2.- Restadores : La sustracción de dos números binarios puede realizarse a través del complemento de un número.

Implementemos circuitos que realizan la resta en forma directa.

a) SemiRestadores

Sea :

X el minuendo
Y el Sustraendo
D la Diferencia
B el Préstamo

X	Y	B	D
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	0

$D = \overline{X}Y + X\overline{Y}$
$B = \overline{X}Y$
} Implementar el circuito

b) **Restador Total:** Circuito combinacional que realiza la substracción entre 2 bit tomando en cuenta que un 1 puede ser prestado a una etapa menos significativa.

X = Minuendo
Y = Sustraendo
Z = Préstamo a una etapa menos significativa
B = Préstamo solicitado a una próxima etapa

X	Y	Z	B	D
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Z \ XY	00	01	11	10
0		1		1
1	1		1	

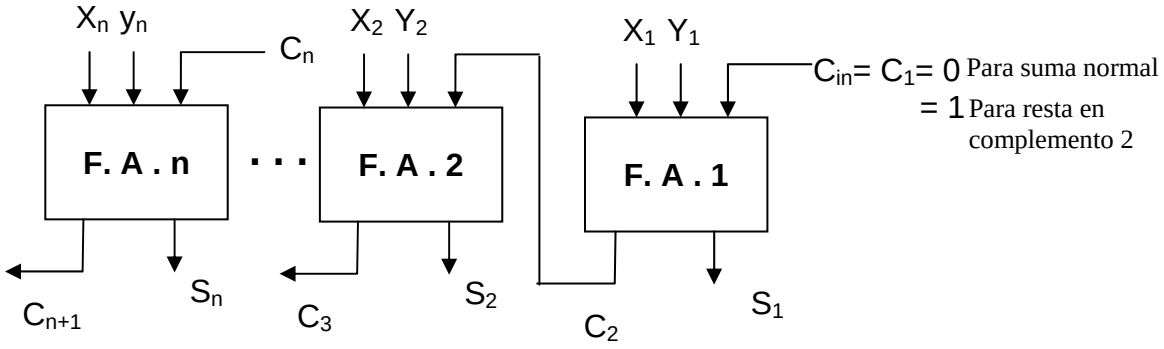
$D = \overline{X}\overline{Y}Z + \overline{X}Y\overline{Z} + X\overline{Y}\overline{Z} + XYZ$

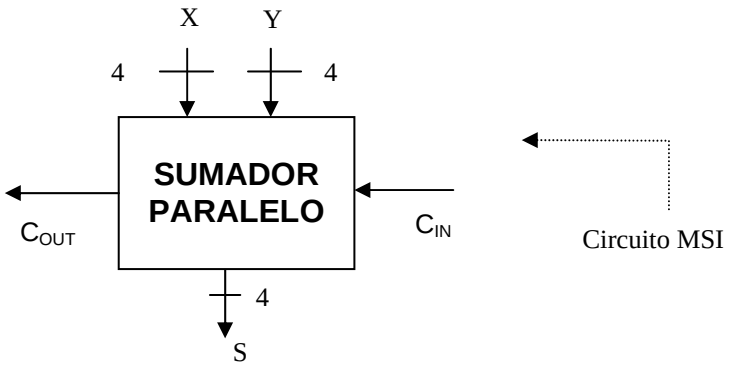
Z \ XY	00	01	11	10
0		1		
1	1	1	1	

$B = \overline{X}Y + \overline{X}Z + YZ$

El sumador completo (F.A.) permite sumar 3 dígitos binarios, así, para sumar dos números binarios de n bit, es necesarios conectar F.A de modo que cada uno realice la suma correspondiente a cada bit de los sumadores.

Para lograr rapidez en el cálculo los F.A se conectan en paralelo de modo que todos los bit de los sumadores ingresan al mismo instante. No obstante la suma de 2 bit requiere tener el carry de la suma anterior. La estructura es la siguiente:





Ej: TTL 74283

El tiempo requerido para obtener la suma completa será el tiempo requerido para la propagación de los carries a los estados siguientes. Así, la suma en el $F.A_i$ no reproducirá el resultado correcto hasta que el C_i no haya sido calculado. El esquema anterior recibe el nombre de ***"Ripple- Carry Adder"***.

"Carry Look a Head Adder" : Es un sumador de tiempo fijo, los acarreos son generados en paralelo, lo cual brinda una mayor repidez de cálculo.

Sabemos que $C_{i+1} = (X_i \oplus y_i)C_i + X_i y_i$

Si definimos:

$P_i = X_i \oplus Y_i$ $\Longrightarrow$ Se propaga el carry

y

$G_i = X_i\, y_i$ $\Longrightarrow$ Se genera el carry

$C_{i+1} = G_i + P_i C_i$

Donde G_i y P_i son señales de acarreo generado y propagado para el estado i-ésimo.

$G_i = 1$ Si el acarreo es generado en el i-ésimo estado es decir si $x_i = y_i = 1$
$P_i = 1$ si x_i ó y_i valen 1 (pero no ambos)

Si $P_i = 1$ y $C_i = 1$ => $C_{i+1} = 1$, esto es, el acarreo del estado i-1 se propaga ininterrumpidamente hacia el estado i+1 a través del estado i

Para generar los acarreos en paralelo es necesario transformar la función del acarreo a una forma no recursiva ,así:

