Tema 4

FUNCIONES DE RUTA DE DATOS

4.1. INTRODUCCIÓN

Hemos visto en el tema anterior que mediante chips MSI podíamos implementar funciones aritméticas y lógicas con un único circuito integrado. En este tema veremos que con estos chips MSI también podemos implementar lo que conocemos como funciones de Ruta de Datos y los Conversores de Código.

Los dispositivos que veremos a continuación son los siguientes:

- **Mux**: selecciona una de entre 2^n entradas en función de n líneas de control.
- **Demux**: lleva la entrada a una de las 2^n salidas en función de n líneas de control.
- **Codificador**: con 2^n entradas, de las cuales sólo una de ellas es activa, genera en las n salidas el código binario asociado a esa línea (código de n bits).
- **Decodificador**: el código binario generado por las n entradas activa una de entre 2^n salidas.
- **Conversor de código**: Con un número arbitrario de entradas y salidas transforma las entradas de un código en salidas de otro.

4.2. MUX O MULTIPLEXO

Es un circuito selector de datos, es decir, la operación de este dispositivo es seleccionar una de entre varias entradas y llevar su valor a la salida. Para realizar esta selección son precisas líneas de control que nos indiquen cual de las entradas es la seleccionada. Si

Líneas de control			Selección
a	b	c	
0	0	0	entrada i_0
0	0	1	entrada i_1
0	1	0	entrada i_2
0	1	1	entrada i_3
1	0	0	entrada i_4
1	0	1	entrada i_5
1	1	0	entrada i_6
1	1	1	entrada i_7

Cuadro 4.1: Tabla de funcionamiento de un MUX 8 a 1.

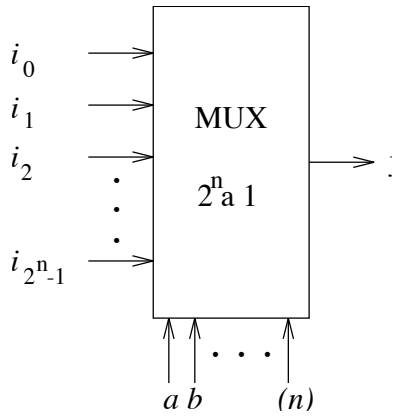


Figura 4.1: Representación de un MUX.

disponemos de 2^n entradas son precisas n líneas de control, para referenciar a cada una de ellas. En la tabla 4.1 vemos un ejemplo para un MUX con n=3, donde tenemos 3 líneas de control (a, b, c) y 8 entradas (desde i_0 hasta i_7).

Por tanto podemos definir el MUX 2^n a 1 como aquel dispositivo con 2^n entradas, una salida y n variables de control, de forma que el código binario contenido en las líneas de control indica cual de las entradas es la que se conecta a la salida.

4.2.1. Construcción de un MUX

En la tabla 4.2 presentamos las tablas de verdad del MUX 4 a 1 y del MUX 8 a 1. Se puede observar que solamente se trasmite a la salida el valor (0 ó 1) de la entrada i seleccionada, no influyendo en la misma las demás entradas, donde hemos puesto "x". Por ejemplo, para el MUX 4 a 1 si $ab = 00$ a la salida el valor de y será el que haya en i_0, independientemente de los valores de i_1, i_2 e i_3, es decir, para $ab = 00$ e $i_0 = 0$ la salida será siempre 0 para cualquier combinación de valores de las otras tres entradas $i_1 i_2 i_3$ desde 000 hasta 111.

a	b	i_0	i_1	i_2	i_3	y
0	0	0	x	x	x	0
0	0	1	x	x	x	1
0	1	x	0	x	x	0
0	1	x	1	x	x	1
1	0	x	x	0	x	0
1	0	x	x	1	x	1
1	1	x	x	x	0	0
1	1	x	x	x	1	1

a	b	c	i_0	i_1	i_2	i_3	i_4	i_5	i_6	i_7	y
0	0	0	0	x	x	x	x	x	x	x	0
0	0	0	1	x	x	x	x	x	x	x	1
0	0	1	x	0	x	x	x	x	x	x	0
0	0	1	x	1	x	x	x	x	x	x	1
0	1	0	x	x	0	x	x	x	x	x	0
0	1	0	x	x	1	x	x	x	x	x	1
0	1	1	x	x	x	0	x	x	x	x	0
0	1	1	x	x	x	1	x	x	x	x	1
1	0	0	x	x	x	x	0	x	x	x	0
1	0	0	x	x	x	x	1	x	x	x	1
1	0	1	x	x	x	x	x	0	x	x	0
1	0	1	x	x	x	x	x	1	x	x	1
1	1	0	x	x	x	x	x	x	0	x	0
1	1	0	x	x	x	x	x	x	1	x	1
1	1	1	x	x	x	x	x	x	x	0	0
1	1	1	x	x	x	x	x	x	x	1	1

Cuadro 4.2: Tablas de verdad para los MUX's 4 a 1 y 8 a 1.

Las expresiónes lógicas de las salidas son las siguientes:

MUX 4 a 1: $y = \overline{a}\,\overline{b}i_0 + \overline{a}bi_1 + a\overline{b}i_2 + abi_3$

MUX 8 a 1: $y = \overline{a}\,\overline{b}\,\overline{c}i_0 + \overline{a}\,\overline{b}ci_1 + \overline{a}b\overline{c}i_2 + \overline{a}bci_3 + a\overline{b}\,\overline{c}i_4 + a\overline{b}ci_5 + ab\overline{c}i_6 + abci_7$

La obtención de estas ecuaciones a partir de la tabla de verdad del MUX es similar a la construcción de funciones en forma de suma de minterm, pero esta vez las variables de entrada que son "x" no intervienen en la formación de los términos producto.

