Capítulo 5

CIRCUITOS MSI (1): Multiplexores y demultiplexores

5.1. Introducción

Los **circutios MSI** son los que están constituidos por un número de puertas lógicas comprendidos entre 12 y 100 (ver apartado 4.2.4). En este capítulo veremos una serie de **circuitos combinaciones** que se utilizan mucho en electrónica digital y que son la base para la creación de diseños más complejos. Aunque se pueden diseñar a partir de puertas lógicas, estos circuitos se pueden tratar como "componentes", asignándoles un símbolo, o utilizando una cierta nomenclatura.

Los circuitos que veremos son los siguientes:

- **Multiplexores** y **demultiplexores**
- **Codificadores** y **decodificadores**
- **Comparadores**

Lo más importante es comprender para qué sirven, cómo funcionan y que bits de entrada y salida utilizan. Estos circuitos los podríamos diseñar perfectamente nosotros, puesto que se trata de *circuitos combinacionales* y por tanto podemos aplicar todo lo aprendido en el capítulo 4.

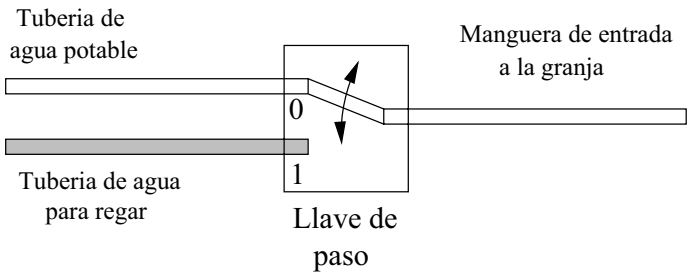


Figura 5.1: Simitud entre un multiplexor y un sistema de agua de una granja

5.2. Multiplexores

5.2.1. Conceptos

Un **Multiplexor** es un **circuito combinacional** al que entran varios canales de datos, **y sólo uno de ellos**, el que hallamos seleccionado, es el que **aparece por la salida**. Es decir, que es un circuito que nos permite **SELECCIONAR** que datos pasan a través de dicho componente.

Vamos a ver un ejemplo NO electrónico. Imaginemos que hay dos tuberías (*canales de datos*) por el que circulan distintos fluidos (*datos*). Una transporta agua para regar y la otra agua potable. Estas tuberías llegan a una granja, en la cual hay una única manguera por la que va a salir el agua (bien potable o bien para regar), según lo que seleccione el granjero posicionando la **llave de paso** en una u otra posición. En la figura 5.1 se muestra un esquema. Las posiciones son la 0 para el agua potable y 1 para el agua de regar.

Moviendo la llave de paso, el granjero puede seleccionar si lo que quiere que salga por la manguera es agua potable, para dar de beber al ganado, o agua para regar los cultivos. Según cómo se posicione esta llave de paso, en la posición 0 ó en la 1, seleccionamos una tubería u otra.

Pero ¿por qué sólo dos tuberías?. Porque es un ejemplo. A la granja podrían llegar 4 tuberías. En este caso el granjero tendría una llave de paso con 4 posiciones, como se muestra en la figura 5.2. Esta llave se podría poner en 4 posiciones distintas para dar paso a la tubería 0, 1, 2 ó 3. Obsérvese que sólo pasa una de las tuberías en cada momento, ¡y sólo una!. Hasta que el granjero no vuelva a cambiar la llave de paso no se seleccionará otra tubería.

Con este ejemplo es muy fácil entender la idea de **multiplexor**. Es como una **llave de paso**, que sólo conecta uno de los canales de datos de entrada con el canal de datos de salida.

Ahora en vez de en tuberías, podemos pensar en canales de datos, y tener un esquema como el que se muestra en la figura 5.3, en la que hay 4 canales de datos, y sólo uno de ellos es seleccionado por el multiplexor para llegar a la salida . En general, en un multiplexor tenemos dos tipos de entradas:

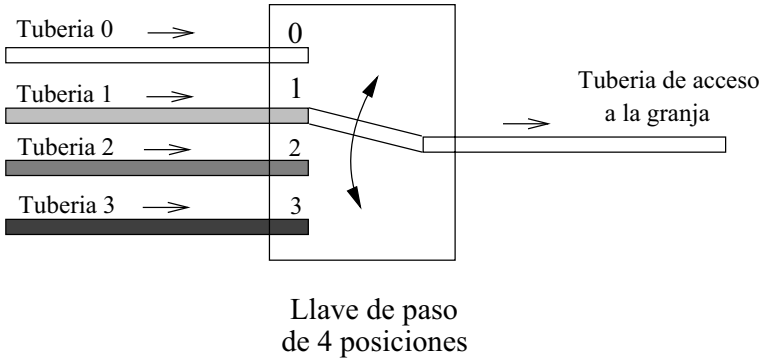


Figura 5.2: Sistema de agua de 4 tuberías

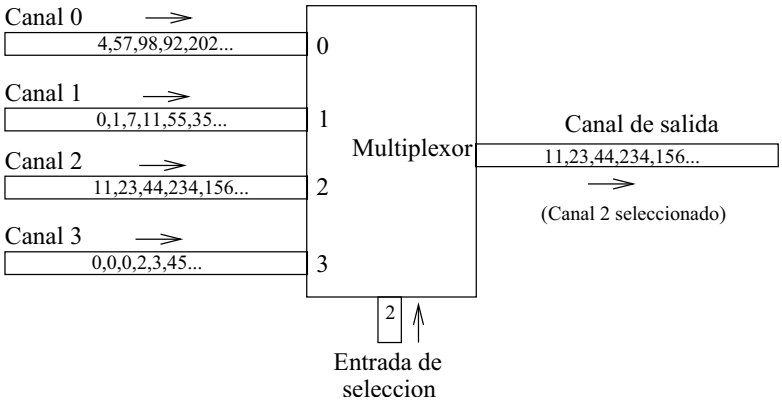


Figura 5.3: Un multiplexor que selecciona entre 4 canales de datos

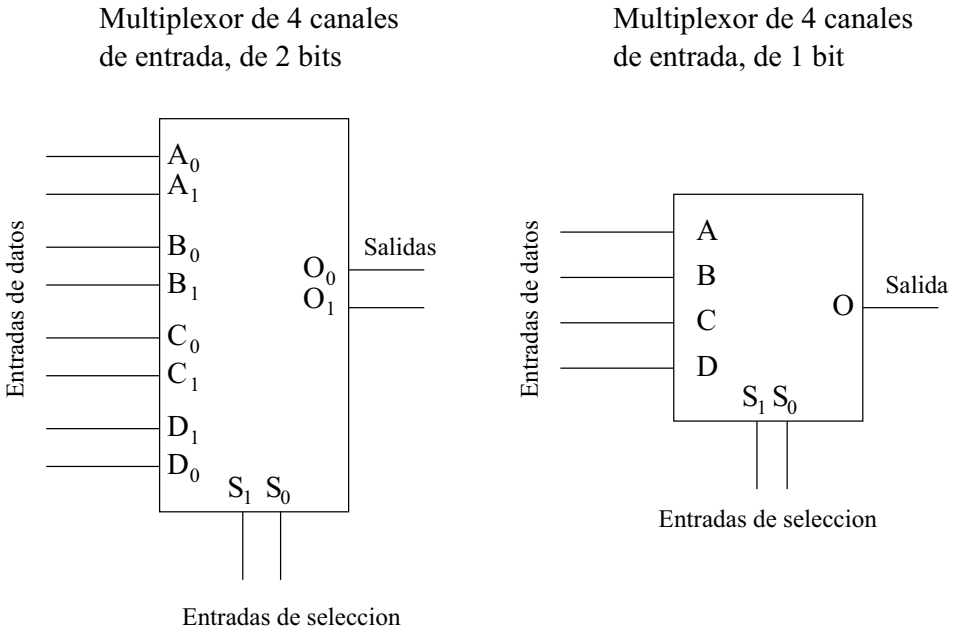


Figura 5.4: Dos multiplexores de 4 canales de entrada

- **Entradas de datos**: (Las tuberías en el ejemplo).
- **Entrada de selección**: Indica cuál de las entradas se ha seleccionado (posición de la llave de paso).

5.2.2. Multiplexores y bits

Hemos visto cómo a un **multiplexor** le llegan **números** por distintas entradas y según el número que le llegue por la **entrada de selección**, lo manda por la salida o no. **¡¡Números!!** Recordemos que los circuitos digitales sólo trabajan con números.

Pero estos números, vimos que siempre vendrán **expresados en binario** y por tanto se podrán expresar mediante **bits**. ¿Cuantos bits? Depende de lo grande que sean los números con los que se quiere trabajar.

En el interior de los microprocesadores es muy normal encontrar multiplexores de 8 bits, que tienen varias entradas de datos de 8 bits. Pero se puede trabajar con multiplexores que tengan 4 bits por cada entrada, o incluso 2, o incluso 1bit. En la figura 5.4 se muestran dos multiplexores que tienen 4 entradas de datos. Por ello la entrada de selección tiene dos bits (para poder seleccionar entre los cuatro canales posibles). Sin embargo, en uno las entradas de datos son de 2 bits y en el otro de 1 bit.

Mirando el número de salidas, podemos conocer el tamaño de los canales de entrada.

Así en los dos multiplexores de la figura 5.4, vemos que el de la izquierda tiene 2 bits de salida, por tanto sus canales de entrada son de 2 bits. El de la derecha tiene 1 bit de salida, por tanto los canales de 1 bit.

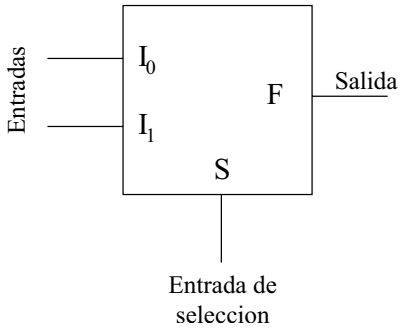
Los multiplexores en lo que principalmente nos centraremos son los que tienen canales de 1 bit. A partir de ellos podremos construir multiplexores mayores, bien con un mayor número de canales de entrada o bien con un mayor número de bits por cada canal.

5.2.3. Multiplexores de 1 bit y sus expresiones booleanas

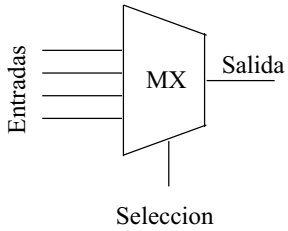
Llamaremos así a los **multiplexores que tienen canales de entrada de 1 bit**, y por tanto **sólo tienen un bit de salida**. Estudiaremos estos multiplexores, comenzando por el más simple de todos, el que sólo tienen una entrada de selección.

Multiplexores con una entrada de selección

El multiplexor más simple es el que sólo tiene una **entrada de selección**, **S**, que permite seleccionar entre dos entradas de datos, según que $S = 0$ ó $S = 1$. Su aspecto es el siguiente:



NOTA: *En esta asignatura representaremos los multiplexores de igual que cualquier otro circuito, mediante una "caja" que tiene unas entradas y unas salidas. No obstante, el símbolo normalmente emlpleado es el siguiente:*



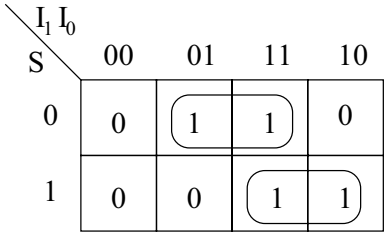
¿Cómo podemos expresar la función de salida F, usando el Algebra de Boole?. Existe una manera muy sencilla y que ya conocemos: hacer la tabla de verdad y obtener la función más simplificada.