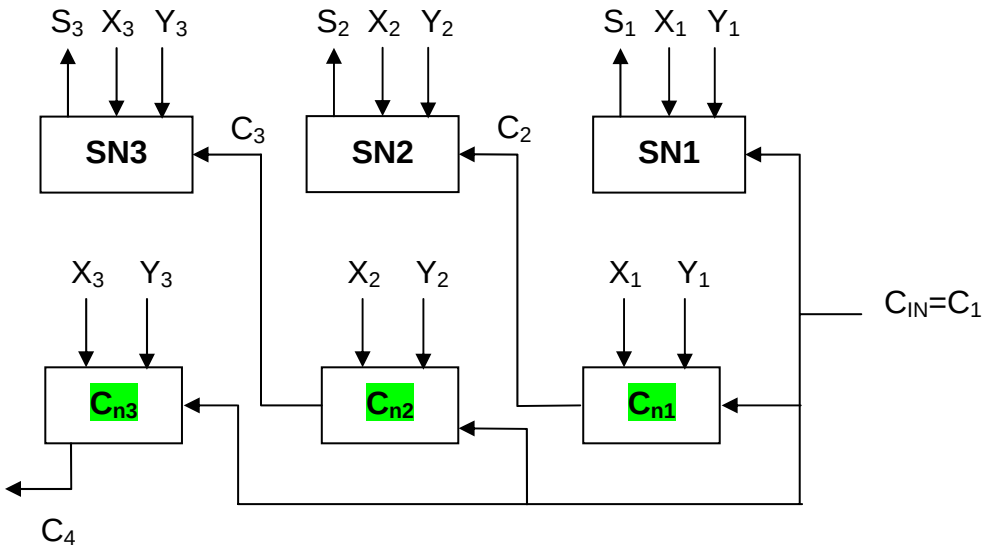
$C_{i+1} = G_i + P_i C_i$ con $1 \leq i \leq n$
$C_i = G_{i-1} + P_{i-1}C_{i-1}$
=> $C_{i+1} = G_i + P_i(G_{i-1} + P_{i-1}C_{i-1})$
$= G_i + P_i G_{i-1} + P_i P_{i-1}(G_{i-2} + P_{i-2}C_{i-2})$

$C_2 = G_1 + P_1 C_1$
$C_3 = G_2 + P_2 C_2 = G_2 + P_2(G_1 + P_1 C_1)$
$C_4 = G_3 + P_3 C_3 = G_3 + P_3\,(G_2 + P_2 G_1 + P_2 P_1 C_1) = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 C_1$

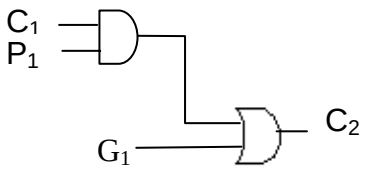
$$C_{i+1} = G_i + P_i G_{i-1} + P_i P_{i-1} G_{i-2} + \ldots + P_i P_{i-1} P_{i-2} \ldots P_1 C_1$$

Esta ecuación define el acarreo generado por el estado i y C_{i+1} = 1 si el carry ha sido generado en el estado i(G_i) ó si ha sido originado en un estado precedente y propagado por los estados subsecuentes.

Un esquema para 3 bit será:

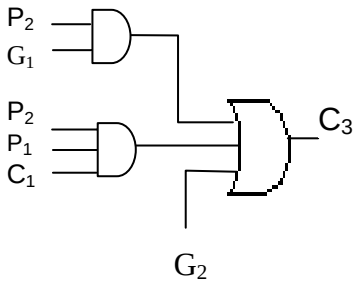


CN1



$C_{N1} = C_2 = G_1 + P_1C_1$

CN2

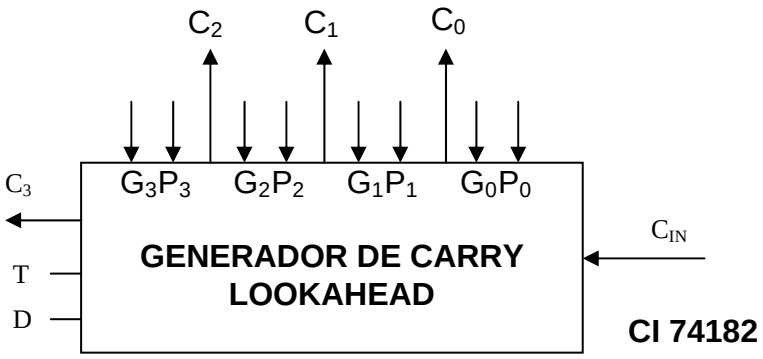


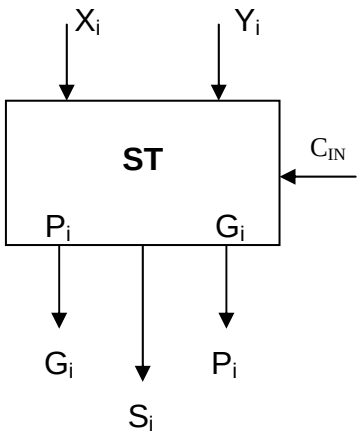
$C_{N2} = C_3 = G_2 + P_2G_1 + P_2P_1C_1$

$C_4 = G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1C_1$, etc.

La realización anterior es muy poco práctica a nivel de compuertas por cuanto se requiere un gran número de estas con un gran número de entradas. Estas limitaciones pueden ser resueltas mediante integración a mayor escala.

Cuando se requiere sumar números de muchos bit (Por ej. 32 ó 64 en un computador), se divide el esquema Lookahead en grupos y se propaga el carry de un grupo a otro => disminución de la rapidez de cálculo.

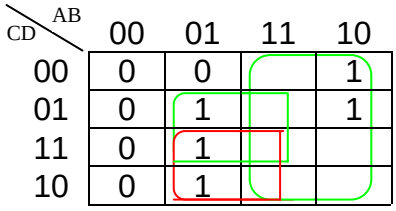




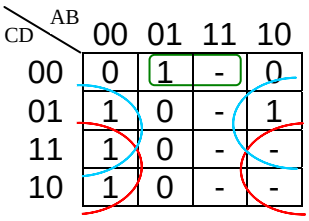
3.- Conversión de Códigos **:** Cuando el código de salida de un sistema difiere del código usado como entrada a otro sistema, un circuito conversor de código debe ser insertado entre ambos.

