2. CODIFICADORES Y DECODIFICADORES

Los sistemas digitales contienen datos o información que está en alguna forma de código binario, los cuales se operan de alguna manera. En este capítulo se examinan circuitos combinatorios cuyas aplicaciones incluyen:

1. Cambio de datos de una forma a otra.

2. Tomar datos y enrutarlos a uno de varios destinos.

3. Decodificación de datos para despliegues visuales.

Muchos de los circuitos lógicos que cumplen estas funciones están ahora como circuitos integrados en la categoría de Mediana Escala de Integración (MSI - Medium Scale Integration). Por esta razón, no nos concentraremos en el diseño de estos circuitos, sino que investigaremos cómo se usan solos o en combinación, para cumplir varias operaciones sobre datos digitales. Algunas de las operaciones que se discuten son *decodificación*, *codificación*, *conversión de códigos*, *multiplexado* y *demultiplexado*.

2.1 Decodificadores

Un ***decodificador*** es un circuito lógico combinacional, que convierte un *código de entrada binario de **N** bits en **M** líneas de salida* (*N* puede ser cualquier entero y *M* es un entero menor o igual a 2^N), tales que cada línea de salida será activada para una sola de las combinaciones posibles de entrada. La **Figura 1**, muestra el diagrama general de un *decodificador* con *N* entradas y *M* salidas. Puesto que cada una de las entradas puede ser **0** ó **1**, hay 2^N posibles combinaciones o códigos de entrada. Para cada una de estas combinaciones de entrada sólo una de las *M* salidas estará activada **1**, para *lógica positiva*; todas las otras salidas estarán en **0**. Muchos decodificadores se diseñan para producir salidas **0** activas, *lógica negativa*, donde la salida seleccionada es **0** mientras que las otras son **1**. Esto último, se indica siempre por la presencia de pequeños círculos en las líneas de salida del diagrama del decodificador.

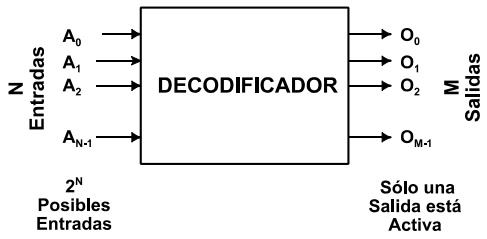


FIGURA 1. DIAGRAMA GENERAL DE UN *DECODIFICADOR*

Algunos decodificadores no usan todos los $\mathbf{2^N}$ códigos posibles de entrada, sino sólo algunos de ellos. Por ejemplo, un decodificador **BCD** a **DECIMAL**, tiene un código de entrada de 4 bits, el cual usa sólo diez grupos codificados **BCD**, *0000* hasta *1001*. Algunos de estos decodificadores se diseñan de tal manera, que si cualquiera de los códigos no usados se aplican a la entrada, ninguna de las salidas se activará.

La **Figura 2**, muestra la circuitería para un decodificador con *3 entradas* y 2^3*=8 salidas*. Como sólo usan compuertas **Y**, las salidas activadas son **1**. Para tener salidas activas **0**, deberían usarse compuertas **No Y**.

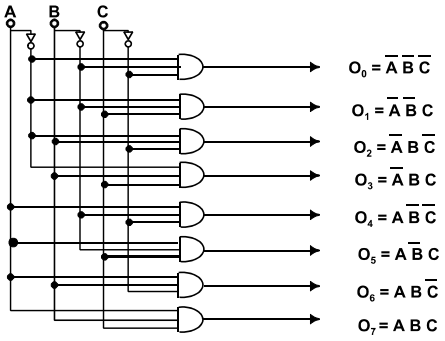


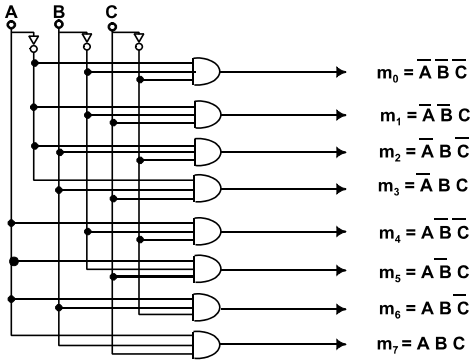
FIGURA 2. DECODIFICADOR BINARIO A OCTAL

Puede hacerse referencia a este codificador de distintas maneras, todas ellas válidas y usuales. Puede llamarse un *decodificador de 3 líneas a 8 líneas* (3 x 8), porque tiene tres líneas de entrada y ocho de salida. También recibe el nombre de *convertidor o decodificador binario a octal*, porque toma un código de entrada binario de tres entradas y produce un **1** en una de las ocho (octal) salidas correspondientes a ese código. A veces se hace referencia al circuito como un *decodificador 1 de 8*, porque **1** de las **8** salidas se activa a la vez. A continuación se muestra la Tabla funcional para este decodificador (**74138**):

TABLA FUNCIONAL

DEC	A	B	C	M_O	M_1	M_2	M_3	M_4	M_5	M_6	M_7
0	0	0	0	1	0	0	0	0	0	0	0
1	0	0	1	0	1	0	0	0	0	0	0
2	0	1	0	0	0	1	0	0	0	0	0
3	0	1	1	0	0	0	1	0	0	0	0
4	1	0	0	0	0	0	0	1	0	0	0
5	1	0	1	0	0	0	0	0	1	0	0
6	1	1	0	0	0	0	0	0	0	1	0
7	1	1	1	0	0	0	0	0	0	0	1

El logigrama correspondiente es:



Si se tiene una función reducida, deberá primero obtenerse su forma canónica para poderla realizar con un decodificador.

La mayoría de estos circuitos tienen sólo dos niveles de conmutación.

La tabla funcional queda en función de minitérminos por utilizarse lógica positiva.

Ejemplo 1: Diseñar un decodificador **BCD** a **DECIMAL**.

SOLUCIÓN

Un decodificador que toma un código BCD de 4 bits en la entrada y produce **10 salidas** correspondientes a los dígitos decimales, se denomina un *decodificador* (o convertidor) *BCD a decimal*. La **Figura 3**, muestra el arreglo lógico básico que usa compuertas **Y**. Cada salida se hace **1** cuando ocurre su grupo codificado *BCD* correspondiente. Por ejemplo, O_5 es **1** sólo cuando *0101* (5 en *BCD*) ocurra en las entradas *ABCD*, respectivamente. Este decodificador se llama también un *decodificador de 4 a 10 líneas* (4x10) o un *decodificador 1 de 10*.

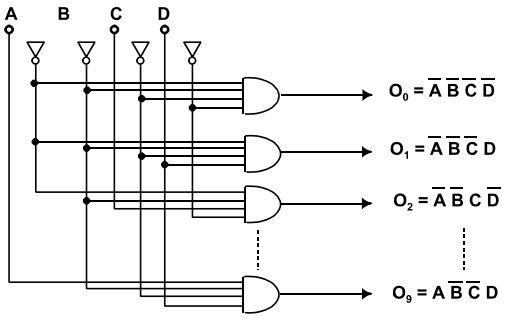


FIGURA 3. LOGIGRAMA PARA UN DECODIFICADOR BCD A DECIMAL

DEC	A B C D	$0_0\ 0_1\ 0_2\ 0_3\ 0_4\ 0_5\ 0_6\ 0_7\ 0_8\ 0_9$

0	0 0 0 0	1 0 0 0 0 0 0 0 0 0
1	0 0 0 1	0 1 0 0 0 0 0 0 0 0
2	0 0 1 0	0 0 1 0 0 0 0 0 0 0
3	0 0 1 1	0 0 0 1 0 0 0 0 0 0
4	0 1 0 0	0 0 0 0 1 0 0 0 0 0
5	0 1 0 1	0 0 0 0 0 1 0 0 0 0
6	0 1 1 0	0 0 0 0 0 0 1 0 0 0
7	0 1 1 1	0 0 0 0 0 0 0 1 0 0
8	1 0 0 0	0 0 0 0 0 0 0 0 1 0
9	1 0 0 1	0 0 0 0 0 0 0 0 0 1
10	1 0 1 0	**todas las salidas = 0**
11	1 0 1 1	
12	1 1 0 0	
13	1 1 0 1	
14	1 1 1 0	
15	1 1 1 1	

Este decodificador es un ejemplo de uno que no usa todas las combinaciones de entrada. Los grupos codificados **1010** hasta **1111** son ilegales para **BCD** y no producen ninguna salida activa. En la familia TTL, el circuito integrado 7442 (Mediana Escala de Integración) es un decodificador **BCD** a decimal con salidas activas **0**.

Sin embargo, hemos visto que cuando se tienen códigos de entrada que no se presentan, como es el caso, se pueden usar dichos códigos como términos ***indiferentes***. Por tanto, de la tabla se obtienen las siguientes funciones de conmutación:

$$D_0 = \sum_m(0) + \sum_x(10-15) \qquad D_5 = \sum_m(5) + \sum_x(10-15)$$

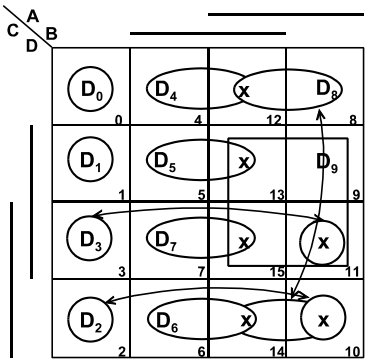
$$D_1 = \sum_m(1) + \sum_x(10-15) \qquad D_6 = \sum_m(6) + \sum_x(10-15)$$

$$D_2 = \sum_m(2) + \sum_x(10-15) \qquad D_7 = \sum_m(7) + \sum_x(10-15)$$

$$D_3 = \sum_m(3) + \sum_x(10-15) \qquad D_8 = \sum_m(8) + \sum_x(10-15)$$

$$D_4 = \sum_m(4) + \sum_x(10-15) \qquad D_9 = \sum_m(9) + \sum_x(10-15)$$

Las funciones de conmutación, se reducen utilizando un solo mapa de Karnaugh, en el cual se coloca la función $\mathbf{D_0}$ a $\mathbf{D_9}$ en lugar del minitérmino correspondiente. Los enlaces se realizan considerando cada una de las funciones con los términos indiferentes. El mapa se presenta en la figura adjunta:



Del mapa se obtienen las funciones reducidas:

$$D_0(A,B,C,D) = \overline{A}\,\overline{B}\,\overline{C}\,\overline{D}$$

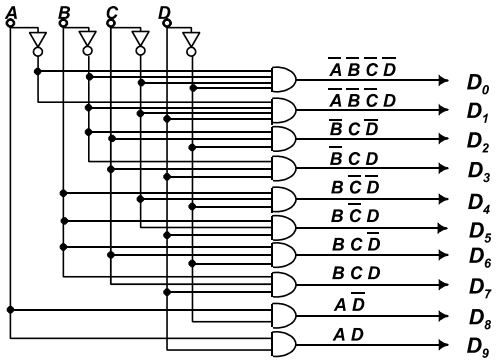
$$D_1(A,B,C,D) = \overline{A}\,\overline{B}\,\overline{C}\,D$$

$$D_2(A,B,C,D) = \overline{B}\,C\,\overline{D} \qquad D_3(A,B,C,D) = \overline{B}\,C\,D \qquad D_4(A,B,C,D) = B\,\overline{C}\,\overline{D}$$

$$D_5(A,B,C,D) = B\,\overline{C}\,D \qquad D_6(A,B,C,D) = B\,C\,\overline{D} \qquad D_7(A,B,C,D) = B\,C\,D$$

$$D_8(A,B,C,D) = A\,\overline{D} \qquad D_9(A,B,C,D) = A\,D$$

El logigrama correspondiente es:



Puede observarse de ambos logigramas que se reducen el números de entradas en algunas de las compuertas **Y**. Esto es válidos siempre y cuando no ocurran los códigos de entrada del **10** al **15**.

EJEMPLO 2: Diseñar un *sumador completo* de **2** bits, con un ***decodificador*** y ***compuertas externas***.

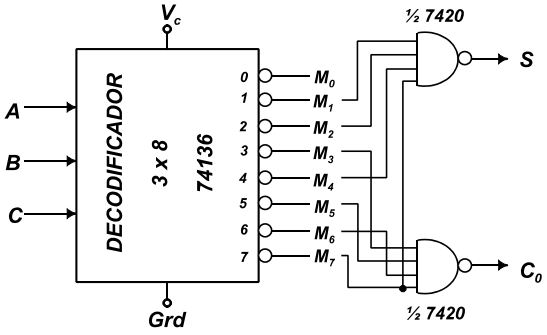
SOLUCIÓN

Las expresiones para la suma y el acarreo del sumador completo de 2 bits, son:

$$S(A,B,C) = \sum_m (1,2,4,7)$$

$$C_0(A,B,C) = \sum_m (3,5,6,7)$$

El logigrama correspondiente es:



El circuito integrado **7420**, contiene dos compuertas **No Y** con cuatro entradas cada una.

EJEMPLO 3: Diseñar un decodificador **BCD** a **7** ***segmentos***.

SOLUCIÓN

Algunos despliegues numéricos usan una configuración de 7 segmentos (**Figura 4. (a)** para producir los caracteres decimales *0-9*. Cada segmento puede ser un *diodo emisor de luz* (LED-Light Emisor Diode). La **Figura 4.(b)**, muestra los patrones de los segmentos que se usan para desplegar los diferentes dígitos. Por ejemplo, para desplegar el número **6**, los segmentos ***c***, ***d***, ***e***, ***f*** y ***g*** se activan mientras los segmentos ***a*** y ***b*** no lo están.

Un **decodificador/manejador** BCD a **7** ***segmentos***, se usa para tomar una

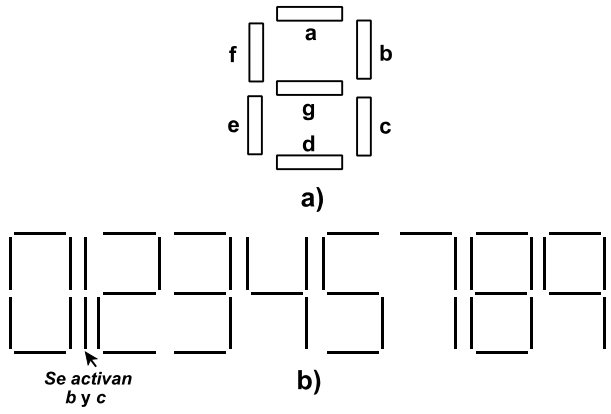


FIGURA 4. a) Arreglo de 7 segmentos. b) Segmentos activos para cada dígito

entrada *BCD de 4 bits* y desplegar el *dígito decimal*, después de pasar corriente por los segmentos apropiados. La lógica para este decodificador es más complicada que aquellas examinadas previamente, porque cada salida se activa por más de una combinación de entradas. Por ejemplo, el segmento *e* debe activarse para cualquiera de los dígitos *0*, *2*, *6* y *8*, lo que ocurre en cualesquiera de los códigos *0000*, *0010*, *0110* ó *1000*. La siguiente tabla funcional, presenta la relación de la entrada en BCD y la activación de los segmentos del desplegado.

DEC	CÓDIGO BCD				EXHIBIDOR DE 7 SEGMENTOS						
	A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	0	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	0	0	1	1
10 | 15					x | x	x | x	x | x	x | x	x | x	x | x	x | x

Una vez establecida la tabla funcional, se obtienen las expresiones booleanas para cada salida y luego se simplifican e implementan usando las técnicas vistas en el capítulo 3. Este procedimiento se deja como ejercicio.

La **Figura 5**, muestra un decodificador **BCD** a **7 *segmentos*** (TTL 7446 o 7447) que se usa para manejar una lectura con diodos emisores de luz de 7 segmentos. Cada segmento consiste de uno o dos diodos emisores de luz. Los ánodos de los diodos están todos conectados a $\mathbf{V_{cc}}$ (+5 volts). Los cátodos de los mismos están conectados a través de resistencias limitadoras de corriente a las salidas apropiadas del decodificador. Este último tiene salidas activas **0** (**BAJAS**), las cuales son transistores de manejo con colector abierto, que pueden absorber corrientes bastante altas. Esto es porque las lecturas con diodos emisores de luz pueden requerir entre **10** y **40** mA por segmento, dependiendo del tipo y tamaño.

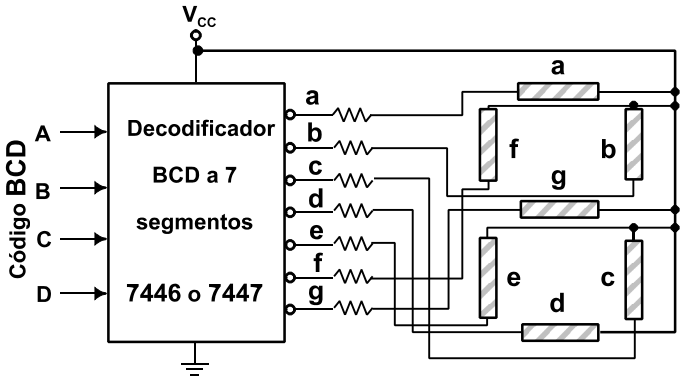


FIGURA 5. Circuito típico para un exhibidor de 7 segmentos

Para ilustrar la operación de este circuito, supóngase que la entrada **BCD** es *A=0*, *B=1*, *C=0* y *D=1*, que es **5** en **BCD**. Con estas entradas las salidas del decodificador $\bar{a}$, $\bar{f}$, $\bar{g}$, $\bar{c}$ y $\bar{d}$ serán conducidas a **0** (conectadas a tierra), permitiendo que la corriente fluya a través de los segmentos ***a***, ***f***, ***g***, ***c*** y ***d*** desplegando por consiguiente el numeral **5**. Las salidas $\bar{b}$ y $\bar{e}$ estarán en **1** (abiertas), así que los segmentos del diodo ***b*** y ***e*** no pueden conducir.

EJEMPLO 4: Decodificador de dos a cuatro líneas con entrada de habilitación (*enable*).

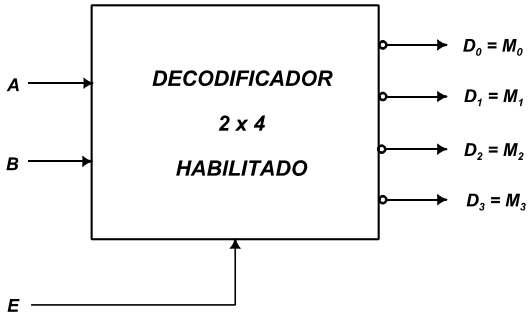
SOLUCIÓN

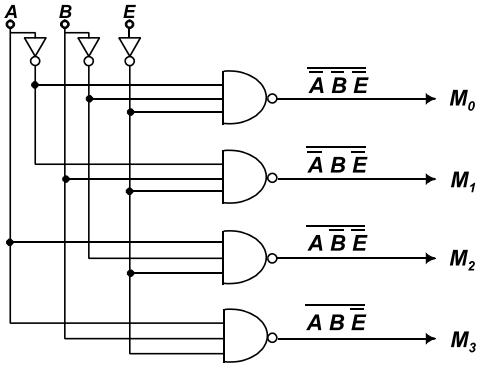
La tabla funcional para este decodificador es:

TABLA FUNCIONAL

E	A	B	D_0	D_1	D_2	D_3
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

El diagrama a bloques y el logigrama se muestran a continuación:





EJEMPLO 5: Diseñar un **decodificador** de **4x16** con dos decodificadores de **3x8**, con entrada **E** de habilitación.