Construyamos la tabla de verdad. Lo primero que nos preguntamos es, ¿Cuantas entradas tengo en este circuito?. En total hay tres entradas. Dos son de datos: I_1, I_0 y una es de selección: S. La tabla de verdad tendrá en total $2^3 = 8$ filas. Para construir esta tabla de verdad sólo hay que entender el funcionamiento del multiplexor e ir caso por caso rellenando la tabla. Por ejemplo, ¿qué ocurre si $S = 1$, $I_1 = 0$ y $I_0 = 1$?. Aplicamos la definición de multiplexor. Puesto que $S = 0$, se está seleccionando la entrada de datos 0, es decir, la entrada I_0. Por tanto, lo que entre por la entrada I_1 será ignorado por el multiplexor. Si la entrada seleccionada es la I_0, la salida tendrá su mismo valor. Y puesto que $I_0 = 1$, entonces $F = 1$. Si hacemos lo mismo para todos los casos, tendremos la siguiente tabla de verdad:

S	I_1	I_0	F
0	0	**0**	**0**
0	0	**1**	**1**
0	1	**0**	**0**
0	1	**1**	**1**
1	**0**	0	**0**
1	**0**	1	**0**
1	**1**	0	**1**
1	**1**	1	**1**

La tabla se ha dividido en dos bloques, uno en el que $S = 0$ y otro en el que $S = 1$. En el primer bloque, se selecciona I_0 que aparecerá en la salida. Se ha puesto en negrita todos los valores de I_0 para que se vea que son los mismos que hay a la salida. En el bloque inferior, lo que se selecciona es I_1 y es lo que se obtiene por la salida.

Apliquemos el **método de Karnaugh** para obtener la expresión **más simplificada de F**. El diagrama que se obtiene es el siguiente: (Se aconseja al lector que lo haga por su propia cuenta, sin mirar los apuntes, así le sirve además para practicar :-)



Obtenemos la siguiente expresión:

$$F = \overline{S} \cdot I_0 + S \cdot I_1 \tag{5.1}$$

Y si ahora "escuchamos" lo que la ecuación nos dice, veremos que tiene mucho sentido:

"Si S=0, $F = I_0$ y si S=1, $F = I_1$"

¡¡Es justo la definición de un multiplexor!! La salida toma el valor de una de las entradas, según el valor que tome la entrada de selección.

En realidad, el multiplexor lo podríamos haber descrito de una manera más sencilla, y podríamos haber obtenido la ecuación de otra forma. Veamos cómo.

La función F que describe el comportamiento de un multiplexor con una única entrada de selección, la podemos describir mediante la siguiente tabla:

S	F
0	I_0
1	I_1

que lo que nos viene a decir es lo mismo que su ecuación: cuando S=0, por la salida del multiplexor aparece el valor I_0 y cuando S=1, aparece el valor I_1. Estamos considerando las variables I_0 e I_1 como parámetros y NO como variables de entrada del circuito y por tanto estamos considerando como si la función F sólo dependiese de la variable S, es decir, tenemos la función F(S). **¿Cómo podemos obtener la ecuación del multiplexor a partir de esta tabla?**: aplicando el **teorema de expansión**, que vimos en el apartado 3.4 obtenemos los siguiente:

$$F(S) = S \cdot F(1) + \overline{S} \cdot F(0)$$

y F(1) es la salida del multiplexor cuando S=1, es decir, que $F(1) = I_1$ y F(0) es la salida cuando S=0, $F(0) = I_0$. La ecuación del multiplexor es la siguiente:

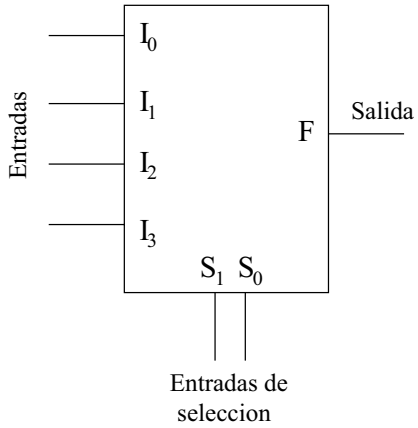
$$F(S) = S \cdot I_1 + \overline{S} \cdot I_0$$

¡¡Que es la misma ecuación que habíamos obtenido por Karnaugh!!

No se asuste el lector por los desarrollos teóricos. Lo importante es comprender cómo funcionan este tipo de multiplexores y cuál es la ecuación que los describe, independientemente de cómo la hallamos obtenido. Aquí, hemos obtenido la ecuación por dos métodos diferentes. Veremos que con los multiplexores de dos entradas de selección sólo lo podremos hacer por el segundo método.

Multiplexores con dos entradas de selección.

El siguiente multiplexor en complejidad es el que tenga **2 entradas de selección**, por lo que se **podrá seleccionar hasta 4 entradas posibles**. Habrá por tanto **4 entradas de datos**. El circuito es como el siguiente:



Hay 4 entradas de datos y 2 entradas de selección, en total 6 entradas. Ahora hacemos lo mismo que antes, construimos la tabla de verdad y aplicamos Karnaugh... pero.... ¿6 variables? **¡¡Hay que hacer una tabla que tenga $2^6 = 64$ filas!! ¡¡Y luego aplicar Karnaugh a una función de 6 variables!!!**

Vemos que este método, aunque fácil, require muchas operaciones. ¡¡Es un método ideal para que lo haga un ordenador!!. Nosotros obtendremos sus ecuaciones de otra manera diferente.

Vamos a describir este multiplexor mediante la siguiente tabla:

S_1	S_0	F
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

que lo que nos está expresando es que la salida del multiplexor valdrá I_0,I_1, I_2 o I_3 según el valor que tomen las variables de entrada S_1 y S_0. Estamos considerando que la función F sólo depende de estas dos variables: $F(S_1, S_0)$ y que I_0,I_1, I_2 e I_3 son parámetros, es decir, valores constantes que pueden valer '0' ó '1'.

Si aplicamos **el teorema de expansión** a la función $F(S_1, S_0)$, desarrollándola por S_1, obtenemos lo siguiente:

$$F(S_1, S_0) = S_1 \cdot F(1, S_0) + \overline{S_1} \cdot F(0, S_0)$$

Y si ahora aplicamos nuevamente el **teorema de expansión** a las funciones $F(1, S_0)$ y $F(0, S_0)$, desarrollándolas por la variable S_0, tenemos lo siguiente:

$$F(1, S_0) = S_0 \cdot F(1, 1) + \overline{S_0} \cdot F(1, 0)$$

$$F(0, S_0) = S_0 \cdot F(0, 1) + \overline{S_0} \cdot F(0, 0)$$

Y ahora, si lo juntamos todo en una única expresión, tenemos:

$$F(S_1, S_0) = S_1 \cdot F(1, S_0) + \overline{S_1} \cdot F(0, S_0) =$$

$$S_1 \cdot \Big[S_0 \cdot F(1, 1) + \overline{S_0} \cdot F(1, 0)\Big] + \overline{S_1} \cdot \Big[S_0 \cdot F(0, 1) + \overline{S_0} \cdot F(0, 0)\Big] =$$

$$S_1 S_0 F(1, 1) + S_1 \overline{S_0} F(1, 0) + \overline{S_1} S_0 F(0, 1) + \overline{S_1 S_0} F(0, 0)$$

¿Cuando vale F(0,0)?, es decir, ¿cuál es la salida del multiplexor cuando $S_0 = 0$ y $S_1 = 0$?. Por la definición de multiplexor, la salida será lo que venga por el canal 0, que es I_0. De la misma manera obtenemos que $F(0, 1) = I_1$, $F(1, 0) = I_2$, $F(1, 1) = I_3$. Sustituyendo estos valores en la ecuación anterior y reordenándola un poco tenemos la **expresión final para un multiplexor de dos entradas de selección**:

$$F = \overline{S_1} \cdot \overline{S_0} \cdot I_0 + \overline{S_1} \cdot S_0 \cdot I_1 + S_1 \cdot \overline{S_0} \cdot I_2 + S_1 \cdot S_0 \cdot I_3 \tag{5.2}$$

Olvidémonos ahora de cómo hemos obtenido esa ecuación. Lo importante es entenderla y

saber utilizarla. Vamos a comprobar si efectivamente esta ecuación describe el funcionamiento de un multiplexor de 2 entradas de selección y 4 entradas de datos.

Si $S_1 = 0$ y $S_0 = 0$, sabemos por el comportamiento de un multiplexor que se seleccionará la entrada I_0 para que aparezca por la salida. Vamos a comprobarlo. En la ecuación del multiplexor sustituimos S_1 por 0 y S_0 por 1. Obtenemos:

$$F = \overline{0} \cdot \overline{0} \cdot I_0 + \overline{0} \cdot 0 \cdot I_1 + 0 \cdot \overline{0} \cdot I_2 + 0 \cdot 0 \cdot I_3 =$$

$$= 1 \cdot 1 \cdot I_0 + 1 \cdot 0 \cdot I_1 + 0 \cdot 1 \cdot I_2 + 0 \cdot 0 \cdot I_3 =$$

$$= I_0$$

Se deja como ejercicio el que se compruebe la ecuación para el resto de valores de las entradas de selección.

Multiplexor con cualquier número de entradas de selección

Si ahora tenemos un multiplexor con 3 entradas de selección, que me permitirá seleccionar entre 8 entradas de datos, la ecuación que lo describe es la generalización de la ecuación 5.2. En total habrá 8 sumandos y en cada uno de ellos se encontrarán las variables S_2, S_1, y S_0, además de los correspondientes parámetros I_0,I_1, ..., I_7.

La ecuación será:

$$F = \overline{S_2} \cdot \overline{S_1} \cdot \overline{S_0} \cdot I_0 + \overline{S_2} \cdot \overline{S_1} \cdot S_0 \cdot I_1 + \overline{S_2} \cdot S_1 \cdot \overline{S_0} \cdot I_2 + \overline{S_2} \cdot S_1 \cdot S_0 \cdot I_3 +$$

$$+ S_2 \cdot \overline{S_1} \cdot \overline{S_0} \cdot I_4 + S_2 \cdot \overline{S_1} \cdot S_0 \cdot I_5 + S_2 \cdot S_1 \cdot \overline{S_0} \cdot I_6 + S_2 \cdot S_1 \cdot S_0 \cdot I_7$$

Y lo mismo podemos hacer para cualquier multiplexor con un número de entradas de selección mayor, lo que ocurre que la ecuación tendrá muchos más términos.

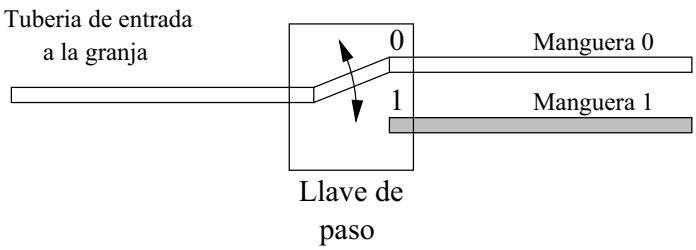


Figura 5.5: Similitud entre un demultiplexor y un sistema de agua de una granja

5.3. Demultiplexores

5.3.1. Conceptos

El concepto de demultiplexor es similar al de multiplexor, viendo las entradas de datos como salidas y la salida como entradas. En un multiplexor hay varias entradas de datos, y sólo una de ellas se saca por el canal de salida. En los demultiplexores hay un único canal de entrada que se saca por una de las múltiples salidas (y sólo por una!!!).

Si utilizamos el símil de la granja y las tuberías, podemos imaginar el siguiente escenario. Supongamos que ahora a la granja le llega una única tubería con agua, pero en el interior de la granja hay varias mangueras, cada una para limpiar una zona del establo o dar de beber a los animales de esa zona. Cómo sólo hay un granjero, sólo podrá usar una de las mangueras cada vez (el granjero no podrá usar a la vez dos mangueras, porque están en sitios diferentes!!).

Para seleccionar qué manguera quiere usar en cada momento, hay una llave de paso, de manera que si la sitúa en una posición, el agua que viene por la entrada saldrá por la manguera 0, mientras que si la sitúa en la otra posición, el agua saldrá por la manguera 1 (ver figura 5.5)

De la misma manera que en los multiplexores puede haber varias entradas, en los demultiplexores puede haber varias salidas. Por ejemplo en la figura 5.6 se muestra el mismo sistema de tuberías de la granja, pero ahora hay 4 mangueras, para llegar a 4 zonas distintas de la granja. Ahora el granjero tendrá que posicionar la llave de paso en una de las 4 posiciones posibles, para que el agua salga por la manguera seleccionada.

