Tema 11: Sistemas combinacionales

Objetivo:

- Introducción
- Generador – Comprobador de paridad
- Comparadores
- Semisumador (HA)
- Sumador Completo (FA)
- Expansión de sumadores
  - –Sumador paralelo con arrastre serie
  - –Sumador con acarreo anticipado
- Semirestador (HB)
- Restador completo (FB)
- Circuito restador paralelo de cuatro bits por complemento a 2
- Codificadores
- Conversores de códigos.
- Decodificadores.
- Multiplexores

INTRODUCCIÓN

•Según su comportamiento los sistemas lógicos se clasifican en dos grandes grupos:

»Circuitos combinacionales.

»Circuitos secuenciales.

•Los **circuitos combinacionales** se caracterizan por generar unas salidas que son función, exclusivamente, del estado lógico actual de las entradas (por ejemplo sumador)

INTRODUCCIÓN

•En los **circuitos secuenciales**, el estado de la salida depende no sólo del estado lógico de las entradas, sino también de la historia pasada del circuito (por ejemplo Contador).

•Para el diseño de un sistema combinacional cualquiera seguiremos los siguientes pasos:

–Definición de la función a realizar y especificación de las entradas y de las salidas.

–Tabla de verdad

–Ecuaciones lógicas de las salidas.

–Simplificación de las ecuaciones lógicas de las salidas.

–Implementación de las funciones simplificadas.

Generador-comprobador de paridad

•El método más sencillo para detectar un error en un sistema digital es añadir al bloque de información un bit adicional llamado bit de paridad que puede ser de dos tipos:

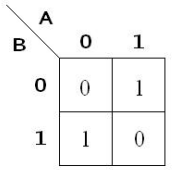
–**Paridad Par(Even)** Si el número de 1 en el bloque de información es par, se añade un cero como dígito de paridad; en caso contrario se añadirá un 1. En todo caso siempre quedará un número par de 1.

–**Paridad Impar (Odd)** Si el número de 1 en el bloque de información es impar, se añadirá un 0 como bit de paridad; en caso contrario se añadirá un 1. En todo caso siempre quedará un número impar de 1.

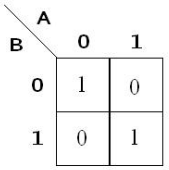
•Circuito de paridad para dos bits de información:

- **Tabla de verdad:**

A	B	PE	PO
0	0	0	1
0	1	1	0
1	0	1	0
1	1	0	1



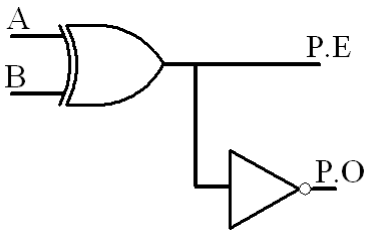
$P.E. = \overline{A}B + A\overline{B}$



$P.O. = \overline{A}\overline{B} + AB$

$P.E = \overline{A}B + A\overline{B} = A \oplus B$

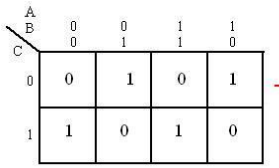
$P.O = \overline{A}\overline{B} + AB = \overline{A \oplus B}$



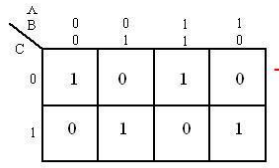
Generador-comprobador de paridad

- **Circuito de paridad para tres bits de información:**

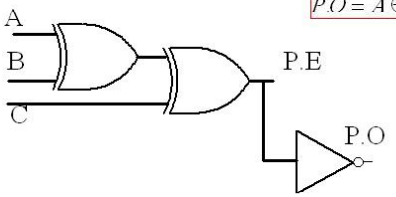
A	B	C	PE	PO
0	0	0	0	1
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	0



$$P.E = \overline{A}\cdot\overline{B}\cdot C + \overline{A}\cdot B\cdot\overline{C} + A\cdot B\cdot C + A\cdot\overline{B}\cdot\overline{C}$$
$$P.E = \overline{B}\cdot(\overline{A}\cdot C + A\cdot\overline{C}) + B\cdot(\overline{A}\cdot\overline{C} + A\cdot C)$$
$$P.E = \overline{B}\cdot(A\oplus C) + B\cdot\overline{(A\oplus C)}$$
$$\boxed{P.E = A\oplus B\oplus C}$$

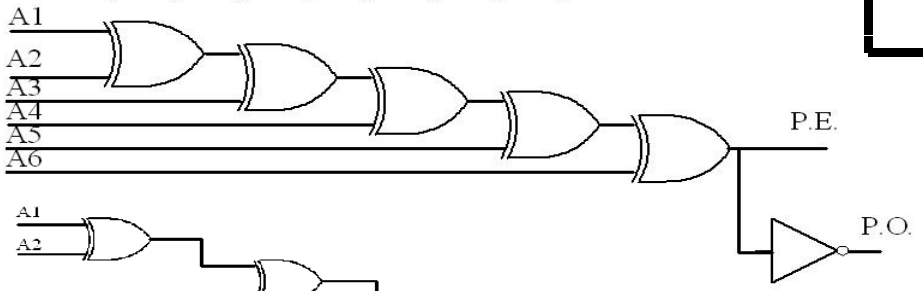


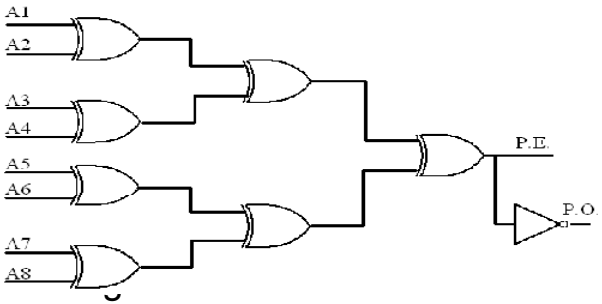
$$P.O = \overline{A}\cdot\overline{B}\cdot\overline{C} + \overline{A}\cdot B\cdot C + A\cdot B\cdot\overline{C} + A\cdot\overline{B}\cdot C$$
$$P.O = \overline{A}\cdot(\overline{B}\cdot\overline{C} + B\cdot C) + A\cdot(B\cdot\overline{C} + \overline{B}\cdot C)$$
$$P.O = \overline{A}\cdot\overline{(B\oplus C)} + A\cdot(B\oplus C)$$
$$\boxed{P.O = \overline{A\oplus B\oplus C}}$$



En general para 8 variables será:

$$P.E. = A_1 \oplus A_2 \oplus A_3 \oplus A_4 \oplus A_5 \oplus A_6 \oplus A_7 \oplus A_8$$
$$P.O. = \overline{A_1 \oplus A_2 \oplus A_3 \oplus A_4 \oplus A_5 \oplus A_6 \oplus A_7 \oplus A_8}$$





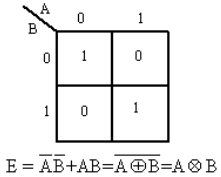
- El primer circuito emplea un número menor de puertas XOR, pero su tiempo total de propagación es inferior

Comparadores

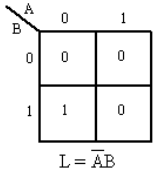
Los comparadores son circuitos digitales que detectan si dos números binarios formados por n bits cada uno son iguales, y en caso contrario cual es mayor y cual es menor.

- El comparador más sencillo es el de cifras de 1 bit
  - –Si A > B activamos una salida del comparador que llamaremos G.
  - –Si A < B activamos una salida del comparador que llamaremos L.
  - –Si A = B activamos una salida del comparador que llamaremos E.

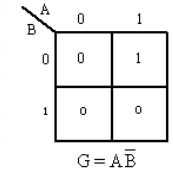
A	B	G	L	E
0	0	0	0	1
0	1	0	1	0
1	0	1	0	0
1	1	0	0	1



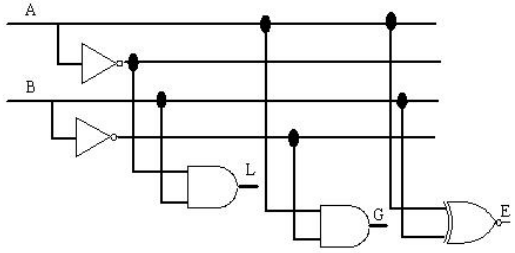
$E = \overline{A}\overline{B}+AB=\overline{A \oplus B}=A \otimes B$



$L = \overline{A}B$



$G = A\overline{B}$



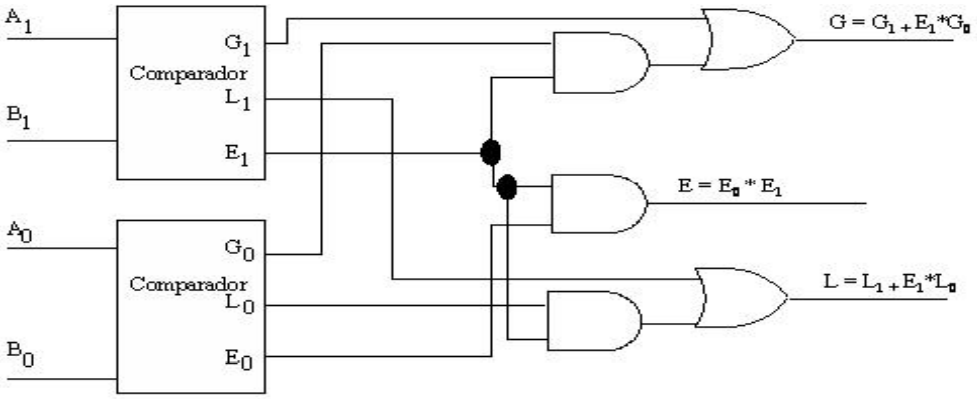
Comparadores

$A = A_1A_0$
$B = B_1B_0$

- Para comparar palabras de mayores de un bit por ejemplo de 2 bits:
- Comparamos los bits de igual peso en ambas cifras (A_o y B_o con el circuito diseñado anteriormente.
- Analizamos cuando debemos activar los bits G, L y E de la comparación de ambos números de 2 bits, es decir, los G,L ó E totales
  - Activaremos G cuando A > B, esto ocurrirá en las siguientes condiciones:
    - $A_1 > B_1$ ó
    - $A_1 = B_1$ y $A_0 > B_0$ $\longrightarrow$ $G_T = G_1 + E_1 \cdot G_0$

  Activaremos E cuando A = B, esto ocurrirá en las siguientes condiciones:
    - $A_1 = B_1$ y $A_0 = B_0$ $\longrightarrow$ $E = E_1 * E_0$

  Activaremos L cuando A < B, esto ocurrirá en las siguientes condiciones:
    - $A_1 < B_1$
    - $A_1 = B_1$ y $A_0 < B_0$ $\longrightarrow$ $L = L_1 + E_1 \cdot L_0$



Comparadores

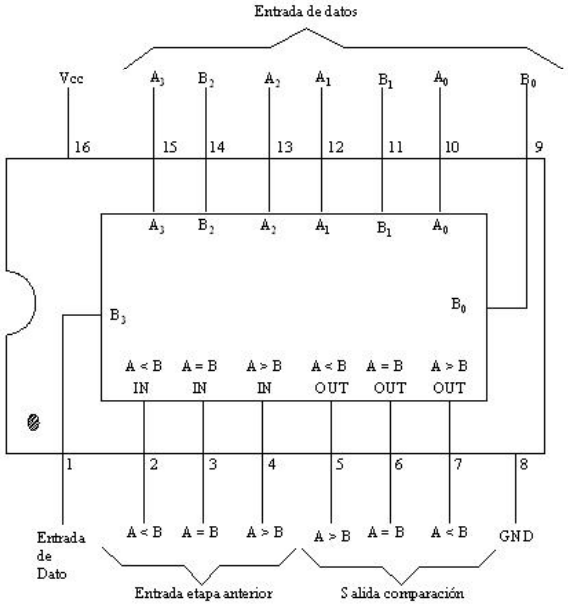
•Para un mayor número de bits haremos un desarrollo similar al que acabamos de realizar para cifras de dos bits.

