Tema 38. Principios de electrónica digital. Álgebra de Boole. Puertas lógicas. Funciones básicas combinacionales: decodificadores, codificadores, multiplexores y otras. Simbología, tipología, función y aplicación en los sistemas de control automático.

38.1. Principios de electrónica digital

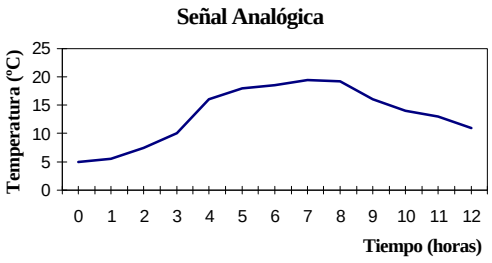
38.1.1. Sistemas digitales y analógicos

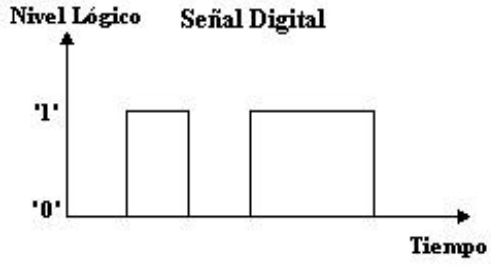
La rápida evolución experimentada por la tecnología electrónica permite diseñar y construir sistemas para procesado y tratamiento de información de bajo coste, reducido volumen, gran capacidad de almacenamiento y unas altas prestaciones. Esto hace que los sistemas electrónicos aparezcan cada vez más y en un mayor número de áreas; desde productos de uso doméstico hasta en complejos procesos de producción industrial.

Los sistemas electrónicos pueden clasificarse en dos grandes grupos: analógicos y digitales, en función de las señales que manipulan; es decir, de los valores que pueden tomar las diferentes variables que intervienen en el sistema.

Los **sistemas analógicos** van a trabajar con señales analógicas; es decir, señales cuya magnitud toma valores continuos. Ejemplos de señales analógicas son: temperatura, altura, sonido, … En los sistemas analógicos, los dispositivos electrónicos que los constituyen trabajan en zona lineal.

Los **sistemas digitales** son sistemas para procesamiento, tratamiento o transmisión de la información, en el que dicha información está limitada a tomar valores en un conjunto discreto. Estas señales, cuya magnitud sólo puede tomar un valor de entre un conjunto discreto de valores son las señales digitales. A diferencia de los sistemas analógicos, en los digitales los dispositivos que los constituyen van a funcionar como interruptores.





38.1.2. Tratamiento digital de la información

Muchos sistemas analógicos están siendo sustituidos por sistemas digitales que realizan funciones similares debido a sus ventajas inherentes:

- Mayor fiabilidad, propia de los circuitos integrados.
- Mayor facilidad de diseño.
- Flexibilidad, debido al carácter programable de muchos circuitos digitales.

- Procesado y transmisión de datos de una forma más eficiente y fiable.
- Facilidad de almacenamiento.
- Menor coste en general.

En los sistemas electrónicos digitales hay dos posibles valores de magnitud; es decir, se trabaja con señales binarias. Estos estados se representan con los dígitos '0' y '1', y van a estar asociados a unos niveles de tensión. En la figura anterior el '1' está asociado al valor más alto de tensión (V_H) y el '0' al nivel bajo (V_L), se habla de lógica positiva. Si se asocia el '0' al valor más elevado de tensión, y el '1' al más bajo, se habla de lógica negativa.

Los sistemas digitales se clasifican en dos grandes grupos:

- Combinacionales: las salidas en cualquier instante de tiempo dependen del valor de las entradas en ese mismo instante de tiempo (salvo los retardos propios de los dispositivos electrónicos). Son, por tanto, sistemas sin memoria.
- Secuenciales: la salida del sistema va a depender del valor de las entradas en ese instante de tiempo y del estado del sistema; es decir, de la historia pasada del sistema. Son sistemas con memoria.

38.2. Álgebra de Boole

38.2.1. Definición

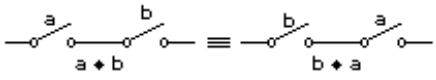
George Boole formuló en el siglo XIX un álgebra de aplicación en la teoría de conjuntos. Lo realmente interesante del álgebra de Boole es que permite expresar y analizar de una forma muy eficiente las operaciones de los circuitos lógicos.