Ya comprendemos cómo funcionan los demultiplexores. Si lo aplicamos al mundo de la electrónica, en vez de tuberías tendremos canales de datos. Habrá un único canal de entrada, por el que llegarán números, que saldrán sólo por uno de los canales de salida, el que tengamos seleccionado, como se muestra en la figura 5.7.

En general en un demultiplexor tendremos:

- **Una entrada de datos**

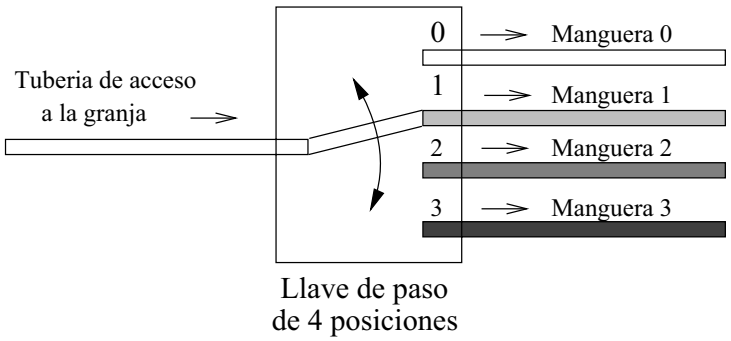


Figura 5.6: Sistema de agua de 4 mangueras

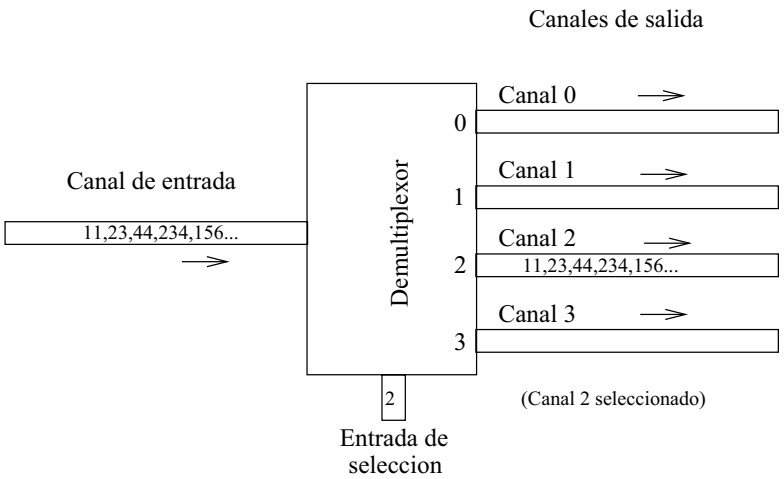


Figura 5.7: Un demultiplexor que selecciona entre 4 canales de datos

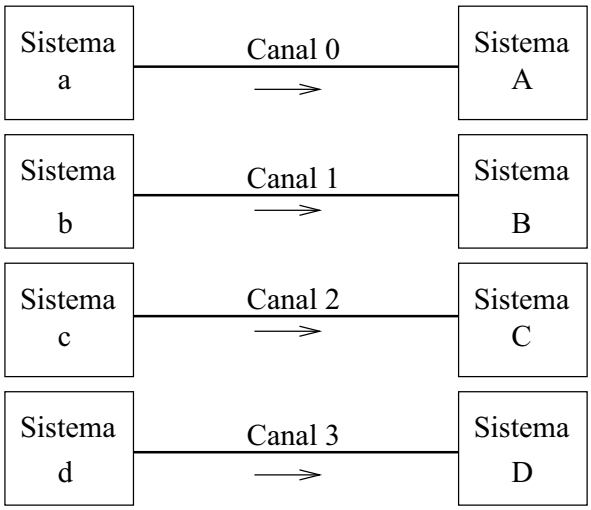


Figura 5.8: Una alternativa para comunicar sistemas

- **Una entrada de selección**: que indica a cuál de las salidas se manda la entrada
- **Varios canales de datos de salida**. Sólo estará activo el que se haya seleccionado.

5.3.2. Juntando multiplexores y demultiplexores

Vamos a ver una aplicación típica de los multiplexores y los demultiplexores. Imaginemos que tenemos 4 sistemas, que los llamaremos **a**,**b**,**c** y **d**, y que necesitan enviar información a otros 4 dispositivos **A**,**B**,**C** y **D**. La comunicación es uno a uno, es decir, el sistema **a** sólo envía información al sistema **A**, el **b** al **B**, el **c** al **C** y el **d** al **D**.

¿Qué alternativas hay para que se produzca este envío de datos? Una posibilidad es obvia, y es la que se muestra en la figura 5.10. Directamente se tiran cables para establecer los canales de comunicación.

Pero esta no es la única solución. Puede ser que podamos tirar los 4 cables, porque sean muy caros o porque sólo haya un único cable que comunique ambas parte, y será necesario llevar por ese cable todas las comunicaciones.

La solución se muestra en la figura 5.9. Vemos que los sismteas **a**, **b**, **c** y **d** se conectan a un multiplexor. Un circuito de control, conectado a las entradas de selección de este multiplexor, selecciona periódicamente los diferentes sistemas, enviando por la salida el canal correspondiente. Podemos ver que a la salida del multiplexor se encuentra la información enviada por los 4 sistemas. Se dice que esta información está **multiplexada en el tiempo**. Al final de esta línea hay un demultiplexor que realiza la función inversa. Un circuito de control selecciona periódicamente

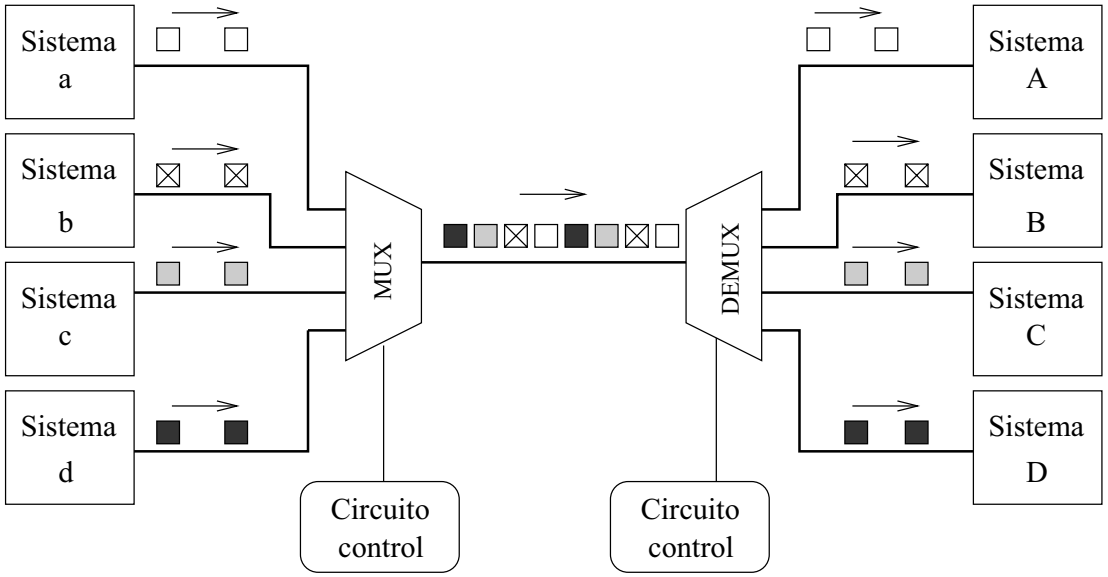


Figura 5.9: Uso de un multiplexor y demultiplexor para transmisión de datos por un único cable

por qué salidas debe salir la información que llega por la entrada.

Lo que hemos conseguido es que toda la información enviada por un sistema, llega a su homólogo en el extremo anterior, pero sólo hemos utilizado un único canal de datos.

5.3.3. Demultiplexores y bits

Un demultiplexor, como cualquier otro circuito digital trabaja sólo con **números**. Pero estos números vendrán expresados en **binario**, por lo que los **canales de datos de entrada y salida**, y **la entrada de selección** vendrán e**xpresados en binario** y tendrán un número determinado de **bits**.

Una vez más nos hacemos la pregunta, ¿Cuantos bits?. Depende. Depende de la aplicación que estemos diseñando o con la que estemos trabajando. En la figura 5.10 se muestran dos demultiplexores de 4 canales, por lo que tendrán 2 bits para la entrada de selección. El de la izquierda tiene canales de 2 bits y el de la derecha de 1 bit.

Los multiplexores que vamos a estudiar son lo que tienen canales de 1 bit. A partir de ellos podremos construir multiplexores con un mayor número de bits por canal.

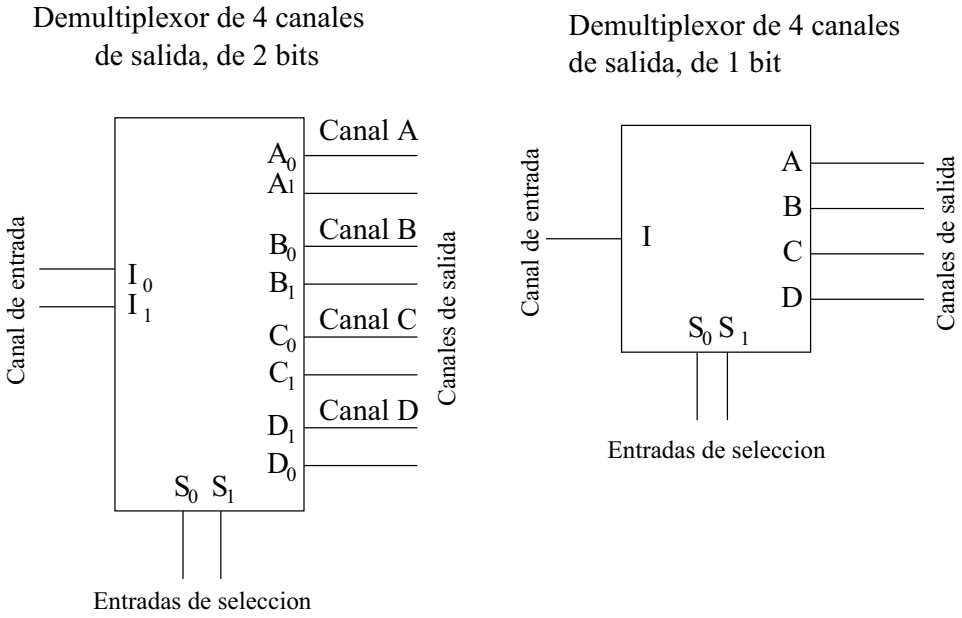
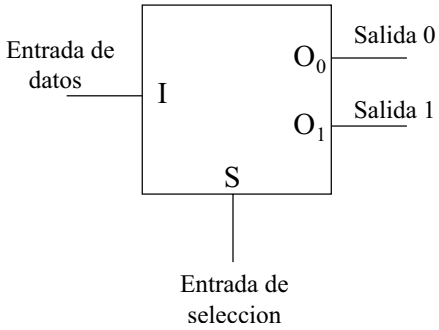


Figura 5.10: Dos demultiplexores de 4 canales de salida

5.3.4. Demultiplexores de 1 bit y sus expresiones booleanas

Demultiplexor de una entrada de selección

El demultiplexor más simple es el que tiene una entrada de selección, una entrada de datos y dos salidas. Según el valor de la entrada de selección, la entrada de datos se sacará por la salida O_0 o por la O_1:



Nos hacemos la misma pregunta que en el caso de los multiplesore: ¿Cómo podemos expresar las funciones de salida usando el Algebra de Boole?. Podemos escribir la tabla de verdad y obtener las expresiones más simplificadas. Para tener la tabla aplicamos la definición de demultiplexor y vamos comprobando caso por caso qué valores aparecen en las salidas. Por ejemplo, si S=1 e I=1, se estará seleccionando la salida O_1, y por ella saldrá el valor de I, que es 1. La salida

O_0 no estará seleccionada y tendrá el valor 0.