•En circuitos integrados comerciales tenemos, por ejemplo, comparadores binarios de 4 bits, por ejemplo el 4063 en tecnología CMOS y el 7485 en tecnología TTL.

•Estos comparadores tienen **entradas en cascada** E, G y L para hacer comparaciones de palabras mayores de 4 bits, si queremos comparar palabras de sólo 4 bits, colocaremos E = 1 y G = L = 0 (suponiendo que son activas a nivel alto). De la misma forma se puede obtener comparadores de palabras de 16 bits asociando dos de 8 o cuatro de 4 bits …etc.

Comparadores

•El C.I 74HC85 puede comparar dos números de cuatro bits cada uno de ellos, El primero de ellos $A=A_3A_2A_1A_0$, se introducirá por las patillas 15, 13, 12 y 10 del integrado, y el segundo $B=B_3B_2B_1B_0$, se introducirá por las patillas 1, 14, 11 y 9. El 74HC85 compara ambas cifras y activa una de las salidas de la comparación, si A > B activa la patilla 5, si A < B activa la patilla 7, y si A = B activa la patilla 6.



•Si sólo deseamos comparar dos cifras de 4 bits, colocamos la entrada E a 1 y las entradas G y L a 0.

Comparadores

•En la hoja de características facilitada por el fabricante del 74HC85, entre otros datos, tenemos acceso a la tabla de verdad que rige el funcionamiento de dicho C.I:

FUNCTION TABLE

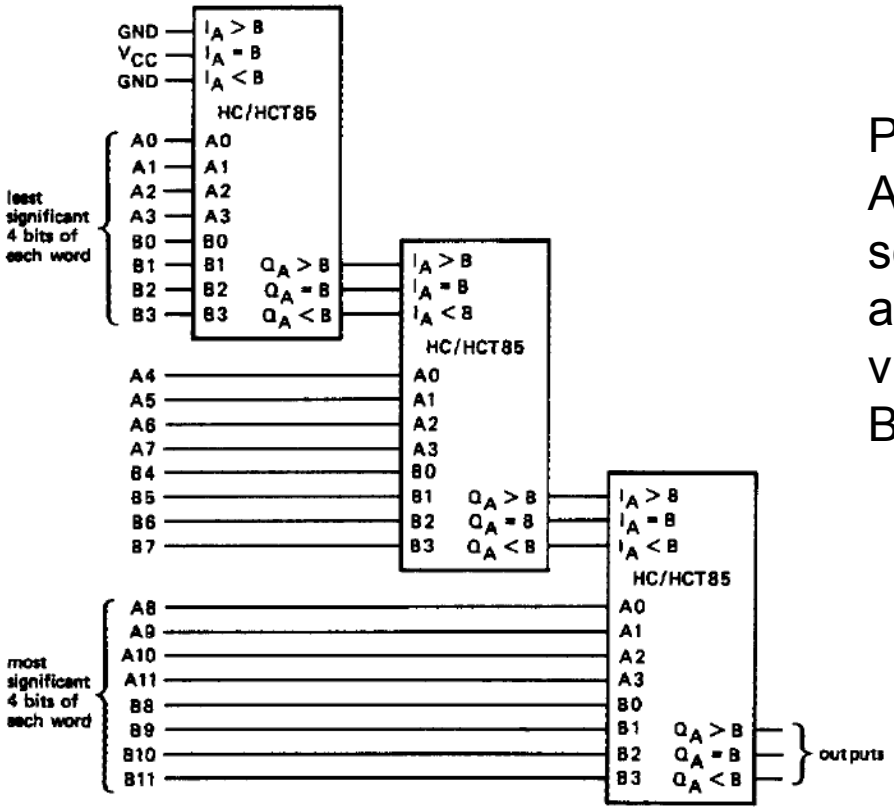
COMPARING INPUTS				CASCADING INPUTS			OUTPUTS		
A_3, B_3	A_2, B_2	A_1, B_1	A_0, B_0	$I_{A>B}$	$I_{A<B}$	$I_{A=B}$	$Q_{A>B}$	$Q_{A<B}$	$Q_{A=B}$
$A_3>B_3$	X	X	X	X	X	X	H	L	L
$A_3<B_3$	X	X	X	X	X	X	L	H	L
$A_3=B_3$	$A_2>B_2$	X	X	X	X	X	H	L	L
$A_3=B_3$	$A_2<B_2$	X	X	X	X	X	L	H	L
$A_3=B_3$	$A_2=B_2$	$A_1>B_1$	X	X	X	X	H	L	L
$A_3=B_3$	$A_2=B_2$	$A_1<B_1$	X	X	X	X	L	H	L
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0>B_0$	X	X	X	H	L	L
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0<B_0$	X	X	X	L	H	L
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	H	L	L	H	L	L
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	L	H	L	L	H	L
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	L	L	H	L	L	H
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	X	X	H	L	L	H
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	H	H	L	L	L	L
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	L	L	L	H	H	L

También se deduce de la tabla en sus tres últimas filas que la entrada $I_{A=B}$ es prioritaria frente a las otras dos entradas en cascada

Observar que las entradas de comparación son prioritarias frente a las de cascada o dicho de otra forma : las entradas en cascada sólo son atendidas si $A_0A_1A_2A_3 = B_0B_1B_2B_3$

Comparadores

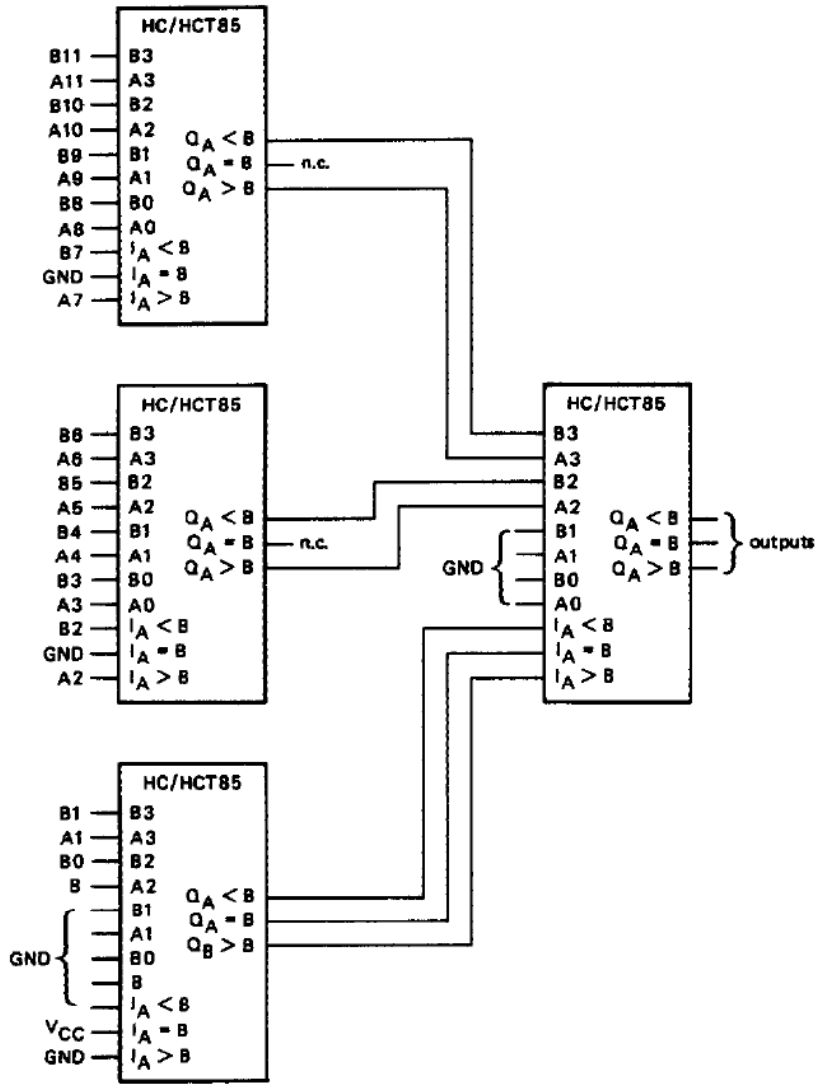
•También de la hoja de características facilitada por el fabricante del 74HC85, obtenemos el conexionado para ampliar la capacidad de comparación de 4 a 12 bits.



Primero se comparan los MSB $A_8A_9A_{10}A_{11}$ con $B_8B_9B_{10}B_{11}$, y solamente cuando son iguales se atienden las entradas en cascada que vienen de comparar $A_4A_5A_6A_7$ con $B_4B_5B_6B_7$ …etc.

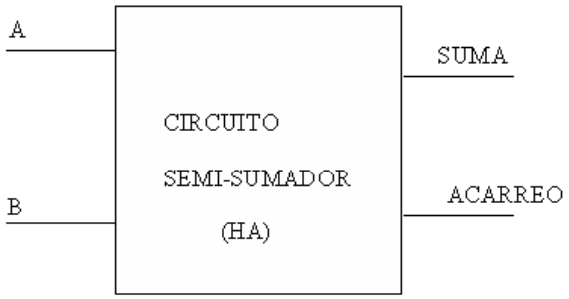
Comparadores

•Otra posibilidad sería:



Semisumador (HA)

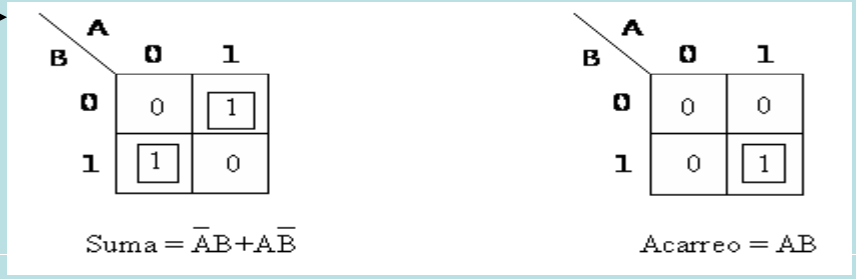
- El semisumador (HA) será el circuito capaz de sumar dos datos binarios de un bit cada uno de ellos



•El circuito a diseñar debe cumplir con la tabla de verdad para la suma de dos bits

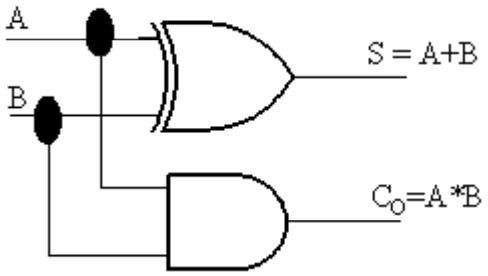
Primer Bit	Segundo Bit	Suma	Acarreo
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Pasando a mapas de karnaugh, uno para cada una de las dos salidas



Semisumador (HA)

- Con las expresiones ya deducidas, podemos implementar el circuito HA:



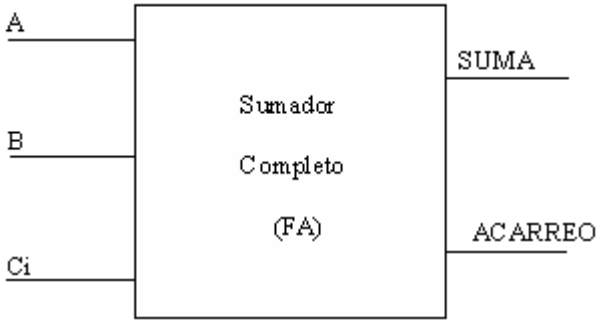
Sumador Completo (FA)

- Para sumar cifras mayores de un bit, necesitamos circuitos que sumen cada pareja de bits del mismo peso por separado, pero también debemos tener en cuenta los acarreos de sumas anteriores.