Un álgebra de Boole es toda clase o conjunto B de elementos que pueden tomar dos valores perfectamente diferenciados (representados por 0 y 1) y que están relacionados por dos operaciones binarias, denominadas suma lógica + y producto lógico · que cumplen los siguientes postulados:

(a) Ambas operaciones son conmutativas:

$a + b = b + a$

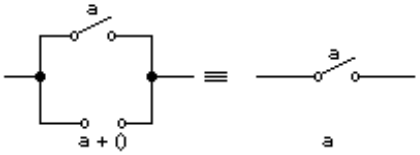
$a \cdot b = b \cdot a \qquad \forall\, a, b \in B$



(b) Existen elementos neutros dentro de B para ambas operaciones:

$a + 0 = 0 + a = a \qquad \forall\, a \in B$

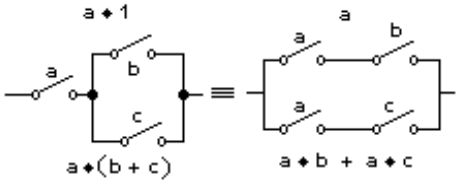
$a \cdot 1 = 1 \cdot a = a \qquad \forall\, a \in B$



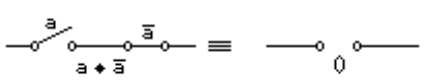
(c) Las operaciones son distributivas cada una respecto a la otra:

$a + (b \cdot c) = (a + b) \cdot (a + c)$

$a \cdot (b + c) = (a \cdot b) + (a \cdot c) \quad \forall\, a, b, c \in B$



(d) Cada elemento de B tiene su complemento:



$a + a' = 1$
$a \cdot a' = 0 \qquad \forall\, a \in B$

Se puede recurrir a los diagramas de contactos (fueron los primeros circuitos digitales utilizados) para comprender mejor estas leyes. La operación suma se asimila a la conexión en paralelo de contactos y la operación producto se asimila a la conexión serie. El inverso de un contacto es otro cuyo estado es siempre el opuesto del primero (está cerrado cuando aquel está abierto, y viceversa). El elemento 0 es un contacto siempre abierto y el 1 un contacto siempre cerrado.

38.2.2. Teoremas

Basándose en los postulados anteriores se deducen los teoremas que se exponen a continuación:

Teorema 1: Ley de *Idempotencia*

$a = a + a$
$a = a \cdot a$

Teorema 2:

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

Teorema 3: Ley de absorción

$a + ab = a$
$a\,(a + b) = a$

Teorema 4 : En un álgebra de Boole las operaciones + y · son asociativas:

$a + (b + c) = (a + b) + c = a + b + c$
$a \cdot (b \cdot c) = (a \cdot b) \cdot c = a \cdot b \cdot c$

Teorema 5: se verifica que para todo elemento a de un álgebra de Boole:

$a'' = a$

Teorema 6: Leyes de De Morgan

$(a + b)' = a' \cdot b'$
$(a \cdot b)' = a' + b'$

En general:

$(b_1 + b_2 + \ldots b_n)' = b_1' \cdot b_2' \cdot \ldots \cdot b_n'$
$(b_1 \cdot b_2 \cdot \ldots \cdot b_n)' = b_1' + b_2' + \ldots b_n'$

Teorema 7: Principio de dualidad: si en una expresión válida intercambiamos las operaciones (+, ·) y los elementos neutros (0, 1), la nueva expresión así obtenida también sigue siendo válida.

38.3. Puertas lógicas

38.3.1. Funciones lógicas

Una función lógica es una variable binaria F, cuyo valor es igual al de una expresión algebraica de variables lógicas (complementadas o no) unidas por las operaciones lógicas (+,·). Ejemplos de funciones lógicas serían:

$F_1 = F_1(a,b,c) = ab + c$
$F_2 = F_2(a,b,c,d) = a' + bc + ad'$

Donde $F_1(a,b,c)$ indica que la función depende de las variables a, b y c. Las funciones lógicas se clasifican en:

a) Completamente especificadas: si a cada una de las posibles combinaciones de las variables de entrada corresponde un valor único y definido de la función.

b) Incompletas: si a una o más combinaciones de entrada se le puede asignar el valor 0 ó 1 indistintamente (por ejemplo: si nunca van a aparecer determinadas combinaciones de entrada o si aparecen me da igual el valor que tome la función). Se habla de indiferencias.

38.3.2. Representación de funciones digitales

Además de las expresiones algebraicas vistas en el apartado anterior las funciones digitales se van a representar de otras formas:

a) Tabla de verdad: se indica el valor 0 ó 1 que toma la función para cada una de las combinaciones posibles de las variables de la función.
La tabla del ejemplo se correspondería a F(a,b) = a+b´. En el caso de una función incompleta, las salidas no definidas se representan mediante un guión o una x en la tabla de la verdad.

a	b	F(a,b)
0	0	1
0	1	0
1	0	1
1	1	1

b) Forma canónica: representación basada en el teorema de Shannon que definiremos a continuación:

Minterm: dadas n variables, un *mimterm* es un término producto en el que aparecen todas las variables solo una vez, complementadas o sin complementar.
Maxterm: dadas n variables, un *maxterm* es un término suma en el que aparecen todas las variables sólo una vez, complementadas o sin complementar.

Teorema de expansión de Shannon: Toda función digital de n variables se puede expresar como suma de *mimterms* o producto de *maxterms*.