S	I	O_1	O_0
0	**0**	0	**0**
0	**1**	0	**1**
1	**0**	**0**	0
1	**1**	**1**	0

Para obtener las expresiones de O_0 y O_1 no hace falta aplicar Karnaugh puesto que cada salida sólo toma el valor '1' para un caso y '0' para todos los restantes. Desarrollando por la primera forma canónica:

$$O_1 = S \cdot I$$

$$O_0 = \overline{S} \cdot I$$

Y podemos comprobar que si hemos seleccionado la salida 0 ($S = 0$), entonces $O_0 = I$ y $O_1 = 0$, y si hemos seleccionado la salida 1 ($S = 1$), $O_0 = 0$ y $O_1 = I$.

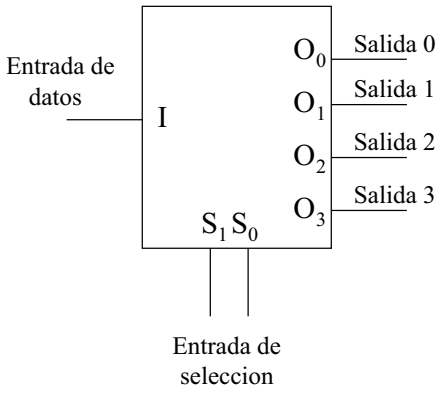
De la misma manera que hicimos con los multiplexores, podemos considerar que las funciones O_0 y O_1 sólo dependen de la entrada de Selección (S), tomando la entrada I como un parámetro. Así podemos describir este demultiplexor mediante la siguiente tabla:

S	O_1	O_0
0	I	0
1	0	I

Esta descripción será la que empleemos, ya que es más compacta.

Demultiplexor de dos entradas de selección

Este demultiplexor tiene dos entradas de selección y cuatro salidas:



La tabla de verdad "abreviada" la podemos expresar así:

S_1	S_0	O_3	O_2	O_1	O_0
0	0	0	0	0	I
0	1	0	0	I	0
1	0	0	I	0	0
1	1	I	0	0	0

La entrada I se saca por la salida indicada en las entradas de selección. Las ecuaciones de las funciones de salida son:

$$O_0 = \overline{S_1} \cdot \overline{S_0} \cdot I$$

$$O_1 = \overline{S_1} \cdot S_0 \cdot I$$

$$O_2 = S_1 \cdot \overline{S_0} \cdot I$$

$$O_3 = S_1 \cdot S_0 \cdot I$$

Si analizamos la ecuación de O_0 lo que nos dice es lo siguiente: "$O_0 = I$ sólo cuando $S_1 = 0$ y $S_0 = 0$". Para el resto de valores que pueden tomar las entradas de selección S_1 y S_0, O_0 siempre será 0.

Demultiplexor con cualquier número de entradas de selección

Para demultiplexores con mayor número de entradas de selección, las ecuaciones serán similares. Por ejemplo, en el caso de un demultiplexor que tenga tres entradas de selección: S_2, S_1 y S_0, y que por tanto tendrá 8 salidas, la ecuación para la salida O_5 será:

$$O_5 = S_2 \cdot \overline{S_1} \cdot S_0 \cdot I$$

y la ecuación de la salida O_0 será:

$$O_0 = \overline{S_2} \cdot \overline{S_1} \cdot \overline{S_0} \cdot I$$

Se deja como ejercicio al lector el que obtenga el resto de ecuaciones de salida.

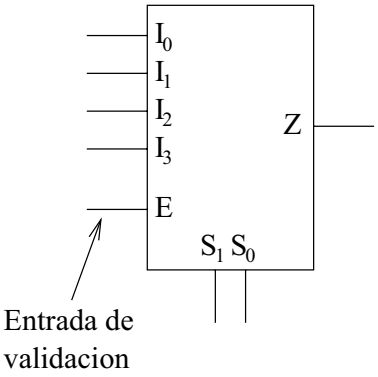
5.4. Multiplexores con entrada de validación (ENABLE)

Los multiplexores, y en general la mayoría de circuitos MSI, disponen de una **entrada adicional**, llamada **entrada de validación** (en inglés **Enable**). Esta entrada funciona como un interruptor de encendido/apagado para el circuito MSI. Si la **entrada de validación está activada**, **el circuito funcionará normalmente**. Pero si esta está desactivada, el circuito sacará el valor '0' por todas sus salidas, independientemente de lo que llegue por sus entradas. Se dice que está deshabilitado (no está en funcionamiento).

Las entradas de validación se les suele llamar E (del inglés Enable) y pueden ser de dos tipos: activas a nivel alto ó activas a nivel bajo.

5.4.1. Entrada de validación activa a nivel alto

Si esta entrada se encuentra a '1' (E=1) el multiplexor funciona normalmente (está conectado). Si se encuenrta a '0' (E=0) entonces su salida será '0' (estará desconectado). A continuación se muestra un multiplexor de 4 entradas de datos, 2 entradas de selección y una entrada de validación activa a nivel alto:



La tabla de verdad es la siguiente:

E	S_1	S_0	Z
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	I_0
1	0	1	I_1
1	1	0	I_2
1	1	1	I_3

Sólo en los casos en los que E=1, el multiplexor se comporta como tal. Cuando E=0, la salida Z siempre está a '0'. Esta tabla de verdad se suele escribir de una manera más abreviada de la siguiente manera:

E	S_1	S_0	Z
0	**x**	**x**	**0**
1	0	0	I_0
1	0	1	I_1
1	1	0	I_2
1	1	1	I_3

Con las 'x' de la primera fila se indica que cuando E=0, independientemente de los valores que tengan las entradas S_1 y S_0 la salida siempre tendrá el valor '0'.

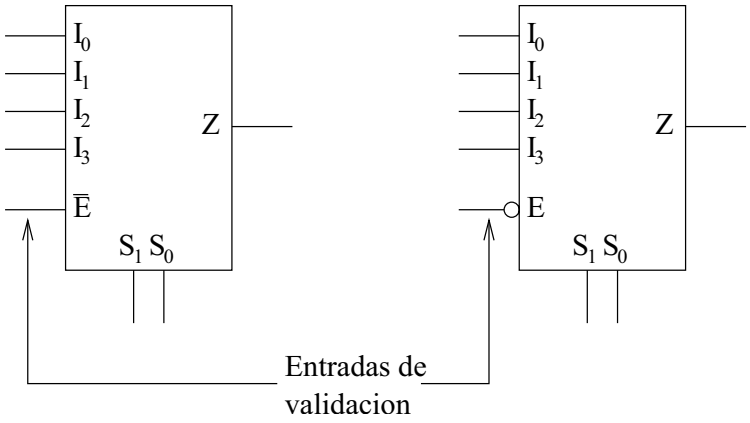
¿Y cual sería la nueva ecuación de este multiplexor? La misma que antes pero ahora multiplicada por E:

$$Z = (\overline{S_1} \cdot \overline{S_0} \cdot I_0 + \overline{S_1} \cdot S_0 \cdot I_1 + S_1 \cdot \overline{S_0} \cdot I_2 + S_1 \cdot S_0 \cdot I_3) \cdot E$$

Si E=0, entonces Z=0. El multiplexor está **deshabilitado**.

5.4.2. Entrada de validación activa a nivel bajo

Otros fabricamentes de circuitos integrados utilizan una **entrada de validación activa a nivel bajo**, que es justamente la inversa de la enterior. Se suele denotar mediante $\overline{E}$. Cuando la entrada E está a '0' el multiplexor funciona normalmente, y cuando está a '1' está desconectado. En la siguiente figura se muestran dos multiplexores de 4 entradas, dos entradas de selección y una entrada de validación activa a nivel bajo. Ambos multiplexores son iguales, pero se han utilizado notaciones distintas. En el de la izquierda se utiliza $\overline{E}$ y en el de la derecha E pero con un pequeño círculo en la entrada:



La tabla de verdad es la siguiente:

E	S_1	S_0	Z
0	0	0	I_0
0	0	1	I_1
0	1	0	I_2
0	1	1	I_3
1	**x**	**x**	**0**

Y la nueva ecuación es:

$$Z = (\overline{S_1} \cdot \overline{S_0} \cdot I_0 + \overline{S_1} \cdot S_0 \cdot I_1 + S_1 \cdot \overline{S_0} \cdot I_2 + S_1 \cdot S_0 \cdot I_3) \cdot \overline{E}$$

Cuando E=1, $\overline{E} = 0$ y entonces Z=0, con lo que el multiplexor se encuentra **deshabilitado**.

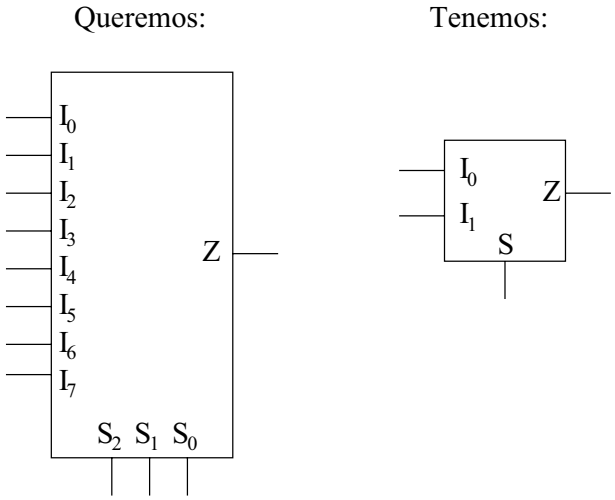
5.5. Extensión de multiplexores

La idea es poder conseguir tener **multiplexores más grandes a partir de otros más pequeños**. Y esto es necesario porque en nuestros diseños podemos necesitar unos multiplexores grandes, sin embargo en el mercado nos encontramos con multiplexores menores. Tenemos que saber cómo construir los multiplexores que necesitamos para nuestra aplicación a partir de los multiplexores que encontramos en el mercado.

La extensión puede ser bien **aumentando el número de entradas**, bien **aumentando el número de bits por cada canal de datos** o bien ambos a la vez.

5.5.1. Aumento del número de entradas

La solución es **conectarlos en cascada**. Lo mejor es verlo con un ejemplo. Imaginemos que necesitamos una multiplexor de 8 canales, pero sólo disponemos de varios de 2 canales:

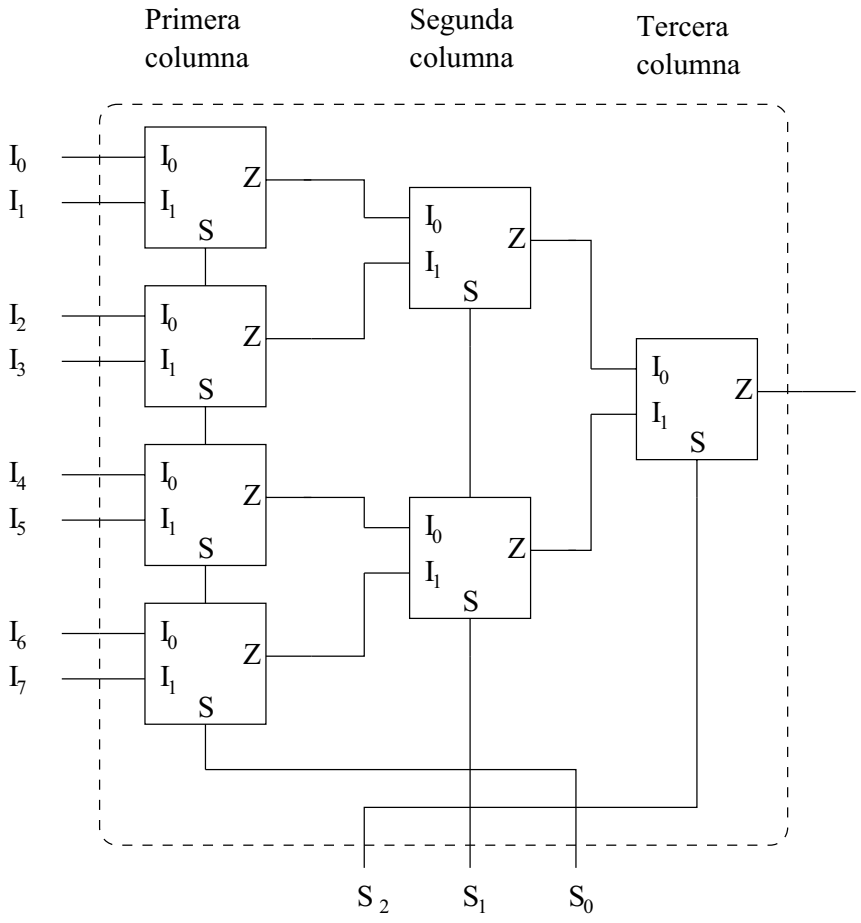


La solución es **conectarlos en cascada**. Primero colocamos una columna de 4 multiplexores de dos entradas, para tener en total 8 entradas. Todas las entradas de selección de esta primera columna se unen. Por comodidad en el dibujo, esto se representa mediante una línea vertical que une la salida S de un multiplexor con el de abajo.