BCD - EXCESO 3

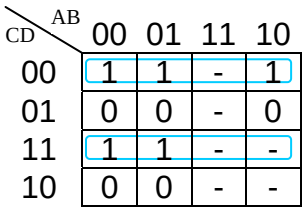
Entrada				Salida			
A	**B**	**C**	**D**	**W**	**X**	**Y**	**Z**
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0



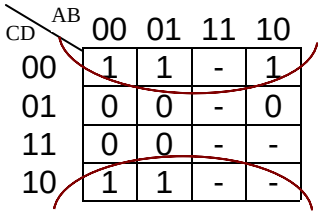
$W = A + BD + BC$



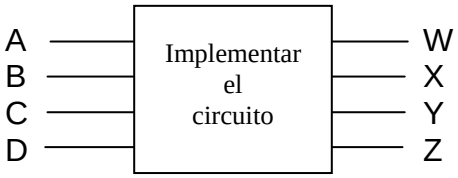
$X = B\bar{C}\bar{D} + \bar{B}C + \bar{B}D$



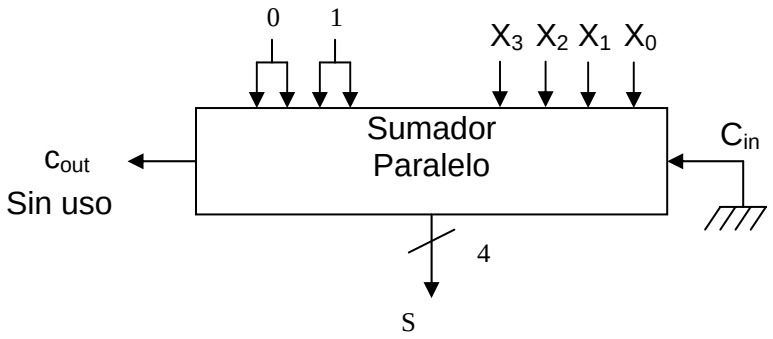
$Y = \bar{C}\bar{D} + CD$



$Z = \bar{D}$



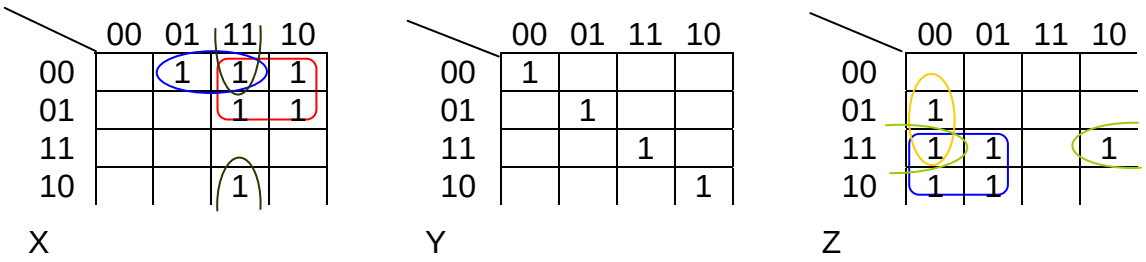
A nivel de MSI



4.- <u>Comparadores</u> **:** Un comparador es un circuito que compara dos números A y B y determina sus magnitudes relativas. El resultado es entregado en 3 salidas que indican si A > B , A = B ó A < B

Ej: Considere los números A y B de 2 bits

Entradas				Salidas		
A		B		A > B	A = B	A < B
A_1	A_0	B_1	B_0	X	Y	Z
0	0	0	0	0	1	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	0	1
0	1	0	0	1	0	0
0	1	0	1	0	1	0
0	1	1	0	0	0	1
0	1	1	1	0	0	1
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	0	1	0
1	0	1	1	0	0	1
1	1	0	0	1	0	0
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	0	1	0



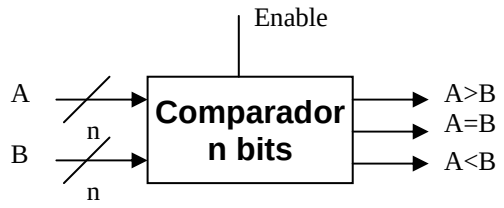
$X = A_1\bar{B}_1 + \bar{A}_0\bar{B}_1\bar{B}_0 + A_1\bar{A}_0\bar{B}_0$

$Y = \Sigma\ (0, 5, 10, 15)$

$Z = \bar{A}_1B_1 + \bar{A}_1\bar{A}_0B_O + \bar{A}_0B_1B_0$

} Implementar el circuito

A nivel de MSI



La tabla de verdad para la comparación de números binarios para más de 3 dígitos es inmanejable, pues requiere 2^{2n} casilleros, para esto se manejan normalmente métodos computacionales. Sin embargo, los circuitos que poseen una cierta simetría, pueden ser algunas veces diseñados por medio de un algoritmo.

Ej: Considere 2 números A(A_2, A_1, A_0) y B(B_2, B_1, B_0) con 3 dígitos cada uno.