La forma canónica de una función digital sería su expresión como suma de *mimterms* o como producto de *maxterms*. La siguiente tabla nos indica como obtenerla:

Tipo de ecuación	Método de obtención	Convenio a aplicar
Ecuación *minterms*	Obtener suma de *mimterms* que hacen 1 la función	0 Variable negada 1 Variable sin negar
Ecuación *maxterms*	Obtener producto de *maxterms* que hacen 0 la función	0 Variable sin negar 1 Variable negada

Filas	x	y	z	f
0	0	0	0	0
1	0	0	1	0
2	0	1	0	1
3	0	1	1	1
4	1	0	0	1
5	1	0	1	0
6	1	1	0	1
7	1	1	1	1

Lo ejemplificaremos con la siguiente función dada por la siguiente tabla:

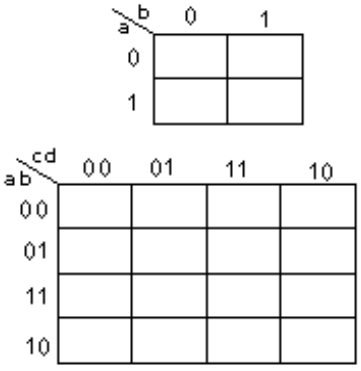
- Suma de *mimterms*: f(x,y,z) = x´yz´ + x´yz + xy´z´ + xyz´ + xyz
  Que en forma abreviada sería: f(x,y,z) = $\sum$m(2,3,4,6,7)
- Producto de *maxterms*: f(x,y,z) = (x+y+z)·(x+y+z´)·(x´+y+z´)
  Que en forma abreviada sería: f(x,y,z) = $\prod$M (0,1,5)

Filas	x	y	z	f
0	0	0	0	1
1	0	0	1	0
2	0	1	0	1
3	0	1	1	0
4	1	0	0	X
5	1	0	1	0
6	1	1	0	X
7	1	1	1	1

Si hay indiferencias, el proceso anterior se repite, como se puede ver en el siguiente ejemplo:

- Expresiones válidas son:
  f = x´y´z´ + x´yz´ + xyz
  f = x´y´z´+ x´yz´ + xyz + xy´z´ + xyz´ considerando las indiferencias.
- De forma abreviada: f = $\sum$m(0,2,7) + $\sum$d(4,6) ó f = $\prod$M (1,3,5) + $\prod$d (4,6)

c) Mapa de Karnaugh: Esta forma de representación va a permitir al diseñador lógico simplificar las funciones, logrando que el circuito digital resultante sea más sencillo y económico a la hora de implementarse. El mapa de Karnaugh para una función de n variables está constituido por un rectángulo dividido en 2^n celdas, que representan cada una de las posibles combinaciones de las variables de la función. Se coloca un 1 en aquellas celdas tales que, para la combinación de las variables correspondiente, la función tome el valor 1 y un guión en las indiferencias; en las restantes no se coloca nada. En la siguiente figura, se muestran a modo de ejemplo los mapas de Karnaugh para dos funciones; una de dos variables y otra de cuatro.



38.3.3. Puertas lógicas: tipología y funciones

En este apartado se van a definir las funciones lógicas básicas, que son las que se utilizan fundamentalmente en la realización de sistemas digitales. Estas funciones existen implementadas en electrónica digital, denominándose puertas lógicas a los circuitos que las realizan.

En la siguiente figura aparecen las funciones básicas, su nombre, su tabla de verdad y su simbología. Se incluye la nueva simbología (símbolos rectangulares, norma IEEE Std. 91-1984) y la simbología antigua (IEEE Std. 91-1973), debido a que ambas son ampliamente usadas y coexisten en la actualidad.

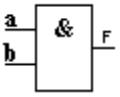
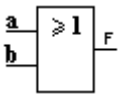
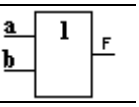
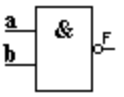
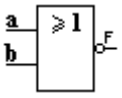
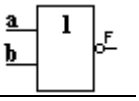
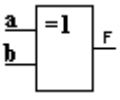
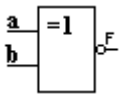
Se han representado las funciones AND, OR, NAND, NOR, EXOR y NEXOR de dos variables; ya que, para más variables lo único que cambia es que las puertas tendrán más entradas y en las tablas de verdad aparecen más variables; pero su

comportamiento será como el indicado.

Hay que señalar que aunque las funciones EXOR y NEXOR se pueden realizar asociando otras puertas (AND, OR y NOT por ejemplo), el perfeccionamiento de las tecnologías de fabricación ha permitido fabricar circuitos integrados que realizan estas funciones.

Este último apunte sirve para definir lo que se conoce como conjunto suficiente de conectores o funcionalmente completo: viene a ser un conjunto de conectores (puertas) unitarios y/o binarios tal que cualquier otra función de n variables se puede expresar relacionando las n variables a través de este conjunto de conectores. Por ejemplo, con el conjunto {OR,AND,NOT} se va a poder implementar cualquier función digital; por tanto, sería un conjunto suficiente de conectores. Otros conjuntos son {NAND} y {NOR}, lo que nos quiere decir que cualquier circuito digital se puede realizar usando solo puertas NAND, o solo puertas NOR.