Normalmente se suele incluir una señal de "enable" o "strobe" (s) para la inhibición del dispositivo, con el siguiente funcionamiento:

- $s = 0$: El circuito está inhibido y la salida es siempre cero ($y = 0$).
- $s = 1$: Funcionamiento normal, la salida es igual a la entrada seleccionada.

La inclusión de esta entrada en las expresiónes lógicas se realiza simplemente multiplicando cada término producto por s:

MUX 4 a 1: $y = s\overline{a}\,\overline{b}i_0 + s\overline{a}bi_1 + sa\overline{b}i_2 + sabi_3$

MUX 8 a 1: $y = s\overline{a}\,\overline{b}\,\overline{c}i_0 + s\overline{a}\,\overline{b}ci_1 + s\overline{a}b\overline{c}i_2 + s\overline{a}bci_3 + sa\overline{b}\,\overline{c}i_4 + sa\overline{b}ci_5 + sab\overline{c}i_6 + sabci_7$

La construcción de estos MUX's a partir de puertas lógicas se muestra en la figura 4.2.

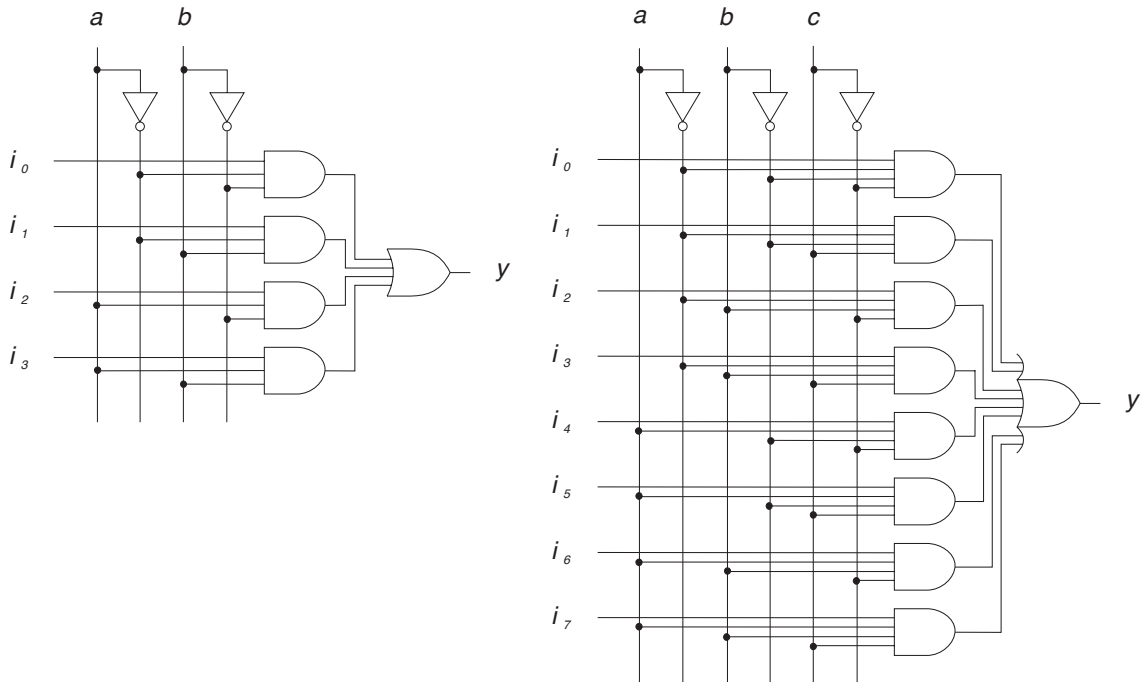


Figura 4.2: Construcciones del MUX 4 a 1 y del MUX 8 a 1.

4.2.2. Árboles multiplexores

El mayor MUX comercial disponible en forma de chip es de tamaño 16 a 1, pero podemos construir MUX's de cualquier tamaño interconectando varios MUX en una estructura de árbol. Por ejemplo, podemos realizar un MUX 32 a 1 a partir de cuatro MUX 8 a 1 y un MUX 4 a 1, tal como se muestra en la figura 4.3.

Cada MUX del primer nivel selecciona una de sus 8 entradas dependiendo de los bits de control comunes c, d y e. El MUX del segundo nivel selecciona una de las salidas de los MUX's del primer nivel en función de los bits de control a y b. El resultado final es que la salida toma el valor de una de las 32 entradas en función de las cinco líneas de control a, b, c, d y e. Notar que al MUX del segundo nivel (MUX 4 a 1) van las líneas de control más significativas.

El tamaño del MUX global se obtiene multiplicando los tamaños de los MUX de los dos niveles. En este caso, MUX's 8 a 1 y un MUX 4 a 1 dan lugar a un MUX 8×4 a 1 (MUX 32 a 1). También se pueden construir árboles multiplexores de cualquier número de entradas sin más que añadir niveles de MUX's.

4.3. DEMUX O DEMULTIPLEXO

Es un circuito deselector de datos, es decir, la operación de este dispositivo consiste en tomar la única entrada, seleccionar una de entre varias salidas y conectarla a la entrada.