SOLUCIÓN

La siguiente figura muestra el diagrama correspondiente al **decodificador** de **4 x 16**:

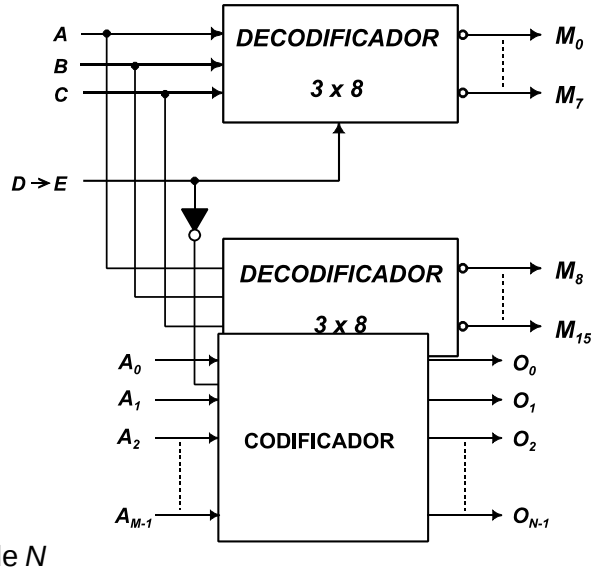


FIGURA 6. DIAGRAMA GENERAL DE UN CODIFICADOR

2.2. Codificadores

Un *decodificador* acepta un código de entrada de *N* bits y produce un **1** ó **0** en *una y sólo una* línea de salida. En otras palabras, se puede decir que un decodificador identifica, reconoce o detecta un código particular. El

opuesto de este proceso de decodificación es llamado **codificación** y es ejecutado por un circuito lógico llamado ***codificador***. Un *codificador* tienen un número de líneas de entrada, de las cuales sólo una es activada en un tiempo dado y produce un código de salida de ***N*** bits, dependiendo de cuál entrada es activada. La **Figura 6**, muestra el diagrama general de un codificador con ***M*** entradas y ***N*** salidas. Todas las entradas y salidas están en **1** cuando están activadas (Note la ausencia de círculos en el diagrama).

Se vio que un decodificador binario a octal acepta un código binario de entrada de 3 bits y activa una de las ocho líneas de salida. Un *codificador octal a binario* opera de la manera opuesta. Acepta *ocho líneas de entrada* y produce un *código binario de 3 bits* a la salida. Su logigrama se muestra en la **Figura 7**, tomando como base la siguiente tabla funcional:

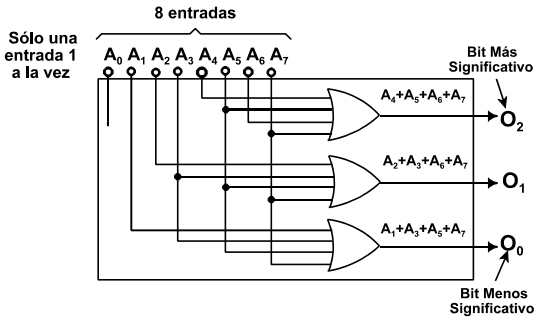


FIGURA 7.CODIFICADOR OCTAL A BINARIO

ENTRADA								CÓDIGO BINARIO		
A_0	A_1	A_2	A_3	A_4	A_5	A_6	A_7	O_2	O_1	O_0
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

Donde las funciones de conmutación son:

$$O_2(A_0,A_1,A_2,A_3,A_4,A_5,A_6,A_7) = A_4 + A_5 + A_6 + A_7$$

$$O_1(A_0,A_1,A_2,A_3,A_4,A_5,A_6,A_7) = A_2 + A_3 + A_6 + A_7$$

$$O_0(A_0,A_1,A_2,A_3,A_4,A_5,A_6,A_7) = A_1 + A_3 + A_5 + A_7$$

Se supone que sólo una de las entradas es **1** cada vez, así que sólo hay *8 condiciones posibles* de entrada. El circuito está diseñado de tal manera que cuando $\mathbf{A_0}$ es **1**, se genera a la salida el código binario **000**; cuando $\mathbf{A_1}$ es **1**, se genera el código binario **001**, cuando $\mathbf{A_2}$ es **1**, se genera el código binario **010** y así sucesivamente. El diseño del circuito es muy simple, puesto que involucra analizar cada bit de salida y determinar para cuáles casos de entrada ese bit es **1** y luego pasar los resultados por una compuerta **O**. Por ejemplo, la tabla funcional muestra que $\mathbf{O_0}$ (bit menos significativo del código de salida) debe ser **1** cuando cualesquiera de las entradas $\mathbf{A_1}$, $\mathbf{A_3}$, $\mathbf{A_5}$ o $\mathbf{A_7}$ sean **1**.

Ejemplo 6: Describir la estructura y operación de un codificador *decimal* a *BCD* con salidas activas **0**.

SOLUCIÓN

Este codificador toma **10** líneas de entrada, una sola de las cuales estará en **1** y produce un código de salida de **4** bits *BCD*. Puesto que hay cuatro salidas, el circuito contiene cuatro

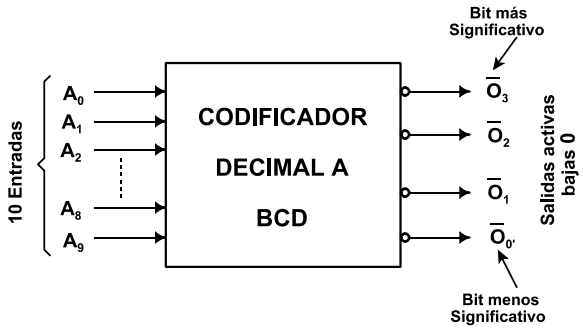


FIGURA 8. CODIFICADOR DECIMAL A BCD

compuertas. Las compuertas usadas son **No O**, porque han de ser normalmente **1** e ir a **0**, cuando una de sus entradas se hace **1**. La **Figura 8**, muestra el logigrama de este codificador.

3. MULTIPLEXOR Y DEMULTIPLEXOR

3.1 Multiplexor

Un ***multiplexor*** o selector de datos es un circuito lógico que acepta *varias entradas de datos* y permite que *sólo una de ellas* pase en un tiempo a la salida. El enrutamiento de la entrada de datos hacia la salida está controlada por las entradas de ***selección*** (a las que se hace referencia a veces como entradas de *dirección*). La **Figura 9** muestra el diagrama general de un ***multiplexor***. En este diagrama las entradas y salidas se dibujan como flechas gruesas para indicar que pueden ser una o más líneas.

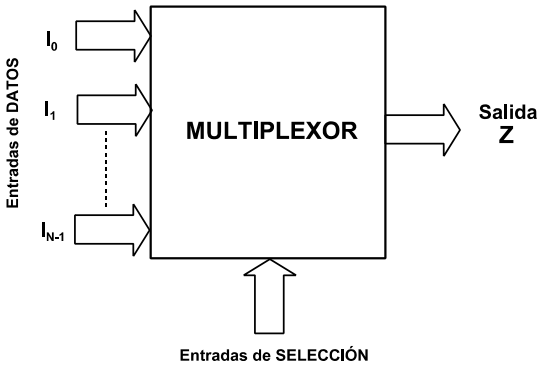


FIGURA 9. Diagrama general de un *MULTIPLEXOR* digital

El *multiplexor* actúa como un *conmutador multiposicional* controlado digitalmente, donde el código digital aplicado a las entradas de *SELECCIÓN*, controla cuáles entradas de datos serán conmutadas hacia la salida. Por ejemplo, la salida **Z** será igual a la entrada de datos I_0 para algún código de entrada particular de *selección*; **Z** será igual a I_1 para otro código particular de *selección* de entrada y así sucesivamente. Establecido de otra manera, un multiplexor selecciona **1** de **N** fuentes de entrada de datos y transmite los datos seleccionados a un solo canal de salida. Esto se llama *multiplexión* o *multiplexaje*.

La **Figura 10**, muestra la circuitería lógica para un ***multiplexor*** de dos entradas (o dos canales) con entradas de datos **A** y **B** y entrada de *selección* **S**. El nivel lógico aplicado a la entrada **S** determina cuál compuerta **Y** se activa, para que su entrada de datos pase a través de la compuerta **O** a la salida

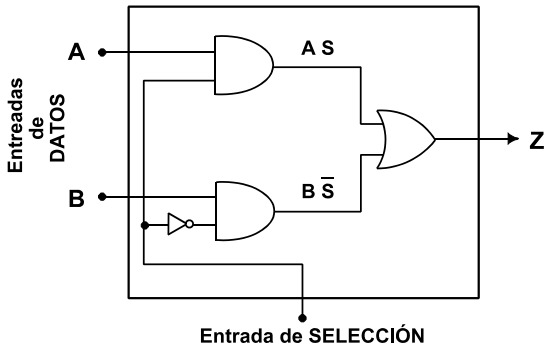


FIGURA 10. Multiplexor de dos entradas

Z. Visto de otra manera, la expresión booleana para la salida se obtiene de la siguiente tabla funcional:

Selección S	Salida Z
0	B
1	A

$$Z = A S + B \overline{S}$$

Con **S = 0**, esta expresión se hace:

$$Z = A \cdot 0 + B \cdot 1 = B$$

lo cual indica que **Z** será idéntico a la señal de entrada **B**, la cual puede ser un nivel lógico fijo o una señal lógica variable con el tiempo.

Con **S = 1**, la expresión se hace:

$$Z = A \cdot 1 + B \cdot 0 = A$$

Mostrando que la salida **Z** será idéntica a la señal de entra **A**.

EJEMPLO: Muestre cómo el multiplexor de la **Figura 10** puede usarse para tomar dos números binarios de 3 bits (X_2, X_1, X_0 y Y_2, Y_1, Y_0) y transmitir uno o el otro número a las salidas Z_2, Z_1 y Z_0, dependiendo de un nivel de selección de entrada.

SOLUCIÓN

La **Figura 11** muestra 3 multiplexores de dos entradas que se usan para cumplir la operación deseada. Note que las entradas **S** de cada multiplexor se conectan a una entrada de selección común. Cuando **S=1** se enrutan las entradas **X** de cada multiplexor individual a través de las salidas **Z**. Cuando **S=0**, las entradas **Y** se enrutan hacia las salidas.

