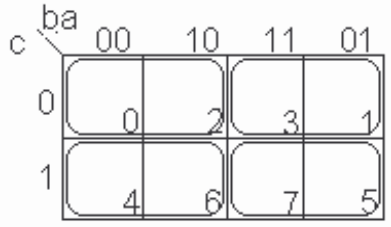


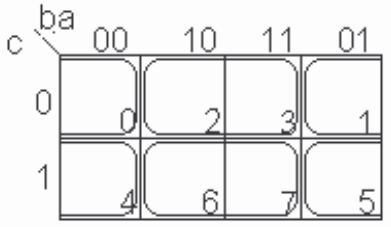
Figura 5.4. Ampliación de las entradas de datos multiplexando a 2 niveles

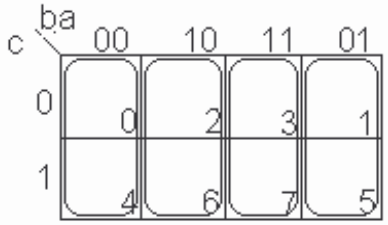
La función multiplexora como generador lógico de funciones

La ecuación booleana para la salida de un multiplexor que tenga 4 entradas de datos es: $f= BAD_0+ BAD_1+ BAD_2+ BAD_3$ las variables A y B se usan como señales para las líneas de control. A y B pueden despejarse como factores en cualquier función de n variables, aplicándose luego a las líneas de datos las funciones residuo de n-2 variables. Por ejemplo, si n es 3, pueden aplicarse 4 señales de 1 variable a cada una de las líneas de datos. Suponiendo C como esa tercera variable, las señales posibles en las líneas de datos son C C 0 y 1.

Para el multiplexor de 4 entradas, hay 3 posibles opciones para las variables de control: AB, AC y BC, pudiéndose asociarse estas combinaciones con líneas de datos individualmente como se muestra en los siguientes mapas de Karnaugh.







Por ejemplo para la función de 3 variables ***f= CBA+CBA+CBA+CBA*** se implementa de las distintas formas-, variables de control AB, AC y BC, en la siguiente figura.

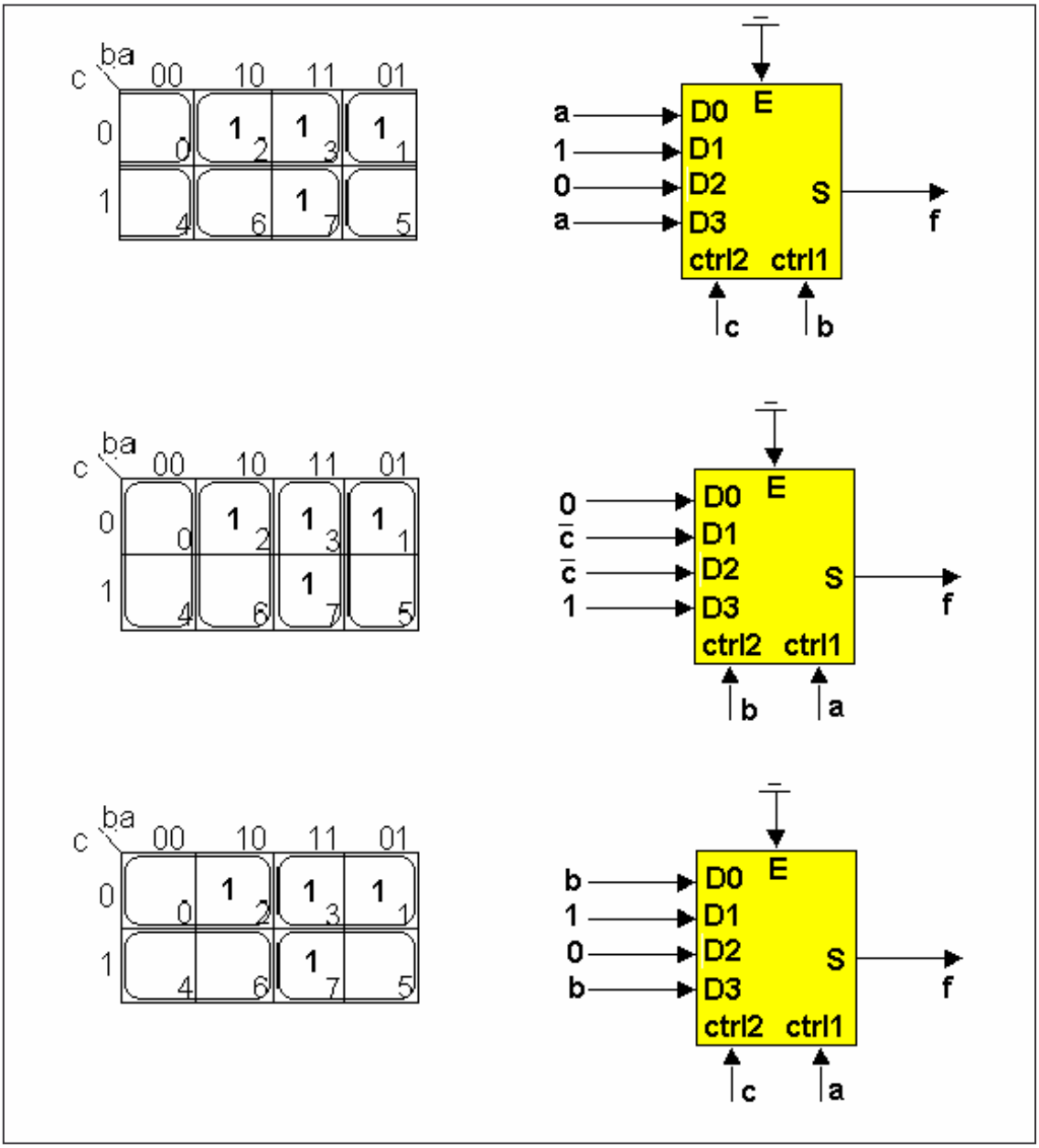


Figura 5.5. Implementación de funciones con multiplexores

Considera ahora la implementación de la función de 4 variables ***f = φ0,2,5,6,10,11,13,15*** utilizando un multiplexor de 4 entradas. Para este caso la

aplicación de 2 variables a sus entradas de control dejará funciones residuo de 2 variables aplicadas a sus entradas de datos.