1) Son Iguales si A2=B2 , B1=A1, A0=B0

$$\Rightarrow (A=B) = \underbrace{(A_2B_2 + \bar{A}_2\bar{B}_2)}_{X_2}\ \underbrace{(A_1B_1 + \bar{A}_1\bar{B}_1)}_{X_1}\ \underbrace{(A_0B_0 + \bar{A}_0\bar{B}_0)}_{X_0}$$

2) Para determinar si A > B ó A < B, se inspeccionarán las magnitudes por pares comenzando por los bits más significativos. Si dos bits son iguales se comparan los próximos dígitos y se continúa hasta que 2 bit no sean iguales. En ese momento si A_x es mayor que B_x entonces A > B y viceversa.

$$(A>B) = A_2\bar{B}_2 + A_1\bar{B}_1\underbrace{(A_2B_2 + \bar{A}_2\bar{B}_2)}_{X_2} + A_0\bar{B}_0(\underbrace{A_2B_2 + \bar{A}_2\bar{B}_2}_{X_2})(\underbrace{A_1B_1 + \bar{A}_1\bar{B}_1}_{X_1})$$

$$(A>B) = \bar{A}_2B_2 + \bar{A}_1B_1\underbrace{(A_2B_2 + \bar{A}_2\bar{B}_2)}_{X_2} + \bar{A}_0B_0(\underbrace{A_2B_2 + \bar{A}_2\bar{B}_2}_{X_2})(\underbrace{A_1B_1 + \bar{A}_1\bar{B}_1}_{X_1})$$

Tarea. Implementar el Circuito

5.- <u>Decodificadores y codificadores</u> **:** Considérese un código binario de n bits capaz de representar $m \leq 2^n$ palabras de información.

Un **Decodificador** es un circuito combinacional que convierte un código binario de **n bits** (variables) a **m** líneas de salidas una por cada palabra de información.

Un **Codificador** es un circuito combinacional que acepta **m** entradas una para cada palabra de información y genera un código binario de **n bits** (salidas)

Ej. <u>Decodificador de Binario a Octal</u>

Entradas			Salidas							
X	**Y**	**Z**	**D_0**	**D_1**	**D_2**	**D_3**	**D_4**	**D_5**	**D_6**	**D_7**
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

$D_0 = \bar{X}\,\bar{Y}\,\bar{Z}$
$D_1 = \bar{X}\,\bar{Y}\,Z$
$D_2 = \bar{X}\,Y\,\bar{Z}$
$D_3 = \bar{X}\,Y\,Z$
$D_4 = X\,\bar{Y}\,\bar{Z}$
$D_5 = X\,\bar{Y}\,Z$
$D_6 = X\,Y\,\bar{Z}$
$D_7 = X\,Y\,Z$

Implementar el circuito.

<u>Codificador de Octal a Binario</u>

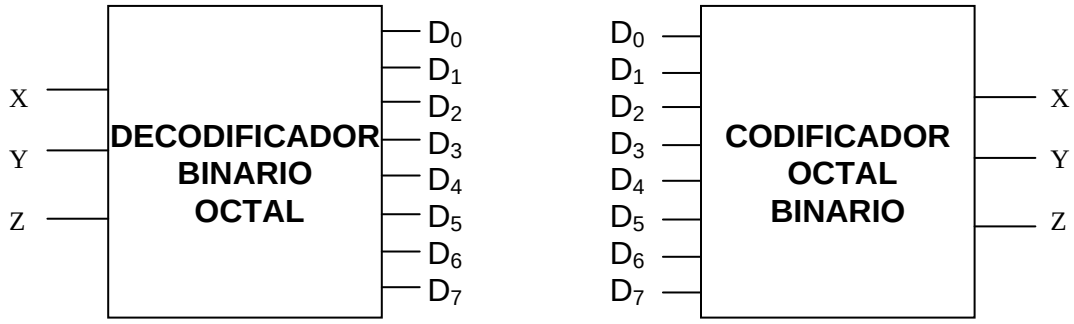
Entradas								Salidas		
D_0	**D_1**	**D_2**	**D_3**	**D_4**	**D_5**	**D_6**	**D_7**	**X**	**Y**	**Z**
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

$X = D_4 + D_5 + D_6 + D_7$

$Y = D_2 + D_3 + D_6 + D_7$ } Implementar Circuito

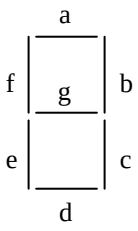
$Z = D_1 + D_3 + D_5 + D_7$

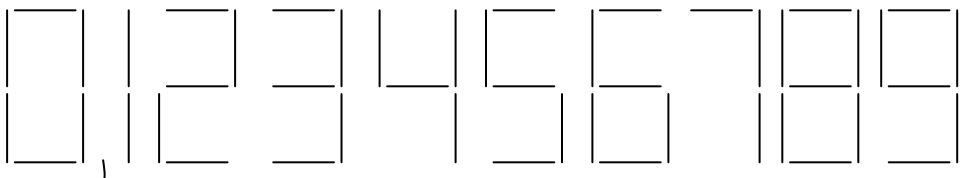
Luego



Decodificador BCD a 7 segmentos

Un decodificador de BCD a siete segmentos (Seven-segments) es un circuito combinacional que acepta palabras decimales en BCD y genera salidas apropiadas para la selección de segmentos en un sistema constituido por led ó cristales usados para dibujar el decimal.



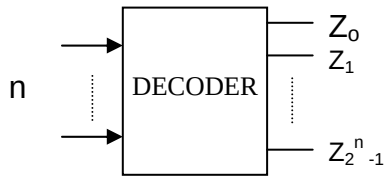


	A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	0	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	0	0	1	1

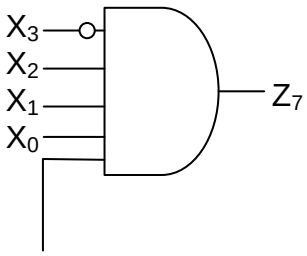
Tarea

1) Implementar el circuito
2) Diseñe el BCD - 7 segmentos usando todas las condiciones superfluas
3) Muestre los números generados cuando las condiciones no permitidas de entradas ocurren.

