2 FUNCIONES BOOLEANAS Y SU SIMPLIFICACION

2.1. Funciones Lógicas
2.2. Simplificación de funciones booleanas: mapas de Karnaugh
2.3. Ejercicios de síntesis y simplificación de funciones booleanas
2.4. Decodificadores y multiplexores; otras formas de configurar funciones

El álgebra de Boole permite expresar, en forma de funciones matemáticas, tanto la realización de cálculos en el sistema binario como la adopción de decisiones a través de la combinación de proposiciones.

Cantidades y cualidades pueden ser representadas por conjuntos de «ceros» y «unos», es decir, mediante palabras binarias cuyos dígitos pueden adoptar solamente los valores 0 y 1; cada dígito o «bit» corresponde a una variable.

Una función booleana establece una dependencia entre una variable de salida "y" y un conjunto de variables de entrada "a b c...": una correspondencia entre el conjunto de valores de las variables de entrada y el valor de la variable de salida.

Las funciones booleanas son «multiformes», es decir, pueden representarse de muy diversas formas: desde el mero enunciado textual que expresa las especificaciones o requisitos que definen la función, hasta su forma algebraica como operaciones entre variables, pasando por su tabla funcional (o «tabla de verdad») que detalla, en forma de listado, el valor de la función para cada conjunto de valores de las entradas.

Precisamente el diseño del circuito digital correspondiente a una función booleana consiste en el «cambio de forma» de la misma, a partir de su enunciado, construyendo su tabla funcional y extrayendo de ella la forma algebraica de la función; dicha expresión algebraica puede ser trasladada directamente a un esquema de puertas lógicas que conforma el circuito digital de dicha función.

En este proceso resulta de mucha importancia la simplificación de la expresión algebraica de la función, de forma que contenga el menor número de términos y el menor número de variables posible. Al reducir la expresión algebraica disminuye el tamaño, la complejidad y el coste (y, en muchos casos, aumenta la velocidad) del circuito digital que permite «obtener» tal función. Con esta finalidad, los «mapas de Karnaugh» constituyen una eficaz herramienta gráfica de simplificación «a mano», mientras que el método de Quine-McCluskey proporciona las bases algorítmicas para programar la simplificación sobre un computador.

También es posible configurar el circuito digital de una función booleana sin llegar a su expresión algebraica, directamente desde su tabla funcional. Para ello pueden utilizarse dos bloques digitales de tipo «selector»: el decodificador y el multiplexor; ambos incluyen todas las posibilidades de valores de sus variables de entrada y permiten activar cada una de dichas posibilidades.

Tomás Pollán Santamaría. Tecnología Electrónica. E.U.I.T.I.Z. Universidad de Zaragoza. tpollan@posta.unizar.es

2.1. Funciones Lógicas

Dentro del Álgebra de Boole de 2 elementos, una función booleana o función lógica es una expresión de operaciones booleanas enlazando variables que solamente pueden adquirir los valores **0** y **1**. Una función booleana es una aplicación que a cada conjunto de valores booleanos de sus variables le asigna un y sólo un valor booleano.

La primera de las dos definiciones anteriores es de tipo «descriptivo»: describe la forma algebraica de una función booleana; mientras que la segunda es de tipo «conceptual»: identifica la función como correspondencia entre el conjunto de valores de las variables y el valor booleano de la variable dependiente.

En una función **f** designaremos con el nombre de *variables de entrada* $\mathbf{x_i}$ al conjunto de sus variables propias y denominaremos *variable de salida* **y** a la variable dependiente o resultado de la propia función $\mathbf{y = f(x_i)}$.

De acuerdo con las definiciones anteriores, las funciones lógicas pueden representarse en dos formas diferentes:

- por su *expresión algebraica* o *fórmula booleana*, como expresión de las operaciones que ligan a sus variables;
- por su *tabla operativa* o *tabla de verdad*, expresando en forma de tabla la correspondencia entre la variable de salida y cada combinación posible de valores de sus variables de entrada.

También puede expresarse una función en forma de enunciado o texto que manifiesta las especificaciones o requisitos que dan lugar a dicha función y en forma gráfica como circuito digital o esquema de puertas lógicas que «produce» los valores de salida de la función al recibir los correspondientes valores en sus entradas.

El proceso de síntesis o «construcción digital» de una función parte del enunciado o especificaciones de la misma, para configurar la «tabla de verdad» de la función y obtener, a través de ella, su expresión algebraica; una vez simplificada, dicha expresión puede ser directamente trasladada a un esquema de puertas como representación gráfica del circuito digital que «hace efectiva» dicha función.

enunciado $\rightarrow$ *tabla funcional* $\rightarrow$ *expresión algebraica* $\rightarrow$ *esquema de puertas*

Dada una función de **m** variables, cada una de las posibles combinaciones de valores de dichas **m** variables recibe el nombre de *vector de entrada*; el número total de vectores de entrada será $\mathbf{2^m}$ y tal será el número de filas que ha de tener la tabla funcional completa.

Para cada vector de entrada podemos construir un *término mínimo*, formado por el producto booleano (operación "**y**") de las **m** variables de entrada, estando cada una de ellas afirmada si su valor en el vector de entrada es **1** y negada cuando vale **0**.

Un término mínimo da resultado **1** al asignar a sus variables los valores del vector de entrada que le corresponde y, en cambio, para cualquier otro vector de entrada da resultado **0**.

Así pues, tal como están construidos, los términos mínimos poseen la propiedad de «seleccionar» o «filtrar» a su propio vector de entrada: a cada vector de entrada le corresponde un término mínimo y cada termino mínimo es un *discriminador* o *selector* del vector de entrada al que corresponde.

*Ejemplos para 4 variables (**a**, **b**, **c**, **d**):*

vector de entrada ***1001***	*término mínimo*	$a.\bar{b}.\bar{c}.d$	$1.\bar{0}.\bar{0}.1 = 1$
vector de entrada ***1100***	*término mínimo*	$a.b.\bar{c}.\bar{d}$	$1.1.\bar{0}.\bar{0} = 1$
vector de entrada ***0010***	*término mínimo*	$\bar{a}.\bar{b}.c.\bar{d}$	$\bar{0}.\bar{0}.1.\bar{0} = 1$
vector de entrada ***0000***	*término mínimo*	$\bar{a}.\bar{b}.\bar{c}.\bar{d}$	$\bar{0}.\bar{0}.\bar{0}.\bar{0} = 1$
vector de entrada ***1111***	*término mínimo*	$a.b.c.d$	$1.1.1.1 = 1$

número de vectores de entrada y de términos mínimos posibles: $\mathbf{2^4 = 16}$

Dualmente se construye el *término máximo* correspondiente a un vector de entrada mediante la suma booleana (operación "**o**") de sus variables, afirmadas cuando su valor es **0** y negadas cuando valen **1**. Un término máximo da resultado **0** al asignar a sus variables los valores del vector de entrada que le corresponde y adopta el valor **1** para cualquier otro vector.