Se han elegido las variables d y c para aplicar a las entradas de control, el mapa de Karnough queda dividido en 4 tablas de 4 celdas y 2 variables, la simplificación sólo puede hacerse dentro de ellas

Si se elige un multiplexor con 8 entradas de datos y se eligen las variables d, c y b para ser asignadas a sus entradas de control, el mapa queda dividido en 8 tablas de 2 celdas y una variable, pudiéndose realizar la simplificación solamente dentro de ellas.

La figura 5.5. refleja la implementación de la función con multplexores de 4 y 8 entradas de datos.

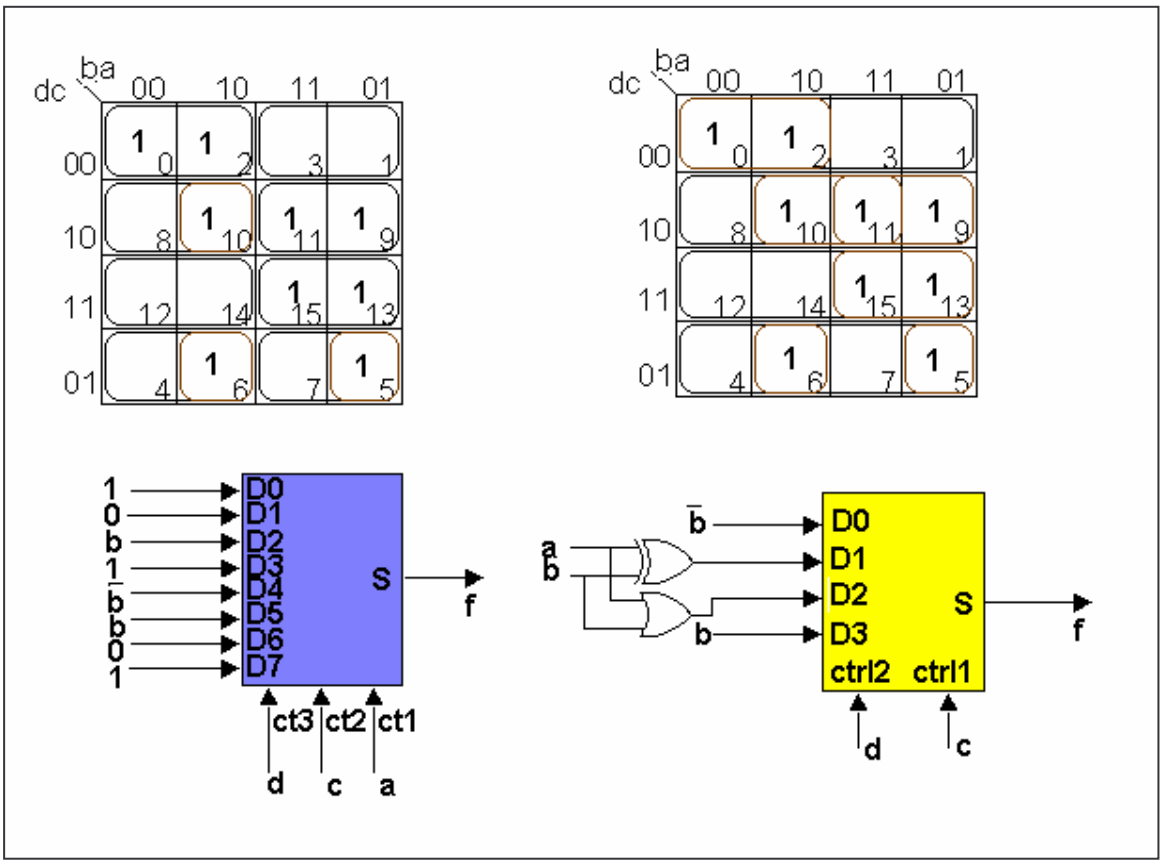


Figura 5.5. Implementación de funciones con multiplexores

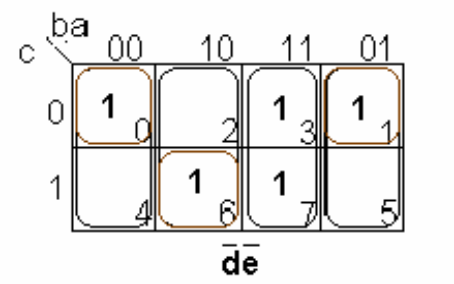
Si el número de variables de la función aumenta, se implementa con más de un nivel de multiplexado. Sea por ejemplo la función de 5 variables ***f = φ0,1,3,6,7,8,14,15,17,18,20,21,22,24,27,28,31*** que implementaremos con multiplexores de 4 entradas de datos.