- La salida de una puerta AND es 1 solo si todas las variables valen 1 a la vez.
- La salida de una puerta OR es 0 cuando todas las variables valen 0 a la vez.
- En un *buffer* la salida sigue a la entrada.
- La salida de una puerta NAND es la negada de la función AND.
- La salida de una puerta NOR es la negada de la función OR.
- En un inversor, obtenemos a la salida la negación de la entrada.
- La puerta EXOR tendrá un 1 en la salida cuando una de sus entradas sea 1 y la otra 0.
- La salida de una puerta NEXOR es la negada de la función EXOR.

TablaVerdad	Función	Nombre	Símbolo	Símbolo
a b | F(a,b) 0 0 | 0 0 1 | 0 1 0 | 0 1 1 | 1	F(a,b) = a·b	AND		
a b | F(a,b) 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 1	F(a,b) = a+b	OR		
a | F(a) 0 | 0 1 | 1	F(a) = a	*BUFFER*		
a b | F(a,b) 0 0 | 1 0 1 | 1 1 0 | 1 1 1 | 0	F(a,b) = $\overline{a \cdot b}$	NAND		
a b | F(a,b) 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 0	F(a,b) = $\overline{a+b}$	NOR		
a | F(a) 0 | 1 1 | 0	F(a) = $\overline{a}$	NOT		
a b | F(a,b) 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 0	F(a,b)=$\overline{a}$·b+a·$\overline{b}$ ó F(a,b) = a ⊕ b	EXOR		
a b | F(a,b) 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 1	F(a,b)=$\overline{a}$·$\overline{b}$+a·b ó F(a,b) = a ? b	NEXOR		

38.4. Funciones básicas combinacionales

38.4.1. Clasificación, tipología, función y aplicación

Las funciones combinacionales básicas están implementadas mediante circuitos MSI (*Medium Scale Integration*, que contienen de 10 a 100 puertas lógicas o de 100 a 1000 transistores); y se clasifican, según la función que desempeñan en el interior de los sistemas digitales, en los siguientes grupos:

a. Circuitos de comunicación

Sirven tanto para transmitir información por una línea como para codificar, decodificar o modificar la estructura de dicha información. Los más importantes son:

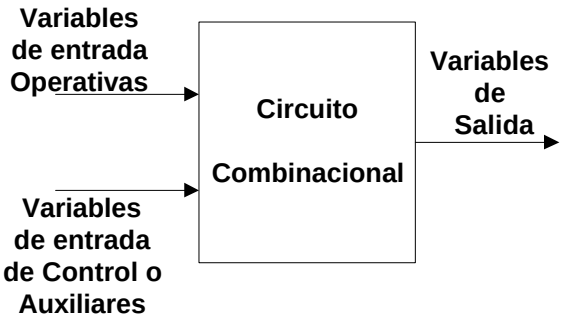
- Multiplexores y demultiplexores.
- Codificadores:
  - Codificadores sin prioridad.
  - Codificadores con prioridad.
- Decodificadores:
  - Decodificadores no excitadores.
  - Decodificadores excitadores: en ánodo común, cátodo común.
- Convertidores de código.

b. Circuitos aritmético-lógicos

Son circuitos que realizan una serie de operaciones aritméticas o lógicas con los datos binarios que procesan. Los principales son:

- Sumadores.
- Generadores de acarreo.
- Restadores.
- Comparadores.

Los bloques funcionales combinacionales responden en general, al diagrama de bloques de la figura, en el que las variables de entrada se dividen en dos grupos:



a. **Variables de entrada operativas**: con las que el circuito realiza una determinada función u operación.
b. **Variables de entrada de control o auxiliares**: influyen en la forma en que el circuito actúa sobre las operativas o modifican el resultado (variables de salida). Las principales variables de control de los bloques combinacionales son:

- Entrada de desinhibición/inhibición (*Enable/Disable*): van a inhibir o desinhibir la acción del circuito sobre las demás entradas y ponen las salidas en un determinado nivel lógico.
- Entradas de control tercer estado (*TriState*): controlan cuando la salida se pone

en tercer estado o estado de alta impedancia.
- Entradas de inversión de variables operativas: hacen que el circuito actúe directamente sobre las variables operativas o sobre sus inversas.
- Entradas de inversión de variables de salida: hacen que a la salida aparezcan las variables en forma directa o inversa.
- Entradas de selección de operación: seleccionan la operación que realiza el circuito.