1 1 1 1 → Acarreos		
1 1 0 1 1 → Primera cifra	→	27
0 1 1 0 1 → Segunda Cifra	→	13
1 0 1 0 0 0		40

Sumador Completo (FA)

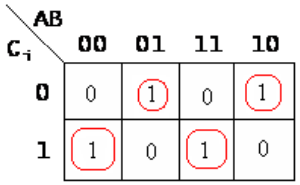
- El circuito capaz de sumar tres bits (2 datos de un bit cada unos mas el acarreo de la suma anterior) se le llamará sumador completo (FA).

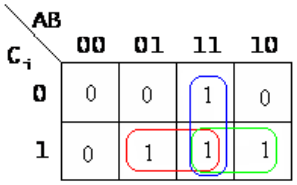


•El circuito a diseñar debe cumplir con la tabla de verdad para la suma de tres bits

A	B	Ci	S	Co
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Pasando a mapas de karnaugh, uno para cada una de las dos salidas



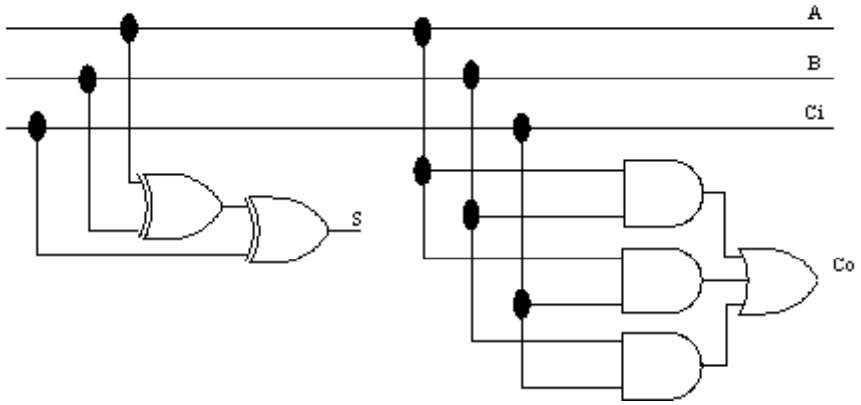


$$S = \bar{A}\bar{B}C_i + \bar{A}B\bar{C}_i + ABC_i + A\bar{B}\bar{C}_i =$$
$$S = \bar{A}(\bar{B}C_i + B\bar{C}_i) + A(BC_i + \bar{B}\bar{C}_i) =$$
$$S = \bar{A}(B \oplus C_i) + A(\overline{B \oplus C_i}) =$$
$$S = A \oplus B \oplus C_i$$

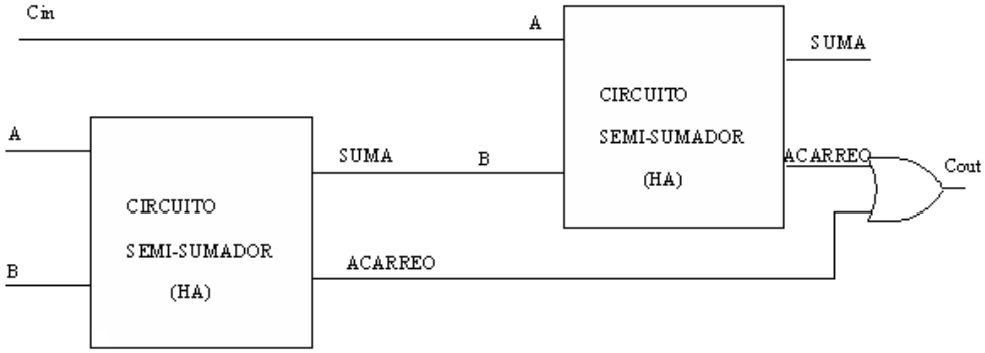
$$C_{out} = AB + AC_i + BC_i$$

Sumador Completo (FA)

- Con las expresiones ya deducidas, podemos implementar el circuito FA:

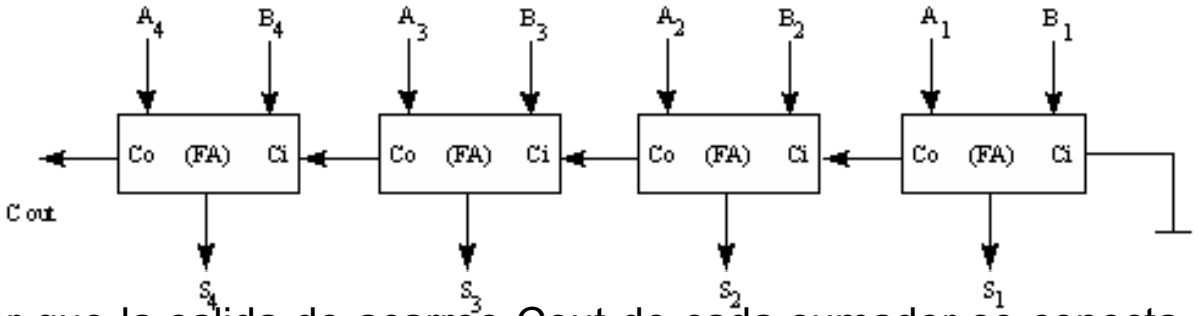


- Otra forma de diseñar un sumador completo, será la de basarnos en los anteriores semisumadores (HA).



Expansión de sumadores

- ***Sumador paralelo con arrastre serie:***
  - El sumador paralelo con arrastre serie consta de varios pasos de sumadores completos (FA) conectados entre si como se muestra en la figura:



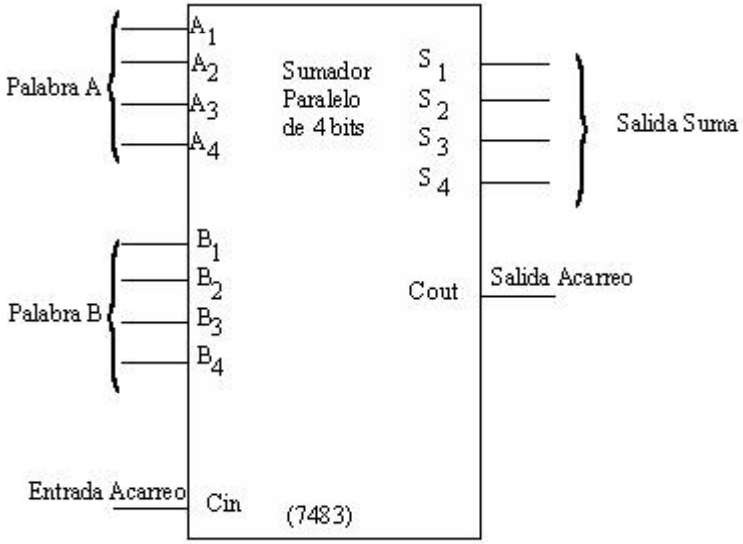
• Observar que la salida de acarreo Cout de cada sumador se conecta a la entrada de acarreo Cin del sumador de peso inmediatamente superior, por lo tanto el bit de acarreo se va propagando de un sumador completo a otro.

•En cada etapa (FA) cada uno de los sumadores realiza la suma tan pronto como están presentes sus entradas Ai y Bi, pero las salidas no son correctas hasta tanto no se procesen todos los acarreos, por lo que se generará una salida de suma provisional incorrecta.

•Por esta razón los sumadores paralelos tienen con frecuencia unas puertas a la salida de las líneas de suma, que estarán desactivadas hasta que el sumador haya dispuesto de tiempo suficiente para propagar las señales de la suma de forma correcta.

Expansión de sumadores

- Por esta razón a este tipo de sumadores se les considera lentos, si bien son suficientemente rápidos para la mayoría de los procesos a excepción de grandes ordenadores.
- Podemos observar que en el sumador completo de menor peso, tenemos la entrada de acarreo conectada permanentemente a cero, ya que no existe suma anterior que nos pueda provocar un acarreo de entrada, por esta razón, podríamos sustituir ese sumador completo (FA) por un semisumador (HA).
- Un ejemplo de CI para sumar dos datos de cuatro bits cada uno es el CI 7483 :



Expansión de sumadores

- ***Sumador con acarreo anticipado:***
  - El problema de lentitud descrito en el sumador con arrastre serie, crece según vamos aumentando el tamaño de los datos a sumar, la técnica del acarreo anticipado, subsana este tema, a costa de complicar la circuitería.
  - Dicha técnica permite anticipar todas las señales de acarreo antes de que sean generadas, utilizando una lógica adicional.
  - Partiremos de la ecuación inicial para el acarreo de salida en un sumador completo sin simplificar:

$$Cout = \overline{A}BCi + A\overline{B}Ci + AB\overline{Ci} + ABCi$$

Sacando factores comunes queda la siguiente ecuación:

$$Cout = Ci(\overline{A}B + A\overline{B}) + AB(\overline{Ci} + Ci)$$

$$Cout = Ci(A \oplus B) + AB$$

Llamaremos:

Término de propagación: $P = A \oplus B$

Término de generación: $G = AB$

Expansión de sumadores

– Por lo que la ecuación quedará: $Cout = PCi + G$

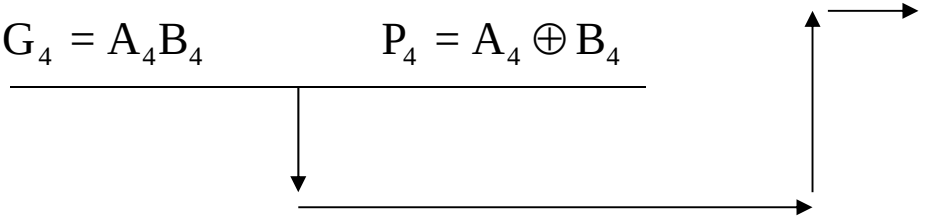
– Analizando esta última expresión, deduciremos que Cout = 1 si se cumple cualquiera de las dos siguientes condiciones:

1. $G = 1 \rightarrow A = 1 \text{ y } B = 1$

2. $P*Ci = 1 \rightarrow P = 1 \text{ y } Ci = 1 \rightarrow Ci = 1 \text{ y } \left\{ \begin{array}{l} A = 1 \\ o \\ B = 1 \end{array} \right\}$

• Para un sumador de cuatro bits:

$$G_1 = A_1B_1 \qquad P_1 = A_1 \oplus B_1$$
$$G_2 = A_2B_2 \qquad P_2 = A_2 \oplus B_2$$
$$G_3 = A_3B_3 \qquad P_3 = A_3 \oplus B_3$$
$$G_4 = A_4B_4 \qquad P_4 = A_4 \oplus B_4$$



$$C_1 = P_1C_0 + G_1$$
$$C_2 = P_2C_1 + G_2$$
$$C_3 = P_3C_2 + G_3$$
$$C_4 = P_4C_3 + G_4$$

Expansión de sumadores

- Por lo que la ecuación quedará: $Cout = PCi + G$
- Analizando esta última expresión, deduciremos que Cout = 1 si se cumple cualquiera de las dos siguientes condiciones:
  - $G = 1 \rightarrow A = 1 \text{ y } B = 1$
  - $P*Ci = 1 \rightarrow P = 1 \text{ y } Ci = 1 \rightarrow Ci = 1 \text{ y } \begin{cases} A = 1 \\ o \\ B = 1 \end{cases}$

• Para un sumador de cuatro bits:

$$G_1 = A_1B_1 \qquad P_1 = A_1 \oplus B_1 \qquad C_1 = P_1C_0 + G_1$$
$$G_2 = A_2B_2 \qquad P_2 = A_2 \oplus B_2 \qquad C_2 = P_2C_1 + G_2$$
$$G_3 = A_3B_3 \qquad P_3 = A_3 \oplus B_3 \qquad C_3 = P_3C_2 + G_3$$
$$G_4 = A_4B_4 \qquad P_4 = A_4 \oplus B_4 \qquad C_4 = P_4C_3 + G_4$$

•Sustituyendo la fórmula de C0 en la de C1

$$C_2 = P_2(P_1C_0 + G_1) + G_2$$
$$C_2 = P_2P_1C_0 + P_2G_1 + G_2$$

Sustituyendo en C_3 y C_4

$$C_3 = P_3P_2P_1C_0 + P_3P_2G_1 + P_3G_2 + G_3$$
$$C_4 = P_4P_3P_2P_1C_0 + P_4P_3P_2G_1 + P_4P_3G_2 + P_4G_3 + G_4$$

Expansión de sumadores

- Recordando la fórmula de la suma en un sumador completo (FA):

$$S = A + B + C_i$$

- Tenemos en nuestro sumador de cuatro bits:

$$S_1 = A_1 \oplus B_1 \oplus C_0 = P_1 \oplus C_0$$
$$S_2 = A_2 \oplus B_2 \oplus C_1 = P_2 \oplus C_1$$
$$S_3 = A_3 \oplus B_3 \oplus C_2 = P_3 \oplus C_2$$
$$S_4 = A_4 \oplus B_4 \oplus C_3 = P_4 \oplus C_3$$

Para La generación de los bits C0, C1, C2 y C3 lo hago por las fórmulas halladas anteriormente que son funciones de Pi y de Gi que a su vez son funciones de Ai, Bi y de C-1.
Para saber el valor de Ci no necesito saber el valor de Ci-1, sino que sólo necesito saber los valores de Ai, Bi y de C-1.
De esta forma obtenemos un sumador más rápido, pero complicamos su circuitería

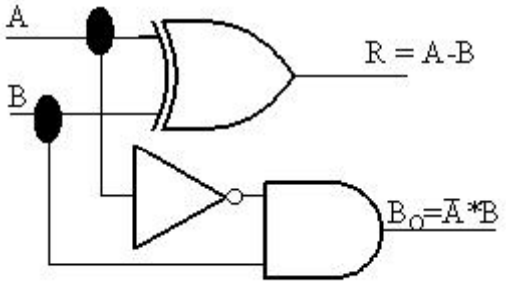
Expansión de sumadores

- Se han visto dos tipos de sumadores, el sumador paralelo de arrastre serie y el sumador de acarreo anticipado, ambos tiene ventajas e inconveniente frente al otro que comentamos a continuación:
  - El sumador paralelo de arrastre serie es claramente más lento que el de acarreo anticipado, y esta diferencia se hace mayor cuando tengamos más bits para sumar.
  - La limitación más importante del sumador paralelo de acarreo anticipado es que su circuitería es más compleja, aumentando esta complejidad según tengamos más bits para sumar, a parte que cada vez las puertas utilizadas necesitan un mayor Fan-In.

Semirestador (HB)

- De la misma forma que se ha diseñado el semisumador (HA), podemos diseñar el semirestador (HB)

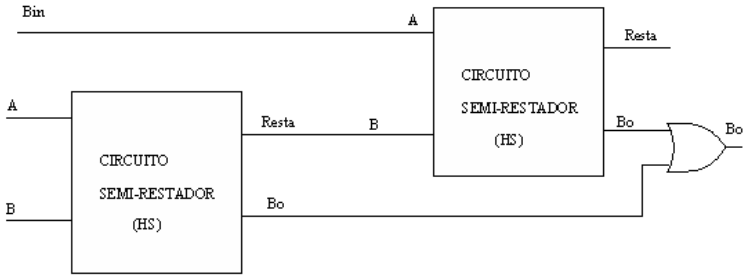
• Realizando el mismo proceso que se realizó en el HA, llegaremos a



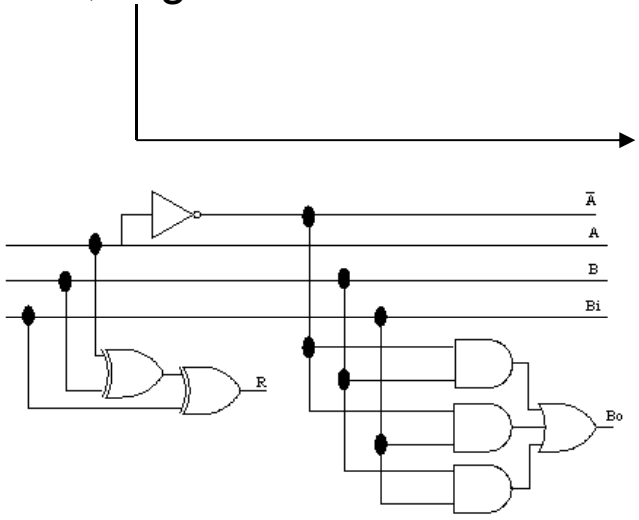
•El circuito a diseñar debe cumplir con la tabla de verdad para la suma de dos bits

Minuendo	Sustraendo	Diferencia	"Préstamo"
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

RESTADOR COMPLETO (FB)

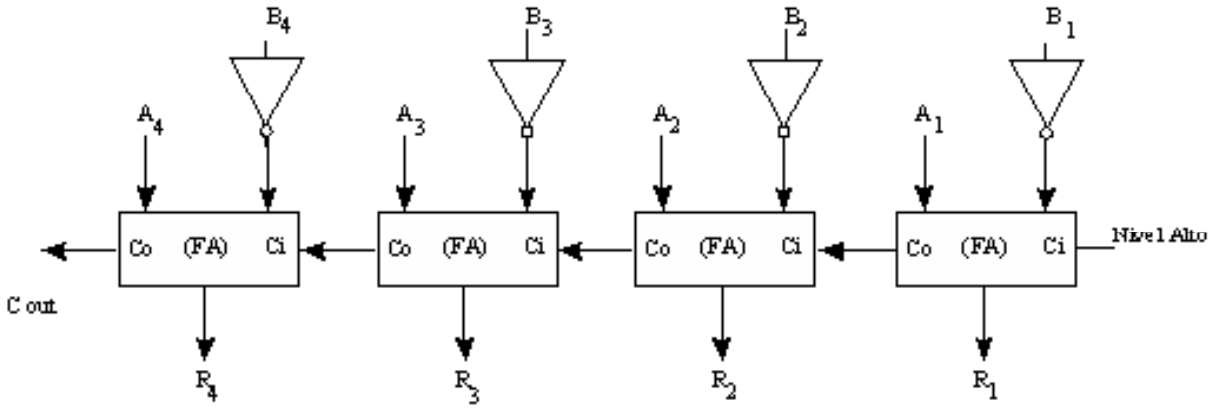


• Realizando el mismo proceso que se realizó en el HA, llegaremos a



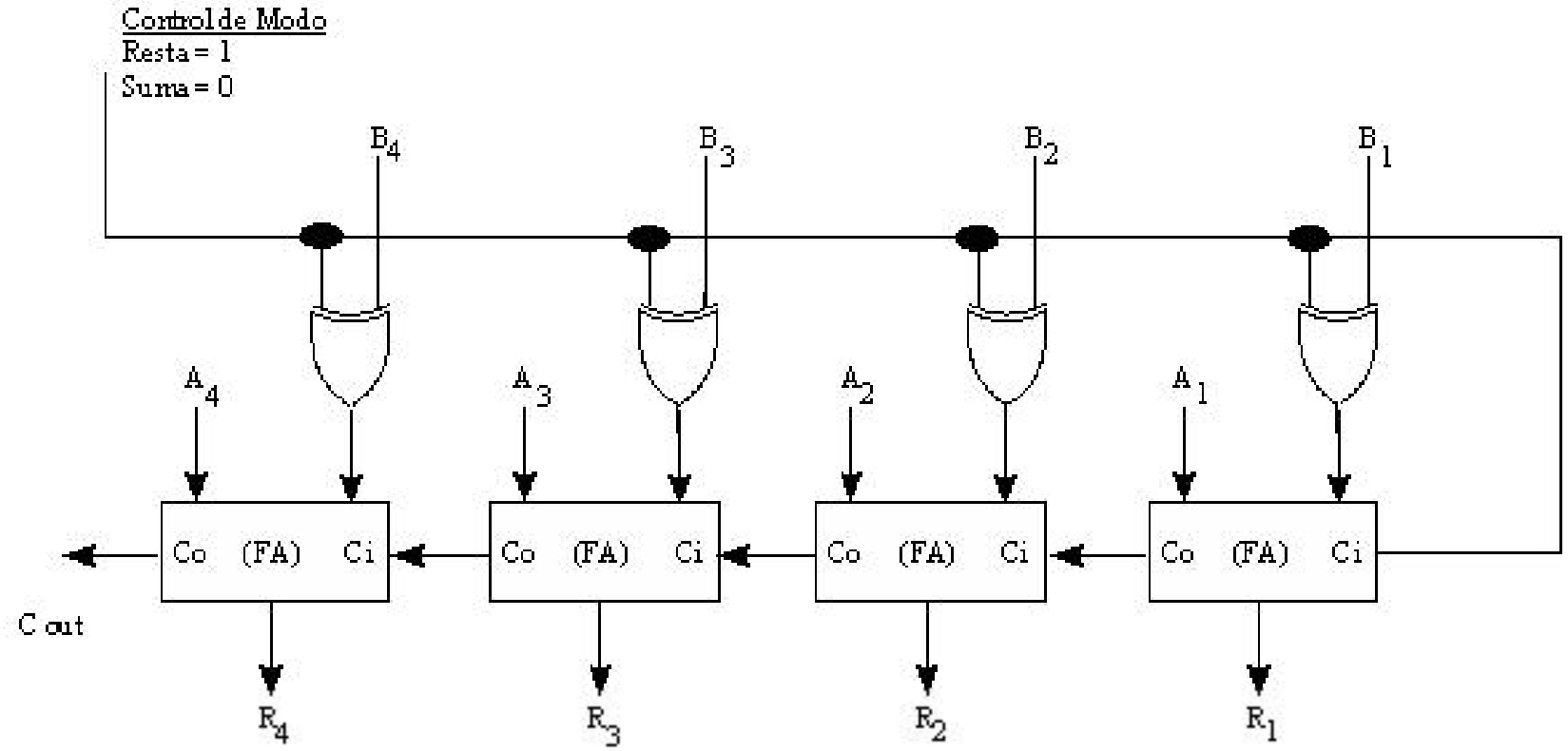
A	B	Bi	RESTA	Bo
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Circuito restador paralelo de cuatro bits por complemento a 2



