Tema 6. Circuitos Combinacionales Lógicos

Objetivos

- Comprender el funcionamiento de los circuitos combinacionales MSI básicos
- Analizar el diseño de estos circuitos
- Comprender la implementación de funciones lógicas mediante decodificadores y multiplexores
- Asimilar la metodología de asociación de multiplexores y demultiplexores.
- Comprender el diseño e implementación de los circuitos generadores y comprobadores de paridad, y de los conversores de código.

Tema 6. Circuitos Combinacionales Lógicos

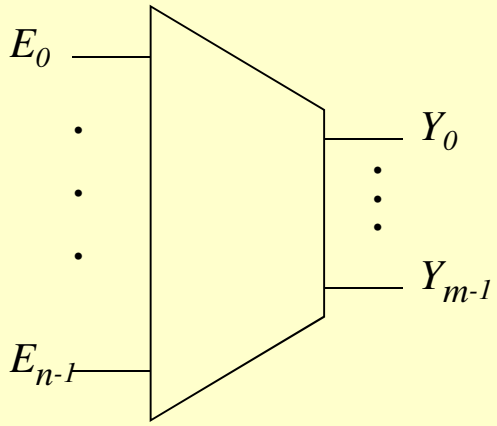
Contenido

- Codificadores
- Decodificadores
- Multiplexores
- Demultiplexores
- Asociación de Multiplexores y demultiplexores
- Conversores de código.
- Generadores y comprobadores de paridad.

Codificadores

Codificadores

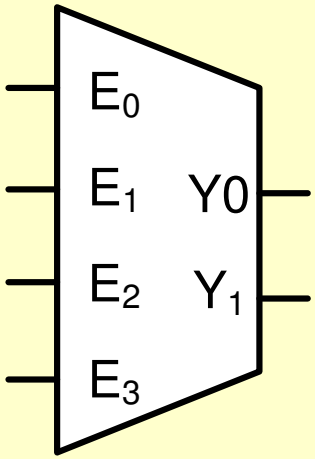
- Un codificador para un determinado código es un bloque combinacional con *n* entradas ($E_{n-1}, \ldots, E_0$) y *m* salidas ($Y_{m-1}, \ldots, Y_0$) siendo *n* el nº de caracteres a codificar y *m* el nº de bits que tiene el código binario.
- *Representación a nivel de bloque.*



Codificadores

Codificadores sin prioridad

- En un codificador **sin prioridad** en cada instante de tiempo **tiene que haber siempre una sola entrada activa**. En la salida se representa en el código correspondiente la única entrada activa.
- **Ejemplo: codificador binario de 4 a 2**

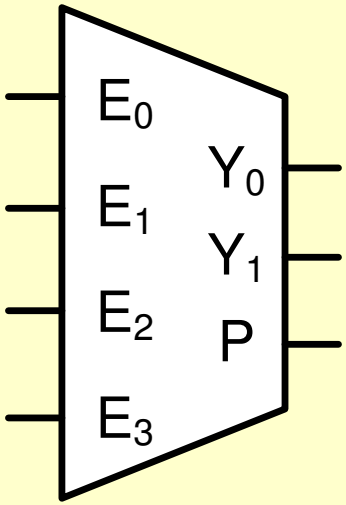


E_3	E_2	E_1	E_0	Y_1	Y_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

Codificadores

Codificadores con prioridad

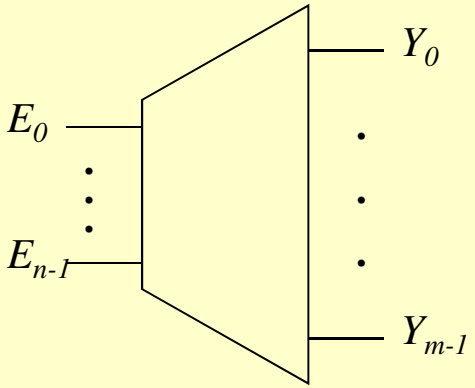
- ***Ejemplo. Codificador con prioridad de 4 a 2.***
  - Esquema de prioridad: E_3 (+) → E_2 ⟶ E_1 ⟶ E_0 (-)
  - La salida P se activa si no hay activa ninguna entrada.