La misma idea básica puede usarse para formar el multiplexor de cuatro entradas que se muestra en la **FIGURA 12**. Aquí hay cuatro entradas, las cuales se transmiten selectivamente a la salida, en base a las cuatro combinaciones posibles de las entradas de selección S_1S_0. Cada entrada de datos pasa por compuertas con una

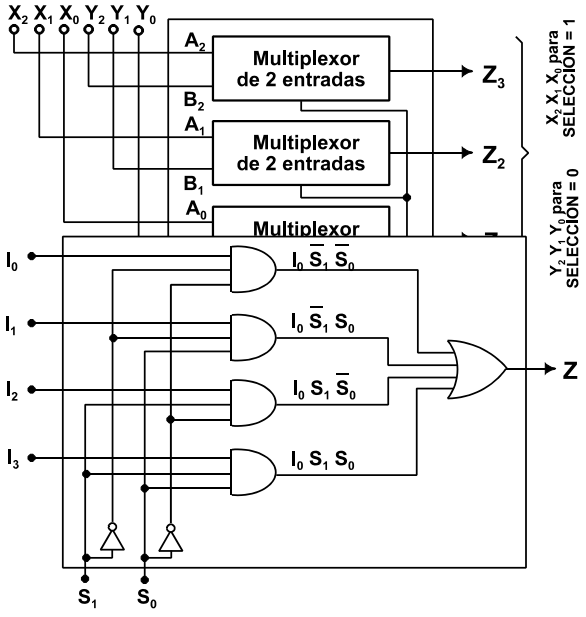


FIGURA 12. Multiplexor de cuatro canales

combinación diferente de los niveles de selección de entrada, como lo muestra la siguiente tabla:

Entradas de Selección		Salida Z
S_1	S_0	
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

I_0 pasará a través de su compuerta **Y** a las salida **Z** sólo cuando **S_1=0** y **S_0=0**. La tabla anterior muestra las salidas para los otros tres códigos de selección de entrada.

Las familias TTL y CMOS disponen de multiplexores de 2, 4, 8 y 16 entradas.

Un tipo muy útil de multiplexor es el de dos canales, 4 bits que se muestra en la **Figura 13**. Este multiplexor opera básicamente como el multiplexor de la **FIGURA 10**, excepto que las entradas **A** y **B** y la salida **Z** son *grupos de datos de 4 bits*. Los cuatro bits de salida Z_3, Z_2, Z_1 y Z_0 aparearán ya sea las cuatro entradas **A** o las cuatro entradas **B**, dependiendo de la entrada de selección **S**.

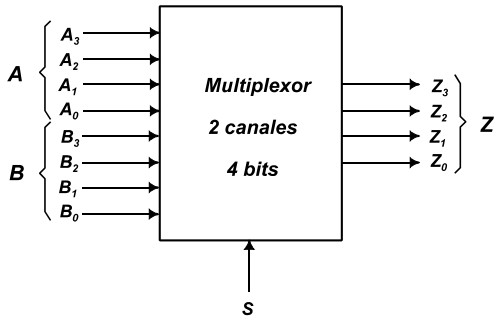


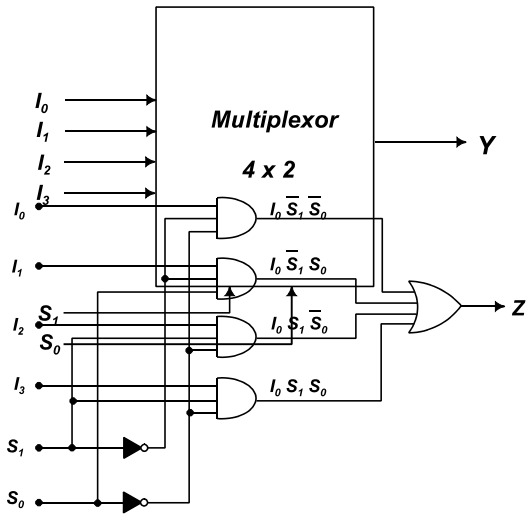
FIGURA 13. Diagrama de un multiplexor de 2 canales, 4 bits

Para un *multiplexor de 4x1*, es decir, dos señales selectoras, se tiene la siguiente tabla funcional:

DEC	S_1	S_0	Y
0	0	0	I_0
1	0	1	I_1
2	1	0	I_2
3	1	1	I_3

Donde: **S_1** y **S_0** son las señales selectoras, **I_0**, **I_1**, **I_2** e **I_3**, las entradas y **Y** la salida.

El diagrama adjunto esquematiza al **multiplexor** de **4x**1.



Usando compuertas, se observa el contenido del diagrama anterior, como lo muestra la figura siguiente:

EJEMPLO: Realizar la siguiente función utilizando un multiplexor con dos variables selectoras.

$$F(A,B,C) = \sum_m (1,3,5,6)$$

SOLUCIÓN

1. Tabla Funcional:

DEC	A	B	C	E_Y
0	0	0	0	**0**
1	0	0	1	**1**
2	0	1	0	**0**
3	0	1	1	**1**
4	1	0	0	**0**
5	1	0	1	**1**
6	1	1	0	**1**
7	1	1	1	**0**

2. Asignación de las variables de la función **F(A, B, C)** a las variables selectoras del multiplexor.

$S_0 \leftarrow C$

$S_1 \leftarrow B$

$? \leftarrow A$

Como n+1=3, entonces **n = 2** y por tanto, se requiere un multiplexor con 2^2 entradas.

3. Se forma una fila con los valores de las señales de entrada del multiplexor.

4. Se forman dos filas más, la primera con el valor complementado de la variable que se está buscando su asignación (el valor más significativo de la función booleana). A la fila 3 se le asigna el valor sin complementar de la variable más significativa de la función booleana.

5. En la fila 2 se escribe todos los minitérminos en donde aparece $\overline{A}$. En la fila 3 se escriben todos los minitérminos en donde **A** está sin complementar.

6. Se encierran en un círculo (o paréntesis) todos los minitérminos que forman parte de la función booleana.

7. Se analiza columna por columna, es decir, cada una de las entradas del multiplexor y se le asigna un **0** si tanto el minitérmino superior, como el minitérmino inferior, no están encerrados en un círculo, este valor de **0** será el valor que tomará la señal de entrada I_0, si en la siguiente columna ambos minitérminos están encerrados en un círculo, se le asigna el valor de **1**, valor que tomará la señal I_1. En la siguiente columna el minitérmino no está encerrado en la fila 2, pero sí lo está en la 3, por lo que se le asigna el valor de **A**. Finalmente, si en la última columna el minitérmino superior está encerrado en un círculo y el inferior no lo está, con lo que se le asigna el valor de $\overline{A}$.

Lo anterior se muestra en la siguiente tabla:

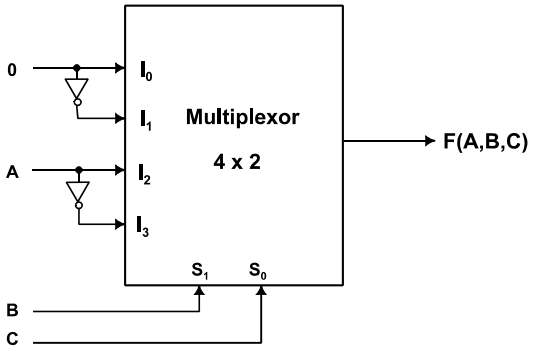
	I_0	I_1	I_2	I_3
$\overline{A}$	0	(1)	(2)	(3)

A	4	(5)	(6)	7
	8	8	8	8
	0	**1**	***A***	$\overline{A}$

Por tanto, los valores asignados a las entradas del multiplexor son:

$I_0 = 0$
$I_1 = 1$
$I_2 = A$
$I_3 = \overline{A}$

La figura adjunta, muestra el logigrama correspondiente:



EJEMPLO: Realizar la siguiente función utilizando un multiplexor.

$$F(A,B,C,D) = \sum_m (0,1,3,4,8,9,15)$$

Como el número de variables de entrada es igual a **4**, entonces **n+1=4**, por lo que **n=3**, es decir, 3 variables selectoras y 2^3 variables de entrada. En base a lo anterior, se requiere un ***multiplexor*** de ***8x1***.

Se hace la siguiente asignación de variables:

$S_0 \leftarrow D$

$S_1 \leftarrow C$

$S_2 \leftarrow B$

$? \leftarrow A$

En la siguiente tabla se determinan los valores de las entradas del multiplexor.

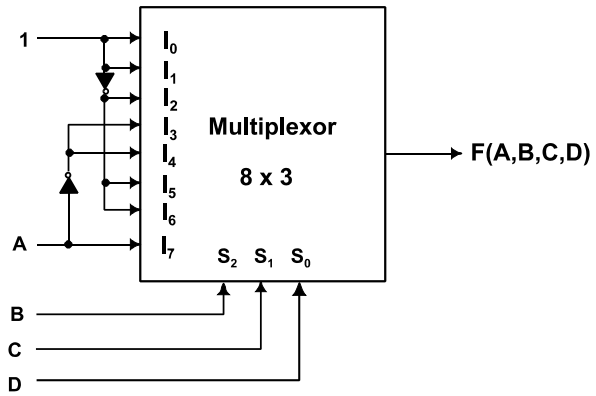
	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7
$\overline{A}$	**(0)**	**(1)**	2	**(3)**	**(4)**	5	6	7
A	**(8)**	**(9)**	10	11	12	13	14	**(15)**
	1	**1**	**0**	$\overline{A}$	$\overline{A}$	**0**	**0**	A

De la tabla se obtienen los siguientes valores a las entradas del multiplexor:

$I_0 \leftarrow 1$
$I_1 \leftarrow 1$
$I_2 \leftarrow 0$
$I_3 \leftarrow \overline{A}$

$I_4 \leftarrow \overline{A}$
$I_5 \leftarrow 0$
$I_6 \leftarrow 0$
$I_7 \leftarrow A$

El logigrama correspondiente es:



3.2. Demultiplexor (distribuidor de datos)

Un *multiplexor* toma varias entradas y transmite una de ellas a la salida. Un ***demultiplexor*** toma una sola entrada y la distribuye sobre varias salidas. La **Figura 14**, muestra el diagrama general para un *demultiplexor*. Las flechas grandes para entradas y salidas pueden representar una o más líneas. El código de entrada *selección* determina a cuál salida será transmitida la entrada *datos*. En otras palabras, el *demultiplexor* toma una fuente de datos de entrada y la distribuye en forma selectiva a **1** de ***N*** canales de salida.