•Los cuatro inversores convierten el sustraendo a su forma en complemento a uno, y el nivel alto a la entrada de Cin del primer FA, es lo mismo que sumar uno al sustraendo.

circuito sumador/restador paralelo de cuatro bits por complemento a 2

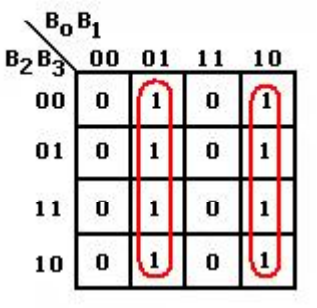


CONVERSORES DE CODIGOS

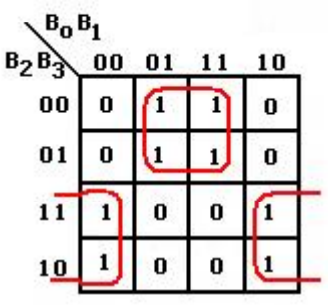
•El diseño de cualquier conversor de código se puede hacer siguiendo los mismos pasos que en el diseño de cualquier circuito combinacional.

•Ejemplo: Diseñar un conversor de binario de 4 bits a código GRAY

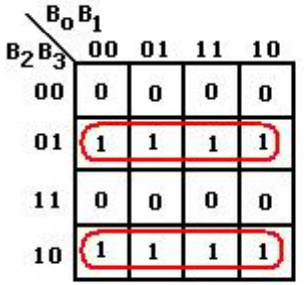
	BINARIO				GRAY			
	B3	B2	B1	B0	G3	G2	G1	G0
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	1
3	0	0	1	1	0	0	1	0
4	0	1	0	0	0	1	1	0
5	0	1	0	1	0	1	1	1
6	0	1	1	0	0	1	0	1
7	0	1	1	1	0	1	0	0
8	1	0	0	0	1	1	0	0
9	1	0	0	1	1	1	0	1
10	1	0	1	0	1	1	1	1
11	1	0	1	1	1	1	1	0
12	1	1	0	0	1	0	1	0
13	1	1	0	1	1	0	1	1
14	1	1	1	0	1	0	0	1
15	1	1	1	1	1	0	0	0



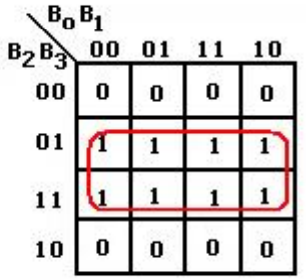
$G_0 = B_0 \oplus B_1$



$G_1 = B_1 \oplus B_2$



$G_2 = B_2 \oplus B_3$



$G_3 = B_3$

CODIFICADORES

- Los circuitos codificadores son sistemas combinacionales de N entradas y “n” salidas cuyo funcionamiento es tal que cuando se activa una de las entradas, en las salidas aparece la combinación lógica en binario de la entrada que se ha activado.
- Para ello se debe cumplir $N \leq 2^n$
- Los circuitos Codificadores se pueden dividir en:
    - Codificadores con prioridad
    - Codificadores sin prioridad

• Un ejemplo claro de necesidad de circuitos codificadores se puede encontrar en el funcionamiento de una calculadora, cada vez que pulsamos un número, la calculadora debe conocer el código binario correspondiente a la tecla pulsada.

Codificadores

•La codificación realizada, puede ser a cualquier código binario (BCD, GRAY, etc..).

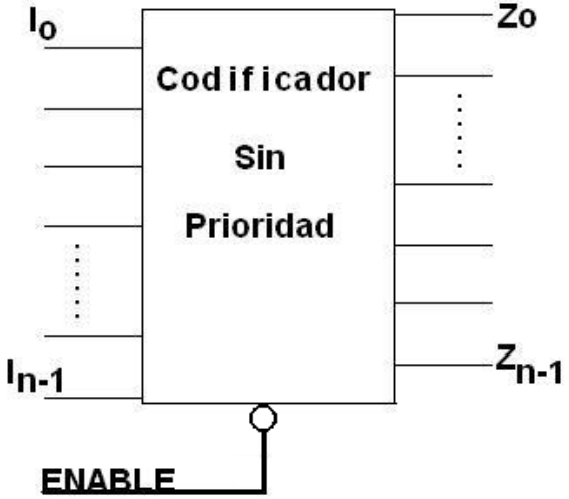
•**CODIFICADOR SIN PRIORIDAD**: Cuando se activan simultáneamente varias entradas, en la salida aparece la combinación correspondiente a la suma lógica de dichas entradas, por lo que sólo se debe permitir activar una entrada en cada momento.

•**CODIFICADOR CON PRIORIDAD**: Cuando se activan simultáneamente varias entradas, en la salida aparece la combinación correspondiente a una sola de las entradas (generalmente la de mas peso).

Codificadores

CODIFICADORES SIN PRIORIDAD

CODIFICADOR GENÉRICO SIN PRIORIDAD



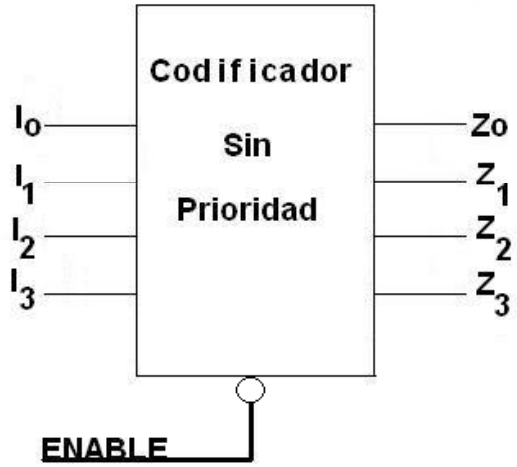
ENTRADAS		SALIDAS			
E	I	Z_{n-1}	...	Z_1	Z_0
1	X	0	0	0	0
0	Io	0	0	0	0
0	I1	0	0	0	1
0	I2	0	0	1	0
0	I3	0	0	1	1
0	I4	0	1	0	0
0	I5	0	1	0	1
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
0	In-1	1	1	1	1
0	Varias entradas	SUMA LÓGICA DE LAS SALIDAS			

•Generalmente tanto los codificadores, como otros sistemas, poseen una entrada de Enable que nos permite activar/desactivar el circuito (Observar que en este caso la activación se realiza con un 0 en la entrada ENABLE)..

Codificadores

EJEMPLO CODIFICADORES SIN PRIORIDAD (8 bits)

CODIFICADOR GENÉRICO SIN PRIORIDAD 8 BITS

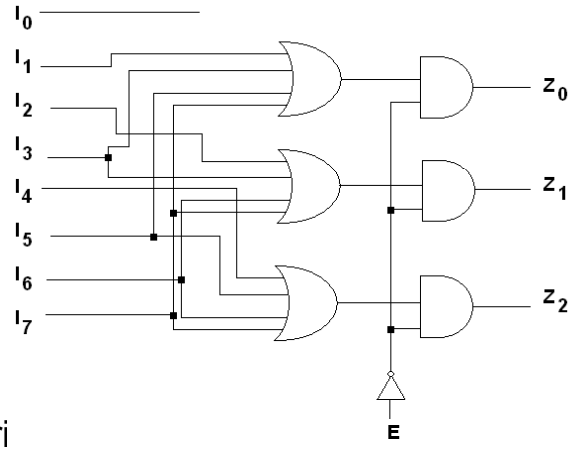


ENTRADAS		SALIDAS		
E	I	Z_2	Z_1	Z_0
1	X	0	0	0
0	Io	0	0	0
0	I1	0	0	1
0	I2	0	1	0
0	I3	0	1	1
0	I4	1	0	0
0	I5	1	0	1
0	I6	1	1	0
0	I7	1	1	1
0	Varias entradas	SUMA LÓGICA DE SALIDAS		

$$Z_0 = \overline{E}(I_1 + I_3 + I_5 + I_7)$$

$$Z_1 = \overline{E}(I_2 + I_3 + I_6 + I_7)$$

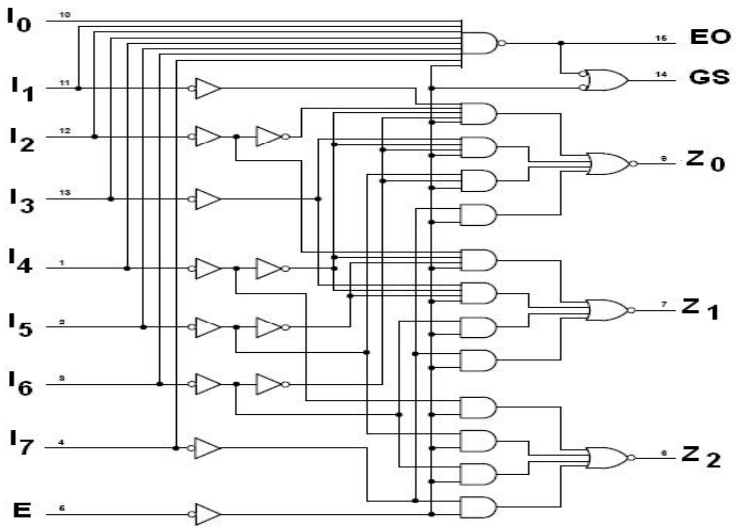
$$Z_2 = \overline{E}(I_4 + I_5 + I_6 + I_7)$$



Codificadores

CODIFICADORES CON PRIORIDAD

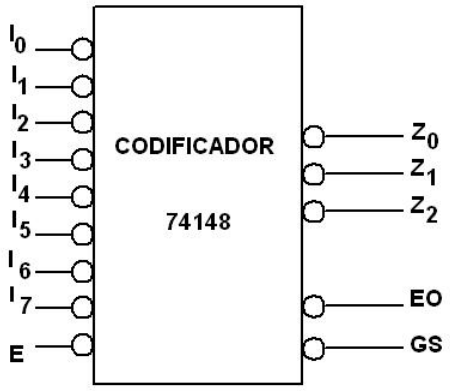
Entradas									Salidas				
E	I_7	I_6	I_5	I_4	I_3	I_2	I_1	I_0	Z_2	Z_1	Z_0	GS	EO
1	X	X	X	X	X	X	X	X	1	1	1	1	1
0	1	1	1	1	1	1	1	0	1	1	1	0	1
0	1	1	1	1	1	1	0	X	1	1	0	0	1
0	1	1	1	1	1	0	X	X	1	0	1	0	1
0	1	1	1	1	0	X	X	X	1	0	0	0	1
0	1	1	1	0	X	X	X	X	0	1	1	0	1
0	1	1	0	X	X	X	X	X	0	1	0	0	1
0	1	0	X	X	X	X	X	X	0	0	1	0	1
0	0	X	X	X	X	X	X	X	0	0	0	0	1
0	1	1	1	1	1	1	1	1	1	1	1	1	0



•En caso de activarse más de una entrada simultáneamente, la salida sólo muestra la de mayor peso

•La salida GS (Group Signal Output) se activará (nivel bajo) siempre y cuando exista alguna entrada activa, esto permite resolver la ambigüedad entre chip deshabilitado o entrada I_o activa.