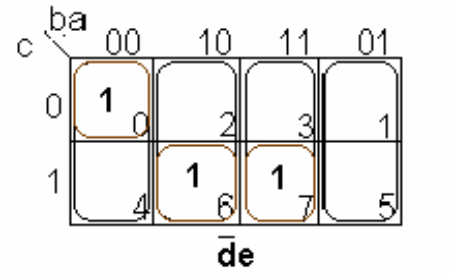
Para el primer nivel elegimos las variables e y d. Formamos una tabla como la mostrada seguidamente, donde la columna de la izquierda muestra los términos producto que conforman la función y las columnas de la derecha, encabezadas por ed, ed, ed y ed muestran el resultado de sacar factor común

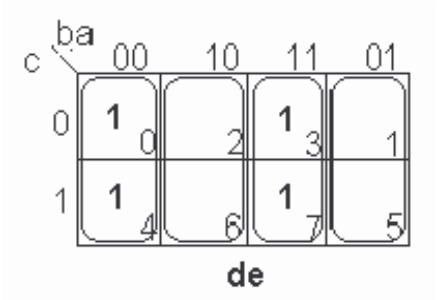
ed, ed, ed y ed en los términos producto de la función. Las funciones de entrada para el multiplexor del primer nivel resultan:

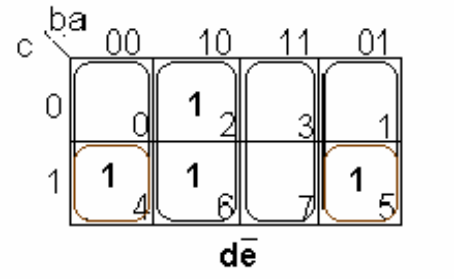
$$D_{01} = cba+cba+cba+cba+cba$$
$$D_{02} = cba+cba+cba+cba+cba$$
$$D_{03} = cba+cba+cba+cba+cba$$
$$D_{04} = cba+cba+cba+cba+cba$$

función	ed	ed	ed	ed
0 edcba	cba			
1 edcba	cba			
3 edcba	cba			
6 edcba	cba			
7 edcba	cba			
8 edcba		cba		
14 edcba		cba		
15 edcba		cba		
17 edcba			cba	
18 edcba			cba	
20 edcba			cba	
21 edcba			cba	
22 edcba			cba	
24 edcba				cba
27 edcba				cba
28 edcba				cba
31 edcba				cba









El segundo nivel puede generar las funciones de 3 variables obtenidas como muestran los mapas anteriores, donde se han tomado b y a como variables asignadas a las entradas de control. La figura 5.6. muestra la implementación de la función.

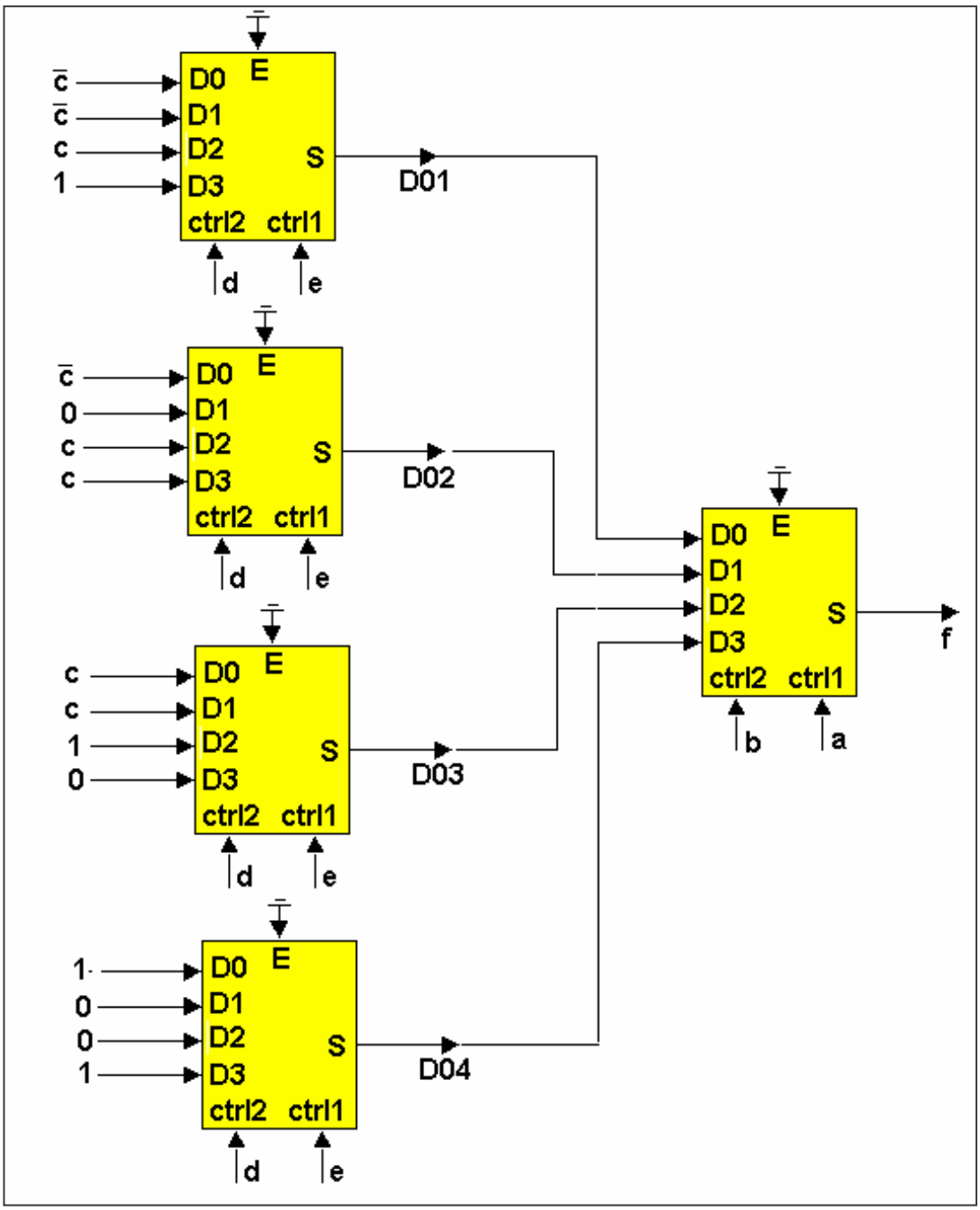


Figura 5.6. Implementación de una función de 5 variables

Demultiplexores

Un demultiplexor realiza la función contraria a la del multiplexor. Consta de una entrada de datos, un número de entradas de control (generalmente 1, 2 o 3), una entrada de habilitación y un número de salidas, $2^{nº \text{ de líneas de control}}$. La función que realiza es la de conectar la línea de entrada de datos con aquella línea de salida que le indique el valor binario de las líneas de control. La figura 5.7. muestra la implementación de un demultiplexor con cuatro salidas y una entrada de habilitación.