A cada vector de entrada le corresponden un término máximo y cada término máximo es, asimismo, un *selector* o *discriminador* del vector al que corresponde.

*Ejemplos para 4 variables (**a**, **b**, **c**, **d**) :*

vector de entrada ***1001***	*término máximo*	$\bar{a}+b+c+\bar{d}$	$\bar{1}+0+0+\bar{1}=0$
vector de entrada ***1100***	*término máximo*	$\bar{a}+\bar{b}+c+d$	$\bar{1}+\bar{1}+0+0=0$
vector de entrada ***0010***	*término máximo*	$a+b+\bar{c}+d$	$0+0+\bar{1}+0=0$
vector de entrada ***0000***	*término máximo*	$a+b+c+d$	$0+0+0+0=0$
vector de entrada ***1111***	*término máximo*	$\bar{a}+\bar{b}+\bar{c}+\bar{d}$	$\bar{1}+\bar{1}+\bar{1}+\bar{1}=0$

número de vectores de entrada y de términos máximos posibles: $\mathbf{2^4 = 16}$

Se denomina **forma canónica** de una función booleana a su expresión como suma (operación "**o**") de términos mínimos de sus variables; para construirla a partir de su tabla funcional bastará tomar todos aquellos términos mínimos que corresponden a vectores de entrada que hacen la función igual a **1** y sumarlos.

Habida cuenta de la propiedad de filtrado o selección que tienen los términos mínimos, dicha suma de términos mínimos asigna resultado **1** para aquellos vectores de entrada cuyo término mínimo se encuentra presente en la misma y resultado **0** para todos los demás vectores de entrada.

Ejemplo: función "ser número primo" para números de 3 dígitos.

c	**b**	**a**	ser nº primo		
0	**0**	**0**	**0**		$\leftarrow c+b+a$
0	**0**	**1**	**1**	$\leftarrow \bar{c}.\bar{b}.a$	
0	**1**	**0**	**1**	$\leftarrow \bar{c}.b.\bar{a}$	
0	**1**	**1**	**1**	$\leftarrow \bar{c}.b.a$	
1	**0**	**0**	**0**		$\leftarrow \bar{c}+b+a$
1	**0**	**1**	**1**	$\leftarrow c.\bar{b}.a$	
1	**1**	**0**	**0**		$\leftarrow \bar{c}+\bar{b}+a$
1	**1**	**1**	**1**	$\leftarrow c.b.a$	

*Para obtener la forma canónica de esta función hemos de tomar los términos mínimos de los vectores de entrada que dan resultado **1**.*

$$y = \bar{c}.\bar{b}.a + \bar{c}.b.\bar{a} + \bar{c}.b.a + c.\bar{b}.a + c.b.a \quad \text{forma canónica}$$

simplificando

$$= \bar{c}.a + \bar{c}.b + c.a = a + \bar{c}.b$$

La **forma canónica dual** de una función booleana corresponde a su expresión como producto (operación "**y**") de términos máximos; para construirla se toman todos aquellos términos máximos que corresponden a vectores de entrada que hacen la función igual a **0** y se realiza el producto de ellos.

La propiedad de filtrado o selección que tienen los términos máximos asegura que su producto asigna resultado **0** para aquellos vectores de entrada cuyo término máximo está presente y resultado **1** para todos los demás vectores de entrada.

*En el ejemplo anterior (función "ser número primo" para números de 3 dígitos), para obtener la forma canónica dual tomamos los términos máximos de los vectores de entrada que dan resultado **0**.*

$$y = (c+b+a).(\bar{c}.+b+a).(\bar{c}+\bar{b}+a) \quad \text{forma canónica dual}$$

simplificando

$$= (b+a).(\bar{c}+a)$$

La forma canónica de una función es única, salvo en lo que se refiere al orden de los términos y al de las variables en los mismos; igualmente es única la forma canónica dual. En cambio, una misma función puede tener expresiones simplificadas muy diversas.

Dos funciones de **m** variables diferirán en los términos mínimos que contenga su forma canónica; el número de funciones diferentes de **m** variables coincide con el de combinaciones posibles de sus $\mathbf{2^m}$ términos mínimos: $\mathbf{2^k}$ siendo $\mathbf{k = 2^m}$.

El número de vectores de entrada, y consiguientemente el de funciones posibles, crece fuertemente al hacerlo el número de variables de la función. Hay 16 funciones de 2 variables, las funciones de 4 variables tendrán 16 vectores de entrada, siendo posibles 2^{16} = 65.536 funciones y para 6 variables existen 64 vectores de entrada con un número de funciones posibles 2^{64} superior a 10^{18} (trillones de ellas).

2.2. Simplificación de funciones booleanas: mapas de Karnaugh

La aplicación de los teoremas del álgebra permite simplificar las funciones, reduciendo el número de puertas necesarias para su configuración; en concreto, los teoremas más útiles para la simplificación son los de idempotencia y absorción:

$a + a = a$ $\quad a + \bar{a} = 1$ $\quad a + a.b = a$ $\quad a + \bar{a}.b = a + b$

$a \,.\, a = a$ $\quad a \,.\, \bar{a} = 0$ $\quad a.(a + b) = a$ $\quad a.(\bar{a} + b) = a.b$

y, en ocasiones, el teorema de consenso: $a.b + \bar{a}.c + b.c = a.b + \bar{a}.c$

Ejemplo: $Y = \bar{d}.c.\bar{b}.\bar{a} + \bar{d}.c.\bar{b}.a + d.\bar{c}.\bar{b}.\bar{a} + d.\bar{c}.b.\bar{a} + d.c.\bar{b}.\bar{a} + d.c.\bar{b}.a + d.c.b.\bar{a}$

($x + \bar{x} = 1$ aplicado a las siguientes parejas de términos: 1º y 2º x=a; 3º y 4º x=b; 5º y 6º x=a ; 7º y 4º x=c , utilizando, en este último caso, también, x=x+x)

$$Y = \bar{d}.c.\bar{b} + d.\bar{c}.\bar{a} + d.c.\bar{b} + d.b.\bar{a}$$

($d + \bar{d} = 1$ *aplicado a los términos 1º y 3º y sacando factor común en los otros dos)*

$$Y = c.\bar{b} + d.\bar{a}.(\bar{c} + b)$$

(aplicando el teorema de Morgan al paréntesis del segundo término)

$$Y = c.\bar{b} + d.\bar{a}.(\overline{\bar{c}.b})$$

(y, finalmente, el teorema de absorción $x + \bar{x}.z = x + z$ *aplicado a ambos términos)*

$$Y = c.\bar{b} + d.\bar{a}.$$

La aplicación directa de teoremas booleanos para simplificar las funciones requiere una cierta habilidad, cuyos resultados dependen de la complejidad de la función y de la experiencia e intuición de quien la realiza.