E_3	E_2	E_1	E_0	Y_1	Y_0	P
1	-	-	-	1	1	0
0	1	-	-	1	0	0
0	0	1	-	0	1	0
0	0	0	1	0	0	0
0	0	0	0	0	0	1

Decodificadores

- Realiza la función contraria a la del codificador. La función de encontrar el carácter representado en un determinado código se denomina ***decodificación***.
- Un decodificador para un determinado código es un bloque combinacional con con *n* entradas ($E_{n-1}, \ldots, E_0$) y *m* salidas ($Y_{m-1}, \ldots, Y_0$) siendo *n* el número de bits que se utiliza en el código y *m* el número de caracteres que se están decodificando.
- ***Representación a nivel de bloque.***



- ***Función general.***
  - En cada instante de tiempo se activa solamente la salida correspondiente al carácter codificado en sus entradas.
- ***Especificación.***
  - Se indica el tamaño según el formato ***n*** a ***m***, siendo n < m, generalmente

Decodificadores

- *Los decodificadores más comunes son los **binarios**.*
  - En las entradas se aplica las palabras del código binario natural.
  - Se cumple que $m = 2^n$. Por tanto, se especifican: **n a 2^n**
  - Las salidas corresponden a los números decimales, que se representan Y_i, donde ***i*** es el número decimal.
  - Para cada valor binario de entrada se activa una única salida.
  - Se activa la salida cuyo subíndice numérico coincide con el equivalente decimal del número binario que hay en sus entradas.
- ***Ejemplo. Decodificador binario de 2 a 4 con salidas activas a nivel alto.***

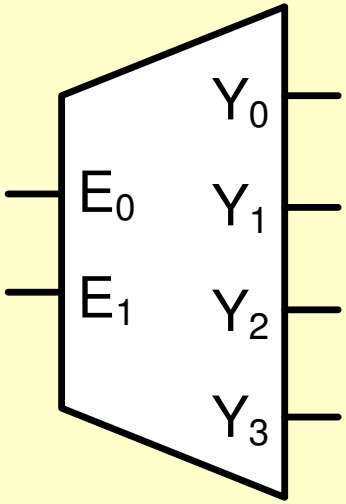
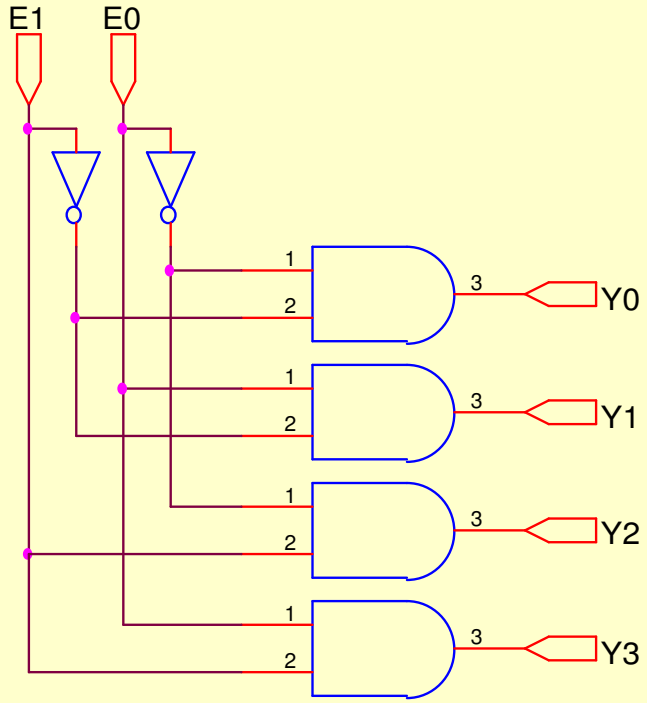


Tabla de verdad

E_1	E_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

Decodificadores binarios

- ***Diagrama lógico del decodificador binario de 2 a 4.***



Funciones lógicas

$$Y_0 = \overline{E}_1 \, \overline{E}_0$$
$$Y_1 = \overline{E}_1 \, E_0$$
$$Y_2 = E_1 \, \overline{E}_0$$
$$Y_3 = E_1 \, E_0$$

- *La estructura interna de un decodificador binario de n a 2^n con salidas activas a nivel alto consta de 2^n puertas lógicas AND de n entradas y n puertas NOT.*
- *Cada puerta AND genera una salida.*

Decodificadores binarios

- **Cada salida se activa solamente para un único minterm de las dos entradas**.
- Esto se puede deducir fácilmente de la tabla de verdad y de las funciones lógicas de las salidas.