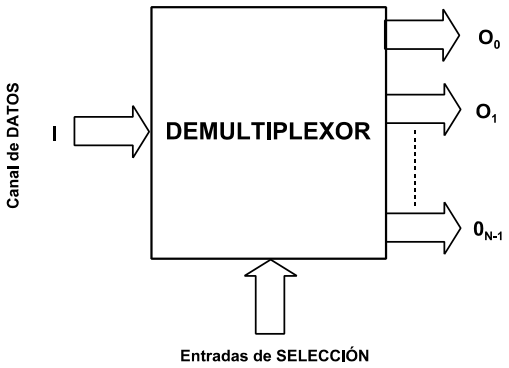


***FIGURA 14. DEMULTIPLEXOR* general**

La **Figura 15**, muestra el logigrama para un *demultiplexor* que distribuye *una línea de entrada a ocho líneas de salida*. La sola línea de entrada de datos **I** se conecta a todas las ocho compuertas **Y**, pero una sola de ellas será capacitada por las líneas de entrada *selección*. Por ejemplo, para $\mathbf{S_2\ S_1\ S_0 = 0\ 0\ 0}$, sólo la primera compuerta **Y** será habilitada y la entrada de datos **I** aparecerá en la salidas $\mathbf{O_0}$**.** Para otros códigos de *selección*, la entrada **I** estará presente en otras salidas.

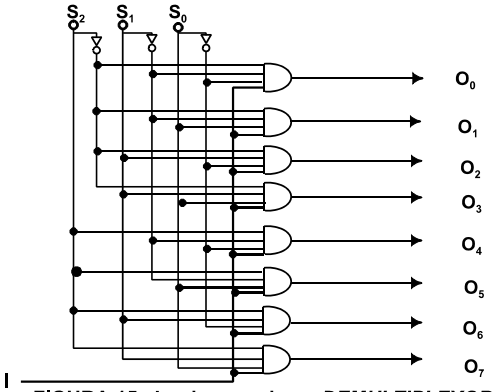


FIGURA 15. Logigrama de un *DEMULTIPLEXOR*

El demultiplexor de la **Figura 15**, es realmente una modificación del circuito decodificador de la **Figura 2**. Si se añade una cuarta entrada a todas las compuertas decodificadoras, esta entrada puede usarse como la entrada común de datos **I** y las entradas **A**, **B** y **C** pueden servir como las líneas de selección. Muchos decodificadores proveen esta entrada común extra (llamada generalmente la entrada *habilitadora*), así el ***decodificador*** puede usarse también como un ***demultiplexor***.

4. MEMORIA DE SOLO LECTURA (ROM - **R**ead **O**nly **M**emory**)**

Como se vio en la sección **2**, un decodificador genera $\boldsymbol{2^n}$ términos mínimos de las ***n*** variables de entrada. Colocando las compuertas **O** para sumar los términos mínimos de las funciones de Boole, se podrá generar cualquier circuito combinacional. Una ***memoria de solo lectura*** (*ROM*) es un elemento que incluye el decodificador y las compuertas **O** dentro de una cápsula de circuito integrado. Las conexiones entre las salidas del decodificador y las entradas a las compuertas **O**, se especifican para cada configuración particular, programando la *ROM*. Esta se usa a menudo para configurar un circuito combinacional complejo en un solo circuito integrado y así eliminar los cables de conexión.

Una *ROM* es esencialmente un dispositivo (o acumulador) de memoria en el cual se almacena un conjunto fijo de información binaria. La información binaria debe especificarse por el usuario y luego enclavarse en la unidad para formar el patrón de interconexión requerida. Las *ROM* vienen con enlaces internos especiales que pueden estar fusionados o abiertos. La interconexión deseada para una aplicación particular requiere que ciertos enlaces estén fusionados para formar los caminos del circuito necesarios. Una vez que se establezca un patrón para una *ROM*, éste permanecerá fijo, aunque se haga un corte de corriente y luego se restablezca.

La **Figura 16** muestra un diagrama a bloques de una *ROM*. Este consiste en ***n*** líneas de entrada y ***m*** líneas de salida. Cada combinación de bits de las variables de entrada se llama un *dirección*. Cada combinación de bits que sale por las líneas de salida se llama una *palabra*. El número de bits por palabra es igual al número de líneas de salida *m*. Una dirección es esencialmente un número binario que denota uno de los términos mínimos de *n* variables. El número de *direcciones* diferentes posibles de ***n*** variables de entrada es 2^n. Una *palabra* de salida puede seleccionarse por una *dirección* única y como hay 2^n direcciones diferentes en una *ROM*, hay 2^n *palabras* diferentes que se dice que están acumuladas en la unidad. La palabra disponible en las líneas de salida, en cualquier momento dado, depende del valor de la dirección aplicada a las líneas de entrada. Una *ROM* se caracteriza por el número de palabras 2^n y el número de bits por palabra ***m***. Esta terminología se usa debido a la similitud entre la memoria de sólo lectura y la memoria de lectura-escritura.

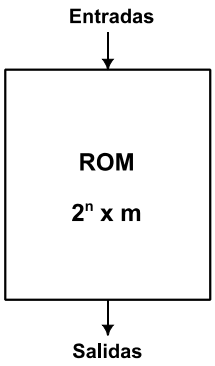


FIGURA 16. Diagrama a bloques de una ROM

Considérese una *ROM* de **32x8**. La unidad consiste de *32 palabras de 8 bits cada una*. Esto significa que hay *8 líneas de salida* y *32 palabras* distintas almacenadas en la unidad. La palabra particular seleccionada que está presente en las líneas de salida se determina a partir de las cinco líneas de entrada. Hay solamente cinco entradas en una *ROM* de 32x8 porque 2^6=32 y con cinco variables se pueden especificar **32** direcciones o términos mínimos. Para cada dirección de entrada hay una palabra única seleccionada. Así, si una dirección de entrada es **00000** se selecciona la palabra **0** y ésta aparece en las líneas de salida. Si la dirección de entrada es **11111**, se selecciona la palabra número **31** y se aplica a las líneas de salida. Entre la primera y la última hay otras 30 direcciones que pueden seleccionar otras 30 palabras.

El número de palabras direccionadas en una *ROM* se determina del hecho de que se necesitan *n* líneas de entrada para especificar 2^n palabras. Una *ROM* se especifica algunas veces por el número total de bits que contiene, el cual será $2^n xm$. Por ejemplo, una *ROM* de **2048** bits puede organizarse como **512** palabras de **4** bits cada una. Esto significa que la unidad tiene **4** líneas de salida y **9** líneas de entrada para especificar $\mathbf{2^9=512}$ palabras. El número total de bits en la unidad es **512x4=2,048**.

Internamente, la *ROM* es un circuito combinacional con compuertas **Y** conectadas como *decodificador* y un número de compuertas **O** igual al número de *salidas* de la unidad. La **Figura 17**, muestra la construcción lógica interna de una *ROM* de **32x4**. Las cinco variables de entrada se decodifican en **32** líneas por medio de **32** compuertas **Y** y *cinco* inversores. Cada salida del decodificador representa uno de los términos mínimos de una función de cinco variables. *Cada una de las **32** direcciones selecciona una y sólo una salida del decodificador*. La dirección es un número de cinco bits aplicado a las entradas y el término mínimo seleccionado por fuera del decodificador es el marcado con el número decimal equivalente. Las **32** salidas del decodificador están conectadas por medio de enlaces a cada compuerta **O**.

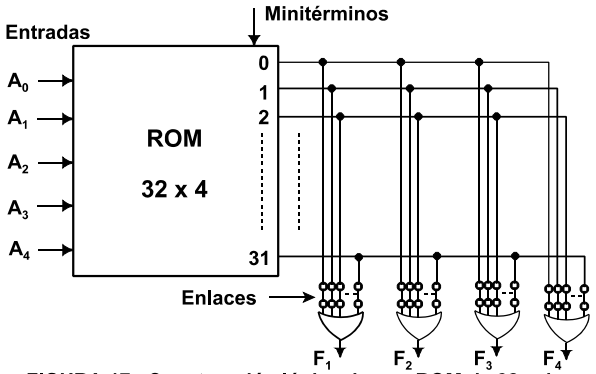


FIGURA 17. Construcción lógica de una ROM de 32 x 4

Solamente **4** de estos enlaces se muestran en el diagrama, pero realmente cada compuerta **O** tiene **32** entradas y cada entrada pasa a través de un enlace que puede estar cortado, si así se desea.

La *ROM* es una configuración de dos *niveles* en forma de suma de términos mínimos. No tiene que ser una configuración **Y-O**, pero puede ser cualquiera otra posible configuración de términos mínimos de dos niveles. El segundo nivel es normalmente una conexión de lógica cableada para facilitar la función de los enlaces.

Las *ROM* tienen muchas aplicaciones importantes en el diseño de sistemas de computadoras digitales. Su uso para la configuración de *circuitos combinacionales complejos* es justamente una de esas aplicaciones.

Del diagrama lógico de la *ROM*, es claro que cada salida produce la suma de todos los términos mínimos de *n* variables de entrada. Recuérdese que una función de Boole puede expresarse en forma de suma de términos mínimos. Al romper los enlaces de aquellos términos mínimos que no se incluyen en la función, cada salida de la *ROM* puede hacerse que represente la función de Boole de una de las variables de salida de un circuito combinacional. Para un circuito combinacional de *n* entradas y *m* salidas se necesita una *ROM* de $2^n xm$. La ruptura de los enlaces se refiere a la programación de la *ROM*. El diseñador necesita solamente especificar una tabla del programa *ROM* que da la información para los caminos necesarios en la *ROM*. La programación real es un procedimiento del material que sigue las especificaciones listadas en la tabla de programación.

Para aclarar el proceso es necesario un ejemplo específico. La siguiente tabla, correspondiente al logigrama de la **FIGURA 18**, especifica un circuito combinacional con dos entradas y dos salidas.