•La salida Eo valdrá 0 sólo en el caso que todas las entradas sean 1 a la vez (desactivadas), permite resolver ambigüedades entre chip deshabilitado o ninguna entrada activa.



Codificadores

EJEMPLO CODIFICADORES CON PRIORIDAD

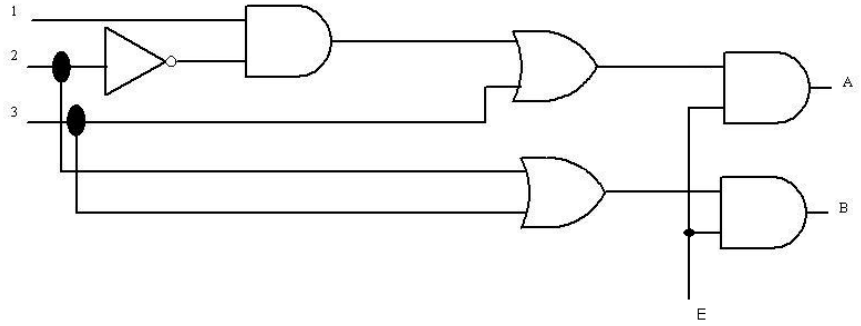
•Para ver una idea sencilla de cómo podemos diseñar un codificador con prioridad, nos limitaremos a codificar los números decimales 0,1,2 y 3 en binario natural, la tabla de verdad será la siguiente suponiendo niveles activos en estado alto.

Entradas			Salidas	
1	2	3	B	A
0	0	0	0	0
1	0	0	0	1
X	1	0	1	0
X	X	1	1	1

$$A = E(I_3 + I_1 \overline{I_2})$$

$$B = E(I_2 + I_3)$$

1	2	3	B	A
0	0	0	0	0
0	0	1	1	1
0	1	0	1	0
0	1	1	1	1
1	0	0	0	1
1	0	1	1	1
1	1	0	1	0
1	1	1	1	1



Extensión de Codificadores

•Los fabricantes, como en el resto de circuitos, limitan el número de entradas que podemos codificar con un solo C.I., para aumentar este valor debemos recurrir a la interconexión entre codificadores.
•En general no se fabrican codificadores con prioridad de más de 10 entradas.
•Los fabricantes a través de sus hojas de características, dan información de como aumentar la capacidad de los codificadores
•En todos aquellos circuitos que disponen de entrada de enable (EI), y salidas (EO) y (GS), la extensión de capacidad es muy simple, pudiendo realizarse de dos formas: serie y paralelo

- •En la forma serie, el número de circuitos lógicos adicionales es mínimo.
- •La forma paralelo ofrece una mayor velocidad

DECODIFICADORES

- Un decodificador puede definirse como aquel circuito que realiza la tarea opuesta a los codificadores.
- Los circuitos decodificadores se pueden dividir en:
    - Decodificadores no excitadores o simplemente decodificadores
    - Decodificadores excitadores o Drivers

DECODIFICADORES

DECODIFICADORES NO EXCITADORES

• Son sistemas combinacionales con “n” entradas y “N” salidas, de tal forma que para cada combinación de entrada se excita una sola salida.

• En aquellos casos en los que el número de salidas “N” sea menor que el posible número de combinaciones de entrada $N \leq 2^n$ existirá un número de combinaciones de entrada 2^n-N que se denominan **condiciones de Indiferencia.**

•Si las condiciones de indiferencia se toman como datos falsos, impidiendo su uso para que ante ellas no se active ninguna salida, se dice que el decodificador rechaza datos falsos de entrada, (tenerlo en cuenta al no agrupar en karnaugh ni McClusKey estas combinaciones)

DECODIFICADORES

DISEÑO DE DECODIFICADORES NO EXCITADORES

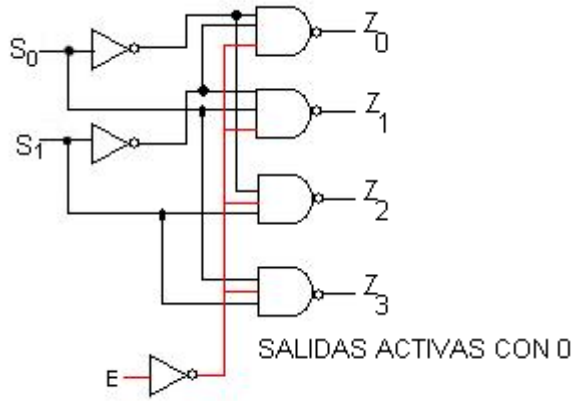
• **Diseñar un decodificador de 2 entradas (S1 y S2) y 4 salidas (Z0, Z1, Z2 y Z3) con entraba de habilitación (Enable) activo con 0, en caso de que Enable = 1 todas las salidas serán 1**.

No existen condiciones de indiferencia ya que $N=4=2^n=2^2$

E \ S_1S_0	00	01	11	10
0	Z_0	Z_1	Z_3	Z_2
1	0	0	0	0

$Z_0 = \bar{E}\,\bar{S}_1\,\bar{S}_0$ $Z_2 = \bar{E}\,S_1\,\bar{S}_0$

$Z_1 = \bar{E}\,\bar{S}_1\,S_0$ $Z_3 = \bar{E}\,S_1\,S_0$

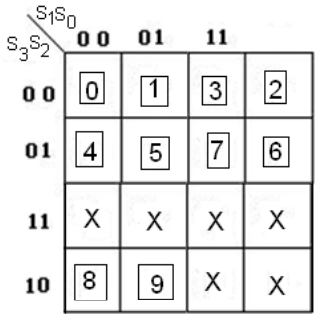


DECODIFICADORES

DISEÑO DE DECODIFICADORES NO EXCITADORES

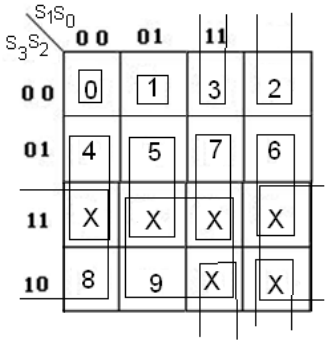
- **Ejemplo 2: Decodificador BCD-DECIMAL con rechazo de datos falsos y otro sin rechazo de datos falsos**

Existen condiciones de indiferencia ya que $N = 10 < 2^n = 2^4$



CON RECHAZO DE DATOS FALSOS

$0 = \bar{S}_0 \cdot \bar{S}_1 \cdot \bar{S}_2 \cdot \bar{S}_3$
$1 = S_0 \cdot \bar{S}_1 \cdot \bar{S}_2 \cdot \bar{S}_3$
$2 = \bar{S}_0 \cdot S_1 \cdot \bar{S}_2 \cdot \bar{S}_3$
$3 = S_0 \cdot S_1 \cdot \bar{S}_2 \cdot \bar{S}_3$
$4 = \bar{S}_0 \cdot \bar{S}_1 \cdot S_2 \cdot \bar{S}_3$
$5 = S_0 \cdot \bar{S}_1 \cdot S_2 \cdot \bar{S}_3$
$6 = \bar{S}_0 \cdot S_1 \cdot S_2 \cdot \bar{S}_3$
$7 = S_0 \cdot S_1 \cdot S_2 \cdot \bar{S}_3$
$8 = \bar{S}_0 \cdot \bar{S}_1 \cdot \bar{S}_2 \cdot S_3$
$9 = S_0 \cdot \bar{S}_1 \cdot \bar{S}_2 \cdot S_3$



SIN RECHAZO DE DATOS FALSOS

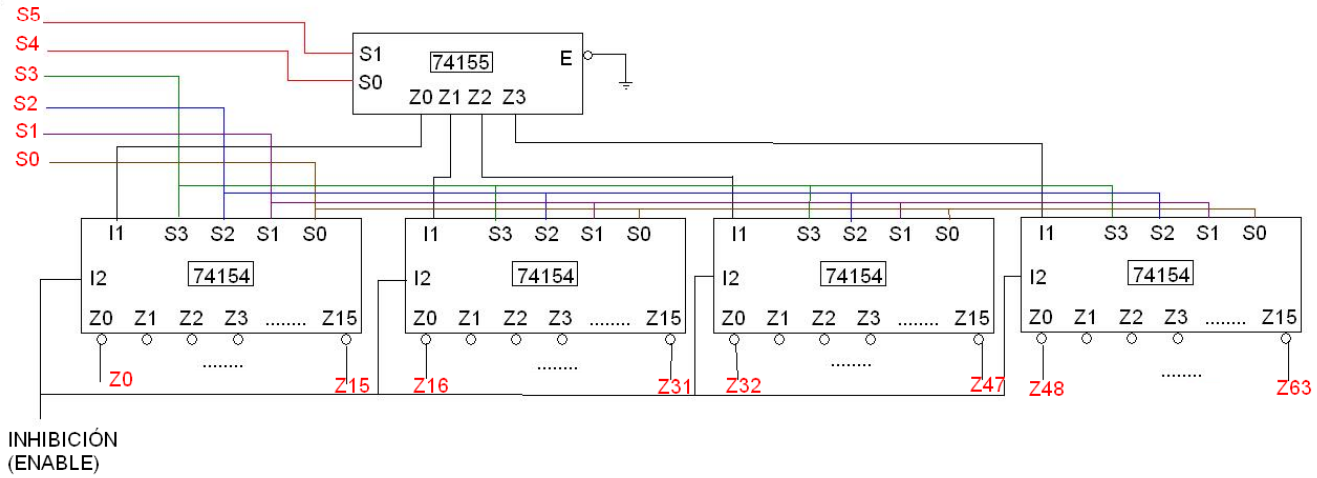
$0 = \bar{S}_0 \cdot \bar{S}_1 \cdot \bar{S}_2 \cdot \bar{S}_3$
$1 = S_0 \cdot \bar{S}_1 \cdot \bar{S}_2 \cdot \bar{S}_3$
$2 = \bar{S}_0 \cdot S_1 \cdot \bar{S}_2$
$3 = S_0 \cdot S_1 \cdot \bar{S}_2$
$4 = \bar{S}_0 \cdot \bar{S}_1 \cdot S_2$
$5 = S_0 \cdot \bar{S}_1 \cdot S_2$
$6 = \bar{S}_0 \cdot S_1 \cdot S_2$
$7 = S_0 \cdot S_1 \cdot S_2$
$8 = \bar{S}_0 \cdot S_3$
$9 = S_0 \cdot S_3$

DECODIFICADORES

EXTENSIÓN DE DECODIFICADORES NO EXCITADORES

• Es posible extender la capacidad de los decodificadores mediante una simple combinación de ellos mismos.