Existen métodos de simplificación que aportan una formulación sistemática del proceso y que aseguran la máxima simplificación; los más utilizados de ellos son el método gráfico de los *mapas de Karnaugh*, que se describe a continuación, y el método algorítmico de *Quine-McCluskey* (cuya descripción se encuentra en el apéndice A1).

Los mapas de Karnaugh son el método habitual de simplificación cuando se hace «a mano» y el número de variables de la función es pequeño (no superior a 6). Para mayor número de variables se recurre a la ayuda del computador, con programas de simplificación automática que suelen estar basados en el algoritmo de Quine-MacCluskey.

La simplificación de una función por medio de los *mapas de Karnaugh* se realiza dibujando su tabla de operación en un diagrama bidimensional según la estructura siguiente:

C \ B A	*0 0*	*0 1*	*1 1*	*1 0*
0				
1				

Mapa de Karnaugh para 3 variables

D C \ B A	*0 0*	*0 1*	*1 1*	*1 0*
0 0				
0 1				
1 1				
10				

Mapa de Karnaugh para 4 variables

E D \ C B A	*0 0 0*	*0 0 1*	*0 1 1*	*0 1 0*	*1 1 0*	*1 1 1*	*1 0 1*	*1 0 0*
0 0								
0 1								
1 1								
1 0								

Mapa de Karnaugh para 5 variables

F E D \ C B A	0 0 0	0 0 1	0 1 1	0 1 0	1 1 0	1 1 1	1 0 1	1 0 0
0 0 0								
0 0 1								
0 1 1								
0 1 0								
1 1 0								
1 1 1								
1 0 1								
1 0 0								

Mapa de Karnaugh para 6 variables

La estructura de los mapas de Karnaugh aprovecha las propiedades del *código Gray*, en el que dos números o vectores sucesivos difieren únicamente en el valor de una variable.

	Binario	*Gray*
0	**0**	**0**
1	**1**	**1**
2	**10**	**11**
3	**11**	**10**
4	**100**	**110**
5	**101**	**111**
6	**110**	**101**
7	**111**	**100**
8	**1000**	**1100**
9	**1001**	**1101**
15	**1111**	**1000**

Una forma sencilla de generar el *código Gray* para números sucesivos a partir del 0 es la representada en el siguiente esquema:

0 1 | 11 10 | 110 111 101 100 | 1100 1101 …

*Las líneas verticales señalan la necesidad de añadir un nuevo dígito y actúan «a manera de espejo», de forma que tras cada línea vertical los números empiezan por un primer dígito adicional **1** y el resto de sus dígitos son iguales a los del número que se encuentra en posición simétrica respecto a dicha línea vertical, «simetría especular», añadiendo, en su caso, los ceros que fueran necesarios.*

El código Gray aparece con frecuencia en los sistemas digitales por sus prestaciones en relación con la simplificación y con la seguridad funcional, derivadas ambas de la propiedad de que dos números sucesivos solamente difieren en el valor de un dígito (son iguales salvo en un bit).

Los términos mínimos correspondientes a dos vectores sucesivos, según el código Gray, son simplificables entre sí pues difieren solamente en el valor de una de sus variables. Por ello, los mapas de Karnaugh tienen la propiedad de que dos cuadros adyacentes se pueden simplificar entre sí.

El proceso de simplificación en un mapa de 4 variables consiste en agrupar los cuadros para formar rectángulos que contengan un número de cuadros potencia de 2: 1, 2, 4, 8; los cuadros contenidos en cada uno de dichos rectángulos son simplificables entre sí y conducen a un solo término:

DC \ B A	0 0	0 1	1 1	1 0
0 0				
0 1		**1**	**1**	
1 1		**1**	**1**	
1 0				

y = C.A

D C \ B A	0 0	0 1	1 1	1 0
0 0		**1**		
0 1		**1**		
1 1		**1**		
1 0		**1**		

y = $\overline{B}$.A

DC \ B A	0 0	0 1	1 1	1 0
0 0			**1**	**1**
0 1				
1 1				
1 0			**1**	**1**

y = $\overline{C}$.B

D C \ B A	0 0	0 1	1 1	1 0
0 0	**1**			**1**
0 1				
1 1				
1 0	**1**			**1**

y = $\overline{C}.\overline{A}$

(téngase en cuenta que el diagrama conecta consigo mismo por sus bordes vertical y horizontal)

DC \ BA	00	01	11	10
00				
01				
11	**1**	**1**		
10				

y = D.C.$\overline{B}$

DC \ BA	00	01	11	10
00			**1**	
01			**1**	
11				
10				

y = $\overline{D}$.B.A

En el caso de 5 variables, el mapa de Karnaugh está conformado por dos hojas de dimensión 4 x 4 y son simplificables entre sí aquellos rectángulos que se encuentran en posición simétrica respecto a la separación de las dos hojas:

ED \ CBA	000	001	011	010	110	111	101	100
00								
01	**1**	**1**					**1**	**1**
11	**1**	**1**					**1**	**1**
10								

y = D.$\overline{B}$

Para 6 variables, el mapa de Karnaugh consta de 4 hojas de dimensión 4 x 4 e igualmente la simplificación entre hojas actúa por simetría especular:

FED \ CBA	000	001	011	010	110	111	101	100
000								
001				**1**	**1**			
011				**1**	**1**			
010								
110								
111				**1**	**1**			
101				**1**	**1**			
100								

y = D.B.$\overline{A}$

Si bien en todos los casos anteriores se ha aplicado la simplificación a cuadros con valor **1**, lo cual conduce a términos en forma de productos, *términos producto*, por dualidad el mismo procedimiento de simplificación puede aplicarse en forma análoga a cuadros con valor **0**, dando lugar a *términos suma.*

Dada una función y representada su tabla de verdad en forma de mapa de Karnaugh la expresión más simple de dicha función como suma de términos producto se obtiene agrupando todos los cuadros de valor **1** en el menor número de rectángulos simplificables y expresando la suma de los términos que corresponden a dichos rectángulos.

Dualmente, la forma más simple de tal función como producto de términos suma se obtiene agrupando los cuadros con valor **0** en el menor número de rectángulos y expresando el producto de los términos duales que les corresponden.