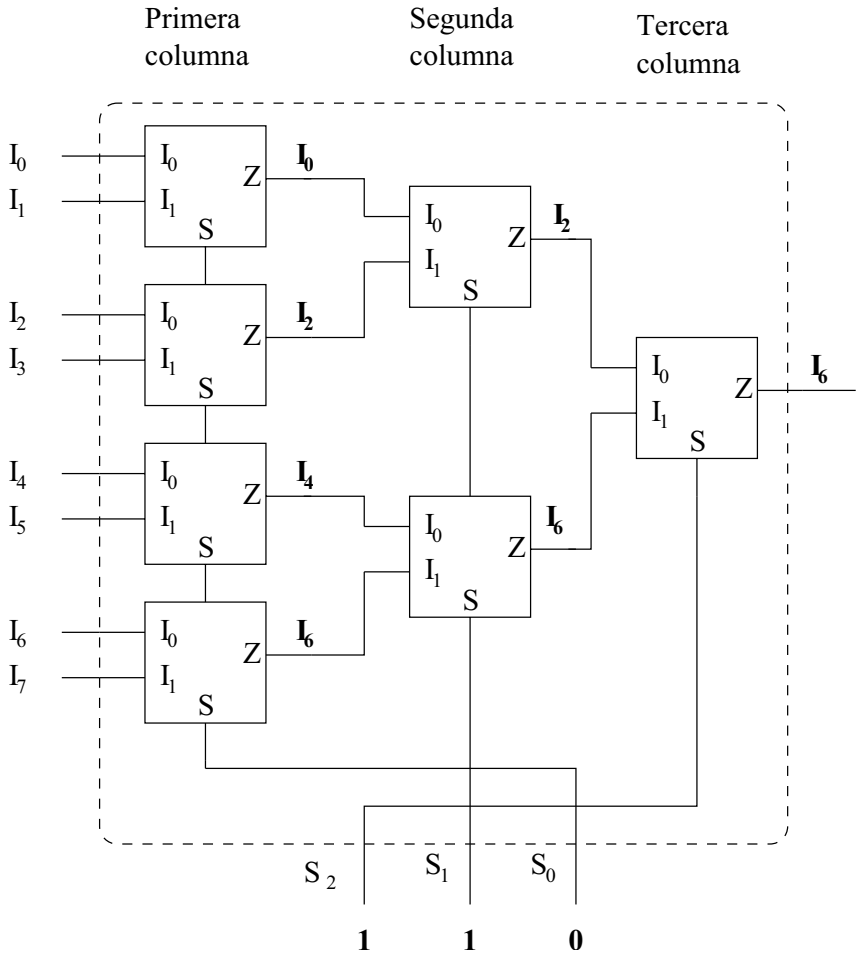
A continuación colocamos una segunda columna de 2 multiplexores de 2 entradas, también con sus entradas de selección unidas. Finalmente colocamos una última columna con un único multiplexor de 2 entradas.

Colocados de esta manera, conseguimos tener un multiplexor de 8 entradas y tres entradas de selección. La única consideración que hay que tener en cuenta es que la entrada de selección de

los multiplexores de la primera columna tiene peso 0, la segunda peso 1 y la última peso 2:



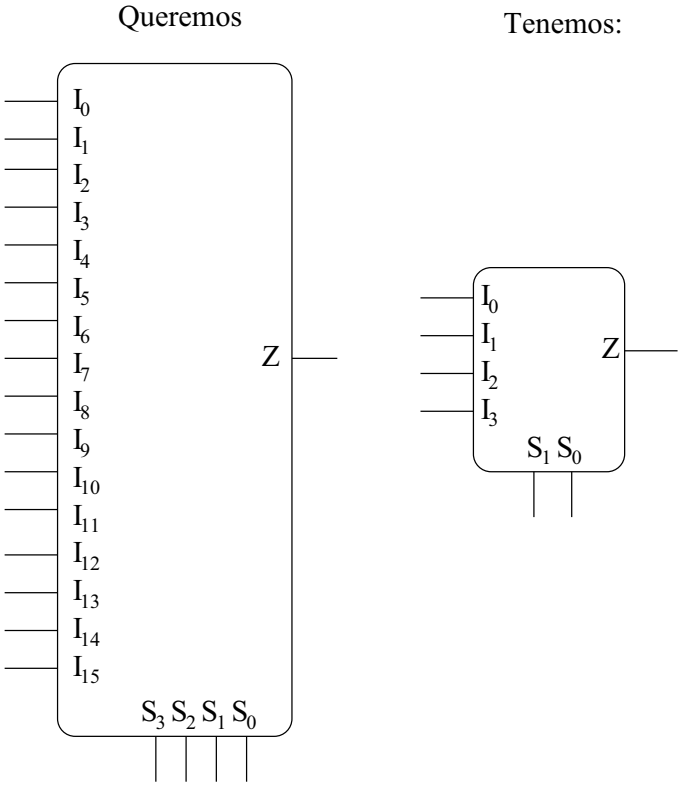
¡¡Vamos a comprobarlo!! (Siempre que se hace un diseño hay que comprobar si es correcto). Vamos a comprobar qué ocurre si seleccionamos el canal 6. Introducimos en binario el número 6 por las entradas de selección: $S_2 = 1$, $S_1 = 1$ y $S_0 = 0$. Por la entrada S de los multiplexores de la primera columna se introduce un '0', por lo que estos multiplexores sacan por sus salidas lo que hay en sus entradas I_1: I_0, I_2, I_4 e I_6. Por la entrada de selección de los multiplexores de la segunda columna se introduce un '1' por lo que están seleccionando su canal I_1. A la salida de estos multiplexores se tendrá: I_2 e I_6. Finalmente, el multiplexor de la última columna está seleccionando su entrada I_1, por lo que la salida final es I_6 (Recordar la idea de multiplexor como una llave de paso que conecta tuberías de agua):



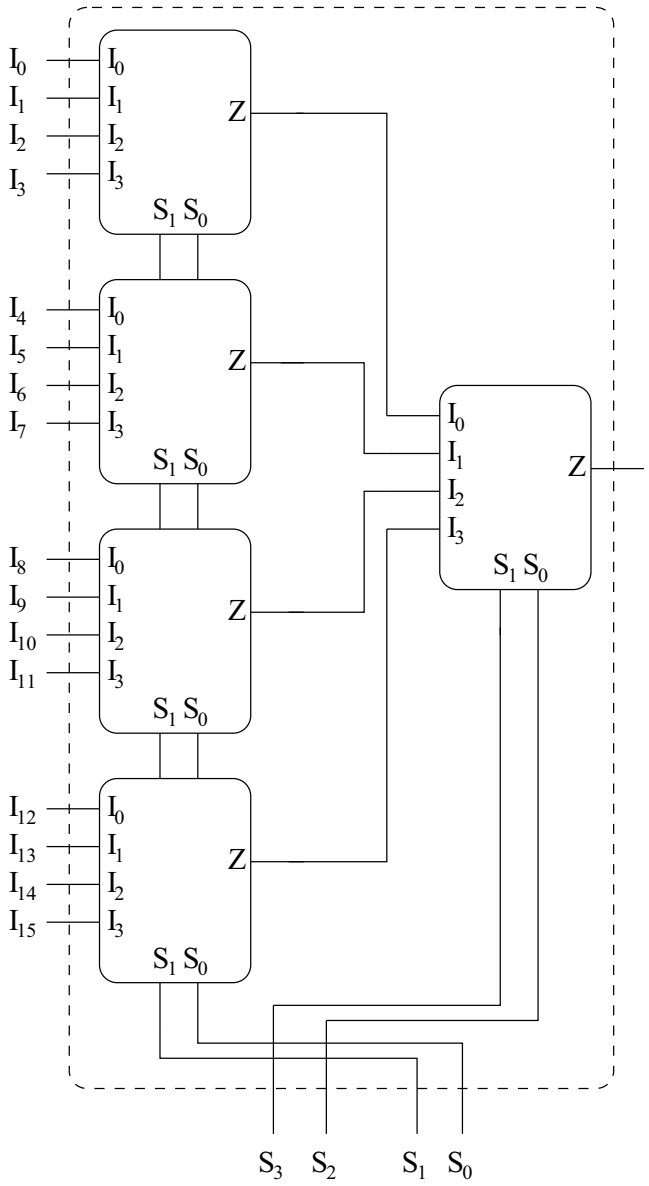
Ejemplo:

Construir un multiplexor de 16 entradas usando multiplexores de 4.

En este caso lo que *queremos* y lo que *tenemos* es lo siguiente:



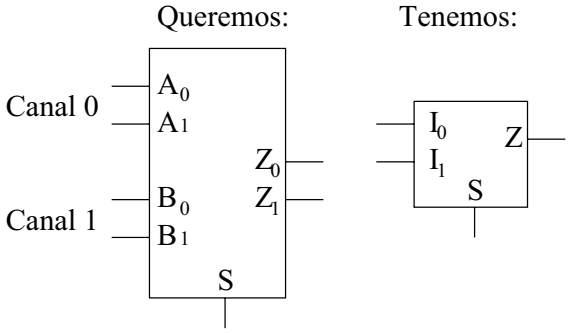
Los conectamos en cascada, para lo cual necesitamos una primera columna de 4 multiplexores de 4 entradas, con entradas S_0 de todos ellos unidos, así como las S_1. En la segunda fila hay un único multiplexor de 4 entradas:



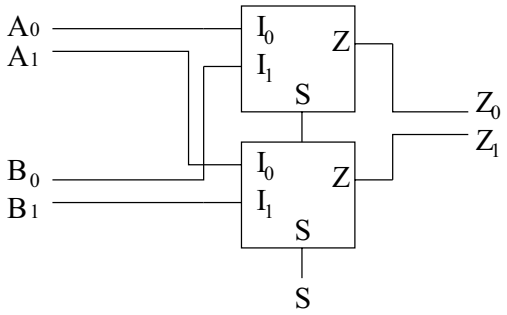
Se deja como ejercicio la comprobación de este diseño.