Tabla de verdad del decodificador binario de 2 a 4

m_i	E_1	E_0	Y_3	Y_2	Y_1	Y_0
m_0	0	0	0	0	0	1
m_1	0	1	0	0	1	0
m_2	1	0	0	1	0	0
m_3	1	1	1	0	0	0

Funciones lógicas de las salidas del decodificador binario de 2 a 4

$$Y_0 = \overline{E}_1\,\overline{E}_0 = m_0$$
$$Y_1 = \overline{E}_1\,E_0 = m_1$$
$$Y_2 = E_1\,\overline{E}_0 = m_2$$
$$Y_3 = E_1\,E_0 = m_3$$

- *De forma genérica un decodificador binario de n a 2^n con salidas activas a nivel alto genera los 2^n minterms de sus n variables de entrada, y se cumple* $\boldsymbol{Y_i = m_i}$
  - Cada salida genera el correspondiente minterm.
  - Por tanto, un decodificador se puede usar para implementar una función lógica.

Decodificadores binarios

- **Implementación de una función lógica mediante decodificadores con salidas activas a nivel lógico alto partiendo de la Suma de minsterms.**
  - Dado un decodificador de ***n a 2^n*** con salidas activas a nivel alto genera en sus salidas los **2^n minterms** de sus **n entradas**:
    - Si se conectan las variables de entrada de la F. L. a las entradas del decodificador, éste genera en cada una de sus salidas uno de los 2^n minterms de la función lógica.
    - De forma que si sumamos los minterms para los que la función lógica vale 1, podremos implementar la función.

$$F(x_{n-1},\ldots,x_0)=\sum m_i=\sum Y_i$$ (Sólo para los minterms que valen 1)

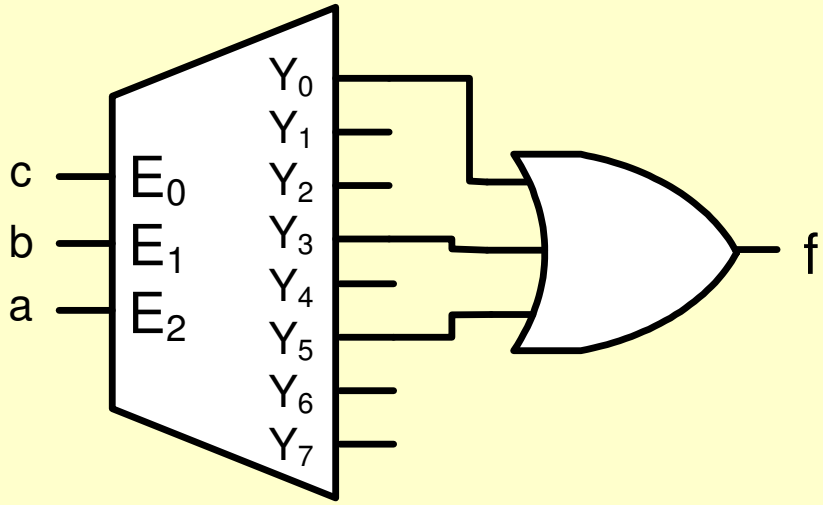
    - Por tanto, una función lógica de n variables de entrada se podrá implementar mediante un decodificador ***n a 2^n*** con salidas activas a nivel alto y una puerta OR.
  - ***Procedimiento.***
    - Expresar la función lógica como suma de minterms.
    - Seleccionar el decodificador necesario. Tendrá tantas entradas como variables de entrada la función.
    - Se conectan las variables de entrada de la función a las entradas del decodificador teniendo en cuenta los pesos.
    - Se conecta a la puerta OR las salidas del decodificador correspondientes a los minterms-1 de la función.

Decodificadores binarios

- ***Ejemplo. Implementación de la función lógica:***

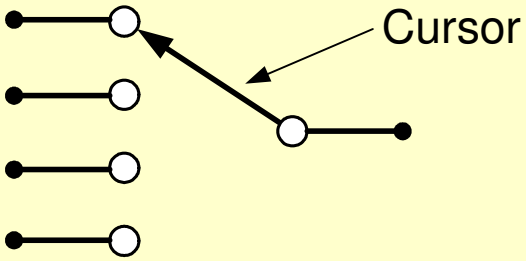
$$f(a,b,c) = \sum m(0,3,5) + d(2,6) = m_0 + m_3 + m_5 = Y_0 + Y_3 + Y_5$$

  - La función lógica está representada como suma de minterms.
  - La función tiene 3 variables de entrada, por lo que será necesario un decodificador de 3 a 8.
  - Las variables a, b y c se conectan respectivamente a E_2, E_1 y E_0.
  - Se conecta a la puerta OR las salidas del decodificador correspondientes a los minterms-1 de la función, es decir los minterms 0, 3 y 5.