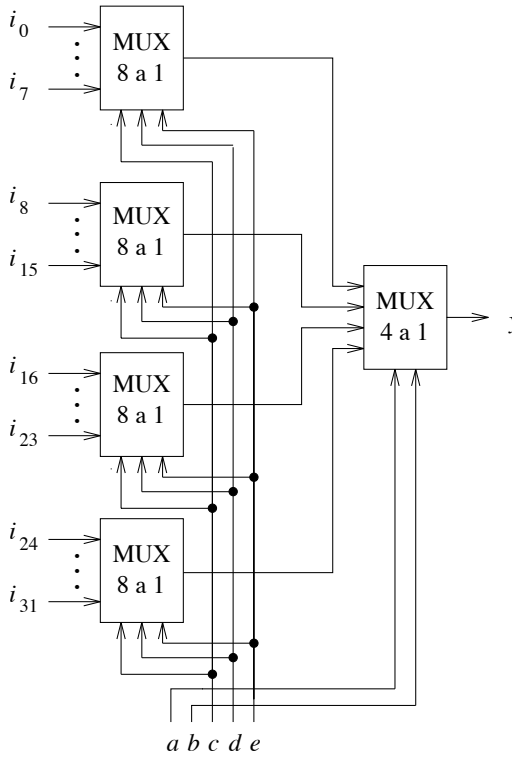


Figura 4.3: Árbol multiplexor 32 a 1.

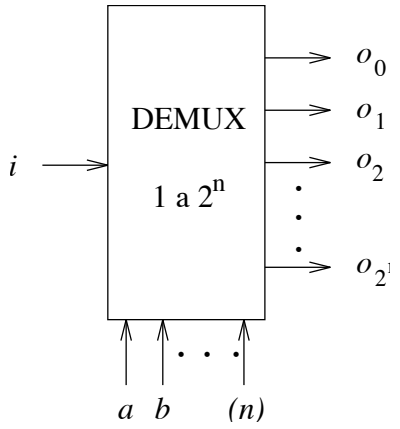


Figura 4.4: Representación de un DEMUX.

Para realizar esta selección son precisas líneas de control que nos indiquen cual de las salidas es la seleccionada. Si disponemos de 2^n salidas son precisas n líneas de control, para referenciar cada una de ellas (DEMUX 1 a 2^n).

Por tanto podemos definir el DEMUX 1 a 2^n como aquel dispositivo con 1 entrada, 2^n salidas, y n variables de control, de forma que el código binario contenido en las líneas de control indica cual de las salidas es la que se conecta a la entrada. El resto de las salidas toman un valor inactivo ("0" si son activas a tensión alta ó "1" si son activas a tensión baja).

4.3.1. Construcción de un DEMUX

Las tablas de verdad del DEMUX 1 a 4 y del DEMUX 1 a 8 (lógica positiva) son las siguientes:

a	b	i	o_0	o_1	o_2	o_3
0	0	0	0	0	0	0
0	0	1	1	0	0	0
0	1	0	0	0	0	0
0	1	1	0	1	0	0
1	0	0	0	0	0	0
1	0	1	0	0	1	0
1	1	0	0	0	0	0
1	1	1	0	0	0	1

a	b	c	i	o_0	o_1	o_2	o_3	o_4	o_5	o_6	o_7
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0	0
0	0	1	1	0	1	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0
0	1	0	1	0	0	1	0	0	0	0	0
0	1	1	0	0	0	0	0	0	0	0	0
0	1	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	0	0	0	0
1	0	1	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	0	0	0
1	1	0	1	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0	1

Las expresiones lógicas de las salidas son:

DEMUX 1 a 4: $o_0 = \overline{a}\,\overline{b}i$, $o_1 = \overline{a}bi$, $o_2 = a\overline{b}i$, $o_3 = abi$.

DEMUX 1 a 8: $o_0 = \overline{a}\,\overline{b}\,\overline{c}i$, $o_1 = \overline{a}\,\overline{b}ci$, $o_2 = \overline{a}b\overline{c}i$, $o_3 = \overline{a}bci$, $o_4 = a\overline{b}\,\overline{c}i$, $o_5 = a\overline{b}ci$, $o_6 = ab\overline{c}i$ y $o_7 = abci$.

Al igual que en el caso del MUX, normalmente se suele incluir una señal de "enable" o "strobe" (s) para la inhibición del dispositivo, con el siguiente funcionamiento:

- $s = 0$: El circuito está inhibido y todas las salidas son siempre cero ($o_i = 0$, para todo i).

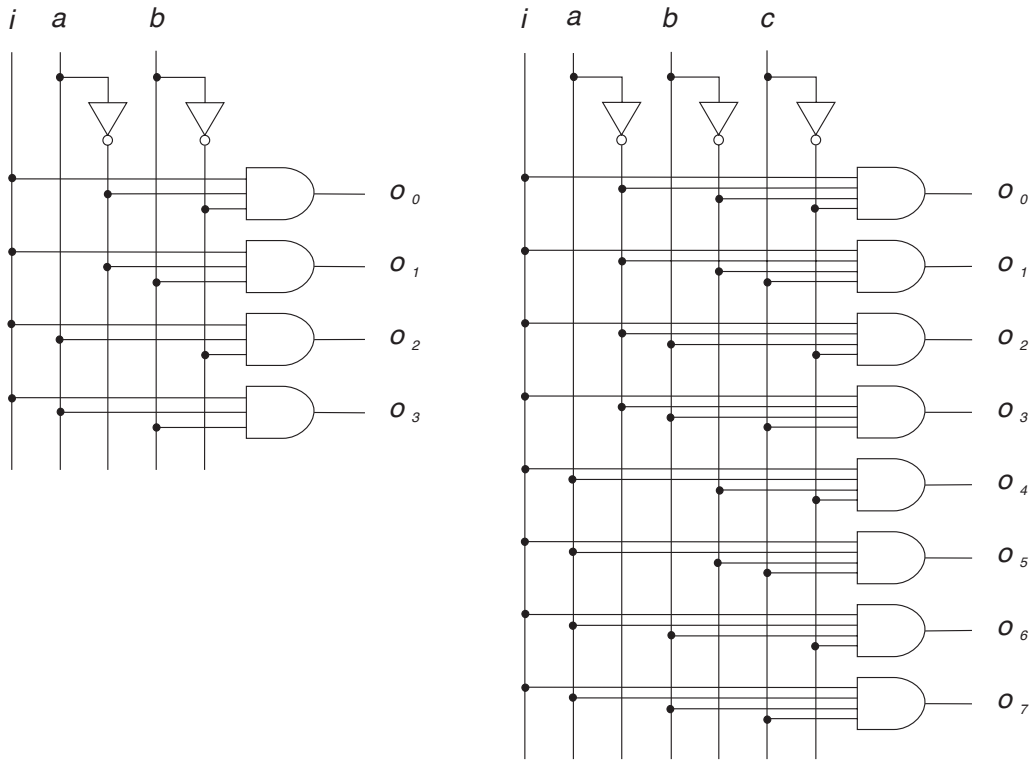


Figura 4.5: Construcciones del DEMUX 1 a 4 y del DEMUX 1 a 8.