Ejemplo:

Sea la función "ser número primo" en el caso de números binarios de 5 dígitos ***edcba****.*

ed \ cba	000	001	011	010	110	111	101	100
00	**0**	**1**	**1**	**1**	**0**	**1**	**1**	**0**
01	**0**	**0**	**1**	**0**	**0**	**0**	**1**	**0**
11	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**0**
10	**0**	**1**	**1**	**0**	**0**	**1**	**0**	**0**

Recorriendo los cuadros con valor 1 para formar rectángulos simplificables desde la izquierda hacia la derecha y de arriba hacia abajo :

$$y = \overline{e}.\overline{d}.a + \overline{e}.\overline{d}.\overline{c}.b + \overline{e}.\overline{c}.b.a + \overline{d}.\overline{c}.a + d.c.\overline{b}.a + e.c.b.a$$

Simplificando en forma de producto de sumas, para lo cual han de recorrerse los cuadros con valor 0 formando rectángulos simplificables:

$$y = (b + a).(\overline{d} + c + b).(\overline{d} + a).(\overline{e} + \overline{d} + c).(\overline{e} + a).(\overline{c} + a).(e + \overline{d} + \overline{c} + \overline{b}).(\overline{e} + d + \overline{c} + b)$$

Aunque esta segunda (producto de sumas) parece más extensa que la anterior (suma de productos), ambas contienen el mismo número de variables (22 variables).

En aquellos casos en que algún vector de entrada no puede presentarse nunca o bien cuando no importa (*da igual*) el valor que adquiera la función booleana para algún vector de entrada, se anota con **X** el valor que corresponde a tales vectores de entrada. El símbolo **X** (*don´t care*) expresa que es indiferente el valor que la función pueda tener y, posteriormente, se utiliza en lugar de **X** el valor **0** o el valor **1** según interese en el proceso de simplificación.

*Consideremos las nueve cifras decimales codificadas en binario (código **BCD**) y la misma función booleana "ser número primo" aplicada a ellas.*

*El número binario para representar las cifras decimales, 0 - 9, ha de disponer de 4 dígitos **dcba**; son números primos las siguientes cifras: 1, 2, 3, 5 y 7. El mapa de Karnaugh correspondiente a la función que determina si una cifra decimal **BCD** es número primo o no será el siguiente:*

D C \ B A	*0 0*	*0 1*	*1 1*	*1 0*
0 0	**0**	**1**	**1**	**1**
0 1	**0**	**1**	**1**	**0**
1 1	**X**	**X**	**X**	**X**
1 0	**0**	**0**	**X**	**X**

*El símbolo **X** en un cuadro indica que resulta indiferente el valor booleano que adopte la función para el correspondiente vector de entrada; en este caso ello es debido a que no van a presentarse entradas con valor decimal superior a 9.*

*Los cuadros con valor **X** son tomados como **1** o como **0** según interese a efectos simplificativos. De esta forma podemos agrupar los cuadros con valor **1** en dos cuadrados (asignando valor **1** a los dos cuadros con valor **X** de la última fila), cuyos términos producto son, respectivamente:*

$$y = \bar{d}.a + \bar{c}.b \quad = \overline{\overline{\bar{d}.a + \bar{c}.b}} = \overline{(\bar{d}.a).(\bar{c}.b)} = (\bar{d} * a) * (\bar{c} * b)$$

2.3. Ejercicios de síntesis y simplificación de funciones booleanas

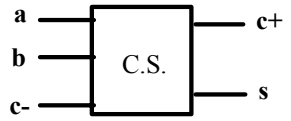
2.3.1. Expresar en forma de funciones booleanas la suma aritmética de dos números binarios de varios dígitos

La suma aritmética de dos bits requiere dos funciones booleanas, una para expresar el dígito resultante y la otra para el posible dígito de *acarreo* o arrastre (*carry*):

1+1 = 10 ; resultado **0** y «me llevo» **1**.

Para configurar un sumador de números de varios dígitos, podemos proceder modularmente, empleando una celda para cada dígito: dicha celda ha de tener, además de las entradas correspondientes a los dos dígitos de ambos números, una entrada adicional de acarreo que reciba el arrastre (*me llevo*) resultante de la suma de los dígitos anteriores.

Célula Sumadora



Se indica con la notación **c+** el acarreo resultante de la suma y con **c-** el acarreo anterior, que participa como sumando; la tabla funcional será la siguiente:

c-	**b**	**a**	suma: **s**	acarreo: **c+**
0	**0**	**0**	**0**	**0**
0	**0**	**1**	**1**	**0**
0	**1**	**0**	**1**	**0**
0	**1**	**1**	**0**	**1**
1	**0**	**0**	**1**	**0**
1	**0**	**1**	**0**	**1**
1	**1**	**0**	**0**	**1**
1	**1**	**1**	**1**	**1**

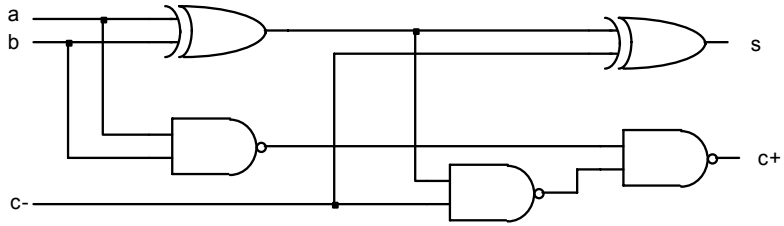
suma:

$$s = \bar{c}_-.(\bar{b}.a + b.\bar{a}) + c_-.(\bar{b}.\bar{a} + b.a) = \bar{c}_-.(b \oplus a) + c_-.(b \otimes a) =$$
$$= \bar{c}_-.(b \oplus a) + c_-.(\overline{b \oplus a}) = c_- \oplus (b \oplus a)$$

arrastre:

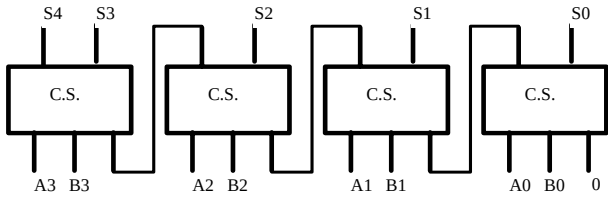
$$c_+ = \bar{c}_-.b.a + c_-.(\bar{b}.a + b.\bar{a} + b.a) = b.a + c_-.(\bar{b}.a + b.\bar{a}) =$$
$$= b.a + c_-.(b \oplus a) = (b * a) * \left[c_- * (b \oplus a)\right]$$

Para configurar estas funciones se requieren 3 puertas "**y-negada**" y 2 puertas "**o-exclusiva**" conectadas según la figura siguiente:



Si se desea utilizar únicamente puertas "**y-negada**" serán necesarias 11 puertas (ya que una puerta "**o-exclusiva**" puede configurarse con 4 puertas "**y-negada**").