5.5.2. Aumento del número de bits por canal

Para conseguir esto hay que **conectarlos en paralelo**. Imaginemos que tenemos queremos construir un multiplexor de dos canales de entrada, cada uno de ellos de 2 bits, y para ello disponemos de multiplexores de 2 canales de un bit:



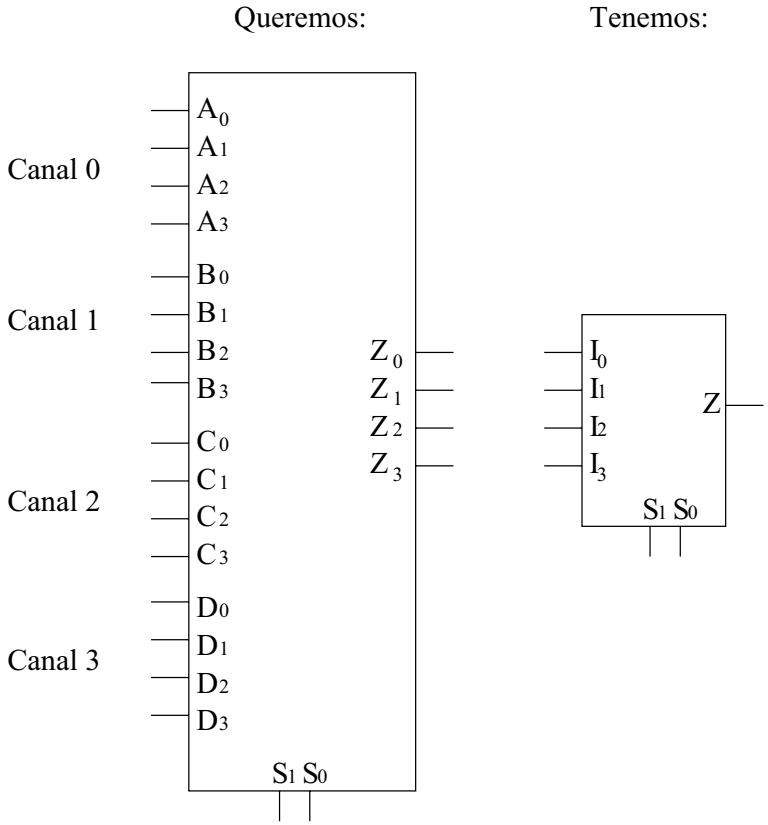
Utilizaremos dos multiplexores de lo que tenemos, uno por cada bit que tengamos en el nuevo canal de salida. Como los canales en el nuevo multiplexor son de 2 bits, necesitaremos 2 multiplexores de canales de 1 bit. Uno de estos multiplexores será al que vayan los bits de menos peso de los canales de entrada y el otro los de mayor peso. Las entradas de selección de ambos están unidas:



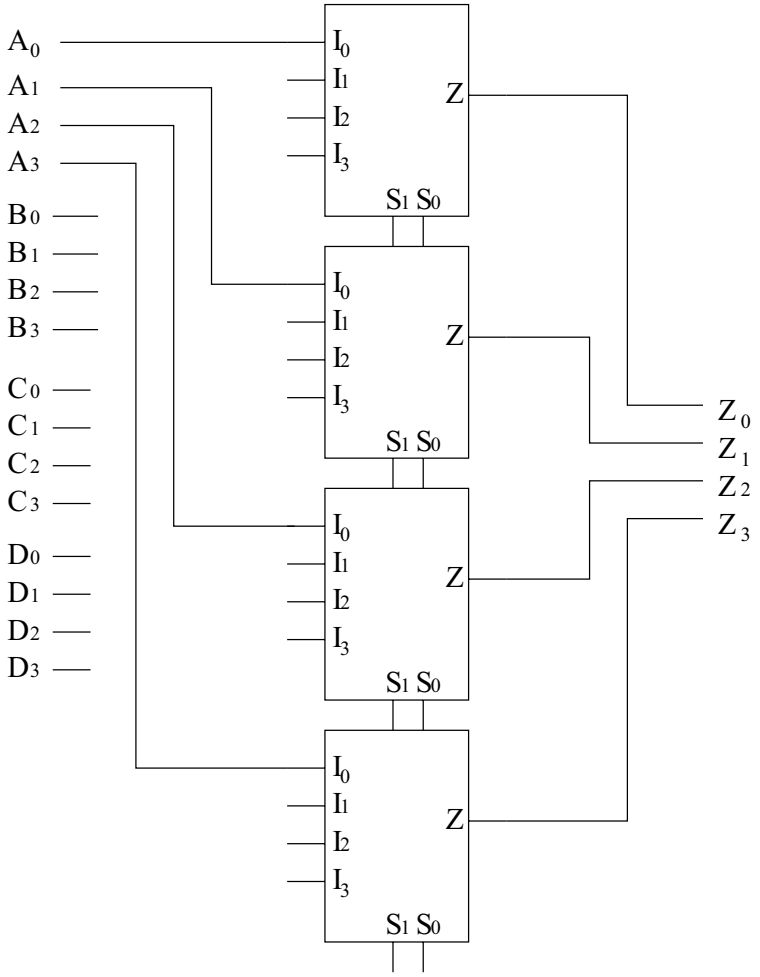
Si con en este nuevo multiplexor hacemos S=0, las salidas serán $Z_0 = A_0$ y $Z_1 = A_1$. Y si hacemos S=1, entonces obtenemos $Z_0 = B_0$ y $Z_1 = B_1$. ¡¡Es lo que andábamos buscando!!. Por la salida obtenemos bien el número que viene por el canal 0 $(A_1 A_0)$ ó bien el número que viene por el canal 1$(B_1 B_0)$.

Ejemplo:

Construir un muliplexor de 4 canales de 4 bits, usando multiplexores de 4 entradas de 1 bit.



Ahora necesitaremos 4 multiplexores de los que tenemos, a cada uno de los cuales les llegan los bits del mismo peso de los diferentes canales. Por el primer multiplexor entran los bits de menor peso (A_0, B_0, C_0 y D_0) y por el último los de mayor (A_3, B_3, C_3 y D_3). En el dibujo no se muestran todas las conexiones para no complicarlo:



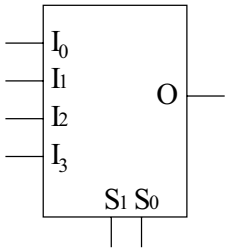
5.6. Implementación de funciones con MX's

Utilizando multiplexores es posible implementar funciones booleanas. En general, ***cualquier función de n variables se puede implementar utilizando un multiplexor de n-1 entradas de selección***.

Por ejemplo, dada la función:

$$F = \overline{x} \cdot y \cdot z + x \cdot \overline{y} + \overline{x} \cdot \overline{y} \cdot \overline{z}$$

que tiene 3 variables, se puede implementar utilizando un multiplexor de 2 entradas de control, como el mostrado a continuación:



Existen dos maneras de hacerlo. Una es emplear el algebra de boole y la ecuación de este tipo de multiplexores. Por lo general este método es más complicado. La otra es utilizar un método basado en la tabla de verdad.

5.6.1. Método basado en el Algebra de Boole

La ecuación de un multiplexor de 2 entradas de control y 4 entradas es la siguiente:

$$O = \overline{S_1} \cdot \overline{S_0} \cdot I_0 + \overline{S_1} \cdot S_0 \cdot I_1 + S_1 \cdot \overline{S_0} \cdot I_2 + S_1 \cdot S_0 \cdot I_3$$

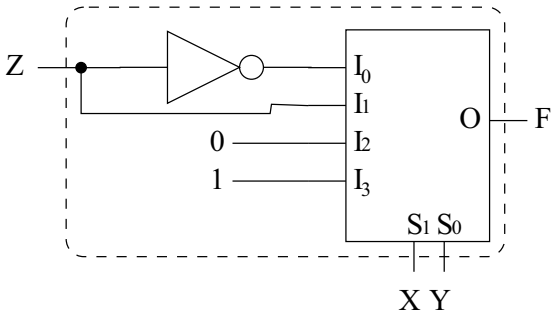
La ecuación de la función que queremos implementar la podemos expresar de la siguiente forma:

$$F = \overline{x} \cdot \overline{y} \cdot \overline{z} + \overline{x} \cdot y \cdot z + x \cdot \overline{y} \cdot 1 + x \cdot y \cdot 0$$

¡¡Que es muy parecida a Z!!. Si igualamos términos, obtenemos que por las entradas del multiplexor hay que introducir:

- $I_0 = \overline{z}$
- $I_1 = z$
- $I_2 = 1$
- $I_3 = 0$
- $S_0 = y$
- $S_1 = x$

La función se implementa así:



Vamos a comprobarlo. Para ello sustituimos en la ecuación del multiplexor los valores que estamos introduciendo por las entradas:

$$O = \overline{S_1} \cdot \overline{S_0} \cdot I_0 + \overline{S_1} \cdot S_0 \cdot I_1 + S_1 \cdot \overline{S_0} \cdot I_2 + S_1 \cdot S_0 \cdot I_3 = \overline{x} \cdot \overline{y} \cdot I_0 + \overline{x} \cdot y \cdot I_1 + x \cdot \overline{y} \cdot I_2 + x \cdot y \cdot I_3 =$$

$$= \overline{x} \cdot \overline{y} \cdot z + \overline{x} \cdot y \cdot z + x \cdot \overline{y} \cdot 0 + x \cdot y \cdot 1 = \overline{x} \cdot \overline{y} \cdot z + \overline{x} \cdot y \cdot z + x \cdot y = F$$

5.6.2. Método basado en la tabla de verdad

Este método se basa en lo mismo, pero se usan las tablas de verdad en vez de utilizar las ecuaciones del multiplexor, por ello es más sencillo e intuitivo. Además tiene otra ventaja: es un método mecánico, siempre se hace igual sea cual sea la función (Aunque como se verá en los ejercicios algunas funciones se pueden implementar de manera más fácil si utilizamos la entrada de validación).

Vamos a realizar este ejemplo con la función anterior. Seguimos los siguientes pasos:

1. Construimos la tabla de verdad de la función F a implementar.

X	**Y**	**Z**	**O**
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	0

2. Dividimos la tabla en tantos grupos como canales de entrada halla. En este caso hay 4 entradas, por lo que hacemos 4 grupos. Las variables de mayor peso se introducen directamente por las entradas de selección S_1 y S_0:

X	Y	Z	O
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	0

   Las variables X e Y son las que se han introducido por las entradas de selección ($S_1 = x$, $S_0 = y$). Vemos que hay 4 grupos de filas. El primer grupo se corresponde con la entrada I_0, el siguiente por la I_1, el siguiente por la I_2 y el último por la I_3.

3. El valor a introducir por las entradas I_0, I_1, I_2, e I_3 lo obtenemos mirando las columnas de la derecha (la columna de Z y de O).

   En el primer grupo, cuando Z=0, O=1 y cuando Z=1, O=0, por tanto $O = \overline{Z}$. Esa será la salida cuando se seleccione el canal 0, por tanto por su entrada habrá que introducir lo mismo: $I_0 = \overline{Z}$.

   Ahora nos fijamos en el siguiente grupo, correspondiente a I_1. En este caso, cuando Z=0, O=0 y cuando Z=1, O=1, por lo que deducimos que $O = I_1 = Z$.

   Vamos a por el tercer grupo. Si Z=0, O=0 y si Z=0, también O=0. Independientemente del valor de Z, la salida vale 0: $I_2 = 0$.

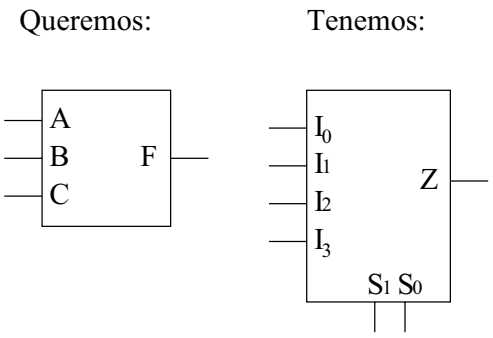
   Y para el último grupo ocurre que si Z=0, O=1, y si Z=1, O=1. Deducimos que $I_3 = 1$.

Si ahora hacemos la conexiones obtenemos el mismo circuito que en el caso anterior.

Ejemplo:

Implementar la función $F = A \cdot B + \overline{A} \cdot B \cdot \overline{C} + A \cdot \overline{B} \cdot \overline{C} + \overline{A} \cdot \overline{B} \cdot C$ utilizando un multiplexor, sin entrada de validación.