- $s = 1$: Funcionamiento normal, la salida seleccionada es igual a la entrada.

La inclusión de esta entrada en las expresiónes lógicas se realiza simplemente multiplicando cada término producto por s:

DEMUX 1 a 4: $o_0 = s\overline{a}\,\overline{b}i$, $o_1 = s\overline{a}bi$, $o_2 = sa\overline{b}i$, $o_3 = sabi$.

DEMUX 1 a 8: $o_0 = s\overline{a}\,\overline{b}\,\overline{c}i$, $o_1 = s\overline{a}\,\overline{b}ci$, $o_2 = s\overline{a}b\overline{c}i$, $o_3 = s\overline{a}bci$, $o_4 = sa\overline{b}\,\overline{c}i$, $o_5 = sa\overline{b}ci$, $o_6 = sab\overline{c}i$ y $o_7 = sabci$.

La construcción de estos DEMUX's a partir de puertas lógicas es la mostrada en la figura 4.5.

4.3.2. Árboles demultiplexores

El mayor DEMUX comercial disponible en forma de chip es de tamaño 1 a 16, pero podemos construir DEMUX's de cualquier tamaño interconectando varios DEMUX en una estructura de árbol. Por ejemplo, podemos implementar un DEMUX 1 a 32 a partir de un DEMUX 1 a 4 y cuatro DEMUX's 1 a 8, tal como se muestra en la figura 4.6.

El DEMUX del primer nivel lleva la entrada a una de sus cuatro salidas dependiendo de los bits de control a y b. Los DEMUX's del segundo nivel llevan cada una de las salidas

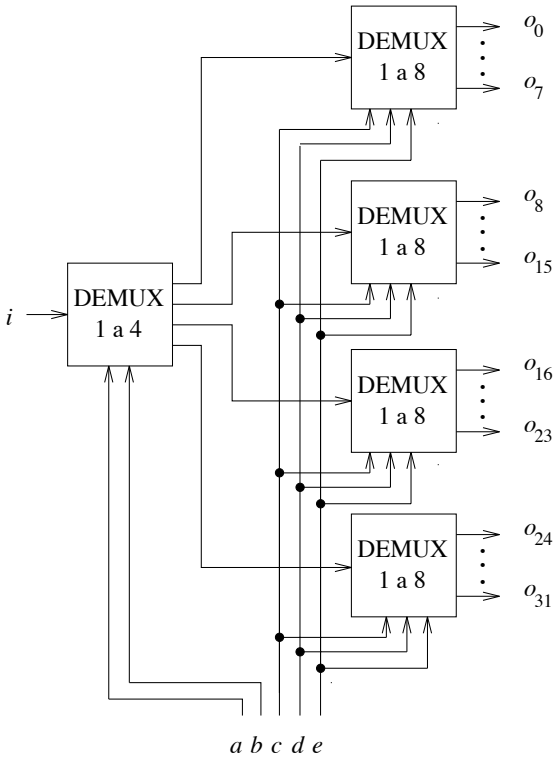


Figura 4.6: Árbol demultiplexor 1 a 32.

del DEMUX del primer nivel a la salida seleccionada en función de los bits de control comunes c, d y e. El resultado final es que la entrada se lleva a una de las 32 salidas en función de las cinco líneas de control a, b, c, d y e. Notar que al DEMUX del primer nivel (DEMUX 1 a 4) van las líneas de control más significativas.

El tamaño del DEMUX global se obtiene multiplicando los tamaños de los DEMUX de los dos niveles. En este caso, el DEMUX 1 a 4 y los DEMUX's 1 a 8 dan lugar a un DEMUX 1 a 4×8 (DEMUX 1 a 32). También se pueden construir árboles demultiplexores de cualquier número de salidas sin más que añadir niveles de DEMUX's.

4.4. DECODIFICADORES

Vamos a referirnos únicamente a los decodificadores binarios. La función de este dispositivo es tomar el código binario de la entrada y activar (poner a 1) la línea de salida que corresponde a ese código binario, dejando el resto de las salidas inactivas (proceso que se denomina decodificación). Un decodificador n a 2^n presentará n entradas y 2^n salidas.

Las tablas de verdad del decodificador binario 2 a 4 y del decodificador binario 3 a 8 las mostramos en la tabla 4.3.

Las expresiones lógicas de las salidas son:

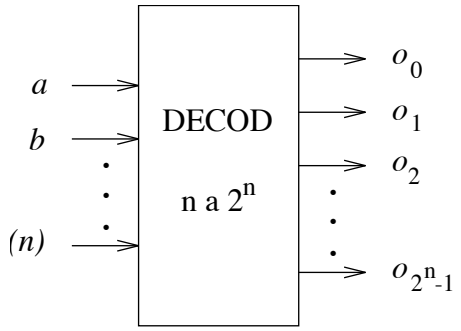


Figura 4.7: Representación de un decodificador.

a	b	o_0	o_1	o_2	o_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

a	b	c	o_0	o_1	o_2	o_3	o_4	o_5	o_6	o_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

Cuadro 4.3: Tablas de verdad para un decodificador 2 a 4 y para un decodificador 3 a 8.