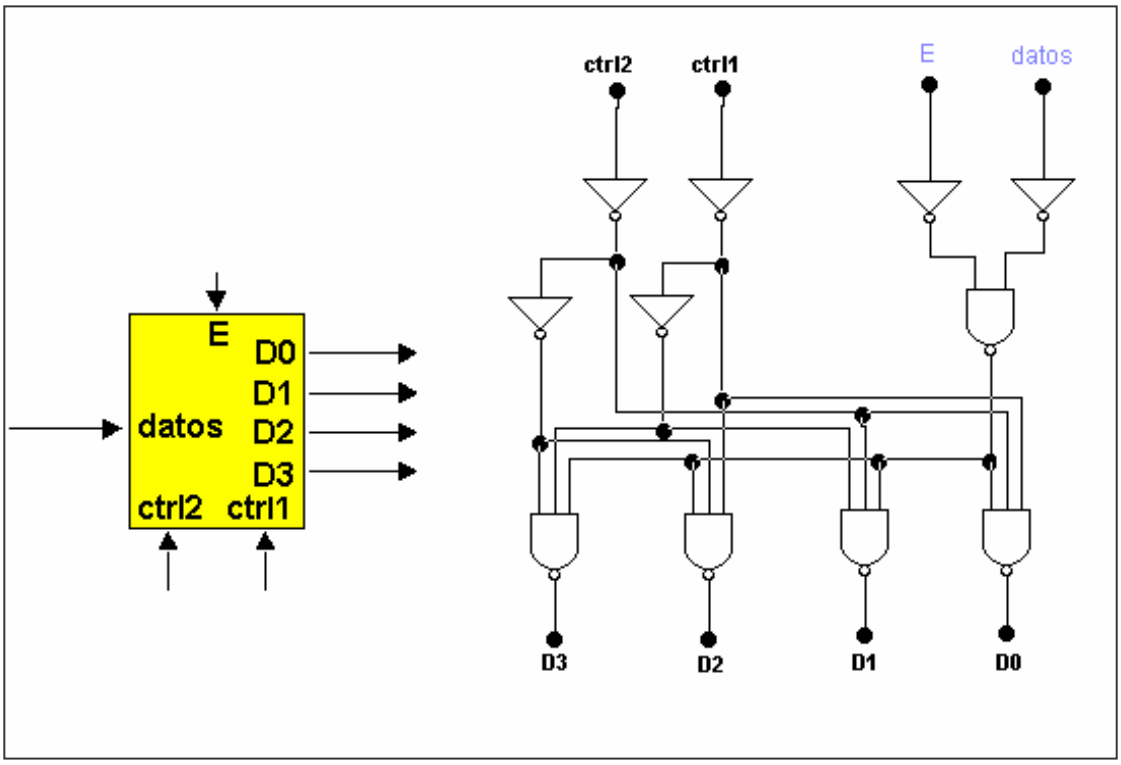


Figura 5.7. Demultiplexor

Decodificadores

Un decodificador de uso común tiene cuatro líneas de entrada y diez líneas de salida, decodificador decimal, de manera que la combinación presente en su entrada se indica a su salida activando, normalmente a nivel bajo, con un 0, la línea correspondiente. Así la combinación dcba = 0000 hace que la línea de salida 0 esté a nivel bajo, con 0, manteniéndose las restantes a nivel alto, con 1. Su bloque representativo se muestra el la figura 5.8.

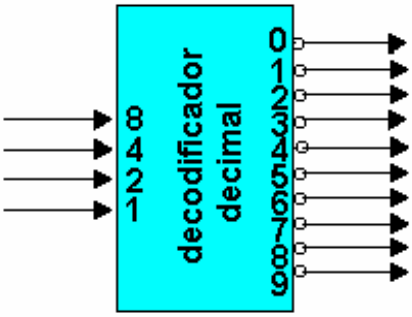
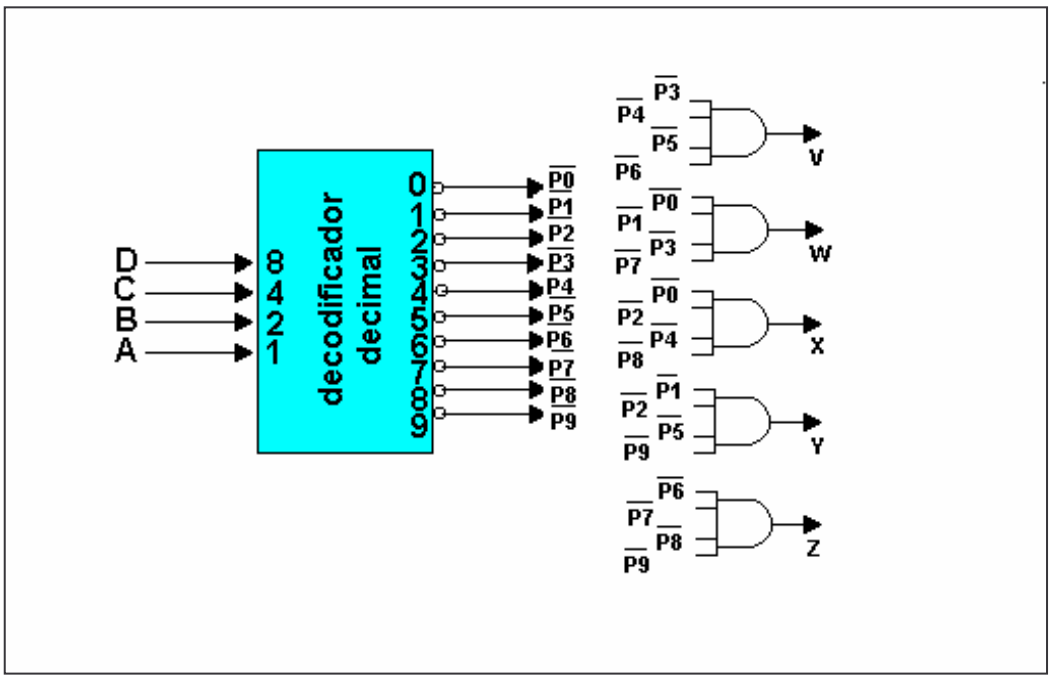