DEC	A_1	A_0	F_1	F_0
0	0	0	0	1
1	0	1	1	0
2	1	0	1	1
3	1	1	1	0

Las funciones de Boole pueden expresarse en suma de términos mínimos:

$$F_1(A_1, A_0) = \sum_m (1,2,3)$$

$$F_2(A_1, A_0) = \sum_m (0,2)$$

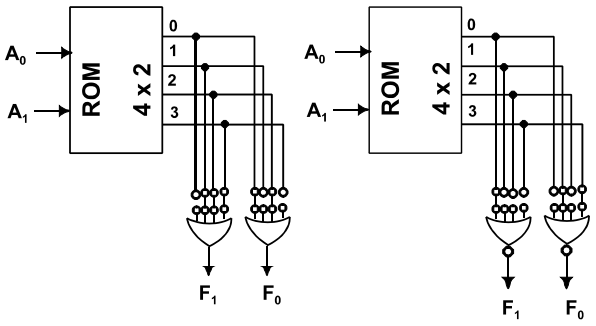


FIGURA 18. Confifuración del circuito combinacional con una ROM de 4 x 2

Cuando se configura un circuito combinacional por medio de una *ROM*, las funciones deben expresarse en *suma de términos mínimos* o mejor aún por una *tabla de verdad*. Si la salida de las funciones se simplifica, se encuentra que el circuito necesita solamente una compuerta **O** y un inversor. Obviamente, éste es un circuito combinacional simple como para ser ejecutado con una *ROM*. La ventaja de las *ROM* es su uso en circuitos combinacionales complejos. Este ejemplo solamente demuestra el procedimiento y no debe considerarse como una situación práctica.

La *ROM* que configura el circuito combinacional debe tener dos entradas y dos salidas, de tal manera que su tamaño deberá ser **4 x 2**. La **Figura 18(a)** muestra la construcción interna de una *ROM*. Es necesario determinar cuáles de los ocho enlaces disponibles deben romperse y cuáles deben dejarse sin tocar. Esto puede hacerse fácilmente de las funciones de salida listadas en la tabla de verdad. Aquellos términos mínimos que especifican una salida de **0** no deben tener un camino a la

salida a través de una compuerta **O**. Así, para este caso particular, la tabla de verdad muestra tres ceros y sus correspondientes enlaces con las compuertas **O** que deben quitarse. Es obvio que se debe asumir que un circuito abierto a una compuerta **O** se comporta como una entrada de **0**.

Algunas *ROM* vienen con un inversor después de cada de las compuertas **O** y como consecuencia se especifica que inicialmente tienen todos **0** en sus entradas. El procedimiento de programación de tales *ROM* requiere que se abran los enlaces de los *términos mínimos* (o direcciones) que especifiquen una salida de **1** en la tabla de verdad. La salida de la compuerta **O** complementa la función una vez más para producir una salida normal. Esto se muestra en la *ROM* de la **Figura 18(b)**.

El procedimiento anterior demuestra el procedimiento general para especificar un circuito combinacional con una *ROM*. A partir del número de entradas y salidas en el circuito combinacional, se determina primero el tamaño de la *ROM* requerido. Luego, se obtiene la tabla de verdad de programación de la *ROM*; no se necesita ninguna otra manipulación o simplificación. Los *ceros* (o *unos*) en las funciones de salida de la tabla de verdad, especifican directamente aquellos enlaces que deben ser removidos, para producir el circuito combinacional requerido, en la forma de suma de términos mínimos.

En la práctica, cuando se diseña un circuito por medio de una *ROM*, no es necesario mostrar los enlaces de las conexiones de las compuertas internas dentro de la unidad, como se hizo en la **Figura 18**; lo cual fue mostrado para propósitos de demostración solamente. Todo lo que el diseñador tiene que hacer es especificar la *ROM* (o su número asignado) y dar la tabla de verdad de la *ROM*, como en la tabla anterior. La tabla de verdad da toda la información para programar la *ROM*. No se necesita un diagrama interno que acompañe a la tabla de verdad.

Ejemplo 7: Diseñar un circuito combinacional usando una *ROM*, el cual acepte un número de 3 bits y genere un número binario a su salida igual a su cuadrado.

SOLUCIÓN

El primer paso es deducir la tabla de verdad para el circuito combinacional. En la mayoría de los casos es todo lo que se necesita, para otros es necesario adicionar una tabla más pequeña, que muestre ciertas propiedades del circuito combinacional.

DEC	ENTRADAS			SALIDAS						DEC
	A_1	A_2	A_0	B_5	B_4	B_3	B_2	B_1	B_0	
0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	1	1
2	0	1	0	0	0	0	1	0	0	4
3	0	1	1	0	0	1	0	0	1	9
4	1	0	0	0	1	0	0	0	0	16
5	1	0	1	0	1	1	0	0	1	25
6	1	1	0	1	0	0	1	0	0	36
7	1	1	1	1	1	0	0	0	1	49

La tabla anterior corresponde a la tabla de verdad del circuito combinatorio. Se requieren 3 entradas y siete salidas para generar todos los números posibles. Se observa que la salida $\mathbf{B_0}$ es siempre igual a la entrada $\mathbf{A_0}$, de tal manera que no es necesario obtener $\mathbf{B_0}$ con la ROM. Asimismo, $\mathbf{B_1}$ es siempre igual a **0**, por lo que siempre es conocida. Por tanto, sólo se requieren generar cuatro salidas con la ROM; las otras dos se obtienen fácilmente. El tamaño mínimo de la ROM debe ser de 3 entradas y 4 salidas. Las 3 entradas especifican 8 palabras, de tal manera que el tamaño de la ROM debe ser de **8x4**. La configuración de la ROM se muestra en la figura previa. Las 3 entradas determinan 8 palabras de 4 bits cada una. Las otras dos salidas de los circuitos combinacionales son iguales a **0** y $\mathbf{A_0}$.

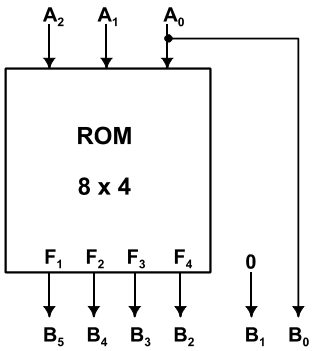


Diagrama a bloques

La siguiente tabla de verdad especifica toda la información necesaria para programar la ROM y el diagrama a bloques muestra las conexiones requeridas.

DEC	A_2	A_1	A_0	F_1	F_2	F_3	F_4
0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0
2	0	1	0	0	0	0	1
3	0	1	1	0	0	1	0
4	1	0	0	0	1	0	0
5	1	0	1	0	1	1	0
6	1	1	0	1	0	0	1
7	1	1	1	1	1	0	0

Los caminos necesarios en una ROM pueden programarse de dos maneras diferentes. La primera se llama *programación por máscara* y la hace el fabricante durante el último proceso de fabricación de la unidad. El procedimiento para fabricar esta ROM, requiere que el usuario llene la tabla de verdad en función de lo que desea que realice la ROM. El fabricante hace la máscara correspondiente para que los caminos produzcan unos y ceros de acuerdo a la tabla de verdad del usuario. Este procedimiento es muy costoso, razón por la cual sólo es conveniente si se van a fabricar grandes cantidades con el mismo tipo de configuración.

Para pequeñas cantidades, es más económico usar un segundo tipo de ROM llamado *memoria programable de solo lectura* o **PROM** (Prograble Read Only Memory). Cuando se adquieren, las unidades PROM contienen ceros (o unos) en cada bit de las palabras almacenadas. Los enlaces en la PROM se rompen por medio de pulsos de corriente a través de las terminales de salida. Un enlace

roto define un estado binario y uno no roto el otro estado. Esto le permite al usuario programar la unidad en su propio laboratorio, para lograr la relación deseada entre las direcciones de entrada y las palabras almacenadas. Comercialmente, se dispone de dispositivos especiales llamados *programadores de PROM*, para facilitar este procedimiento.

El procedimiento para programar las *ROM* o *PROM* es irreversible, por lo que una vez programado el patrón éste es permanente y no puede alterarse. Un tercer tipo de unidad es la llamada *memoria programable de solo lectura borrable* o **EPROM** (**E**rasable **P**rogramable **R**ead **O**nly **M**emory). Las *EPROM* pueden recuperarse a su valor inicial (todos unos o todos ceros) aunque se hayan cambiado previamente. Cuando una *EPROM* se coloca bajo una luz ultravioleta especial por un período dado de tiempo, la radiación de onda corta descarga los puentes internos que sirven de contactos, con lo cual regresa a su estado inicial para ser reprogramada.

Las **ROM** se usan ampliamente para ejecutar circuitos combinatorios complejos, directamente de sus tablas de verdad. Son muy útiles para convertir de un código binario a otro (tal como ASCII a EBCDIC o viceversa), para funciones aritméticas como multiplicadores, para mostrar caracteres en un tubo de rayos catódicos y en cualquier otra aplicación que requiera un gran número de entradas y salidas. Se emplean también en el diseño de unidades de control para sistemas digitales. Como tales, se usan para almacenar patrones fijos de bits que representan una secuencia de variables de control, necesarios para habilitar las diferentes operaciones en el sistema. Una unidad de control que utiliza una *ROM*, para almacenar información de control binario, se llama *unidad de control microprogramada*.

5. EJERCICIOS

1. Realice las siguientes funciones Booleanas, utilizando para cada caso **a)** un ***decodificador*** y compuertas externas y **b)** un ***multiplexor***.

a) $f(A,B,C,D) = \sum_m (0,4,6,10,11,13)$

b) $f(w,x,y,z) = \prod_M (3,4,5,7,11,12,14,15)$

c) $f(a,b,c,d) = \sum_m (3,5,7,11,15)$

d) $f(A,B,C,D,E) = \prod_M (0,1,2,8,9,11,15-17,19,24,25,29-31)$

e) $f(A,B,C,D,E,F) = \sum_m (0,2,4,5,7,8,16,18,24,32,36,40,48,56)$

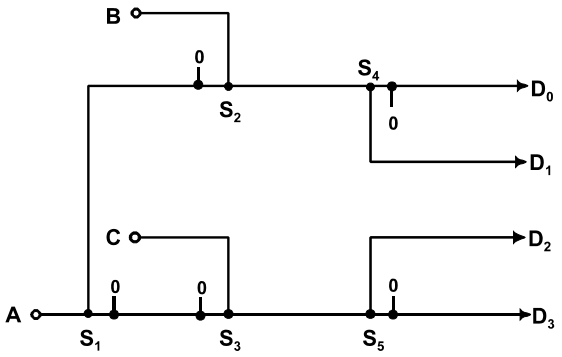
2. Un número primo es aquel que sólo es divisible entre sí mismo y la unidad. Diseñe un circuito lógico que detecte todos los números ***primos*** entre **0** y **31**. La salida ***F(A, B, C, D, E)***, donde ***A*** es la variable de mayor peso binario, será igual a <**1**>, si y sólo si, los cinco bits de entrada representan un número primo. Realice el logigrama utilizando un ***multiplexor***.