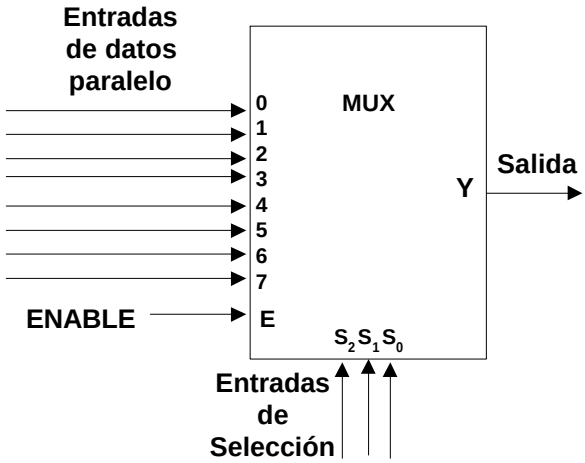
38.5. Multiplexores o selectores de datos

Por definición, el **multiplexor** es un circuito combinacional diseñado para seleccionar la información binaria de una línea entrada de entre varias líneas de entrada y dirigirla a una sola línea de salida. Las entradas y salidas que poseen estos circuitos son las siguientes:

- N entradas de información o canales, denotadas por $I_0,..I_{N-1}$.
- n entradas de selección o control, $S_0,...,S_{n-1}$.
- Una entrada de autorización /habilitación/desinhibición o *enable*, E.
- Una salida de información, Y (en muchos MUX comerciales también aparece Y').

Los canales de entrada (N) con las entradas de selección (n) están relacionados por la ecuación: $N = 2^n$.

En la figura se presenta el esquema de un MUX 8 a 1, en el que se puede observar la relación $N=2^n$, donde N=8 canales y n=3 variables de control.



La función lógica que realiza el multiplexor viene dada por la siguiente ecuación, que a modo de ejemplo se particulariza para un MUX 4 a 1 (4 entradas, 1 salida):

$$Y(I_0, I_1, ... I_{N-1}, S_0, ..., S_{n-1}, E) = E \cdot (I_{N-1} S_{n-1} \cdot ... \cdot S_0 + ... + I_0 \overline{S}_{n-1} \cdot ... \cdot \overline{S}_0)$$

$$Y(I_0, I_1, I_2, I_3, S_0, S_1, E) = E \cdot (I_3 S_1 S_0 + I_2 S_1 \overline{S}_0 + I_1 \overline{S}_1 S_0 + I_0 \overline{S}_1 \overline{S}_0)$$

Como bloque de diseño, los MUX son circuitos muy versátiles ya que aparte de la función de selector de datos, su estructura le permite el ser utilizado como módulo lógico universal en la implementación de una función digital combinacional cualquiera de n variables. Por tanto, tienen **aplicación** desde dos perspectivas diferentes:

- Como **selectores de datos**, permitiendo seleccionar una de entre las N entradas. En este tipo de aplicaciones pueden servir para pasar información en formato paralelo a formato serie, sin más que seleccionar secuencialmente cada una de las líneas paralelo (entradas del multiplexor) con los valores de

selección adecuados.

- Como **módulos de diseño** para implementar cualquier función lógica.

38.5.1. Implementación de una función lógica empleando multiplexores

a) Empleo de MUX de igual número de entradas de selección que variables de la función

La expresión lógica de la salida de un multiplexor de N $=2^n$ entradas, se puede interpretar como:

$$Y = m_0 I_0 + m_1 I_1 + ... + m_{N-1} I_{N-1}\ ,$$

en la que los m_j son los 2^n *mimterms* que pueden construirse con las n variables binarias de selección de entrada. Como tenemos una entrada separada asociada a cada uno de ellos, el hecho de conectar cada una de ellas a un nivel lógico dado nos dará la posibilidad de realizar cualquier tabla de verdad de n variables binarias. Dada una función cualquiera, expresada en su forma canónica como suma de *mimterms*, bastará con fijar a 1 las entradas seleccionadas por *mimterms* pertenecientes a la expresión de la función y a 0 las entradas seleccionadas por *mimterms* no pertenecientes a la función.

b) Empleo de multiplexores con un número de entradas de selección inferior en una unidad al de variables de la función a implementar

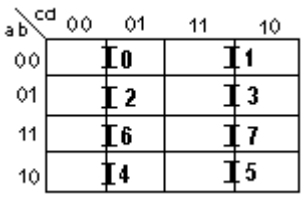
Es posible implementar funciones lógicas de n variables con multiplexores de n-1 entradas de selección empleando solo un inversor adicional, con el consiguiente ahorro económico que ello supone.

Se analiza el método de diseño con un ejemplo: supongamos que se trata de implementar la función de 4 variables definida como:

$$F(a,b,c,d) = \Sigma m(0,3,4,5,9,10,12,13)$$

utilizando un multiplexor de 3 entradas de control.

El punto de partida es el mapa de Karnaugh de la función, en donde hay que tener en cuenta que cada entrada del multiplexor está asociada a una determinada región del mapa (ver figura). El contenido de las casillas asociadas a cada entrada del MUX, va a definir el valor de dicha entrada.