En general con tecnología MSI



Ej: Decodificador para el binario 7



El decodificador BCD-7 segmentos sólo sirve como ejemplo demostrativo de su diseño, pues se puede encontrar un decodificador 7 segmentos integrado por ej. El Decoder BCD 7448 con el display FND-500

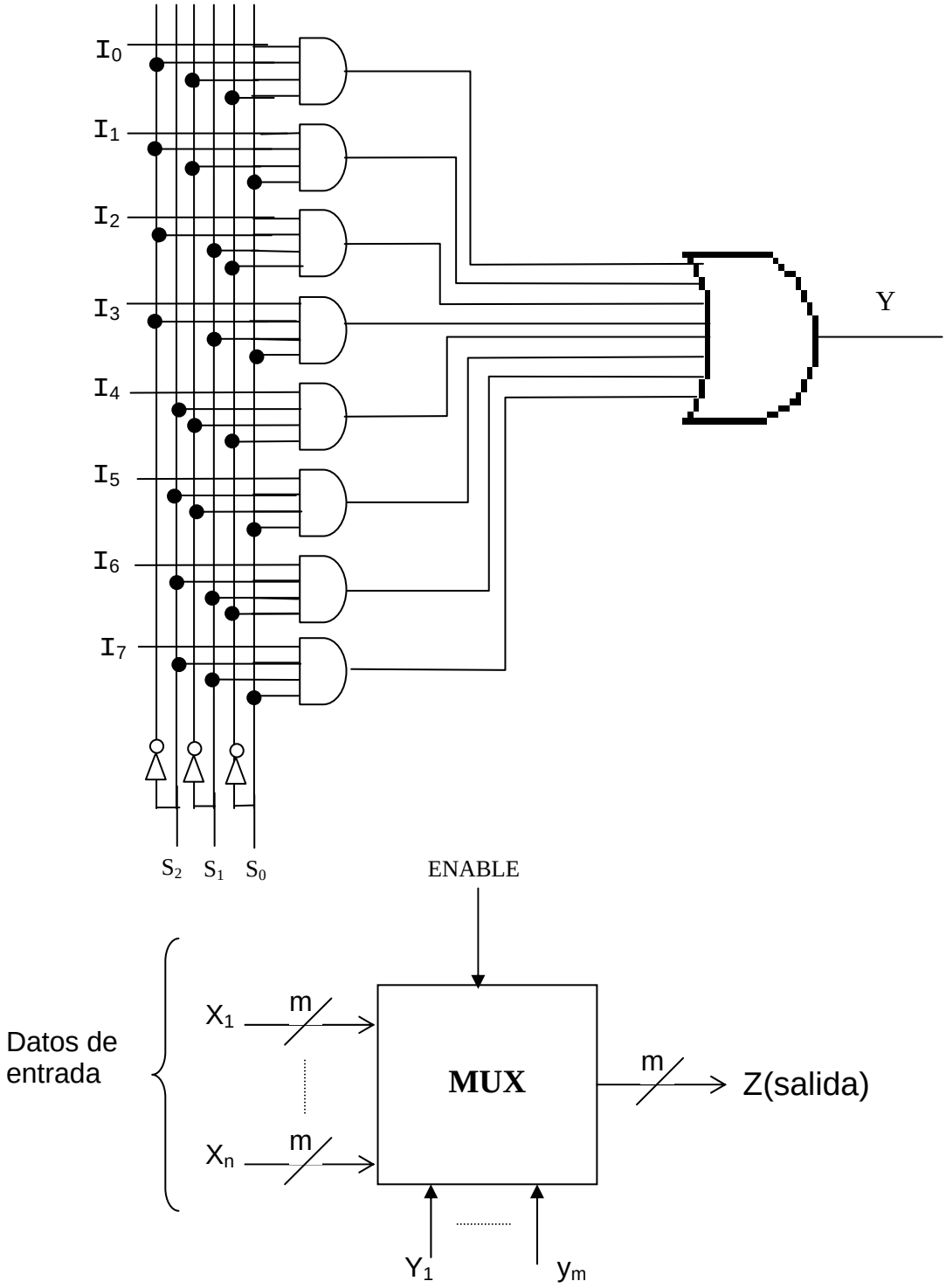
6.- <u>Multiplexores y Demultiplexores</u> **:** Multiplexer significa enviar un gran número de unidades de información a través de un número pequeño de líneas ó canales.

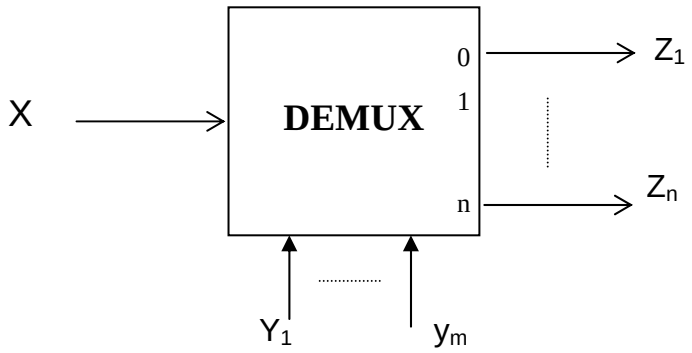
Demultiplexing es la operación contraria, consiste en recibir información en un pequeño número de canales y distribuirlas sobre un gran número de destinatarios.

Un mux digital es un circuito que selecciona información desde 2^n líneas de entrada y las dirige a una sola línea de salida. La selección es controlada por un conjunto de líneas de selección de entrada.