Figura 5.9 Representación de un decodificador decimal

Aplicaciones de los demultiplexores

Además de su aplicación normal, como decodificador, si se utiliza con una cierta cantidad de lógica añadida, puede utilizarse como cambiador de código como muestra el ejemplo de la siguiente figura en la que se utiliza para cambiar entre el código NBCD y el código 3 entre 5.

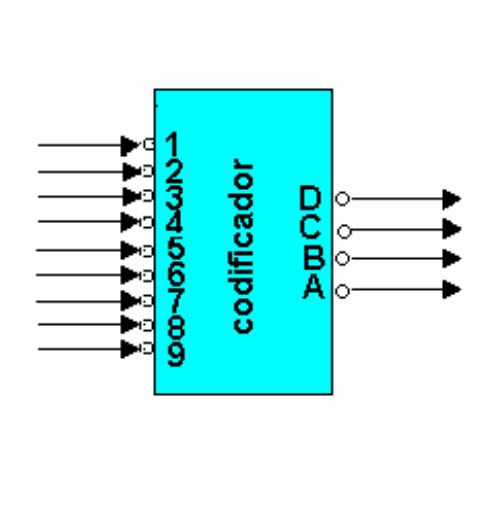
	NBCD				Código 3 entre 5				
	d	c	b	a	v	w	x	y	z
P_0	0	0	0	0	1	0	0	1	1
P_1	0	0	0	1	1	0	1	0	1
P_2	0	0	1	0	1	1	0	0	1
P_3	0	0	1	1	0	0	1	1	1
P_4	0	1	0	0	0	1	0	1	1
P_5	0	1	0	1	0	1	1	0	1
P_6	0	1	1	0	0	1	1	1	0
P_7	0	1	1	1	1	0	1	1	0
P_8	1	0	0	0	1	1	0	1	0
P_9	1	0	0	1	1	1	1	0	0



Codificadores

La función que realiza un codificador es la de asignar una combinación binaria a sus salidas para cada una de sus entradas. Se ha de cumplir que el número de entradas sea igual a $2^{nº\ de\ salidas}$

La definición anterior implica que no puede haber simultáneamente más de una entrada activa. Este inconveniente se solventa en los llamados *codificadores con prioridad*, de manera que al serle asignado a sus entradas una prioridad, estas pueden estar activas simultáneamente ya que solamente se codificará la más prioritaria de las activas. La siguiente tabla expresa este funcionamiento para un codificador con nueve entradas, siendo la prioridad mas alta la del 9 y la mas baja la del 1.



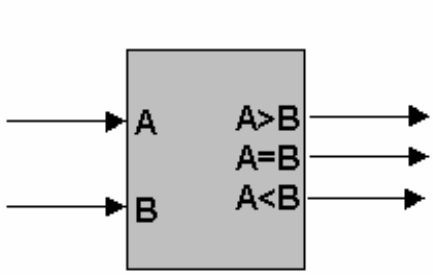
entradas									salidas
1	2	3	4	5	6	7	8	9	
H	H	H	H	H	H	H	H	H	H H H H
X	X	X	X	X	X	X	X	L	L H H L
X	X	X	X	X	X	X	L	H	L H H H
X	X	X	X	X	X	L	H	H	H L L L
X	X	X	X	X	L	H	H	H	H L L H
X	X	X	X	L	H	H	H	H	H L H L
X	X	X	L	H	H	H	H	H	H L HH
X	X	L	H	H	H	H	H	H	H H L L
X	L	H	H	H	H	H	H	H	H H L H
L	H	H	H	H	H	H	H	H	H H H L

Comparadores

Comparar dos números es indicar en que situación se encuentra uno respecto del otro, si uno es mayor, igual o menor que el otro.