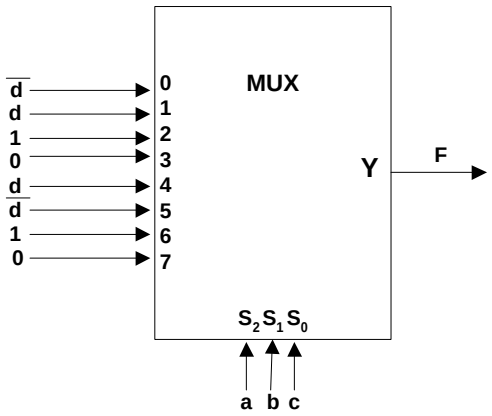


ab \ cd	00	01	11	10
00	1	0	1	0
01	1	1	0	0
11	1	1	0	0
10	0	1	0	1

Para determinar el valor de las entradas se recurre a la tabla de los residuos; en donde la columna etiquetada con "otras variables" (residuos) presenta los valores de la cuarta

variable, obtenidos a partir del mapa de Karnaugh, en cada combinación de entradas existente en la función. Las entradas al multiplexor pueden leerse directamente de esta tabla:

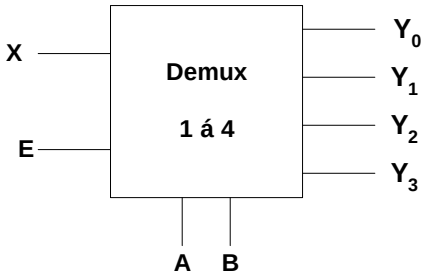
Entrada	Dirección	Otras Variables (residuos)
I_0	a'b'c'	d'
I_1	a'b'c	d
I_2	a'bc'	d+d' = 1
I_3	a'bc	0
I_4	ab'c'	d
I_5	ab'c	d'
I_6	abc'	d+d' = 1
I_7	abc	0



38.6. Demultiplexores

Un demultiplexor de 1 a n es un sistema con una entrada de datos y n entradas de control que permite llevar la entrada a una de las 2^n salidas.

La figura muestra el diagrama lógico de un demultiplexor 1 a 4, con una señal adicional de hahilitación o *enable* (E), activa en alta.

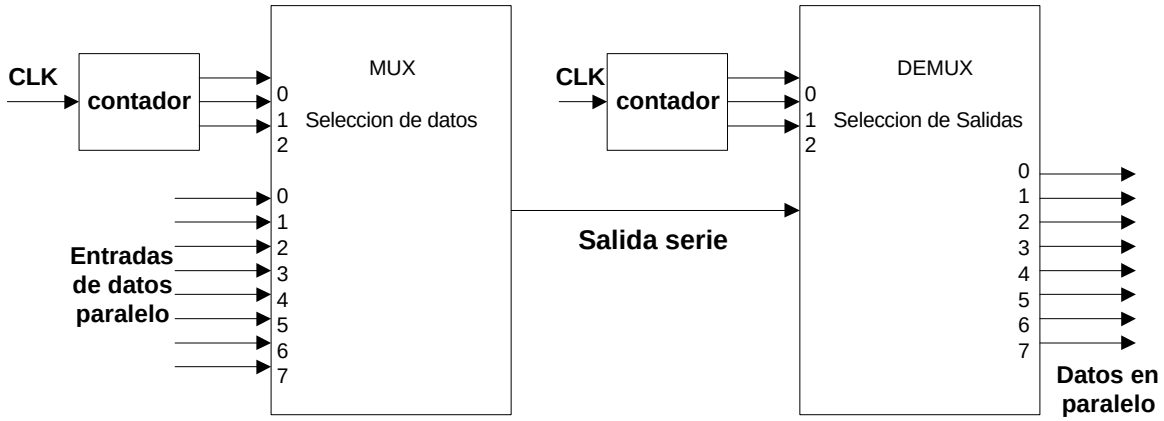


Las funciones lógicas asociadas a las salidas pueden expresarse de la siguiente manera:

$$Y_0 = E(\overline{A}\cdot\overline{B}\cdot X) \quad Y_2 = E(A\cdot\overline{B}\cdot X)$$

$$Y_1 = E(\overline{A}\cdot B\cdot X) \quad Y_3 = E(A\cdot B\cdot X)$$

Una **aplicación** de los demultiplexores es el pasar información de serie a paralelo (lo contrario que un multiplexor) a la salida de un canal de transmisión serie, direccionando secuencialmente las lineas de control. Hay que indicar que los DEMUX no se encuentran comercializados, y se diseñan a partir de los decodificadores, de ahí que en los catálogos de los fabricantes se nombren por decodificadores/demultiplexores.



38.7. Decodificadores

El decodificador es un circuito combinacional provisto de n entradas y un número de salidas menor o igual a 2^n, que activa una sola de sus salidas al aparecer una combinación binaria en sus entradas. Se encuentran comercializados decodificadores que funcionan en lógica negativa (la salida activa está a '0' y el resto a '1') y en lógica positiva (salida activa a '1' y el resto de las salidas a '0').