Conectando **n** de estas células sumadoras «en cadena» (salida **c+** unida a entrada **c-** de la celda siguiente) se obtendrá un sumador de dos números de **n** dígitos.



2.3.2. *Sean dos números binarios de dos dígitos; deducir las tres funciones que realizan la comparación entre ambos números (mayor, menor e igual), expresándolas mediante operaciones "y-negada" (Nand) e inversores*

Sean los dos números: $\mathbf{a_1a_0}$ y $\mathbf{b_1b_0}$; la forma sistemática de sintetizar las funciones de comparación entre ellos requiere escribir la tabla de verdad de las tres funciones, pero resulta más sencillo acudir a un razonamiento directo:

- para comparar dos números hay que comenzar por la cifra más significativa y, en el caso de que ambos dígitos sean iguales, irse desplazando hacia la derecha efectuando la comparación cifra a cifra
- dos dígitos son iguales cuando ambos valen **0** o ambos valen **1**: $\bar{a}.\bar{b} + a.b$.

Los dos números propuestos serán iguales cuando tengan iguales su primer y su segundo dígito:

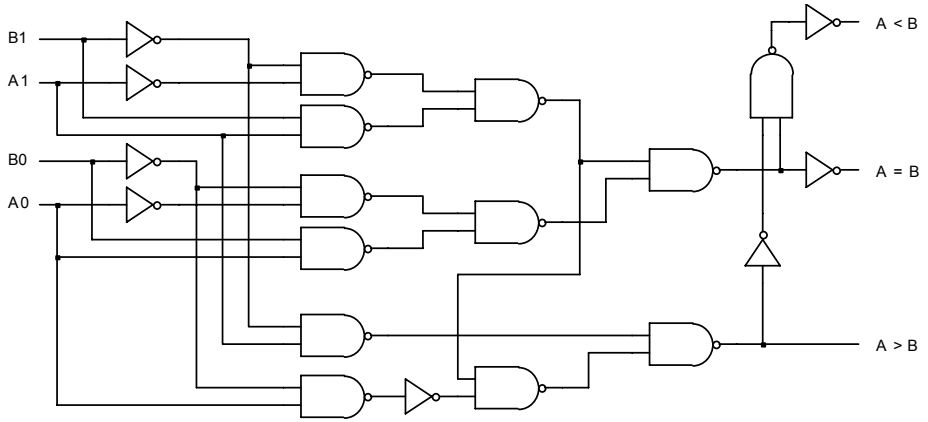
$$y_1 = (\overline{a_1}.\overline{b_1} + a_1.b_1).(\overline{a_0}.\overline{b_0} + a_0.b_0) = \overline{\left[(\overline{a_1} * \overline{b_1}) * (a_1 * b_1)\right] * \left[(\overline{a_0} * \overline{b_0}) * (a_0 * b_0)\right]}$$

función que requiere 7 puertas "**y-negada**" y 5 inversores.

El número **A** será mayor que el **B** en caso de que el dígito $\mathbf{a_1}$ sea **1** y el $\mathbf{b_1}$ sea **0**, o también en caso de que, siendo ambos dígitos iguales, $\mathbf{a_0}$ tenga valor **1** y $\mathbf{b_0}$ sea **0**:

$$y_2 = a_1.\overline{b_1} + (\overline{a_1}.\overline{b_1} + a_1.b_1).(a_0.\overline{b_0}) = (a_1 * \overline{b_1}) * \left[\left((\overline{a_1} * \overline{b_1}) * (a_1 * b_1)\right) * \overline{(a_0 * \overline{b_0})}\right]$$

función que necesita otras 7 puertas "**y-negada**" y otros 5 inversores, pero 3 de las puertas y 4 inversores coinciden con puertas utilizadas en la función anterior.



La función **A < B** ha sido construida a partir de las dos anteriores (**A=B** y **A>B**): dicha función debe adoptar el valor **1** cuando las otras dos valgan **0** (es decir cuando no sea ni **A=B** ni **A>B**): $y_3 = \overline{y_1}.\overline{y_2} = \overline{y_1 + y_2} = y_1 \Delta y_2 = \overline{\overline{y_1} * \overline{y_2}}$

Se necesitan en total 12 puertas "**y-negada**" (*Nand*) y 8 inversores.

2.3.3. *Sean dos números binarios de tres dígitos; expresar la función booleana que corresponde a* $\boldsymbol{A \leq B}$ *y generalizarla para números de 5 dígitos*

Sean los dos números: $\mathbf{a_2a_1a_0}$ y $\mathbf{b_2b_1b_0}$; **A** será menor o igual que **B**

- si el dígito más significativo de **A** $\mathbf{a_2}$ es **0** y el de **B** $\mathbf{b_2}$ es **1**
- o también, si dichos dígitos son iguales y el siguiente dígito de **A** $\mathbf{a_1}$ es **0** y el de **B** $\mathbf{b_1}$ es **1**
- o también, si son iguales $\mathbf{a_2}$ y $\mathbf{b_2}$ y, también, $\mathbf{a_1}$ y $\mathbf{b_1}$ y, además, el dígito de las unidades de **A** $\mathbf{a_0}$ es **0** y el de **B** $\mathbf{b_0}$ es **1** o bien son iguales $\mathbf{a_0}$ y $\mathbf{b_0}$.

$$y = \overline{a_2}.b_2 + (\overline{a_2}.\overline{b_2} + a_2.b_2).\left[\overline{a_1}.b_1 + (\overline{a_1}.\overline{b_1} + a_1.b_1).(\overline{a_0}.b_0 + \overline{a_0}.\overline{b_0} + a_0.b_0)\right] =$$
$$= \overline{a_2}.b_2 + (\overline{a_2}.\overline{b_2} + a_2.b_2).\left[\overline{a_1}.b_1 + (\overline{a_1}.\overline{b_1} + a_1.b_1).(\overline{a_0} + b_0)\right]$$

La expresión anterior se puede simplificar, aplicando el teorema de absorción:

$$y = \overline{a_2}.b_2 + (\overline{a_2} + b_2).\left[\overline{a_1}.b_1 + (\overline{a_1} + b_1).(\overline{a_0} + b_0)\right]$$

Y su generalización para números de cinco dígitos, $\mathbf{a_4a_3a_2a_1a_0}$ y $\mathbf{b_4b_3b_2b_1b_0}$, es directa:

$$\overline{a_4}.b_4 + (\overline{a_4} + b_4).\left\{\overline{a_3}.b_3 + (\overline{a_3} + b_3).\left\{\overline{a_2}.b_2 + (\overline{a_2} + b_2).\left[\overline{a_1}.b_1 + (\overline{a_1} + b_1).(\overline{a_0} + b_0)\right]\right\}\right\}$$

De la misma forma esta función puede generalizarse para números de más dígitos.