Para un dispositivo lógico que realice la función de comparador de números con un solo bit, podemos expresar su funcionamiento y diseño de la siguiente forma:

A	B	A>B	A=B	A<B
0	0	0	0	1
0	1	0	1	0
1	0	1	0	0
1	1	0	0	1



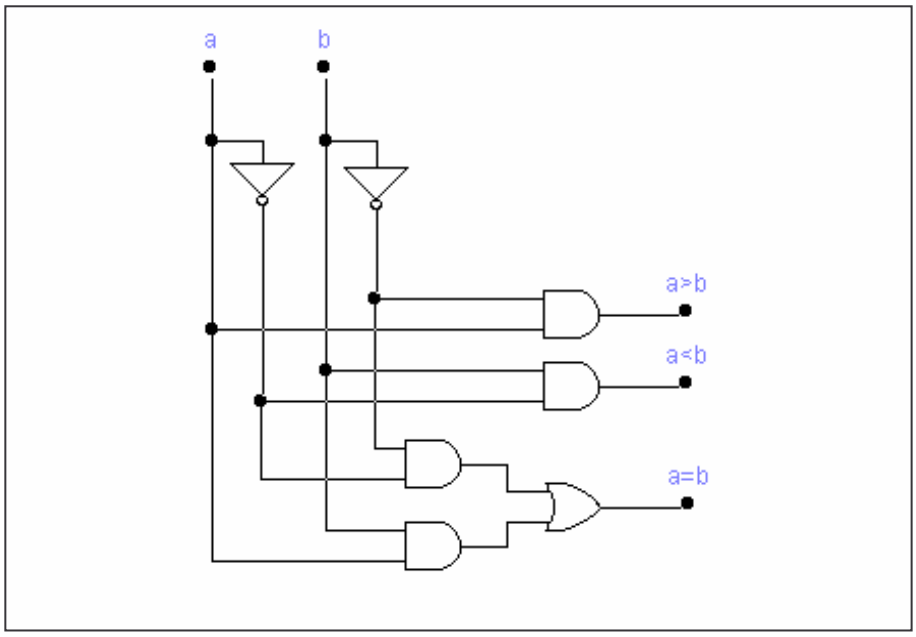


Figura 5.10. Comparador de 2 números de 1 bit

Para realizar comparaciones de números mayores, mas de un bit, se compara el bit más significativo de ambos números, y sólo cuando son iguales se continúa la comparación con el siguiente bit en peso, así hasta terminar con el bit menos significativo, caso de dos números iguales.

La siguiente figura muestra gráficamente este concepto para números de 4 bits.

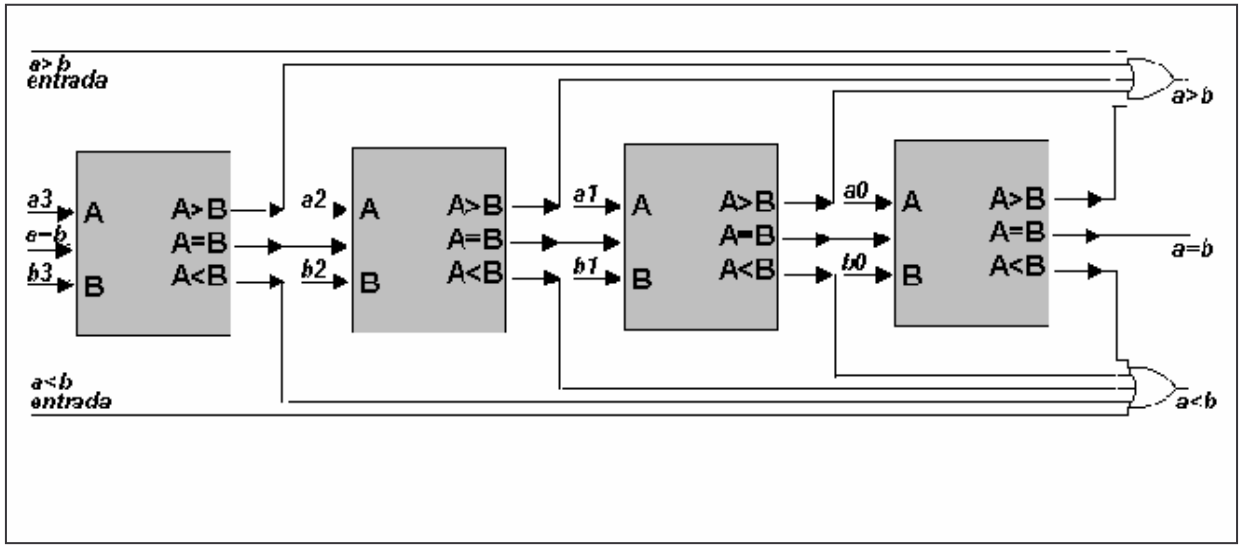
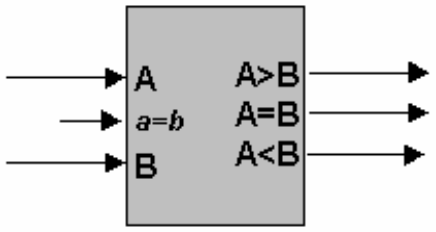


Figura 5.11 Comparador de números de 4 bits

El análisis del esquema indica la posibilidad de encontrarse todas las salidas a 1 lógico, es necesario por tanto inhibir el funcionamiento de las etapas de comparación, de manera que solamente si los bits comparados son iguales, la siguiente etapa de comparación actúe. Además el diseño debe contemplar la posibilidad de conexión en cascada del dispositivo para comparaciones de números de mayor longitud de bits. Esto se consigue añadiendo al diseño las entradas externas *a>b, a=b y a<b.*

Con las modificaciones hechas que recoge la figura anterior, los bloques quedarían de la forma indicada en la siguiente tabla:

a=b	A	B	A>B	A=B	A<B
0	X	X	0	0	0
1	0	0	0	1	0
1	0	1	0	0	1
1	1	0	1	0	0
1	1	1	0	1	0



La figura 5.12 de la siguiente página muestra la implementación de este bloque mediante puertas. En la misma figura se representa el bloque funcional del CI 7485, un comparador de números de 4 bits con la posibilidad de conexión en cascada.