3. En uno de los laboratorios de una compañía químico farmacéutica se elaboran 14 distintas soluciones a partir de las componentes **W**, **X**, **Y**, **Z**. Estas sustancias pesan **800**, **400**, **200** y **100** mg., respectivamente. Las soluciones depositadas en frascos se transportan por medio de una banda hasta una báscula. Si el peso indicado en la báscula es uno de los siguientes: **200**, **500**, **700**, **800**, **1100**, **1400** y **1500** mg., entonces un dispositivo electromecánico **F**, después de

agregar al compuesto la sustancia **Q**, sellará el frasco sobre la báscula y lo apartará de la banda; de otro modo, el frasco permanecerá abierto y la banda lo transporta hacia otra etapa del proceso. Además, por las condiciones previas del proceso, no es posible que lleguen a la báscula ni frascos vacíos, ni frascos que contengan las siguientes sustancias: **WY**, **YZ**, **WX** y **WZ**; todas las demás combinaciones si pueden llegar hasta la báscula.

Determinar la función Booleana del circuito combinatorio *L* que acciona el dispositivo *F* que incluya las condiciones irrelevantes. Realizar el circuito mediante un *decodificador* y compuertas externas.

4. En la torre de control de un patio de ferrocarril, un controlador debe seleccionar la ruta de los furgones de carga que entran a una sección del patio, mismos que provienen del punto **A**, como puede verse en el tablero de control de la siguiente figura. Dependiendo de las posiciones de los conmutadores, un furgón puede llegar a uno cualesquiera de los cuatro destinos. Otros furgones pueden llegar desde los puntos **B** o **C**.



Diseñe un circuito, con ***multiplexores***, que reciba como entradas las señales $\mathbf{S_1}$ a $\mathbf{S_5}$, indicadores de las posiciones de los conmutadores correspondientes, y que encienda una lámpara $\mathbf{D_0}$ a $\mathbf{D_3}$, indicando el destino al que llegará el furgón proveniente de ***A***.

*Para los casos en que los furgones puedan entrar de **B** o **C** ($\mathbf{S_2}$ o $\mathbf{S_3}$ en la posición **0**), todas las lámparas de salida deben encenderse, indicando que un furgón proveniente de **A**, no puede llegar con seguridad a su destino.*

NOTA: $\mathbf{S_1}$ bit de mayor peso binario

5. Un circuito lógico tiene **5** entradas **A**, **B**, **C**, **D**, **E** (donde **A** es la de mayor peso binario). Cuatro de las entradas representan un dígito decimal en **BCD** (decimal codificado en binario). ***La primera entrada es de control***.

Cuando el control está es **0** lógico, la salida **Z** es igual a **0** si el número decimal es **impar** y **1** si es **par**.

Cuando el control está en **1** lógico, la salida **Z** es igual a **1** cuando la entrada es múltiplo de **3**, en caso contrario es **0**.

Diseñe un circuito utilizando un ***decodificador*** y compuertas externas, considerando lógica negativa.

NOTA: Considere al **0** como número par.

6. Un técnico de un laboratorio químico tiene **4** productos **A**, **B**, **C** y **D**. *Cada producto debe encontrarse en uno cualesquiera de dos recipientes de almacenamiento.*

Periódicamente, se requiere cambiar uno o más productos de un recipiente a otro. La naturaleza de los productos es tal, que es peligroso guardar **A** y **B** juntos a menos que **D** esté en el mismo recipiente. También es peligroso almacenar **B** y **C** juntos a menos que **D** esté presente.

Obtener el circuito de la expresión de una variable **Z** que deberá tener el valor de **0** para cada situación peligrosa de almacenamiento, utilizando un ***multiplexor*.**

NOTA: Considere a **A** como la variable de mayor peso binario.

7. Un codificador de posición de eje, proporciona una señal de 4 bits que indica la posición de un eje en pasos de 30°. Utilizando el código de Gray, el cual se muestra a continuación, diseñe un circuito que produzca una salida que indique en dónde se encuentra el eje.

POSICIÓN DEL EJE	SALIDA DEL DECODIFICADOR
0°# P #30°	0 0 1 1
30°< P #60°	0 0 1 0
60°< P #90°	0 1 1 0
90°< P #120°	0 1 1 1
120°< P #150°	0 1 0 1
150°< P #180°	0 1 0 0

POSICIÓN DEL EJE	SALIDA DEL DECODIFICADOR
180°< P #210°	1 1 0 0
210°< P #240°	1 1 0 1
240°< P #270°	1 1 1 1
270°< P #300°	1 1 1 0
300°< P #330°	1 0 1 0
330°< P #360°	1 0 1 1

Obtenga el logigrama utilizando un ***decodificador*** y compuertas externas, considerando lógica negativa.

8. Haciendo referencia al código ***flexowriter*** (código de 6 bits), diseñe un circuito utilizando un ***multiplexor*** que emita una señal si se recibe un código que no sea alguno de los **36** códigos alfanuméricos que se enumeran a continuación:

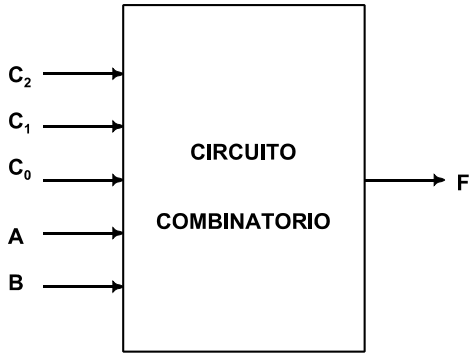
CARÁCTER	CÓDIGO OCTAL
A	30
B	23
C	16
D	22
E	20
F	26
G	13
H	05
I	14
J	32
L	26
L	11

CARÁCTER	CÓDIGO OCTAL
M	07
N	06
O	03
P	15
Q	35
R	12
S	24
T	01
U	34
V	17
W	31
X	27

CARÁCTER	CÓDIGO OCTAL
Y	25
Z	21
0	37
1	52
2	74
3	70
4	64
5	62
6	66
7	72
8	60
9	33

9. Obtener el diagrama lógico, por medio de un ***decodificador*** y compuertas externas, de un circuito de cinco entradas: Dos de datos **A** y **B** y tres de control $\mathbf{C_2}$, $\mathbf{C_1}$ y $\mathbf{C_0}$**.**

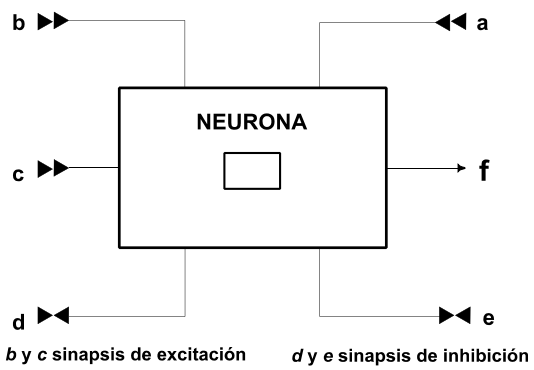
La función de salida **F** depende de los ocho posibles estados de las señales de control, de acuerdo a la siguiente tabla:



CONTROL (DECIMAL)	F
0	**1**
1	$A + B$
2	$\overline{A \cdot B}$
3	$A \oplus B$
4	$\overline{A \oplus B}$
5	$A \cdot B$
6	$\overline{A + B}$
7	**0**

NOTA: Considere a $\mathbf{C_2}$ y **A** como las variables de mayor y menor peso binario, respectivamente.

10. El sistema nervioso humano, incluyendo el cerebro, está hecho de células especializadas llamadas neuronas. Cada neurona tiene *sinapsis* (puntos de interconexión, como se muestra en la figura adjunta) de ***excitación*** y *sinapsis* de ***inhibición***. Una neurona produce una salida <**1**> si el número de sinapsis de ***excitación*** con pulsos <**1**> excede el número de sinapsis de ***inhibición*** con pulsos <**1**> por al menos el valor del umbral de la neurona.



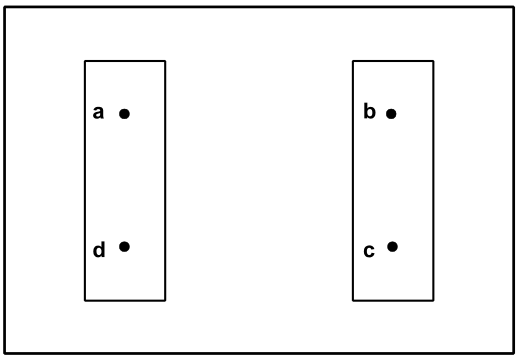
Determine la función booleana **f(a, b, c, d, e)** de emisión de pulsos a través del *canal de salida* (axón) en el modelo de la figura bajo las siguientes condiciones**:**

($\mathbf{C_1}$) ***Valor de umbral = 1*** [es decir, se produce una salida <**1**> si el número de sinapsis de *excitación* con pulsos <**1**>, **excede por al menos uno** el número de sinapsis de *inhibición* con pulsos <**1**>, y

($\mathbf{C_2}$) Siempre que haya al menos un pulso <**1**> en alguna sinapsis del puerto de *excitación*, habrá al menos un pulso <**1**> en alguna sinapsis del puerto de *inhibición* [es decir, no es posible -en este modelo restringido- que existan pulsos <**1**> en el puerto de *excitación* si no existe al menos un pulso <**1**> en el puerto de *inhibición*].

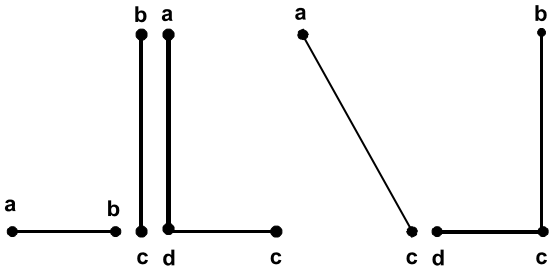
Obtenga **f(a, b, c, d, e)**, que incluya las condiciones irrelevantes **($\mathbf{C_2}$)**. Realizar el logigrama utilizando un ***multiplexor***.