2.3.4. *El Concejo Municipal de una localidad pequeña está formado por un alcalde con dos votos, un secretario con otros dos votos y tres delegados de barrio con un voto cada uno. Los acuerdos se toman por mayoría simple, pero el voto en contra simultáneo de los tres delegados supone un veto al acuerdo. Sintetizar y simplificar la función booleana que expresa el resultado de las votaciones*

Dada la complejidad de la función (32 vectores de entrada) y para obtener su expresión simplificada, se expresa la tabla de verdad directamente en forma de mapa de Karnaugh:

d1 d2 d3 / a s	*0 0 0*	*0 0 1*	*0 1 1*	*0 1 0*	*1 1 0*	*1 1 1*	*1 0 1*	*1 0 0*
0 0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
0 1	**0**	**0**	**1**	**0**	**1**	**1**	**1**	**0**
1 1	**0**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
1 0	**0**	**0**	**1**	**0**	**1**	**1**	**1**	**0**

En el anterior mapa de Karnaugh todos los cuadrados con valor **1** pueden agruparse de 4 en 4 y así resultan los siguientes términos producto, comenzando de arriba hacia abajo y de izquierda a derecha:

$$y = s.d_2.d_3 + a.s.d_1 + a.s.d_2 + a.d_2.d_3 + s.d_1.d_2 + a.d_1.d_2 + s.d_1.d_3 + a.d_1.d_3 + a.s.d_3$$

y agrupando términos análogos:

$$y = s.(d_2.d_3 + d_1.d_2 + d_1.d_3) + a.s.(d_1 + d_2 + d_3) + a.(d_2.d_3 + d_1.d_2 + d_1.d_3) =$$
$$= a.s.(d_1 + d_2 + d_3) + (s + a).(d_2.d_3 + d_1.d_2 + d_1.d_3)$$

Solución a la que podría haberse llegado por simple razonamiento directo:

Se aprobará un acuerdo cuando voten a favor el alcalde y el secretario y uno cualquiera de los delegados, o también cuando lo hagan el alcalde o el secretario y dos delegados.

Esta frase constituye la lectura en lógica proposicional de la función anterior.

2.3.5. *En una instalación se controla la presión, la temperatura y la intensidad eléctrica consumida, de forma que debe activarse una alarma cuando alguno de estos parámetros sobrepase un valor límite detectado por un transductor con salida digital ("1" por encima de dicho valor y "0" por debajo). Se controla también la tensión que alimenta la instalación, de forma que la alarma también se active cuando ésta sea inferior a un valor mínimo. Construir la función booleana que debe accionar la alarma*

V I \ T P	*0 0*	*0 1*	*1 1*	*1 0*
0 0	**1**	**1**	**1**	**1**
0 1	**1**	**1**	**1**	**1**
1 1	**1**	**1**	**1**	**1**
1 0	**0**	**1**	**1**	**1**

El *mapa* de Karnaugh de esta función está completamente cubierto por el valor 1 salvo en una posición; por ello, resulta más apropiada la síntesis de la función por términos máximos, pues solamente es preciso recoger uno de ellos.

"alarma" = $\overline{V} + I + T + P$

es decir, *la alarma suena cuando la tensión está por debajo del valor umbral o cuando la intensidad, la temperatura o la presión están por encima del valor límite*.

2.3.6. *Sea la función booleana siguiente, expresada en forma de suma de productos; obtener la expresión algebraica de dicha función en forma de producto de sumas*

$$y = \overline{d}.\overline{c}.\overline{b} + \overline{d}.\overline{b}.\overline{a} + d.\overline{c}.b + d.b.\overline{a}$$

Se trata, en primer lugar, de obtener el mapa de Karnaugh de dicha función; para ello tomamos cada término producto, identificamos las casillas que le corresponden y anotamos en ellas el valor **1**:

- al término $\overline{d}.\overline{c}.\overline{b}$ le corresponden las dos casillas en que d = 0, c = 0 y b = 0, es decir, las dos primeras casillas horizontales del mapa;
- al término $\overline{d}.\overline{b}.\overline{a}$ le corresponden las dos casillas en que d =0, b = 0 y a = 0, es decir, las dos primeras casillas verticales;
- al término $d.\overline{c}.b$ le corresponden las dos casillas en que d = 1, c = 0 y b = 1, es decir, las dos últimas casillas horizontales del mapa;
- al término $d.b.\overline{a}$ le corresponden las dos casillas en que d = 1, b = 1 y a = 0, es decir, las dos últimas casillas verticales del mapa.

Se completa el mapa de Karnaugh con «ceros» en las casillas vacías, resultando el de la figura siguiente:

d c \ b a	*0 0*	*0 1*	*1 1*	*1 0*
0 0	**1**	**1**	**0**	**0**
0 1	**1**	**0**	**0**	**0**
1 1	**0**	**0**	**0**	**1**
1 0	**0**	**0**	**1**	**1**

Los «ceros» pueden ser agrupados en 3 cuadrados, que dan lugar a una expresión algebraica en forma de producto de sumas más reducida que la expresión inicial en suma de productos:

$$y = (d + \overline{b}).(\overline{d} + b).(\overline{c} + \overline{a})$$

2.3.7. *La combinación ganadora de la «primitiva» del jueves día 23 de enero de 2003 puede expresarse mediante la función booleana siguiente; ¿cuáles son los números que forman dicha combinación ganadora escritos en base 10 (decimal)?*

$$y = (f + \bar{d} + c).(\bar{f} + \bar{d} + b).(\bar{f} + \bar{d} + \bar{c}).(\bar{c} + \bar{b} + \bar{a}).$$
$$.(e + d).(\bar{e} + b).(d + a).(b + a)$$

Recuérdese que la lotería primitiva utiliza los números del 1 al 49 y deben seleccionarse 6 de dichos números para formar una combinación; a estos efectos, el número 0 y los números superiores a 49 no interesan (valor **X**).

Se trata de obtener la tabla de dicha función para seleccionar sobre ella los vectores de entrada que hacen la función igual a **1**; la forma más rápida de obtener dicha tabla funcional consiste en rellenar el correspondiente mapa de Karnaugh, a partir de los términos suma de la función (las casillas que corresponden a estos vectores suma deben tener valor **0**, la casilla 0 y las superiores a 49 deben tener valor **X** y el resto valor **1**).