Ejemplo: Mux de 8 entradas(I_0 a I_7) y un canal de salida. Las líneas de selección son S_2 , S_1 , S_0. La función de salida en álgebra de boole de un mux de 8 entradas sería:

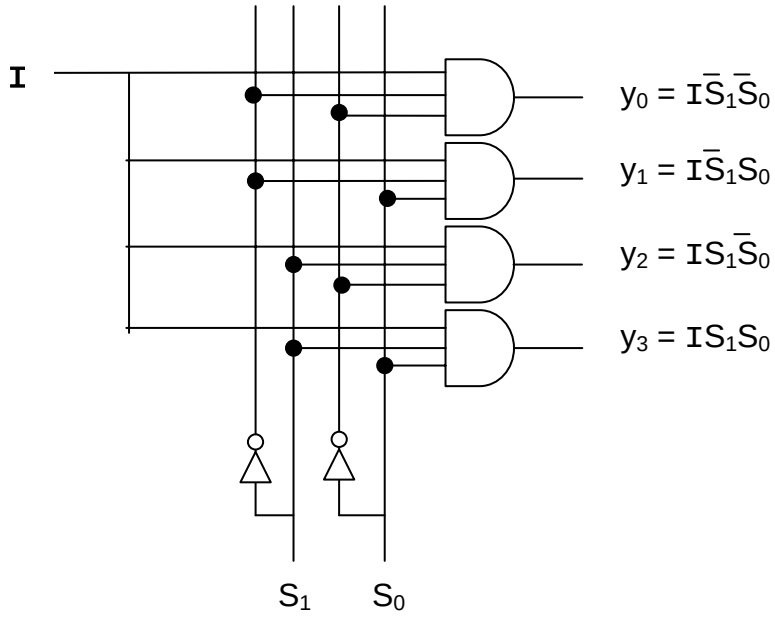
$$Y = I_0\bar{S}_2\bar{S}_1\bar{S}_0 + I_1\bar{S}_2\bar{S}_1S_0 + I_2\bar{S}_2S_1\bar{S}_0 + I_3\bar{S}_2S_1S_0 + I_4S_2\bar{S}_1\bar{S}_0 + I_5S_2\bar{S}_1S_0 + I_6S_2S_1\bar{S}_0 + I_7S_2S_1S_0$$





Líneas de selección

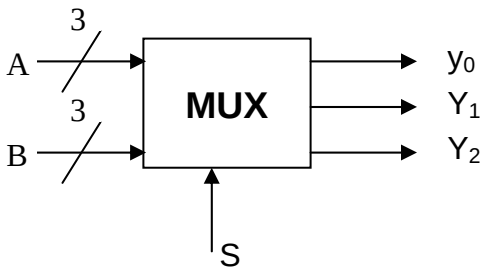
Ej.: Demux para 4 salidas



Ej.: Otro tipo de Mux es aquel que selecciona una de dos entradas de información A y B cada una de las cuales consiste de 3 bits de información. Una línea de selección determinaa que entradas, A ó B es aplicada a las salida.

Las funciones de salidas son :

$$Y_0 = A_0\bar{S} + B_0S$$
$$Y_1 = A_1\bar{S} + B_1S$$
$$Y_2 = A_2\bar{S} + B_2S$$



Mux digital de dos entradas de tres bits

7. ***Funciones OR-EX y equivalencia*** :

OR-EX $\oplus$
Equivalencia $\odot$

Se definen por las siguientes funciones

$X \oplus Y = X\bar{Y} + \bar{Y}X$
$X \odot Y = XY + \bar{X}\bar{Y}$

Una es el complemento de la otra

Ambas son asociativas y conmutativas, debido a ello las funciones de varias variables pueden expresarse sin paréntesis.

$$(A \oplus B) \oplus C = A \oplus (B \oplus C) = A \oplus B \oplus C$$

por lo que existe la posibilidad de usar compuertas con varias entradas, sin embargo las entradas múltiples son antieconómicas desde el punto de vista "Hardware".

Estas funciones son especialmente útiles en operaciones aritméticas y detección y corrección de errores.

Una expresión OR-EX de n variables es igual a una función booleana con $2^n / 2$ mintérminos cuyos números binarios equivalentes tienen un número impar de unos.

Ej.: Para 4 variables => 8 mintérminos

$$A \oplus B \oplus C \oplus D = (A\bar{B} + \bar{A}B) \oplus (C\bar{D} + D\bar{C})$$

$$= \sum m\,(1, 2, 4, 7, 8, 11, 13, 14)$$

CD \ AB	00	01	11	10
00		1		1
01	1		1	
11		1		1
10	1		1	

$F = A \oplus B \oplus C \oplus D$

CD \ AB	00	01	11	10
00	1		1	
01		1		1
11	1		1	
10		1		1

$F = A \odot B \odot C \odot D$

C \ AB	00	01	11	10
0		1		1
1	1		1	

$F = A \oplus B \oplus C$

$F = A \odot B \odot C$

C \ AB	00	01	11	10
0	1		1	
1		1		1

$\overline{A \oplus B \oplus C} = A \oplus B \odot C$

$\overline{A \odot B \odot C} = A \odot B \oplus C$

$F = A \oplus B \odot C = A \odot B \oplus C$

8.- <u>Generador y Detector de Paridad</u>

Un circuito que genera el bit de paridad en el transmisor se llama *Generador de Paridad* y el circuito que detecta la paridad se llama *Detector de Paridad.*

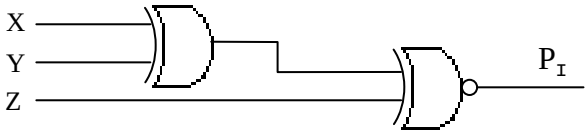
Ej.: Considere un mensaje de 3 bit que desea transmitir con un bit de paridad impar. La siguiente es la tabla de verdad para un generador de paridad impar.