Decodificador 2 a 4: $o_0 = \overline{a}\,\overline{b}$, $o_1 = \overline{a}b$, $o_2 = a\overline{b}$, $o_3 = ab$.

Decodificador 3 a 8: $o_0 = \overline{a}\,\overline{b}\,\overline{c}$, $o_1 = \overline{a}\,\overline{b}c$, $o_2 = \overline{a}b\overline{c}$, $o_3 = \overline{a}bc$, $o_4 = a\overline{b}\,\overline{c}$, $o_5 = a\overline{b}c$, $o_6 = ab\overline{c}$ y $o_7 = abc$.

Estas expresiones son exactamente iguales a las de los DEMUX, pero con la diferencia de que no incluyen la entrada i. Por tanto los decodificadores binarios no se suelen construir como tales; lo que se hace es partir de un DEMUX y hacer la entrada dato $i = 1$. También se puede considerar un DEMUX como un decodificador con señal de strobe, donde la entrada i estaría haciendo esta función. Teniendo en cuenta esto, también podemos concluir que decodificadores mayores de 4 a 16 pueden ser construidos a partir de árboles demultiplexores poniendo la primera entrada a uno ($i = 1$).

4.5. CODIFICADORES

Vamos a referirnos únicamente a los codificadores binarios. Un codificador es el dispositivo inverso a un decodificador. La función de este dispositivo es generar el código binario de la única línea de entrada que está activa en cada instante de un conjunto de varias entradas (proceso denominado codificación). Un codificador 2^n a n presentará 2^n entradas y n salidas. En principio sólo se podrá poner a 1 una de las 2^n entradas.

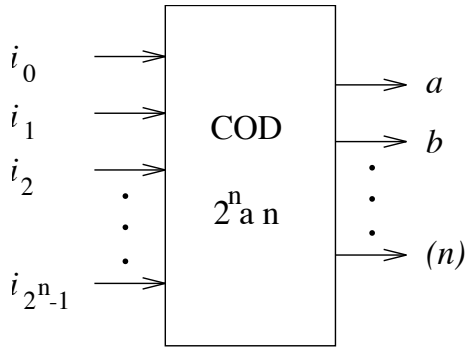


Figura 4.8: Representación de un codificador.

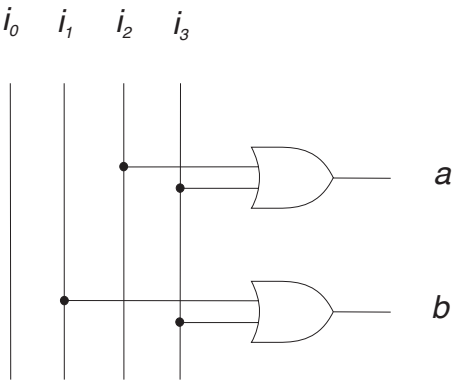


Figura 4.9: Codificador 4 a 2.

Por ejemplo, las tablas de un codificador binario 4 a 2 y de un codificador 8 a 3 son las siguientes, donde sólo hemos hemos incluido las combinaciones de entrada permitidas:

i_0	i_1	i_2	i_3	a	b
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1
otras comb.				–	–

i_0	i_1	i_2	i_3	i_4	i_5	i_6	i_7	a	b	c
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1
otras combinaciones								–	–	–

La expresión lógica de las salidas es la siguiente:

Codificador 4 a 2: $a = i_2 + i_3$, $b = i_1 + i_3$.

Codificador 8 a 3: $a = i_4 + i_5 + i_6 + i_7$, $b = i_2 + i_3 + i_6 + i_7$, $c = i_1 + i_3 + i_5 + i_7$.

Como puede observarse las expresiones de las salidas son la suma lógica de los términos de las líneas de entrada a 1 que ponen dicha salida a 1. Estas expresiones sencillas se deben al gran número de indiferencias que presentan las salidas.

4.5.1. Codificadores con prioridad

Cabe preguntarse qué sucede en el diseño anterior cuando se ponen varias de las líneas de entrada a 1, cuál de los códigos binarios asociados a cada una de esas líneas de entrada es el que se tomará como salida. Tal como hemos diseñado el dispositivo (poniendo indiferencias en las salidas no permitidas) no podemos decir nada sobre esta cuestión.

Es posible imponer prioridades a las líneas de entrada, de tal forma que si varias de ellas están activas el codificador sólo tendrá en cuenta a la más prioritaria. Supongamos que queremos construir un codificador 4 a 2 tal que las líneas de mayor peso sean las más prioritarias: $i_3 > i_2 > i_1 > i_0$. El orden $i_3 > i_2 > i_1 > i_0$ es el orden de prioridad más usual y éstos van a ser los circuitos codificadores que se encuentren en el mercado.

La tabla de verdad de un codificador de este tipo es la siguiente:

i_0	i_1	i_2	i_3	a	b
x	x	x	1	1	1
x	x	1	0	1	0
x	1	0	0	0	1
1	0	0	0	0	0
0	0	0	0	0	0

$$a = i_3 + i_2\bar{i}_3 = i_3 + i_2$$

$$b = i_3 + i_1\bar{i}_2\bar{i}_3 = i_3 + i_1\bar{i}_2$$

En la tabla de verdad, sólo ha de tenerse en cuenta la línea más prioritaria a uno. Así, por ejemplo, si $i_2 = 1$ e $i_3 = 0$ sabemos que la salida ha de ser 2, independientemente de los valores de las líneas i_0 e i_1 (segunda fila de la tabla). Por otro lado, la obtención de estas ecuaciones a partir de la tabla de verdad es similar a la construcción de funciones en forma de suma de minterm, pero teniendo en cuenta que las variables de entrada que son "x" no intervienen en la formación del término producto. A diferencia con un codificador sin prioridad, en un codificador con prioridad todas las combinaciones de entrada tienen definido un valor de salida y, por lo tanto, no hay indiferencias en las funciones de salida del codificador.