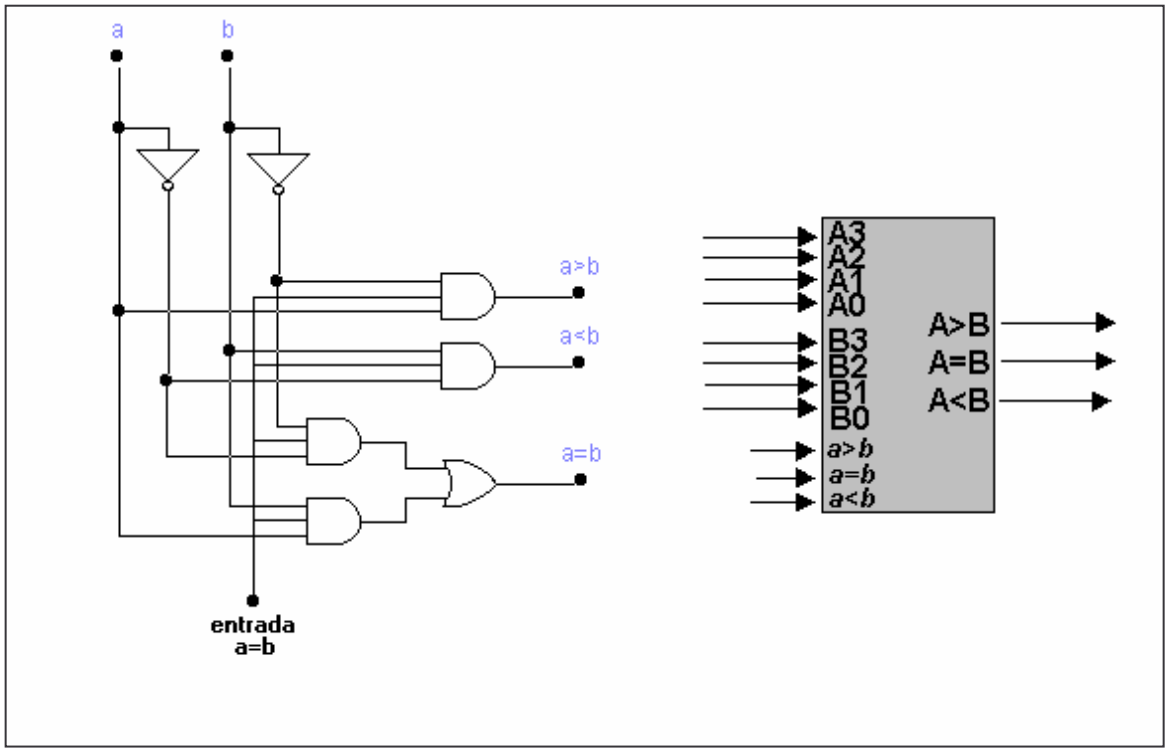
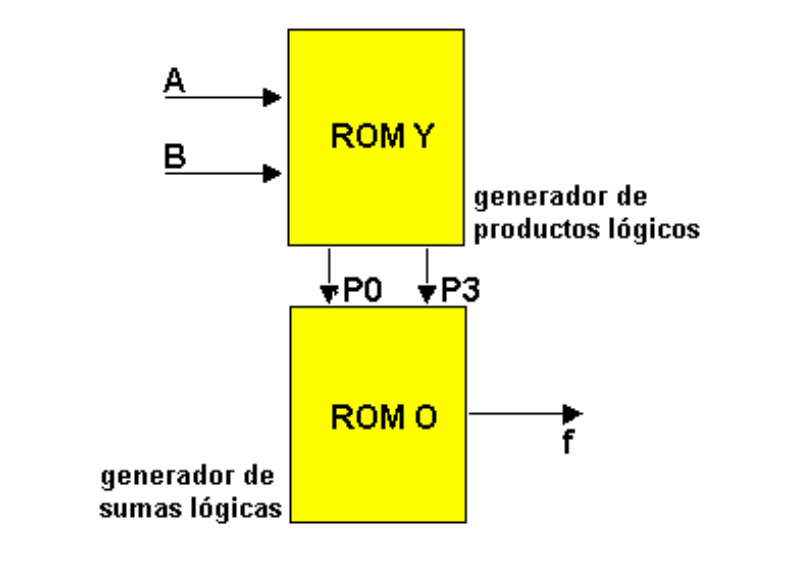


Figura 5.12. Circuito Comparador

Pla's

Un array lógico programaba (PLA, en inglés Programmable Logic Array) es un chip MSI que puede considerarse constituida por dos memorias ROM por separado. Una se denomina ROM Y, o generador de producto lógico, y la otra se denomina ROM 0, o generador de suma lógica. La siguiente figura muestra el esquema en bloques de un PLA sencillo.



En el esquema las líneas A y B son entradas a la ROM Y y las salidas son los términos producto P_0 y P_3 (AB y $\overline{AB}$) que se generan en líneas separadas. Estos términos producto son entradas a la ROM 0 donde se genera la suma lógica. Así en el esquema se ha generado la función $f = P_0 + P_3 = AB + \overline{AB} = A \veebar B$
El esquema en bloques puede representarse mediante una matriz de conexiones como se muestra en la figura siguiente.