11. Textura es la organización de una superficie como un conjunto de elementos repetidos. En un proceso automático para clasificar texturas artificiales, un sensor de 4 puntos (figura anexa) envía señales a un circuito combinatorio cuya tarea es discriminar (emitiendo pulsos <**1**>) los siguientes elementos:

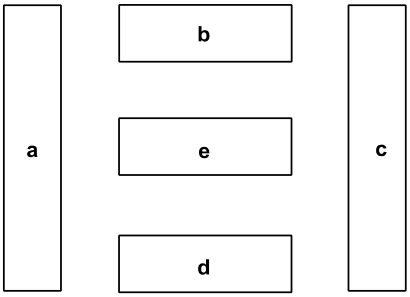


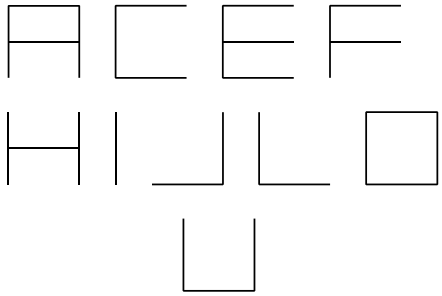
En todos los casos que inspecciona el sensor se activan al menos dos puntos de la rejilla (es decir, no se presentan casos en los cuales se activa tan solo un punto ni casos en los que no se activa ningún punto).

Obtener la función booleana ***f(a, b, c, d)*** a la salida del circuito discriminador haciendo uso de condiciones irrelevantes. Realizar el circuito mediante un ***decodificador*** y compuertas externas.



12. En un fábrica un dispositivo con 5 fotoceldas (figura anexa), registra los caracteres formados abriendo pequeñas ranuras en una tarjeta de control. Si en la tarjeta registrada hay uno de los símbolos:





(Para el símbolo **I** son válidas las dos posiciones) entonces el dispositivo acciona un taladro.

En el proceso no hay tarjetas con ninguno de los caracteres adjuntos:

(Todos los caracteres restantes sí entran en juego)

¿Cuál es la **función booleana** a la salida del dispositivo que acciona el taladro? Obtener la función y realizar el logigrama utilizando un ***multiplexor***.

13. Se desea diseñar e instrumentar un circuito combinatorio de dos entradas con dos bits cada una, sobre las cuales se codifican dos de los cuatro tipos de sangre existentes y a su salida se obtenga una señal que informe sobre la posibilidad o

imposibilidad de la transfusión de uno de ellos sobre el otro, dadas las siguientes reglas de compatibilidad entre ellos.

Los tipos de sangre son **4**: ***A***, ***B***, ***AB*** y ***O***.

El tipo ***O*** puede donar a cualquier otro tipo, pero sólo puede recibir de él mismo.

El tipo ***AB*** puede recibir de cualquier otro tipo pero sólo puede donar a ***AB***.

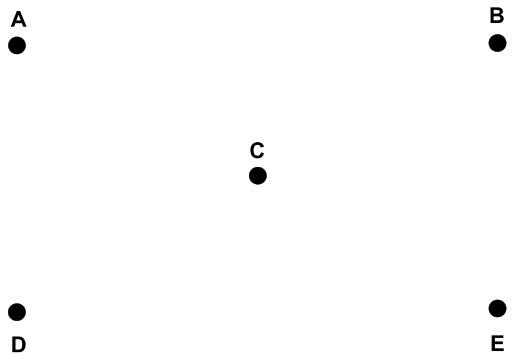
La clase ***A*** puede donar a ***A*** o ***AB*** y recibir de ***A*** u ***O*** únicamente.

Por último el tipo ***B*** puede donar al mismo ***B*** o al tipo ***AB*** y recibir de ***B*** u ***O***.

La señal de salida deberá ser **1** cuando la transfusión propuesta en las entradas sea permitida.

Realizar el logigrama utilizando un **decodificador** y compuertas externas.

14. En un sistema de detección luminosa que tiene el arreglo mostrado en la figura adjunta, se genera una señal de salida con valor de **1** únicamente cuando dos fotoceldas adyacentes están activadas, siempre y cuando la fotocelda del centro esté también activada.



NOTA: No es posible en este sistema que exista señal de salida **0** o **1** si no hay menos de tres fotoceldas activadas.

Considerando a **A** como la variable más significativa y obtenga la función de salida que incluya las condiciones indiferentes, y realice el logigrama utilizando un ***multiplexor***.

15. Un robot de juguete -*llamado* ***U-2***- está diseñado para ser capaz de seguir una trayectoria (previamente programada por medio de controles que el robot tiene en la espalda) avanzando cuadro por cuadro en una área de 5x6 cuadros. El robot *U-2* puede realizar una de las cuatro acciones siguientes:

(D) Girar (sobre su eje vertical) **90°** a la derecha y luego avanzar al centro del siguiente cuadro si su pequeño cerebro recibe la señal binaria **01**.

(I) Girar **90°** a la izquierda y luego avanzar al centro del siguiente cuadro si su diminuto cerebro percibe la señal binaria **10**.

(F) Avanzar al frente un cuadro si su cerebro recibe la señal **00**.

(A) Hacer alto si su cerebro recibe la señal **11**.

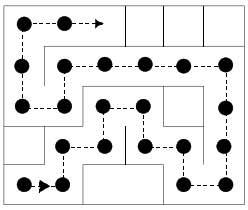


Figura (a)

0	3	4	15	16	29
1	2	5	14	17	28
8	7	6	13	18	27
9	10	11	12	19	26
20	21	22	23	24	25

Figura (b)

Programar el robot para que recorra el laberinto de la *figura (a)*. Determinar las funciones booleanas del par de estímulos binarios que recibe el

minicerebro del robot durante este recorrido y realizarlas mediante un ***decodificador*** y compuertas externas. (*En este problema hay condiciones irrelevantes -parte de la solución consiste en identificarlas*).

Los controles en la espalda del *U-2* están localizados en tres áreas: En el área **-I** se indicará el cuadro inicial mediante los controles de dos posiciones **a**, **b**, **c**, **d** y **e** [como se muestra en la *figura (c)*]; si el control a se presiona del lado derecho, el peso de la variable a se contabilizará para determinar el número asignado al cuadro inicial (lo mismo ocurrirá para el resto de las variables). En el área **-II** se programa la trayectoria por medio de 30 controles de tres posiciones cada uno:

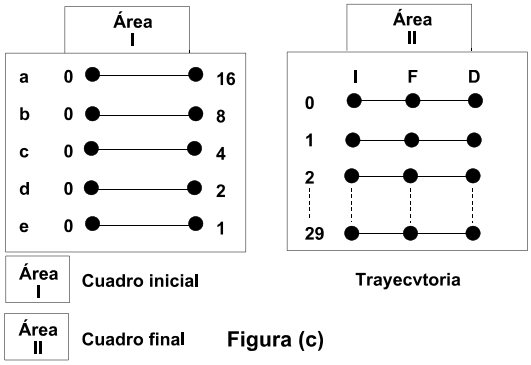


Figura (c)

16. Obtener el diagrama lógico de un sumador completo de dos variables **A** y **B**, usando: un ***decodificador*** y compuertas externas.

17. Partiendo del código **BCD** de 4 bits, diseñe un circuito combinacional que genere el código **exceso** en **4**, utilizando ***multiplexores***.

18. Realice un circuito convertidor de código de *GRAY* a *BINARIO* para 4 bits, por medio de un ***decodificador*** y compuertas externas.

19. Realice los siguientes conversores de código, empleando ***multiplexores***:

a) De **BCD** a **8 4-2-1**.

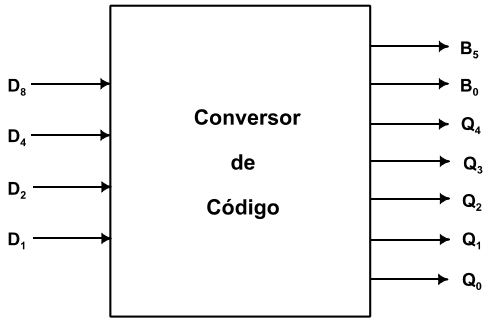
b) De **BCD** a **2 4 2 1**.

BCD	8 4 -2 -1				2 4 2 1			
	A	B	C	D	W	X	Y	Z
0	0	0	0	0	0	0	0	0
1	0	1	1	1	0	0	0	1
2	0	1	1	0	0	0	1	0
3	0	1	0	1	0	0	1	1
4	0	1	0	0	0	1	0	0
5	1	0	1	1	1	0	1	1
6	1	0	1	0	1	1	0	0
7	1	0	0	1	1	1	0	1
8	1	0	0	0	1	1	1	0
9	1	1	1	1	1	1	1	1
10 | 15	X | X	X | X	X | X	X | X	X | X	X | X	X | X	X | X

NOTA: Considere los *términos indiferentes*, si se requieren

20. Obtenga el diagrama lógico con un ***decodificador*** y compuertas externas, del conversor de código de ***exceso 3*** (BCD), a un código *BCD* ***autocomplementario***, cuyas combinaciones *0,1,2,3* y *4* están excedidas en **2** y las restantes tienen un exceso en **4**.

21. Diseñe un circuito, mediante un **decodificador** y compuertas externas, para convertir una entrada *decimal codificada en binario* (BCD) a una salida *biquinaria* (2 de 7). Como se indica en la figura adjunta, deberá contar con 4 entradas y siete salidas. Los códigos para la entrada y la salida correspondientes a los dígitos decimales se dan en la tablas siguiente. Se puede suponer que las seis combinaciones posibles de entrada no anotadas en ella (correspondientes a 10-15) nunca se podrán producir.



DÍGITO	B C D				BIQUINARIO						
	D_8	D_4	D_2	D_1	B_5	B_0	Q_4	Q_3	Q_2	Q_1	Q_0
0	0	0	0	0	0	1	0	0	0	0	1
1	0	0	0	1	0	1	0	0	0	1	0
2	0	0	1	0	0	1	0	0	1	0	0
3	0	0	1	1	0	1	0	1	0	0	0
4	0	1	0	0	0	1	1	0	0	0	0
5	0	1	0	1	1	0	0	0	0	0	1
6	0	1	1	0	1	0	0	0	0	1	0
7	0	1	1	1	1	0	0	0	1	0	0
8	1	0	0	0	1	0	0	1	0	0	0
9	1	0	0	1	1	0	1	0	0	0	0