F E D \ C B A	*000*	*001*	*011*	*010*	*110*	*111*	*101*	*100*
000	**X**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
001	**0**	**0**	**0**	**0**	**1**	**0**	**1**	**0**
011	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**0**
010	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**0**
110	**0**	**0**	**X**	**X**	**X**	**X**	**X**	**X**
111	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**
101	**0**	**0**	**1**	**1**	**0**	**0**	**0**	**0**
100	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

Los vectores de entrada correspondientes a casillas con valor **1** son los siguientes:

001101 001110 010011 011110 101010 101011

números binarios que, expresados en base 10, corresponden a los siguientes números decimales:

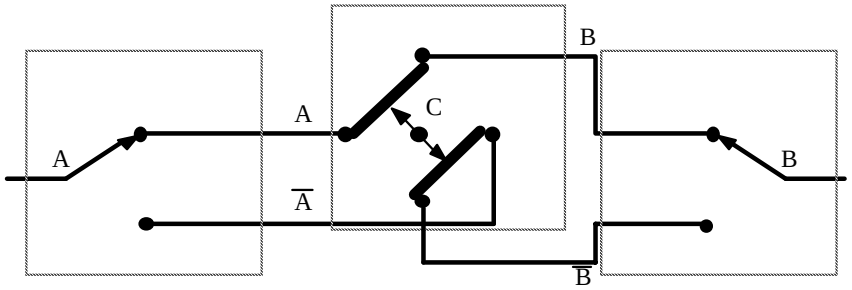
13 14 19 30 42 43

2.3.8. *Se desea controlar una lámpara mediante tres interruptores conmutados, de forma que actuando sobre cualquiera de ellos se pueda cambiar el estado de la lámpara. Determinar la función booleana necesaria*

Suponemos que con todos los interruptores en la posición **0** la lámpara está apagada, si pasa uno cualquiera de ellos a la posición **1** la lámpara se enciende, si son dos los que pasan a posición **1** la lámpara se apaga y, finalmente, con los tres interruptores en posición **1** la lámpara se encontrará encendida.

C	**B**	**A**	luz	
0	**0**	**0**	**0**	
0	**0**	**1**	**1**	$y = \bar{c}.\bar{b}.a + \bar{c}.b.\bar{a} + c.\bar{b}.\bar{a} + c.b.a$
0	**1**	**0**	**1**	
1	**0**	**0**	**1**	
0	**1**	**1**	**0**	$= \bar{c}.(\bar{b}.a + b.\bar{a}) + c.(\bar{b}.\bar{a} + b.a)$
1	**0**	**1**	**0**	
1	**1**	**0**	**0**	
1	**1**	**1**	**1**	$= c \oplus (b \oplus a)$

La función anterior puede realizarse con interruptores mecánicos en la forma que se indica en la siguiente figura:



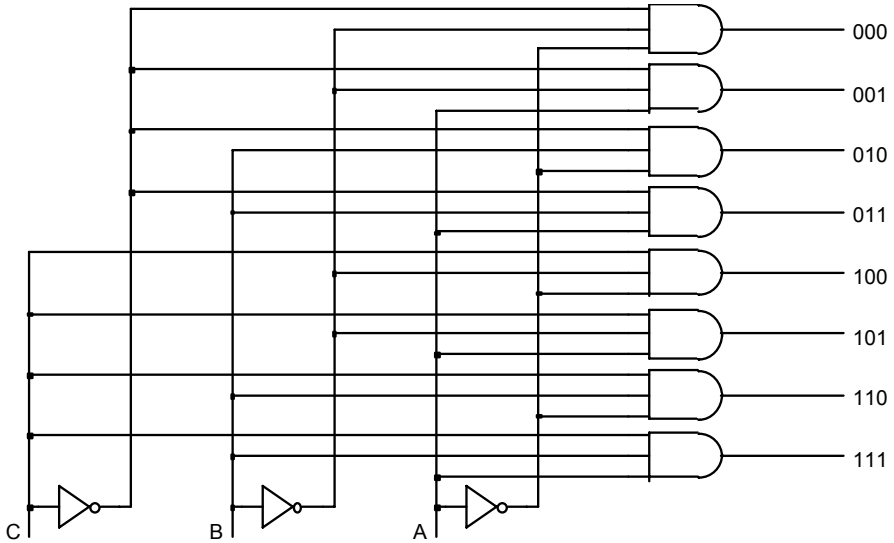
en la que el conmutador C se encuentra en su posición afirmada y negarlo significa efectuar un giro de 90º sobre su centro. Obsérvese que los interruptores de los extremos **A** y **B** son del tipo de interruptores conmutados considerados en el tema anterior (I.3.3 Álgebra de Conmutadores), mientras que el interruptor intermedio **C** realiza una conmutación del tipo «*conexión paralela - conexión cruzada*»:

- conecta A - B y $\overline{A}$ - $\overline{B}$ en una posición y A - $\overline{B}$ y $\overline{A}$ - B en la otra.

Este esquema de interruptores conmutados es ampliable a **n** interruptores, siendo los de los extremos de tipo simple **A**, **B** y los **n-2** intermedios del tipo complejo **C**.

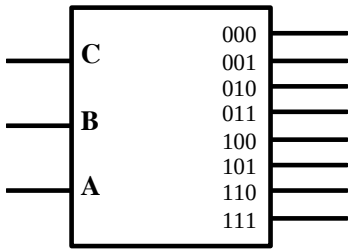
2.4. Decodificadores y multiplexores; otras formas de configurar funciones

Sean tres variables de entrada; podemos construir todos sus términos mínimos: bastarán para ello 8 puertas "**y**" de 3 entradas y 3 inversores.



Este «bloque digital», cuyas salidas corresponden a los diversos términos mínimos de sus entradas, recibe el nombre de *decodificador*; tal denominación se debe a que «decodifica» un número binario de **m** dígitos sobre $\mathbf{2^m}$ líneas, de forma que para cada número o vector de entrada activa una salida diferente.

Un decodificador es un bloque digital con **m** líneas de entrada y $\mathbf{2^m}$ líneas de salida que corresponden a los $\mathbf{2^m}$ posibles vectores de entrada (números binarios de **m** bits): la línea de salida correspondiente al número binario establecido en las entradas se encontrará a **1** y el resto de líneas de salida estará a **0**.



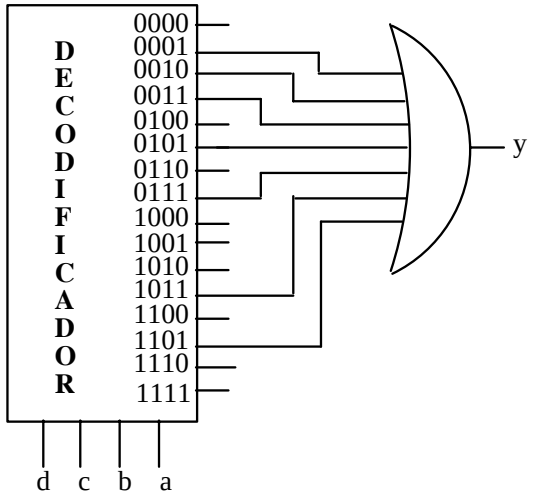
La figura anterior representa un decodificador de 3 líneas de entrada: en cada momento el número binario presente en ellas determina cuál de las 8 líneas de salida se encuentra activada (valor booleano **1**).