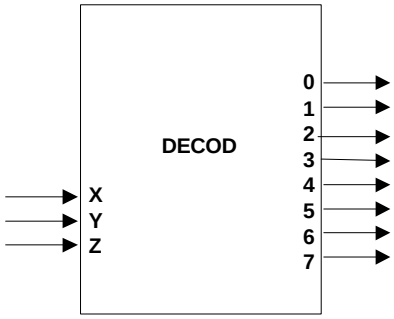
Según sus salidas puedan o no controlar a un indicador numérico, se clasifican:

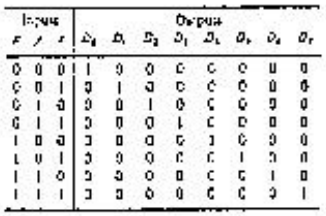
a. Decodificadores no excitadores
b. Decodificadores excitadores (*drivers*)

Los decodificadores tienen las siguientes **aplicaciones**:

- Como **módulos de diseño** permiten implementar funciones digitales.
- Realizar la función de **demultiplexión** debido a que los demultiplexores no se encuentran comercializados.
- Los decodificadores excitadores (***drivers***) permiten extraer la información binaria de un circuito y representarla en sistemas visualizadores (siete segmentos, visualizadores de cristal líquido LCD).

En la figura aparece el diagrama lógico de un decodificador de 3 a 8, con su tabla de verdad. Las tres entradas se decodifican en las 8 salidas, cada una de las cuales representa uno de los *minterms* de tres variables.





38.7.1. Implementación de funciones lógicas con decodificadores

Del mismo modo que en el caso de los multiplexores, los decodificadores pueden utilizarse en la implementación de cualquier función lógica combinacional con un mínimo de elementos adicionales.

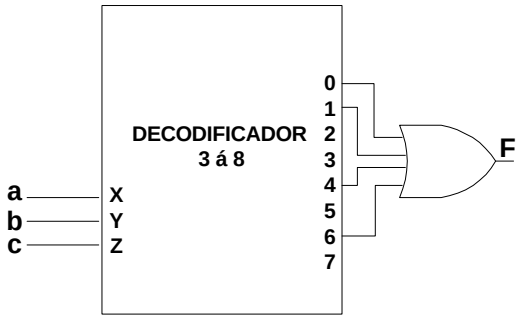
Dado que cualquier decodificador proporciona a sus salidas los 2^n *mimterms* de las n variables de entrada, al poder expresarse cualquier función booleana en su forma canónica como suma de productos, la implementación de un circuito combinacional general con n entradas y m salidas podrá realizarse con un codificador n a 2^n y un conjunto de m puertas OR (lógica positiva) o NAND (lógica negativa).

Al contrario que el MUX, el decodificador requiere colocar al menos una puerta OR que recoja los *mimterms* de la función para decodificadores con salidas activas en nivel alto - la función debe ser activa siempre que se haga 1 uno o varios de los *mimterms*- o NAND, para decodificadores con salidas activas a nivel bajo, ya que, al encontrarse negado cada término activo de la función por el decodificador, la salida se deberá activar sólo cuando uno o varios términos valgan 0.

A modo de ejemplo, en la figura se puede ver la implementación de la función:

$$F(a,b,c) = m(0,1,4,6)$$

mediante un decodificador 3 a 8 con salidas activas en alta. Mediante una puerta OR, se recogen los *minterms* de la función. En caso de tener salidas activas en baja, esta puerta tendría que sustituirse por una puerta NAND.



38.8. Codificadores

Un codificador (*encoder)*, es un circuito con N entradas y n salidas (con $N = 2^n$) de forma que, cuando una determinada entrada está activa, en la salida aparecerá un valor binario representativo del número asociado a dicha entrada.

La función habitual de un codificador es la de convertir cualquier información digitalizada que entra al sistema digital en su equivalente en binario natural o en cualquiera de los códigos binarios existentes. Hay dos tipos de codificadores:

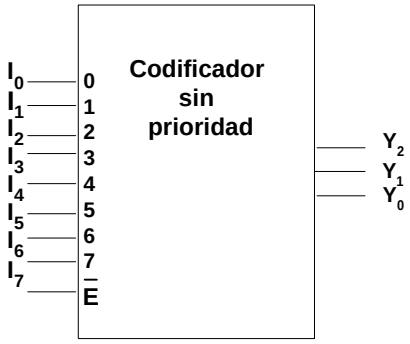
1) Codificadores sin prioridad

Son circuitos en los que no pueden activarse simultáneamente más de una entrada porque, si se activan varias, aparecen códigos binarios erróneos en la salida (se obtiene el OR lógico de las salidas que se obtendrían para cada una de las entradas activadas de forma independiente)

Otro problema asociado a este tipo de codificadores es la ambigüedad asociada al código de salida 0. Puede ser provocado porque el codificador está inhibido, o estando habilitado la entrada activa es la I_0 o no hay ninguna entrada activada.