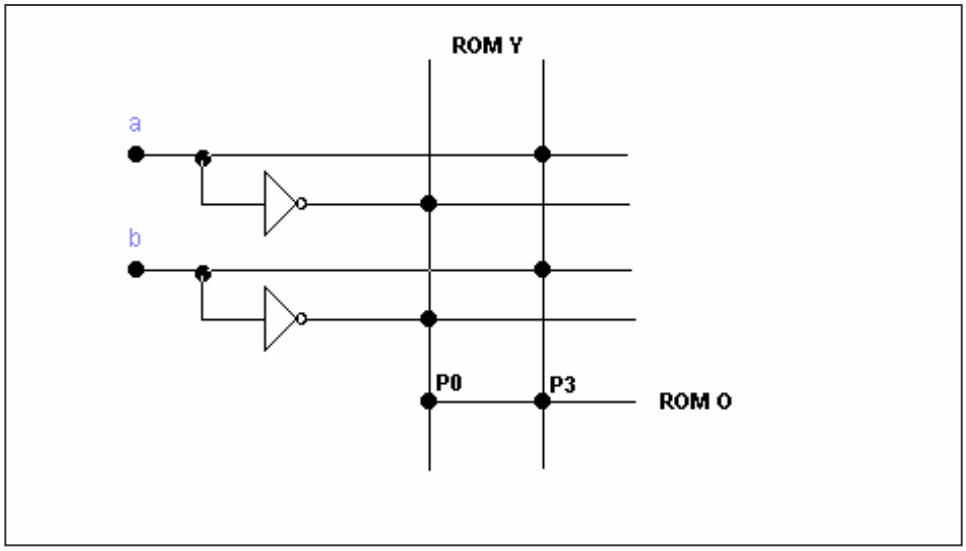


Figura 5.13. Matriz de conexiones

El número de líneas verticales en el array Y determina el número de términos Y que se pueden generar. En el ejemplo sólo hay dos líneas verticales y por tanto sólo se pueden generar dos términos producto posibles. El número de líneas horizontales en el array 0 define el número de funciones de salida, en este caso una.
Las funciones de salida se generan siempre en suma de productos (S de P) de dos niveles, pudiéndose generar con un array lógico programaba tantos términos elementales como términos canónicos.
Un PLA puede usarse como generador de funciones Booleanas. El siguiente ejemplo muestra la implementación de las cuatro funciones de seis variables:

$$f_1 = fedcba+fea+dca+db$$
$$1 \quad 2 \quad 3 \quad 4$$
$$f_1 = feba+dca+db+fe$$
$$5 \quad 6 \quad 4 \quad 7$$
$$f_1 = fedca+dcba+dca+dba$$
$$8 \quad 9 \quad 10 \quad 11$$
$$f_1 = edcba+fea+db+dc+fe$$
$$12 \quad 13 \quad 4 \quad 14 \quad 7$$

Una inspección de las ecuaciones anteriores indica que hay 14 términos producto separados, algunos en forma canónica y otros no. Dado que hay 6 variables de entrada y 14 términos que se deben generar, se requiere un array Y de 12x14 y un array O de 4x12, tal como muestra la figura 5.14.

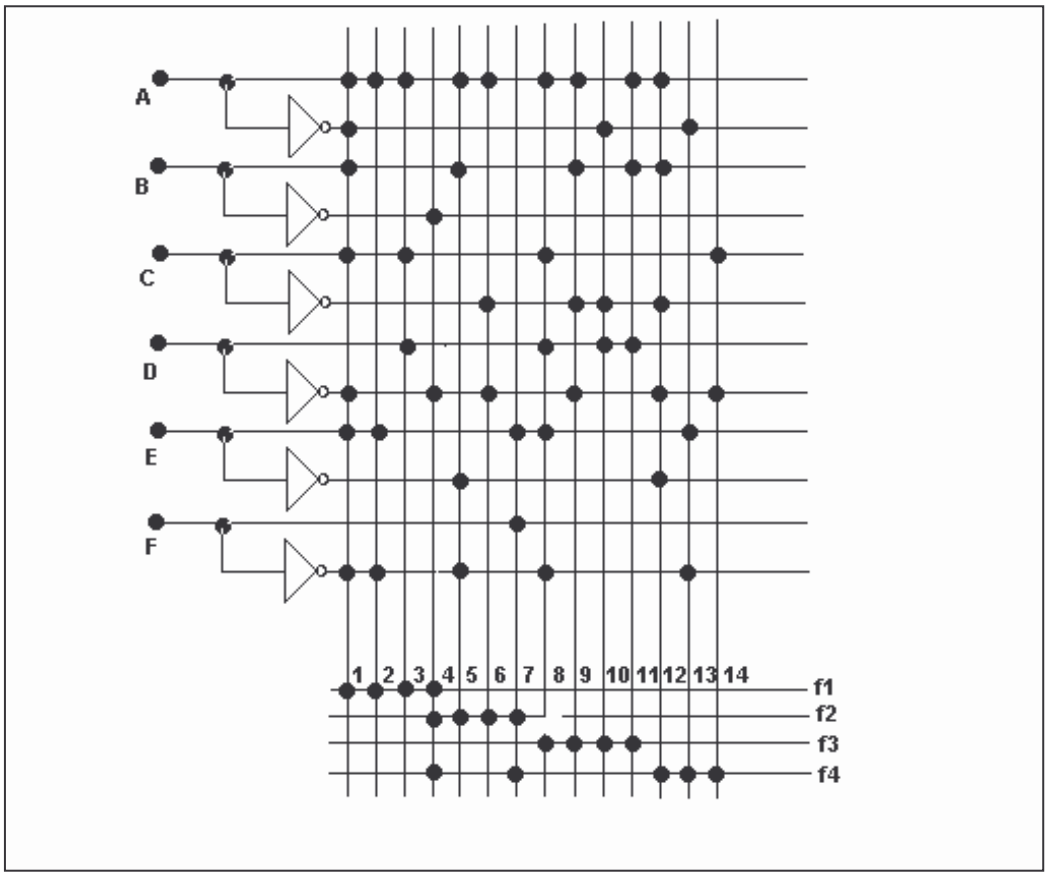


Figura 5.14. Implementación de f1,f2,f3 y f4 con un PLA

En total se necesita un PLA con capacidad de almacenamiento de 224 bits para la implementación de estas 4 funciones.