Utilizaremos el método basado en las tablas de verdad. Lo que queremos implementar es un circuito que tiene 3 entradas y una salida. Como tienen 3 variables de entrada, en general necesitaremos un multiplexor de 2 entradas de control:



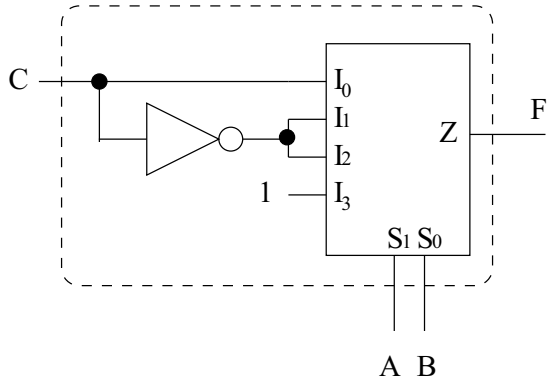
Ahora vamos siguiente los pasos del método. Primero construimos la tabla de verdad a partir de F:

A	**B**	**C**	**F**
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	1
1	1	1	1

Las entradas A y B las conectamos directamente a S_1 y S_0 respectivamente. Fijándonos en las columnas de C y F, deducimos las siguientes conexiones:

- $I_0 = C$
- $I_1 = \overline{C}$
- $I_2 = \overline{C}$
- $I_3 = 1$

El circuito final es el siguiente:



5.6.3. Implementación de funciones con multiplexores con entrada de validación

Para implementar funciones también se puede usar la entrada de validación. En este caso no todas las funciones se pueden implementar con este tipo de multiplexores. La entrada de validación la usamos como si fuese una entrada más.

Ejemplo

Implementar la siguiente función utilizando un multiplexor

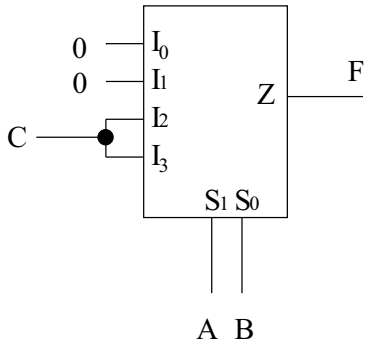
$$F = A \cdot \overline{B} \cdot C + A \cdot B \cdot C$$

Primero utilizaremos un multiplexor sin entrada de validación, utilizando el metodo de las tablas de verdad. Como la función tiene 3 variables, necesitamos un multiplexor de 2 entradas de control.

La tabla de verdad de esta función es:

A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Las entradas A y B se conectan directamente a las entradas S_1 y S_0. Los valores que se introducen por las entradas son: $I_0 = 0$, $I_1 = 0$, $I_2 = I_3 = C$. El circuito es el siguiente:



¿Se podría implementar esta función con un multiplexor con entrada de validación?. Si nos fijamos en la función F vemos que podemos sacar factor común en A:

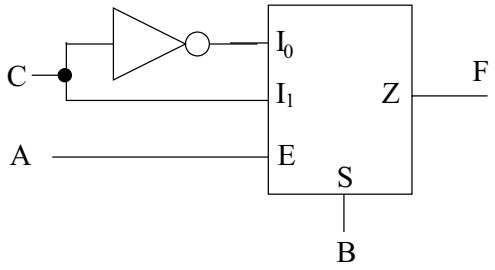
$$F = A \cdot \overline{B} \cdot C + A \cdot B \cdot C = A \cdot (\overline{B} \cdot \overline{C} + B \cdot C)$$

¡¡ Y esa es la ecuación de un multiplexor de una entrada de control y una entrada de validación!!. Si A=0, entonces F=0, y si A=1, se comporta como un multiplexo. Por tanto introducimos A directamente por la entrada de validación y para el resto necesitamos un multiplexor de 1 entrada de selección. Y como la ecuación es tan sencilla, no hace falta ni siquiera hacer el método de las tablas de verdad, fijándonos en su ecuación es suficiente.

La ecuación de un multiplexor con una entrada de selección es:

$$F = \overline{S} \cdot I_0 + S \cdot I_1$$

Si introducimos B por S, $\overline{C}$ por I_0 y C por I_1 ya lo tenemos:



5.7. Resumen

En este capítulo hemos visto los **multiplexores** y los **demultiplexores**, constituidos internamente por puertas lógicas. Los multiplexores nos permiten **seleccionar** entre uno de varios canales de entrada (tuberías) para sacarlo por la salida. Por ello disponen de unas **entradas de datos** (por donde entra el "agua"), **unas entradas de selección** (Llaves de paso) y un **canal de salida**. Estos canales de datos pueden ser de varios bits, sin embargo, en este capítulo nos hemos centrado en los multiplexores que tienen canales de datos de 1 bits, puesto que a partir de ellos podemos construir multiplexores con canales de datos de mayor cantidad de bit, así como multiplexores que tienen mayor cantidad de canales de entrada.

También hemos visto los **demultiplexores**, que realizan la función inversa. Un canal de entrada (tubería) se puede conectar a una de las diferentes salidas, según el valor introducido por las entradas de selección (llaves de paso).

Los multiplexores pueden tener opcionalmente una **entrada de validación**, que puede ser activa a nivel alto o a nivel bajo y actúa como una especie de interruptor que permite que el multiplexor funcione o no. Si está activada, el multiplexor funciona normalmente. Si la entrada de validación está desactivada, por la salida del multiplexor siempre hay un '0'.

Por último hemos visto que con un multiplexor también se pueden **implementar funciones lógicas**, y es otra alternativa que tenemos además de las puertas lógicas. Metiante el **método de las tablas de verdad**, podemos saber facilmente qué variables hay que conectar a las entradas del multiplexor.

5.8. Ejercicios

Capítulo 6

Codificadores, decodificadores y comparadores

6.1. Introducción

En este capítulo veremos otros circuitos MSI: codificadores, decodificares y comparadores.

6.2. Codificadores

6.2.1. Conceptos

Los codificadores nos permiten **"compactar"** la información, **generando un código de salida a partir de la información de entrada**. Y como siempre, lo mejor es verlo con un ejemplo. Imaginemos que estamos diseñando un circuito digital que se encuentra en el interior de una *cadena de música*. Este circuito controlará la cadena, haciendo que funcione correctamente.

Una de las cosas que hará este circuito de control será activar la radio, el CD, la cinta o el Disco según el botón que haya pulsado el usuario. Imaginemos que tenemos 4 botones en la cadena, de manera que cuando no están pulsados, generan un '0' y cuando se pulsan un '1' (Botones digitales). Los podríamos conectar directamente a nuestro circuito de control la cadena de música, como se muestra en la figura 6.1.

Sin embargo, a la hora de diseñar el circuito de control, nos resultaría más sencillo que cada botón tuviese asociado un número. Como en total hay 4 botones, necesitaríamos 2 bits para identificarlos. Para conseguir esta asociación utilizamos un codificador, que a partir del botón que se haya pulsado nos devolverá su número asociado:

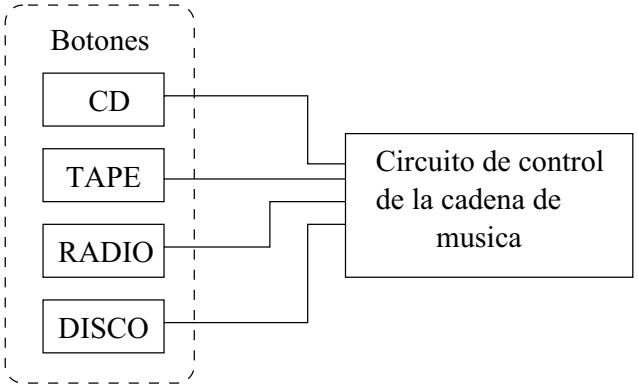
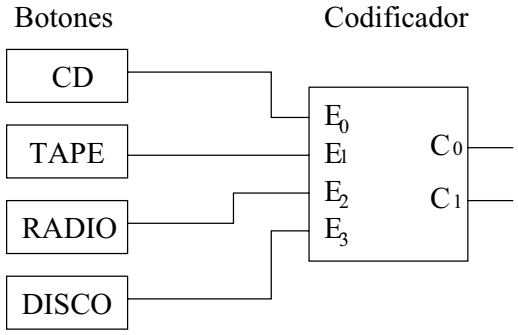


Figura 6.1: Circuito de control de una cadena de música, y 4 botones de selección de lo que se quiere escuchar



Fijémonos en las entradas del codificador, que están conectadas a los botones. **En cada momento, sólo habrá un botón apretado**, puesto que sólo podemos escuchar una de las cuatro cosas. Bien estaremos escuchando el CD, bien la cinta, bien la radio o bien un disco, pero no puede haber más de un botón pulsado[1]. Tal y como hemos hecho las conexiones al codificador, el **CD tiene asociado el número 0**, **la cinta el 1**, **la radio el 2** y **el disco el 3** (Este número depende de la entrada del codificador a la que lo hayamos conectado). **A la salida del codificador obtendremos el número del botón apretado**. La tabla de verdad será así:

E_3	E_2	E_1	E_0	C_1	C_0	**Botón**
0	0	0	1	0	0	**CD**
0	0	1	0	0	1	**TAPE**
0	1	0	0	1	0	**RADIO**
1	0	0	0	1	1	**DISCO**

El circuito de control de la cadena ahora sólo tendrá 2 bits de entrada para determinar el

[1] De hecho, en la cadena de música que tengo en casa, que es un poco antigua, cuando aprietas uno de los botones el otro "salta", y deja de estar apretado.

botón que se ha pulsado. Antes necesitábamos 4 entradas. El codificador que hemos usado tiene 4 entradas y 2 salidas, por lo que se llama **codificador de 4 a 2**. Existen codificadores de mayor número de entradas, como el que vamos a ver en el siguiente ejemplo.

Imaginemos que ahora queremos hacer un circuito para monitorizar la situación de un tren en una vía. En una zona determinada, la vía está dividida en 8 tramos. En cada uno de ellos existe un sensor que indica si el tren se encuentra en ese tramo (el sensor devuelve 1) o fuera de él (valor 0). Se ve claramente que cuando uno de los sensores esté activado, porque que el tren se encuentre en ese tramo, el resto de sensores devolverán un '0' (No detectan al tren).

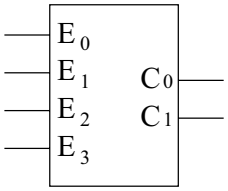
Si conectamos todas las entradas de los sensores a un **codificador de 8 a 3**, lo que tendremos es que a la salida del codificador saldrá un número que indica el tramo en el que se encuentra el tren. El circuito de control que conectemos a las salidas de este codificador sólo necesita 3 bits de entrada para conocer el tramo en el que está el tren, y no es necesario 8 bits. ¡¡Su diseño será más simple!!. La tabla de verdad es:

E_7	E_6	E_5	E_4	E_3	E_2	E_1	E_0	C_2	C_1	C_0	**Tramo**
0	0	0	0	0	0	0	1	0	0	0	**0**
0	0	0	0	0	0	1	0	0	0	1	**1**
0	0	0	0	0	1	0	0	0	1	0	**2**
0	0	0	0	1	0	0	0	0	1	1	**3**
0	0	0	1	0	0	0	0	1	0	0	**4**
0	0	1	0	0	0	0	0	1	0	1	**5**
0	1	0	0	0	0	0	0	1	1	0	**6**
1	0	0	0	0	0	0	0	1	1	1	**7**

6.2.2. Ecuaciones

A continuación deduciremos las ecuaciones de un codificador de 4 a 2, y luego utilizaremos un método rápido para obtener las ecuaciones de un codificador de 8 a 3.

El codificador de 4 a 2 que emplearemos es el siguiente:



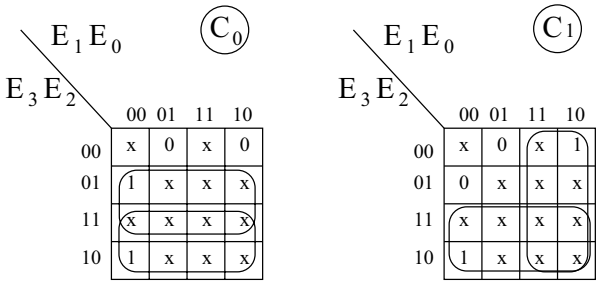
Las ecuaciones las obtenemos siguiendo el mismo método de siempre: primero obtendremos

la tabla de verdad completa y aplicaremos el método de Karnaugh. Con ello obtendremos las ecuaciones más simplificadas para las salidas C_1 y C_0.