Como comentario final, indicar que los codificadores comerciales binarios pueden llegar a ser de 16 a 4. Para diseñar codificadores mayores no es posible construir árboles de decodificadores siguiendo el método para MUX y DEMUX, y habría que estudiar cada caso en particular.

4.6. CONVERSORES DE CÓDIGO

Un coversor de código es un dispositivo que genera la traducción entre dos códigos diferentes. Los números de entradas y salidas de este dispositivo vienen dados respecti-

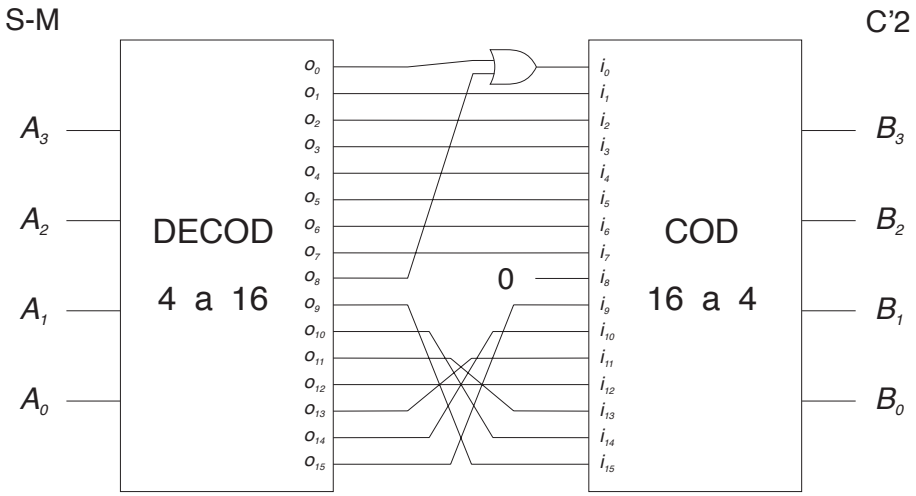


Figura 4.10: Diseño de un conversor S-M a C'2 para números de 4 bits.

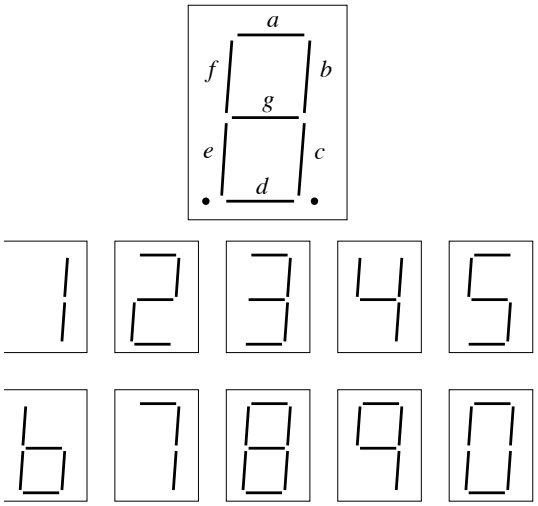


Figura 4.11: Display de siete segmentos.

vamente por la longitud del código de partida y del código traducido. La construcción de estos dispositivos es particular para cada tipo de conversión de códigos elegida. Una forma sencilla de realizar conversores es partiendo de decodificadores y codificadores. En la figura 4.10 podemos ver un ejemplo de un conversor de números de 4 bits en formato signo-magnitud a formato complemento a 2, utilizando un decodificador binario 4 a 16 y un codificador binario 16 a 4. Sin embargo, este método tiene el inconveniente de la limitación del tamaño de los codificadores, con lo cual si el código de salida es de más de cuatro bits, ya habría que estudiar una implementación específica para el conversor.

Un ejemplo interesante es la conversión BCD a siete segmentos. Un visualizador (display) de siete segmentos consta de siete segmentos etiquetados a, b, c, d, e, f y g (figura 4.11), que pueden ser iluminados individualmente mediante LED's. El visualizador incluye una entrada de control para cada segmento de forma que si la entrada corres-

ENTRADAS				SALIDAS							
D	C	B	A	a	b	c	d	e	f	g	DISPLAY
0	0	0	0	1	1	1	1	1	1	0	0
0	0	0	1	0	1	1	0	0	0	0	1
0	0	1	0	1	1	0	1	1	0	1	2
0	0	1	1	1	1	1	1	0	0	1	3
0	1	0	0	0	1	1	0	0	1	1	4
0	1	0	1	1	0	1	1	0	1	1	5
0	1	1	0	0	0	1	1	1	1	1	6
0	1	1	1	1	1	1	0	0	0	0	7
1	0	0	0	1	1	1	1	1	1	1	8
1	0	0	1	1	1	1	0	0	1	1	9
1	0	1	0	–	–	–	–	–	–	–	–
1	0	1	1	–	–	–	–	–	–	–	–
1	1	0	0	–	–	–	–	–	–	–	–
1	1	0	1	–	–	–	–	–	–	–	–
1	1	1	0	–	–	–	–	–	–	–	–
1	1	1	1	–	–	–	–	–	–	–	–

Cuadro 4.4: Conversión BCD a 7 segmentos.

pondiente al segmento a está activa éste se iluminará, mientras que si está inactiva el segmento permanecerá apagado. Igual para los seis segmentos restantes.