• Ejemplo : A partir de 4 decodificadores de 16 salidas (4 entradas) (74154) y uno de 4 salidas (2 entradas) (74155) obtener un decodificador de 64 salidas (6 entradas)



DECODIFICADORES

- **APLICACIONES DE LOS DECODIFICADORES**
- Los decodificadores no excitadores pueden ser empleados para implementar funciones algebraicas
- Generación de funciones lógicas: Un posible procedimiento a seguir para la generación de funciones lógicas usando multiplexores, es el siguiente:
  - Obtener la forma canónica de la función en suma de productos.
  - Seleccionar el decodificador a utilizar, teniendo en cuenta que el número de entradas de selección debe de ser igual al de variables de la función
  - Identificar las variables de la función con las entradas de selección del decodificador.
  - Encontrar las salidas del decodificador que se corresponden con los productos canónicos de la función.
  - Unir las salidas obtenidas en el punto 4 mediante una función OR si las salidas del decodificador se activan con 1 o con una función NAND si se activan con ceros.

DECODIFICADORES

- **APLICACIONES DE LOS DECODIFICADORES**
  - Ejemplo

$$F = A\bar{B}\bar{C}\bar{D} + AB\bar{D} + \bar{B}CD$$

Desarrollo en forma de suma de productos canónicos

$$F = A\bar{B}\bar{C}\bar{D} + ABC\bar{D} + AB\bar{C}\bar{D} + A\bar{B}CD + \bar{A}\bar{B}CD$$

Como la función canónica tiene 4 variables, necesito un decodificador de 16 salidas (por ejemplo : 74LS154)

Llamo S0 = A (LSB), S1 = B, S2 = C, S3 = D

Identifico las salidas

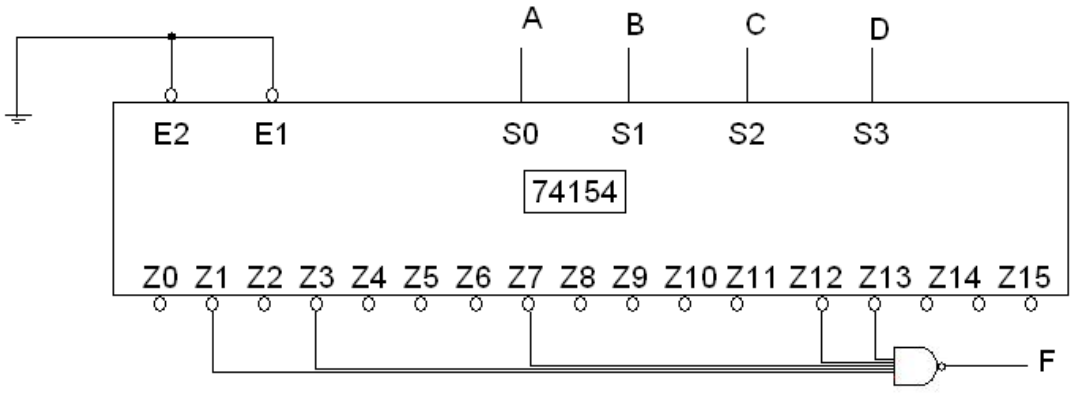
$A\bar{B}\bar{C}\bar{D}$ =Z1

$ABC\bar{D}$ =Z7

$AB\bar{C}\bar{D}$ =Z3

$A\bar{B}CD$ =Z13

$\bar{A}\bar{B}CD$ =Z12



DECODIFICADORES

DECODIFICADORES EXCITADORES O DRIVERS

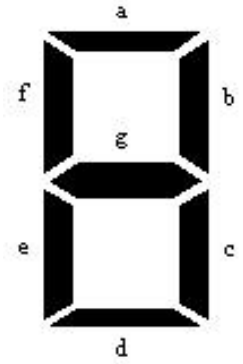
• El funcionamiento de estos decodificadores es similar al de los anteriores, con la salvedad de que para una combinación de entrada se pueden activar simultáneamente varias salidas.

•Los Drivers son encargados de adaptar los niveles eléctricos de las señales a los elementos visualizadores.

DECODIFICADORES

•Como ejemplo del segundo grupo de decodificadores, haremos el ejemplo de un decodificador BCD/Display siete segmentos.

• Los displays siete segmentos son unos dispositivos de salida muy utilizado para mostrar números.

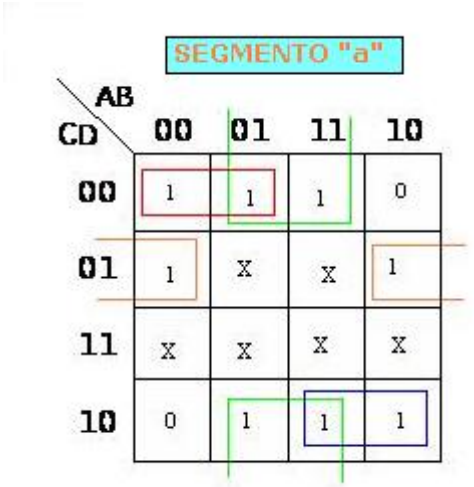


•Los 7 segmentos del display están marcados con las letras de la “a” a la “g”, cada uno de estos segmentos lucirá cuando la entrada correspondiente a su letra tenga un nivel lógico activo que será uno o cero dependiendo de si son activos por unos o por ceros.

•En nuestro ejemplo de decodificador, nos llegará a la entrada un código BCD, y debemos activar las salidas para poder atacar a un display de 7 segmentos.

DECODIFICADORES

ENTRADAS				SALIDAS						
A (2^0)	B (2^1)	C (2^2)	D (2^3)	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	0	1	1	0	0	0	0
0	1	0	0	1	1	0	1	1	0	1
1	1	0	0	1	1	1	1	0	0	1
0	0	1	0	0	1	1	0	0	1	1
1	0	1	0	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
1	1	1	0	1	1	1	0	0	0	0
0	0	0	1	1	1	1	1	1	1	1
1	0	0	1	1	1	1	0	0	1	1



•El caso de los términos “No Importa” del mapa de Karnaugh, son los correspondientes a las combinaciones de 4 bits no existentes en BCD.

•En éste caso, evitaremos agrupar las condiciones de indiferencia (Rechazo de datos falsos)

$$a = B\overline{D} + AC\overline{D} + \overline{A}\overline{C}\overline{D} + \overline{B}\overline{C}D$$

$b = \overline{C}\overline{D} + \overline{B}\overline{C} + AB\overline{D} + \overline{A}\overline{B}\overline{D}$

$c = C\overline{D} + A\overline{D} + \overline{B}\overline{C}$

$d = B\overline{C}\overline{D} + \overline{A}B\overline{D} + \overline{A}\overline{B}\overline{C} + A\overline{B}C\overline{D}$

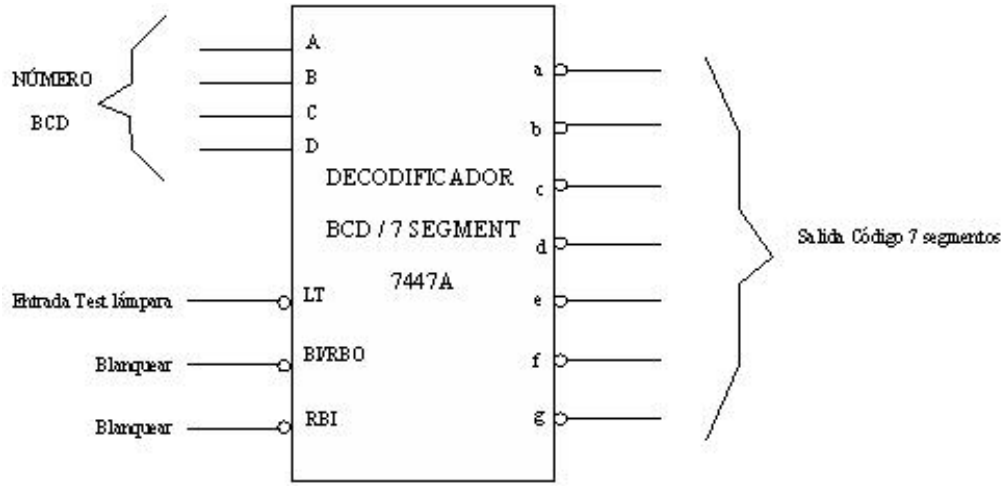
$e = \overline{A}\overline{B}\overline{C} + \overline{A}BD$

$f = \overline{B}C\overline{D} + \overline{A}C\overline{D} + \overline{B}\overline{C}D + \overline{A}\overline{B}\overline{D}$

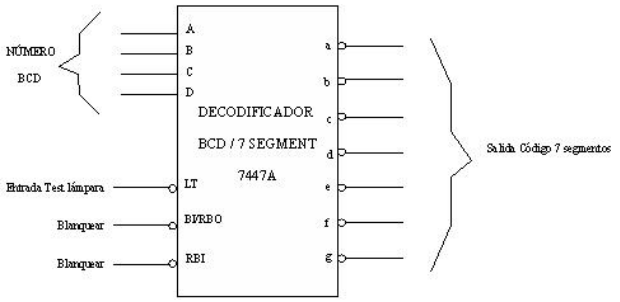
$g = \overline{B}C\overline{D} + B\overline{C}\overline{D} + \overline{B}\overline{C}D + \overline{A}C\overline{D}$

DECODIFICADORES

- Como ejemplo de un decodificador comercial de este segundo grupo, comentaremos el decodificador 7447A que es un decodificador TTL BCD a siete segmentos.



DECODIFICADORES



ENTRADA LE

•En ocasiones existe otro terminal llamado LE (Latch Enable) que da la posibilidad de memorizar el dato de entrada, cuando activamos esa entrada, a la salida tendremos la decodificación del dato de la entrada, y si éste cambia, también cambiará la salida.

• Si esta desactivado, mantendrá activas las salidas de la última decodificación hecha cuando estaba activo

DECODIFICADORES

ENTRADA LT

- La entrada LT “Lamp test” hace activarse a todas las salidas, para comprobar si se encuentran operativos todos los segmentos, siempre y cuando BI/RBO (Entrada de borrado / Salida de borrado de rizado) se encuentre a nivel alto (inactivo).

ENTRADA RBI (Ripple Blanking Input)

- Esta entrada se utiliza para evitar que el display muestre el cero (por ejemplo ceros a la izquierda) cuando está activa, de la misma forma, la salida RBO informa de dicha eventualidad.