En un decodificador están presentes todos los términos mínimos de sus entradas; para construir una función booleana de tales entradas, según su forma canónica, bastará reunir sobre una puerta "**o**" los términos mínimos que corresponden a valor **1** en la tabla de la función, es decir, llevar a una puerta "**o**" las salidas del decodificador que corresponden a vectores de entrada que «activan» la función (dan resultado **1**).

*Por ejemplo, para construir de esta forma la función "ser número primo" de 4 dígitos utilizaremos un decodificador de 4 líneas de control seguido de una puerta "**o**" que recibe aquellas salidas con valor **1** en la «tabla de verdad» de la función: las correspondientes a los números primos 1, 2, 3, 5, 7, 11 y 13.*

Tabla funcional

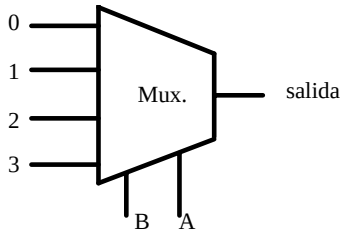
d c b a	y	
0 0 0 0	0	
0 0 0 1	1	<-
0 0 1 0	1	<-
0 0 1 1	1	<-
0 1 0 0	0	
0 1 0 1	1	<-
0 1 1 0	0	
0 1 1 1	1	<-
1 0 0 0	0	
1 0 0 1	0	
1 0 1 0	0	
1 0 1 1	1	<-
1 1 0 0	0	
1 1 0 1	1	<-
1 1 1 0	0	
1 1 1 1	0	



Así como el decodificador selecciona una de entre sus $\mathbf{2^m}$ líneas de salida, un *multiplexor* es un bloque digital que selecciona una de entre $\mathbf{2^m}$ líneas de entrada; para ello dispone de **m** líneas de control y en cada momento el número binario establecido en ellas determina la línea de entrada que «queda conectada» a la salida.

Para seleccionar cada línea de entrada por su número binario bastará realizar la operación "**y**" entre la línea de entrada y el término mínimo que corresponde a dicho número binario; una posterior operación "**o**" en la salida recogerá el resultado de dicha selección (recibirá las salidas de las citadas puertas "**y**").

La figura siguiente representa un multiplexor de 4 líneas de entrada y 2 de control; en cada momento el número binario presente en las entradas de control determina cuál de las 4 líneas de entrada se encuentra «conectada» con la salida.

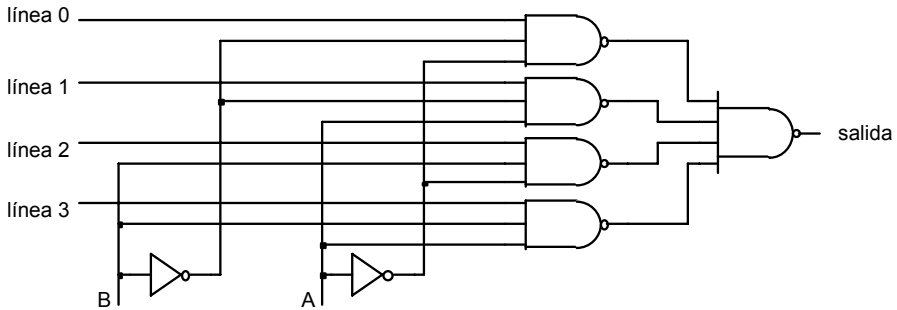


La función booleana que corresponde a este multiplexor es la siguiente:

$$y = L_0.\overline{B}.\overline{A} + L_1.\overline{B}.A + L_2.B.\overline{A} + L_3.B.A$$

que puede construirse con puertas "**y-negada**" (*Nand)* en la siguiente forma:

$$y = \text{Nand}\left[\text{Nand}\left(L_0, \overline{B}, \overline{A}\right), \text{Nand}\left(L_1, \overline{B}, A\right), \text{Nand}\left(L_2, B, \overline{A}\right), \text{Nand}\left(L_3, B, A\right)\right].$$



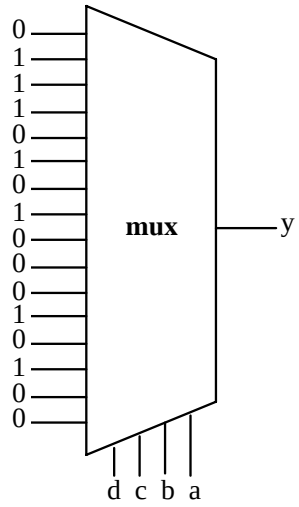
Para construir una función booleana utilizando un multiplexor bastará con fijar en sus líneas de entrada los valores de la «tabla de verdad» de la misma; de esta forma, las variables de control seleccionan sobre la propia tabla de la función el valor booleano que le corresponde al vector de entrada.

Esta forma de configurar funciones booleanas mediante multiplexores, cuyas líneas de entrada reciben los valores de la «tabla de verdad» de la función, se denomina **look-up-table** (LUT): construcción tabular (*mirar sobre su tabla*).

*En el ejemplo de la función "ser número primo" de 4 dígitos, para construirla con un multiplexor de 4 líneas de control, conectaremos a **1** las líneas de entrada del multiplexor que presentan tal valor en la «tabla de verdad» de la función (las correspondientes a los números primos 1, 2, 3, 5, 7, 11 y 13); el resto de las líneas de entrada se conectarán a **0**.*

Tabla funcional

d c b a	y	
0 0 0 0	0	
0 0 0 1	1	<-
0 0 1 0	1	<-
0 0 1 1	1	<-
0 1 0 0	0	
0 1 0 1	1	<-
0 1 1 0	0	
0 1 1 1	1	<-
1 0 0 0	0	
1 0 0 1	0	
1 0 1 0	0	
1 0 1 1	1	<-
1 1 0 0	0	
1 1 0 1	1	<-
1 1 1 0	0	
1 1 1 1	0	



Decodificadores y multiplexores son selectores de línea:

- un decodificador activa una de entre **n** líneas de salida
- y un multiplexor selecciona una de entre **n** líneas de entrada y la pone en comunicación con su línea de salida.

Cada uno de ellos proporciona una forma de configurar una función de sus variables de control:

- la forma canónica a partir de un decodificador (suma de términos mínimos $\sum \mathbf{m}$)
- la forma tabular sobre un multiplexor (*look-up-table* **LUT**).

Ambas formas de construir una función booleana no precisan de la expresión algebraica de la misma; se obtienen directamente de su tabla funcional.