El codigo BCD (código binario decimal) consta de 4 bits en los cuales las combinaciones posibles son las que generan los números binarios 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, que son precisamente los dígitos que se emplean en el sistema decimal. La conversión BCD a 7 segmentos viene dada por la tabla 4.4.

EJERCICIOS

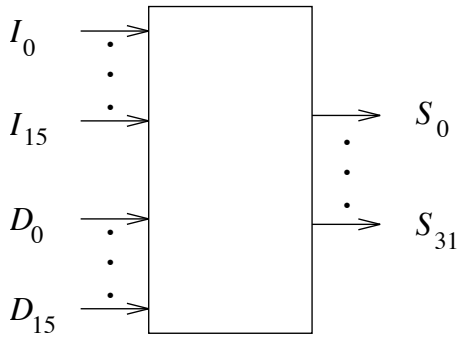
4.1. Construir un sistema que acepte como entradas tres números de cuatro bits y proporcione como salida el número mayor.

4.2. Diseñar con dispositivos MSI (decodificadores, codificadores, MUX y DEMUX), sin utilizar puertas lógicas:

i) Un dispositivo con cuatro entradas, una salida y cuatro señales de control, de modo que si la señal de control i-ésima está activa (valor lógico 1), la salida del dispositivo será igual a la entrada i-ésima. Solamente una de las señales de control estará activa en cada instante.

ii) Un dispositivo que reciba como entrada números comprendidos entre el 0 y el 15 en formato binario puro, y genere una salida que puede tomar 3 valores: igual a 1 si la entrada está comprendida entre el 0 y el 5, igual a 2 si la entrada está entre el 6 y el 10, e igual a 3 si la entrada está entre el 11 y el 15.

4.3. Diseñar un dispositivo con la estructura mostrada en la siguiente figura, donde solamente dos de las entradas tomarán el valor 1 en cada instante (una de cada grupo de entradas), tal que si una de las entradas I se pone a 1 (las demás a 0) y una de las entradas D se pone a 1 (las demás a 0), se active una única línea de salida S, de modo que si $I_x = 1$ y $D_y = 1$ entonces $S_{x\text{PLUS}y} = 1$.



4.4. Contruir el dispositivo definido mediante la siguiente tabla, donde x_0, x_1, x_2 y x_3 son entradas dato, a y b son entradas de control e y_0, y_1, y_2 e y_3 son salidas.

a	b	y_0	y_1	y_2	y_3
0	0	0	0	0	0
0	1	1	1	1	1
1	0	$\overline{x}_0$	$\overline{x}_1$	$\overline{x}_2$	$\overline{x}_3$
1	1	x_0	x_1	x_2	x_3

4.5. Construir un dispositivo con entradas I_i y E_j, $i, j = 0, 1, \ldots, 15$ y salidas S_k, $k = 0, \ldots, 31$, de tal forma que si en cada instante tenemos a una sola de entre cada grupo de 16 entradas a 1 ($I_x = 1$, $E_y = 1$) y el resto a 0 ($I_i = 0\ \forall i \neq x$, $E_j = 0$ $\forall j \neq y$) produzca las siguientes salidas:

i) Si $x > y$ se ha de poner $S_{x\text{MINUS}y}$ a 1 y las otras 31 salidas a 0.

ii) Si $x = y$ se ha de poner $S_{2\times x}$ a 1 y las otras 31 salidas a 0.

iii) Si $x < y$ se ha de poner $S_{y\text{MINUS}x}$ a 1 y las otras 31 salidas a 0.

4.6. La suma en octal de dos dígitos se define de la siguiente forma:

+	0	1	2	3	4	5	6	7
0	00	01	02	03	04	05	06	07
1	01	02	03	04	05	06	07	10
2	02	03	04	05	06	07	10	11
3	03	04	05	06	07	10	11	12
4	04	05	06	07	10	11	12	13
5	05	06	07	10	11	12	13	14
6	06	07	10	11	12	13	14	15
7	07	10	11	12	13	14	15	16

Diseñar un circuito que realice la suma en octal de dos números de 3 bits para dar un resultado de 6 bits, asumiendo la siguiente codificación binaria:

0	011
1	100
2	111
3	000
4	001
5	110
6	010
7	101

así por ejemplo, 4 PLUS 7, con esta codificación sería: 001 PLUS 101 = 100 000.

4.7. Construir un dispositivo que implemente el siguiente algoritmo:

Sean A, B y C enteros de 4 bits en formato binario puro.
If $A > B$ then
    If $A > C$ then (A PLUS B)
    Else (A PLUS C)
Else if $A \leq B$ then
    If $B > C$ then (B PLUS C)
    Else (A PLUS B PLUS C)

4.8. Construir un dispositivo con entradas A, B y C (de 4 bits cada una) y salidas S (de 4 bits) y N (de 2 bits), tal que:

i) La salida S será 0 si las entradas A, B y C son todas distintas.

ii) Si las entradas son todas iguales, S será igual a las entradas.