Al hacer la tabla de verdad, hay que tener en cuenta que muchas de las entradas NO SE PUEDEN PRODUCIR. En las entradas de un decodificador, una y sólo una de las entradas estará activa en cada momento. Utilizaremos esto para simplificar las ecuaciones. Se ha utilizado una X para indicar que esa salida nunca se producirá:

E_3	E_2	E_1	E_0	C_1	C_0
0	0	0	0	x	x
0	0	0	1	0	0
0	0	1	0	0	1
0	0	1	1	x	x
0	1	0	0	1	0
0	1	0	1	x	x
0	1	1	0	x	x
0	1	1	1	x	x
1	0	0	0	1	1
1	0	0	1	x	x
1	0	1	0	x	x
1	0	1	1	x	x
1	1	0	0	x	x
1	1	0	1	x	x
1	1	1	0	x	x
1	1	1	1	x	x

C_1 y C_0 siempre valen 'x' excepto para 4 filas. Los mapas de Karnaugh que obtenemos son:



Las casillas que tienen el valor 'x' podemos asignarles el valor que más nos convenga, de forma que obtengamos la expresión más simplificada. Las ecuaciones de un decodificador de 4 a 2 son:

$$C_0 = E_2 + E_3$$

$$C_1 = E_1 + E_3$$

La manera “rápida” de obtenerlas es mirando la tabla simplificada, como la que se muestra en el ejemplo de la cadena de música. Sólo hay que fijarse en los ’1’ de las funciones de salida (como si estuviésemos desarrollando por la primera forma canónica) y escribir la variable de entrada que vale ’1’. Habrá tantos sumandos como ’1’ en la función de salida.

Las ecuaciones para un codificador de 8 a 3, utilizando el método rápido, son:

$$C_0 = E_1 + E_2 + E_5 + E_7$$

$$C_1 = E_2 + E_3 + E_6 + E_7$$

$$C_2 = E_4 + E_5 + E_6 + E_7$$

6.3. Decodificadores

6.3.1. Conceptos

Un decodificador es un circuito integrado por el que se introduce un número y se activa una y sólo una de las salidas, permaneciendo el resto desactivadas. Y como siempre, lo mejor es verlo con un ejemplo sencillo. Imaginemos que queremos realizar un circuito de control para un semáforo. El semáforo puede estar verde, amarillo, rojo o averiado. En el caso de estar averiado, se activará una luz interna “azul”, para que el técnico sepa que lo tiene que reparar. A cada una de estas luces les vamos a asociar un número. Así el rojo será el 0, el amarillo el 1, el verde el 2 y el azul (averiado) el 3 (Ver figura 6.2).

Para controlar este semáforo podemos hacer un circuito que tenga 4 salidas, una para una de las luces. Cuando una de estas salidas esté a ’1’, la luz correspondiente estará encendida. Sin embargo, ocurre que NO PUEDE HABER DOS O MAS LUCES ENCENDIDAS A LA VEZ. Por ejemplo, no puede estar la luz roja y la verde encendidas a la vez!!!!.

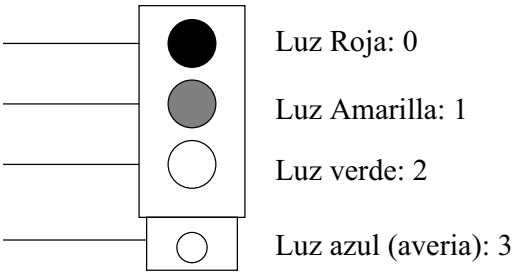


Figura 6.2: El semáforo que se quiere controlar

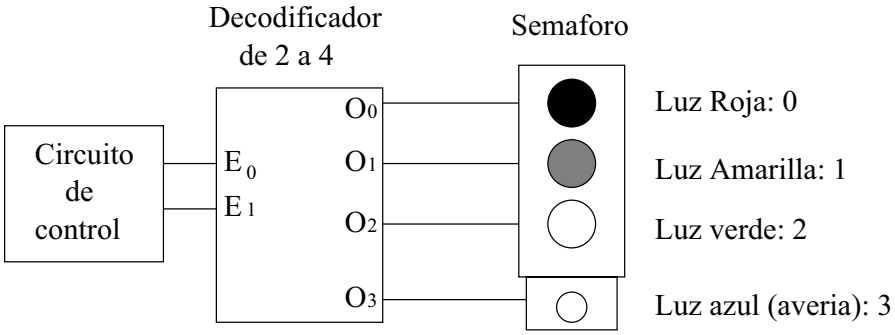


Figura 6.3: Circuito de control del semáforo, usando un decodificador de 2 a 4

Si utilizamos un decodificador de 2 a 4, conseguiremos controlar el semáforo asegurándonos que sólo estará activa una luz en cada momento. Además, el circuito de control que diseñemos sólo tienen que tener 2 salidas. El nuevo esquema se muestra en la figura 6.3.

El funcionamiento es muy sencillo. Si el circuito de control envía el número 2 ($E_1 = 1$, $E_0 = 0$), se encenderá la luz verde (que tiene asociado el número 2) y sólo la luz verde!!!. Un decodificador activa sólo una de las salidas, la salida que tiene un número igual al que se ha introducido por la entrada. En el ejemplo del semáforo, si el circuito de control envía el número 3, se activa la salida O_3 y se encenderá la luz azul (y sólo esa!!).

A la hora de diseñar el circuito de control, sólo hay que tener en cuenta que cada luz del semáforo está conectada a una salida del decodificador y que por tanto tiene asociado un número diferente.

6.3.2. Tablas de verdad y Ecuaciones

Decodificador de 2 a 4

Comenzaremos por el decodificador más sencillo, uno que tiene 2 entradas y 4 salidas, como se muestra en la figura 6.4.

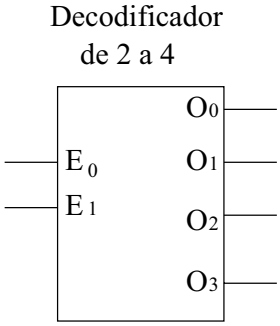


Figura 6.4: Un decodificador de 2 a 4

La tabla de verdad es la siguiente:

E_1	E_0	O_3	O_2	O_1	O_0
0	0	0	0	0	**1**
0	1	0	0	**1**	0
1	0	0	**1**	0	0
1	1	**1**	0	0	0

Y las ecuaciones las podemos obtener desarrollando por la primera forma canónica. Puesto que por cada función de salida sólo hay un '1', no se podrá simplificar (No hace falta que hagamos Karnaugh):

$$O_0 = \overline{\mathrm{E}_1} \cdot \overline{E_0}$$

$$O_1 = \overline{E_1} \cdot E_0$$

$$O_2 = E_1 \cdot \overline{E_0}$$

$$O_3 = E_1 \cdot E_0$$

La tabla de verdad la podemos expresar de forma abreviada de la siguiente manera, indicando la salida que se activa y sabiendo que las demás permanecerán desactivadas.

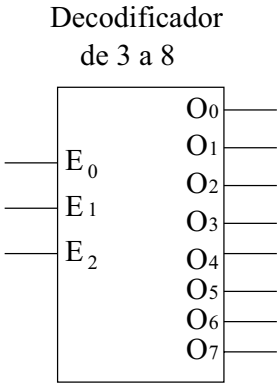


Figura 6.5: Un decodificador de 3 a 8

E_1	E_0	**Salida Activa**
0	0	O_0
0	1	O_1
1	0	O_2
1	1	O_3

Decodificicador de 3 a 8

Tiene 3 entradas y 8 salidas, como se muestra en la figura 6.5.

La tabla de verdad abreviada es la siguiente:

E_2	E_1	E_0	**Salida Activa**
0	0	0	O_0
0	0	1	O_1
0	1	0	O_2
0	1	1	O_3
1	0	0	O_4
1	0	1	O_5
1	1	0	O_6
1	1	1	O_7

Y las ecuaciones son: $O_0 = \overline{E_2} \cdot \overline{E_1} \cdot \overline{E_0}$, $O_1 = \overline{E_2} \cdot \overline{E_1} \cdot E_0$, ... ,$O_7 = E_2 \cdot E_1 \cdot E_0$.

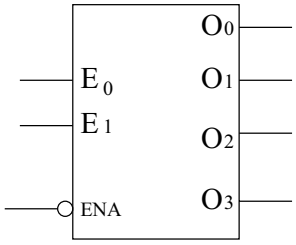


Figura 6.6: Un decodificador de 2 a 4, con entrada de validación activa a nivel bajo

6.3.3. Entradas de validación

Lo mismo que ocurría con los multiplexores y demultiplexores, existe una **entrada de validación** opcional. Si esta entrada está activada, el decodificador funciona normalmente, pero si está desactivada, sus salidas siempre estarán a '0'. Existen dos tipos de entrada de validación, las activas a nivel alto y las activas a nivel bajo.

En la figura 6.6 se muestra un decodificador de 2 a 4 con entrada de validación activa a nivel bajo, por lo el decodificador funcionará siempre que esta entrada esté a '0' y todas sus salidas permanecerán desactivadas cuando la entrada de validación esté a '1'.

Las ecuaciones de este decodificador irán multiplicadas por $\overline{ENA}$, siendo ENA la entrada de validación:

$$O_0 = \overline{\mathrm{E}_1} \cdot \overline{E_0} \cdot \overline{ENA}$$

$$O_1 = \overline{E_1} \cdot E_0 \cdot \overline{ENA}$$

$$O_2 = E_1 \cdot \overline{E_0} \cdot \overline{ENA}$$

$$O_3 = E_1 \cdot E_0 \cdot \overline{ENA}$$

Cuando por la entrada se introduce un '1' ($ENA = 1$), todas las salidas irán multiplicadas por $\overline{ENA}$, que vale '0' y todas ellas valdrán '0'. Si se introduce un '1', las ecuaciones serán las de un decodificador de 2 a 4.

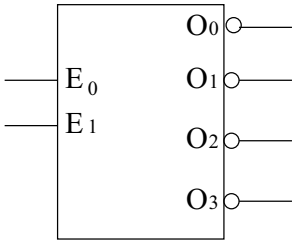


Figura 6.7: Un decodificador de 2 a 4 con salidas activas a nivel bajo

6.3.4. Tipos de decodificadores según sus salidas

Las salidas de los decodificadores pueden ser activas a nivel alto o a nivel bajo. Así, tendremos dos tipos: los **decodificadores con salidas activas a nivel alto** y los **decodificadores con salidas activas a nivel bajo**. Todos los que hemos visto hasta ahora son decodificadores activos a nivel alto, lo que quiere decir que si una salida está activa por ella sale un '1', y si está desactivada un '0'. Sin embargo, en los decodificadores con salidas activas a nivel bajo ocurre justo lo contrario.

En la figura 6.7 se muestra un decodificador de 2 a 4 con salidas a activas a nivel bajo.

La tabla de verdad completa es la siguiente:

E_1	E_0	$\overline{O_3}$	$\overline{O_2}$	$\overline{O_1}$	$\overline{O_0}$
0	0	1	1	1	**0**
0	1	1	1	**0**	1
1	0	1	**0**	1	1
1	1	**0**	1	1	1

6.4. Aplicaciones de los decodificadores

Además del uso normal de los decodificadores, como parte de nuestros diseños, existen otras aplicaciones que veremos a continuación.

6.4.1. Como Demultiplexor

Si examinamos las tablas de verdad, observamos que realmnte un decodificador con una entrada de validación se comporta como un demultiplexor. De hecho no existen circuitos integrados con demultiplexores, sino que se usan decodificadores. Imaginemos que necesitamos utilizar un demultiplexor de dos entradas de selección, como el mostrado en la figura XX.