La figura muestra el símbolo lógico de un codificador de ocho a tres y su tabla

asociada. Este codificador incluye una entrada de *enable* o habilitación, que opera en lógica negativa. La tabla mostrada es la tabla de verdad abreviada, ya que al tener 8 entradas su tabla de verdad tendría 256 filas; de las cuales, solo 8 tienen significado (una única entrada activada), correspondiendo el resto a indiferencias.



Entradas		Salidas		
E'	I_i	Y_2	Y_1	Y_0
H	X	L	L	L
L	ninguna	L	L	L
L	I_0	L	L	L
L	I_1	L	L	H
L	I_2	L	H	L
L	I_3	L	H	H
L	I_4	H	L	L
L	I_5	H	L	H
L	I_6	H	H	L
L	I_7	H	H	H
L	varias	OR lógico		

2) Codificadores con prioridad

Los codificadores con prioridad tienen un funcionamiento similar a los codificadores sin prioridad; pero, en el caso de producirse una activación simultánea de varias entradas del codificador, en la salida aparecerá el código de la entrada de **mayor prioridad**, lo cual exigirá fijar previamente un valor de prioridad. De este modo, en cada instante solo se genera el código de la entrada activa que posea la máxima prioridad.

La siguiente tabla muestra el funcionamiento de un codificador con prioridad, similar a los comercializados actualmente. En caso de activarse más de una entrada, la entrada prioritaria es la de mayor valor decimal. También posee dos salidas adicionales, que van a permitir eliminar la ambigüedad asociada al código de salida 0:

- GS (*Group Signal Output*): se activa a nivel bajo cuando el codificador está habilitado y hay alguna entrada activada.
- EO (*Enable Output*): se activa a nivel bajo cuando el codificador está habilitado y no hay ninguna entrada activada. Se emplea también como salida de expansión, para obtener codificador de mayor capacidad.

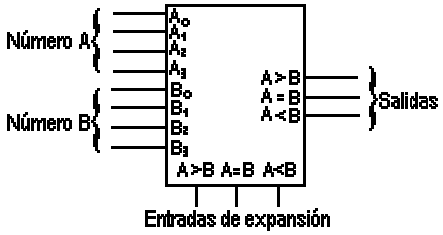
Entradas									Salidas				
E_i'	I_7	I_6	I_5	I_4	I_3	I_2	I_1	I_0	Y_2	Y_1	Y_0	GS'	EO'
H	X	X	X	X	X	X	X	X	L	L	L	H	H
L	L	L	L	L	L	L	L	H	L	L	L	L	H
L	L	L	L	L	L	L	H	X	L	L	H	L	H
L	L	L	L	L	L	H	X	X	L	H	L	L	H
L	L	L	L	L	H	X	X	X	L	H	H	L	H
L	L	L	L	H	X	X	X	X	H	L	L	L	H
L	L	L	H	X	X	X	X	X	H	L	H	L	H
L	L	H	X	X	X	X	X	X	H	H	L	L	H
L	H	X	X	X	X	X	X	X	H	H	H	L	H
L	L	L	L	L	L	L	L	L	L	L	L	H	L

20.7. Comparadores binarios

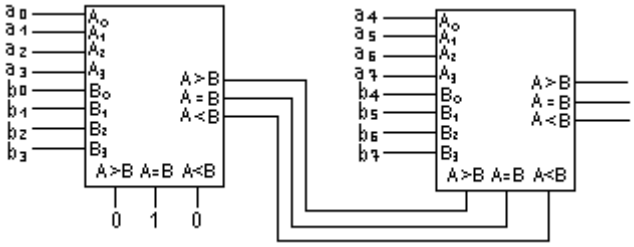
Los comparadores son circuitos combinacionales que indican la relación de igualdad o desigualdad existente entre dos números binarios A y B de n bits cada uno. Además, suelen disponer de una serie de entradas de acoplamiento para poder

comparar palabras de mayor número de bits que los permitidos por el comparador que usamos.

En la figura se muestra el diagrama esquemático de un comparador tipo 7485. En la tabla se puede ver un resumen de su funcionamiento y un esquema de cómo se deben conectar las entradas de ampliación para expandir los comparadores:



AB	<	=	>	A<B	A=B	A>B
A< B	X	X	X	1	0	0
A> B	X	X	X	0	0	1
A= B	1	0	0	1	0	0
A= B	0	1	0	0	1	0
A= B	0	0	1	0	0	1



Como se observa el comparador tiene tres entradas de expansión, que permiten realizar la comparación de combinaciones binarias de un número cualquiera de bits. Conectando las salidas A<B, A=B, A>B del comparador al que se llevan los bits de menor peso, a las entradas de ampliación del comparador al que se conectan los siguientes bits; y así sucesivamente se amplía la capacidad de comparación. Las salidas del comparador asociado a los bits más significativos nos darán el resultado de la comparación.