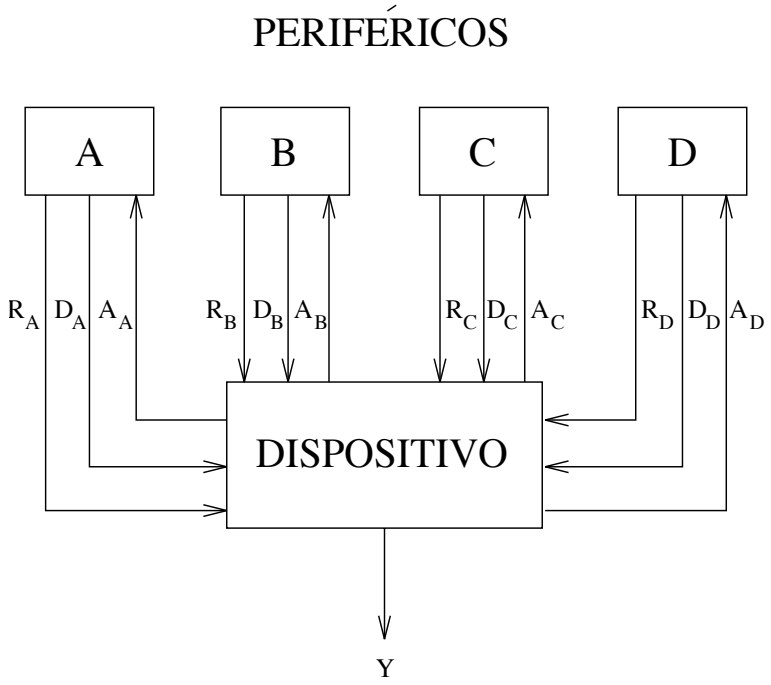


Figura 4.12: Dispositivo controlador de periféricos.

III) Y si hay dos entradas iguales, S será igual a esas dos entradas.

Por otro lado, la salida N proporcionará el número de entradas iguales, es decir, en los casos anteriores valdrá 0, 3 y 2, respectivamente.

4.9. Diseñar un dispositivo encargado de la gestión del envío de datos desde 4 periféricos etiquetados A, B, C y D a un canal de salida Y (ver la figura 4.12), de la siguiente forma:

I) Cuando un periférico desea enviar un dato al canal avisa al dispositivo enviándole una señal de requerimiento (R_A, R_B, R_C o R_D). Como el dispositivo sólo puede atender a un periférico en cada momento, en el caso de que varios periféricos activen simultáneamente su señal de requerimiento, el dispositivo atenderá al periférico de mayor prioridad (con el orden $A > B > C > D$).

II) Una vez que el dispositivo ha decidido atender a un determinado periférico, se lo comunicará a éste activando una señal de aceptación (A_A, A_B, A_C o A_D).

III) Recibida por el periférico la señal de aceptación, debe enviar el dato a la línea dato correspondiente (D_A, D_B, D_C o D_D). El dispositivo procederá a colocar este dato en el canal de salida Y.

4.10. Diseñar un sumador de cuatro bits en formato Signo-Magnitud.

4.11. Diseñar un comparador de dos números de 4 bits que disponga de una entrada de control S de 2 bits que indique el formato en el que están codificadas las entradas del comparador. Los valores posibles de S y su significado son los siguientes:

S	Formato
00	Binario Puro
10	Complemento a 2
11	Complemento a 1

NOTA: Ténganse en cuenta todos los números que pueden ser representados en cada formato.

4.12. Diseñar un circuito combinacional con 16 líneas de entrada (Ei). La activación de cada línea representa un valor entero entre -8 y +7, siendo la entrada E0 la correspondiente al -8, E1 al -7 y así sucesivamente hasta la entrada E15 que corresponde al +7.

El circuito tiene tres salidas: una de 5 bits que proporciona el número de entradas activas; otra de 4 bits que indica el mayor número representado en las entradas activas; y otra, también de 4 bits, que indica el menor número representado en las entradas activas. Los valores de estas dos últimas salidas estarán representados en formato de complemento a dos.

Si no hay ninguna entrada activa, todas las salidas estarán a cero. Si solo hay una entrada activa, las salidas que indican los números mayor y menor tendrán el mismo valor, que será el correspondiente a la entrada activa.

4.13. Disponemos de una memoria M de tamaño 4x4 en la que se almacenan números en complemento a uno. Se pide diseñar un circuito combinacional con dos entradas F y C de 2 bits cada una, que seleccione los números almacenados en la fila F (MF) y en la columna C (MC) de la memoria, y proporcione una salida Q de 1 bit que valga 1 cuando $MF > MC$. Ejemplo: Supóngase que en un momento dado los valores almacenados en la memoria son los siguientes:

F/C	00	01	10	11
00	1	0	1	0
01	0	1	0	0
10	1	1	0	1
11	1	0	0	1

Si F=01 y C=10, entonces MF=0100 y MC=1000, por lo que $MF > MC$ y por tanto Q=1.

Si F=11 y C=00, entonces MF=1001 y MC=1011, por lo que $MF < MC$ y por tanto Q=0.

Nota: Utilícese la notación Mij para indicar el bit almacenado en la fila i, columa j de la memoria y ténganse en cuenta todos los números que pueden ser representados en complemento a uno.

4.14. La empresa portuguesa Çompanhia das Carruagens"quiere automatizar la tarea de clasificar los trenes de mercancias en función de su peso. Esta clasificación se realiza en un cruce al que llegan 2 vías (Ea y Eb) y del que salen 4 (Sa, Sb, Sc y Sd). Un tren que llegue al cruce por una de las vías Ea o Eb saldrá por una de las vías de salida que será escogida en función del peso del tren de acuerdo a la siguiente tabla:

Salida	Peso
Sa	<= 4 T.m.
Sb	> 4 y <= 8 T.m.
Sc	> 8 y <= 12 T.m.
Sd	> 12 T.m.

Diseñar un circuito con dos entradas Pa y Pb de 4 bits, que indican los pesos en toneladas de los trenes que esperan en las entradas Ea y Eb del cruce, y tres salidas:

- Sa y Sb de 2 bits, que indican la salida a la que deben dirigirse los trenes que esperan en las entradas Ea y Eb.
- WAIT/OK de 1 bit, que valdrá 1 cuando Sa y Sb sean diferentes. Si fuesen iguales esta linea valdría 0 para indicarle al tren en Eb que tiene que esperar a que pase el de Ea. Además en este caso el valor final de Sb será también 0.