Las X, Y, Z son entradas y P_I es salida.

X	Y	Z	P_I
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

Z \ XY	00	01	11	10
0	1		1	
1		1		1

$$P_I = X \oplus Y \odot Z = \overline{X \oplus Y \oplus Z}$$



Si ocurre un error, la paridad de los cuatro bits debe ser par. La salida **D** del detector de paridad impar debe ser igual a uno.

X	Y	Z	P_I	D
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

Z \ XY	00	01	11	10
00	1		1	
01		1		1
11	1		1	
10		1		1

$$C = X \odot Y \odot Z \odot P_I$$

NOTA : Implementar el circuito

Ejecución de una Función de Boole Mediante Multiplexores(MSI)

Un mux es un decodificador con una puerta OR ya disponible y es posible configurar cualquier función de Boole de **n** variables con un multiplexor de **2^{n-1} a 1**

Procedimiento

1) Se expresa la función como una sumatoria de mintérminos (se asume la siguiente secuencia ABCD ...) A es la variable del extremo izquierdo y BCD ... son las (n-1) variables restantes.

2) Se conectan las n - 1 variables a las líneas de selección del mux con B conectada a la línea de selección de mayor orden y así sucesivamente hasta S_0. La variable A será complementada en los términos mínimos ó hasta $(2^n/2)-1$ los cuales comprenden la primera mitad. La segunda mitad tendrá la variable A sin complementar.

3) Listar las entradas del mux y bajo ellas los términos mínimos (en dos filas). La primera fila incluye todos los términos mínimos en los que A es complementada y la segunda los términos con A no complementada.

4) Encierre en un círculo todos los mintérminos de la función e inspeccione cada columna separadamente.

5) a) Si 2 mintérminos en una columna no están en círculo, aplíquese un cero a la entrada correspondiente del mux.
   b) Si 2 mintérminos están en círculo, aplíquese un uno a la entrada correspondiente del mux.
   c) Si el mintérmino inferior está encerrado y el superior no, aplíquese A a la entrada correspondiente del mux.
   d) Si el mintérmino superior está encerrado y el inferior no, aplíquese $\bar{A}$ a la entrada correspondiente del mux.

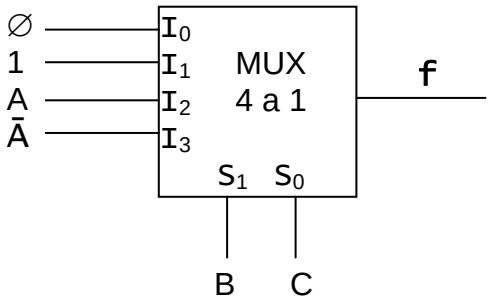
Ej.: Considere $F(A, B, C) = \sum m(1, 3, 5, 6)$

1) 3 variables => Mux de 2^{n-1} a 1, esto es 2^2 a 1 = 4 a 1

	I_0	I_1	I_2	I_3
$\bar{A}$	0	①	2	③
A	4	⑤	⑥	7
	∅	**1**	**A**	**$\bar{A}$**

A	B	C	f
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

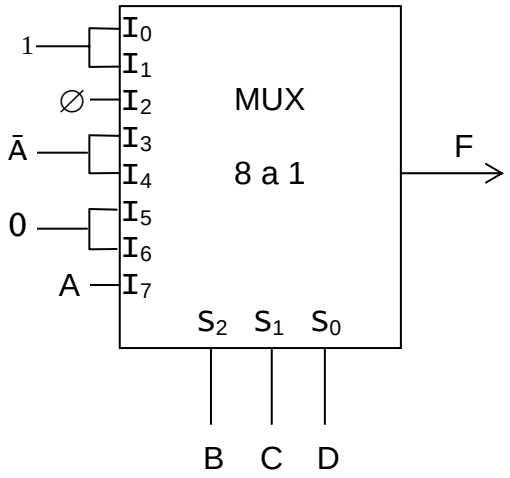
Luego:



2) F = (A, B, C, D) = ∑ (0, 1, 3, 4, 8, 9, 15)

Mux de 8 a 1

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



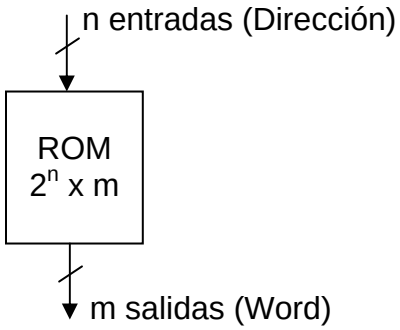
Memoria de Solo Lectura **ROM (Read Only Memory)**

Una ROM, es dispositivo de memoria en el cual se almacena un conjunto fijo de información binaria. Es un elemento que incluye un decodificador y un conjunto de compuertas OR dentro de una sola cápsula de CI.

Las conexiones entre las salidas del decodificador y las entradas de las compuertas OR pueden especificarse para cada configuración particular *"Programando"* la ROM. Una vez que se establezca la información para la ROM, esta permanecerá fija aunque haya un corte de energía a diferencia de una memoria volátil.

Las ROM se usan a menudo para configurar un circuito combinacional complejo en una cápsula de CI y así eliminar los cables de conexión.