DECODIFICADORES

ENTRADA/SALIDA BI/RBO (Blanking Input / Riple Blanking Output) Este terminal puede actuar como entrada o como salida:

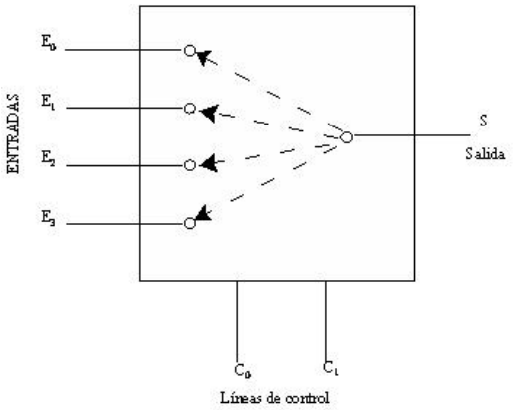
ENTRADA BI : Si se activa apaga todos los segmentos del display.

SALIDA RBO : Esta salida se activa cuando el dato de entrada es el cero, siempre y cuando RBI esté activado, en caso contrario no hace nada.

- Para el normal funcionamiento del decodificador, las tres últimas entradas comentadas deben de estar a nivel alto, en cuyo caso las salidas activadas a nivel bajo serán las correspondientes al código 7 segmentos de las entradas activadas a nivel alto.

MULTIPLEXORES

- Los multiplexores son circuitos combinacionales con varias entradas y una única salida de datos y dotados de varias entradas de control.
- Dependiendo de la combinación que tengamos en la entrada de control se seleccionará una y sólo una de las entradas de datos que se pasará a la única salida.



- A través de las líneas de control podemos seleccionar la entrada que queremos pasar a la salida del multiplexor, si dichas entradas de control están a 00, el valor lógico a la entrada de E0 pasará a la salida, si están a 01, será la entrada E1 la que pase a la salida ...etc.
- Lógicamente el número de entradas de control necesarias depende del número de entradas de datos presentes.

• Si tenemos N entradas de datos necesitaremos n entradas de control de tal forma que se cumpla : $N \leq 2^n$

MULTIPLEXORES

- Dependiendo del tipo de información que se desee manejar a la entrada de datos del multiplexor, diferenciaremos entre:

  - **Multiplexores Analógicos:** Las señales de entrada de datos pueden ser tanto de carácter analógico como digital aunque generalmente es información analógica

  - **Multiplexores Digitales:** Las señales de entrada de datos es siempre de naturaleza analógica.

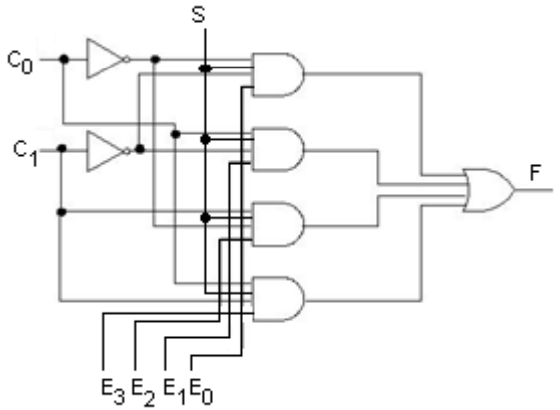
- En ambos casos las señales de control son digitales

MULTIPLEXORES

- **DISEÑO DE MULTIPLEXORES DIGITALES**
  - **Se propone la implementación de un multiplexor de cuatro canales (N=4), con una entrada de "enable" capaz de permitir o impedir la transmisión de información de las entradas a la salida**
  - Con la información disponible se deduce que necesitaremos n = 2 entradas de control ($2^2 = 4 = N$) que llamaremos C_0, C_1, A la salida la llamaremos "F", as entradas de datos serán (E_0, E_1, E_2 y E_3) y "S" será la entrada de habilitación (Enable)

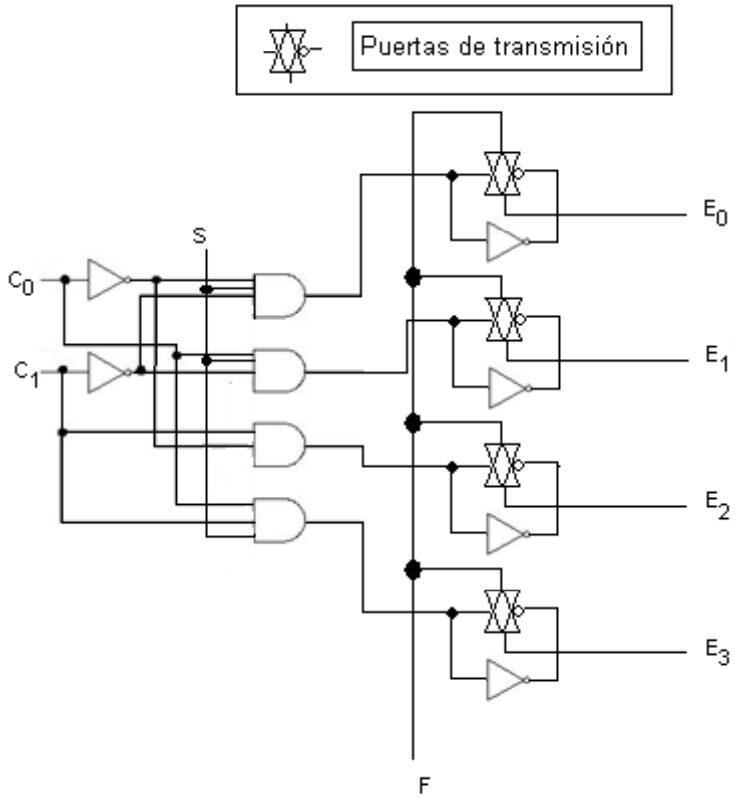
Entradas de control			Entradas de datos	Salida
S	C_1	C_0	E_3 E_2 E_1 E_0	F
0	X	X	X X X X	0
1	0	0	X X X E_0	E_0
1	0	1	X X E_1 X	E_1
1	1	0	X E_2 X X	E_2
1	1	1	E_3 X X X	E_3

$$F = S\,(\overline{C}_1\overline{C}_0E_0+\overline{C}_1C_0E_1+C_1\overline{C}_0E_2+C_1C_0E_3)$$



MULTIPLEXORES

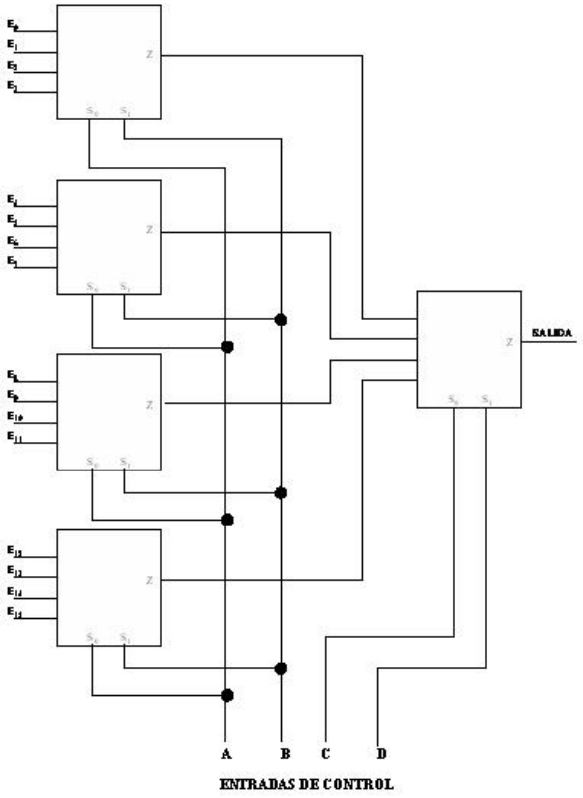
- **DISEÑO DE MULTIPLEXORES ANALOGICOS**
  - Con una pequeña modificación podemos convertir el multiplexor anterior en otro de tipo analógico



Una puerta de transmisión es básicamente un dispositivo que permite o no el paso de una señal entre sus terminales similar a un interruptor que se abre o cierra en función de una señal de control

MULTIPLEXORES

- **EXTENSIÓN DE LA CAPACIDAD DE LOS MULTIPLEXORES**
- En ocasiones podemos necesitar disponer de un multiplexor de mayor número de entradas de las que disponemos en nuestro C.I. Por ejemplo supongamos que sólo disponemos de multiplexores de 4 canales, pero necesitamos trabajar con uno de 16 canales del que no disponemos.



MULTIPLEXORES

- **APLICACIONES DE LOS MULTIPLEXORES**
- Los multiplexores pueden ser empleados para realizar funciones distintas a la de selectores de datos. Algunas de estas funciones son:
    - Generación de funciones lógicas.
    - Convertidor paralelo-serie
- Generación de funciones lógicas: Un posible procedimiento a seguir para la generación de funciones lógicas usando multiplexores, es el siguiente:
    - A partir del número de variables de la función, obtener el tipo de multiplexor a emplear, esto es, si el número de variables de la función es n+1, el Multiplexor debe tener $N=2^n$ entradas de datos.
    - Obtener la forma canónica de la función en suma de productos.
    - Identificar n variables de la función con las entradas de control del multiplexor.
    - Con la información de la forma canónica, deducir los valores a conectar en las entradas de datos del multiplexor

MULTIPLEXORES

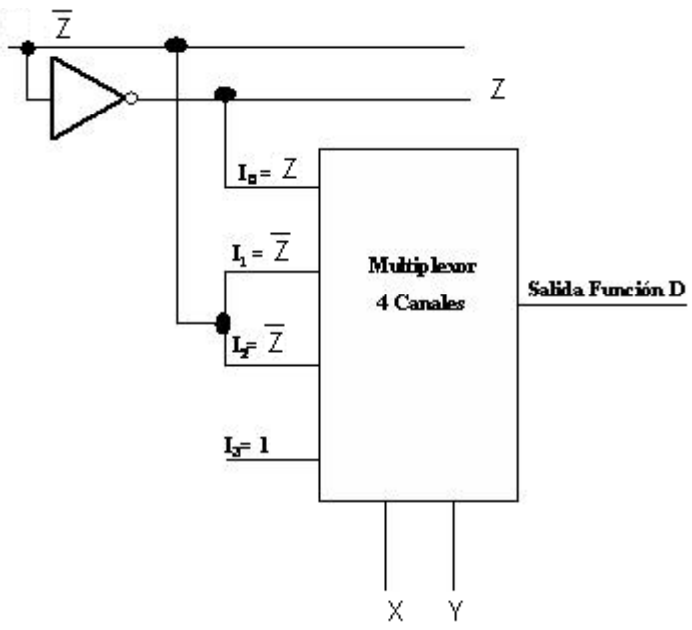
$$f = xy + \bar{x}y\bar{z} + x\bar{y}\bar{z} + \bar{x}yz$$

1º Según la función planteada n+1=3 variables tiene la función. Como $N=2^n$, en este caso tendremos N = 4 entradas de datos deberá tener el multiplexor, lo que implica dos entradas de control.

2º La forma canónica de la función será $f = xyz + xy\bar{z} + \bar{x}y\bar{z} + x\bar{y}\bar{z} + \bar{x}\bar{y}z$

3º Las variables X,Y decidimos que sean las variables de control del multiplexor siendo Y=LSB

	X	Y	F
E_0	0	0	Z
E_1	0	1	$\bar{Z}$
E_2	1	0	$\bar{Z}$
E_3	1	1	1



MULTIPLEXORES

- Conversor paralelo serie: Otra posible aplicación es la de implementar un conversor paralelo serie con la ayuda de un contador.