Diagrama en Bloque de una ROM

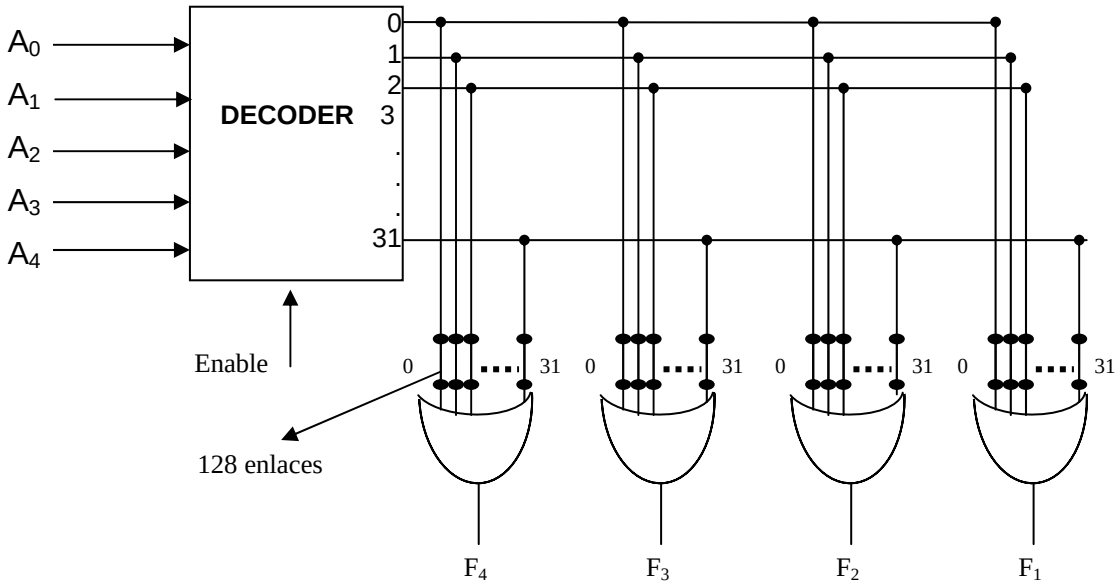


Cada combinación de bits de las variables de entrada se llama ***Dirección***. Cada combinación de las líneas de salida se llama ***Palabra.***

Ej.: Una Rom de 32 x 8 => 32 palabras de 8 bit direccionadas por 5 entradas

Construcción Lógica de una ROM

Ej.: Rom de 32 x 4 (32 palabras de 4 bits c/u)

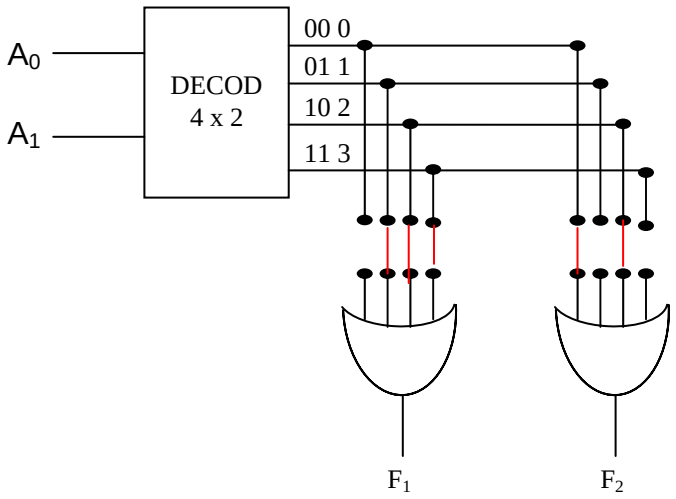


Observar que la salida de la Rom produce la suma de todos los mintérminos de n-variables de entrada. Al romperse los enlaces se puede obtener un circuito combinacional en particular.

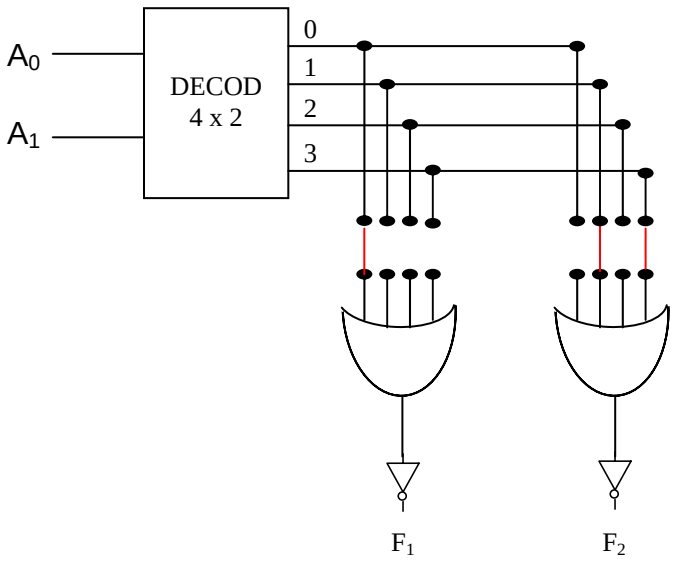
Ej.: $F_1 (A_1, A_0) = \sum m (1, 2, 3)$

$F_2 (A_1, A_0) = \sum m (0, 2)$

A_1	A_0	F_1	F_2
0	0	0	1
0	1	1	0
1	0	1	1
1	1	1	0



También



Tipos de ROM

- **ROM :** La programación la hace el fabricante.

- **PROM :** (ROM Programable) Los enlaces se rompen por medio de pulsos de corriente con instrumentos llamados programadores de PROM. Una vez realizada la programación el proceso irreversible.

- **EPROM :** (Erase PROM, Reprogramable) Su contenido puede ser cambiado borrando con una luz ultravioleta por un cierto periodo de tiempo, la radiación corta los puentes internos que sirven de contacto.

- **EAPROM :** (Eléctrica alterable PROM) Eprom eléctricamente alterable, pueden ser borradas con señales eléctricas