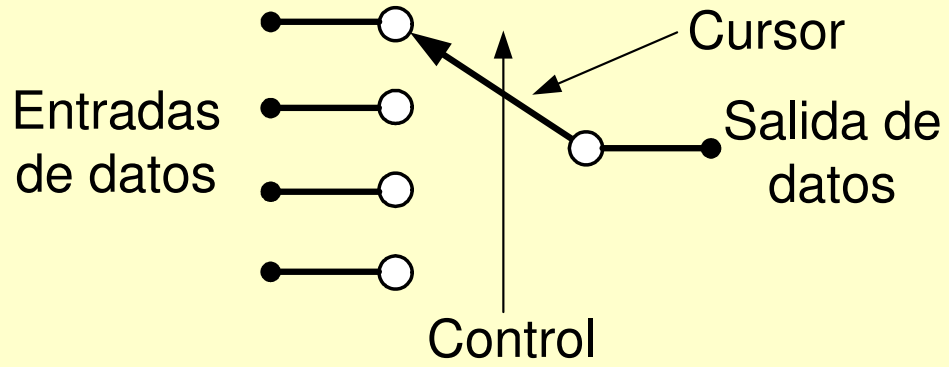
Multiplexores

- Un multiplexor realiza la función de un conmutador de múltiples posiciones.



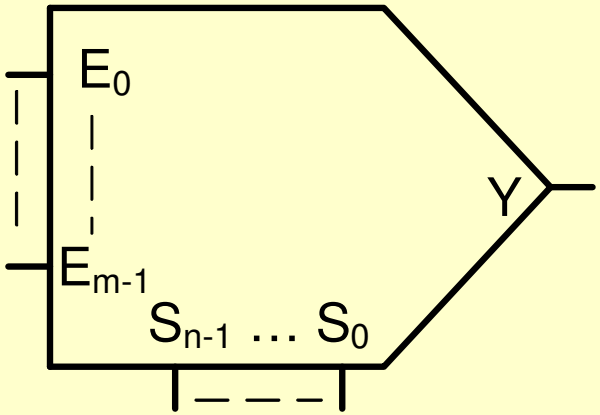
  - Tiene varias entradas y una sola salida.
  - Dependiendo de la posición del cursor, una única entrada se conecta a la salida

- Un multiplexor además tiene unas entradas de control que indican la entrada que se conecta a la salida.



Multiplexores

- *Un multiplexor (mux) es un bloque combinacional que tiene* ***n entradas de control de selección*** *(S_{n-1}, ..., S_0),* ***m entradas de datos*** *(E_0, ..., E_{m-1}) y* ***una única salida de datos*** *Y.*
- *Lo común es que $m = 2^n$. Se denomina multiplexor binario.*
- ***Representación a nivel de bloque.***

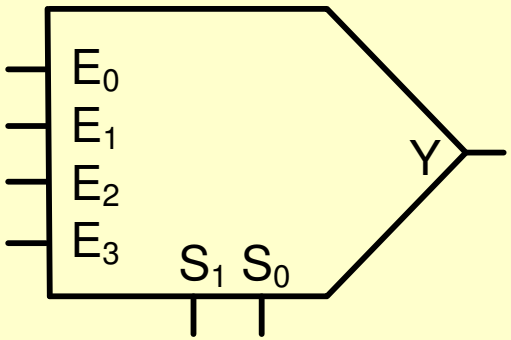


- ***Especificación.***
  - m a 1 ó 2^n a 1
- ***Función.***
  - Un multiplexor conecta a su única salida de datos una y sólo una de las entradas de datos, y en concreto, aquella cuyo subíndice coincide con el equivalente decimal del número binario aplicado en las entradas de control de selección.

Multiplexores

- ***Ejemplo. Mutiplexor de 4 a 1.***
  - Tendrá *4 entradas de datos*: E_3, E_2, E_1, y E_0
  - $4 = 2^2 \quad \Rightarrow n = 2$, es decir *2 entradas de control de selección:* S_1 y S_0
  - *Una salida de datos: Y*

Representación a nivel de bloque



- ***Tabla de verdad.***
  - Como la salida Y tiene el valor de la entrada de datos indicada por S_1 y S_0, se puede simplificar la tabla usando en la columna de salida las variables de las entradas de datos.

S_1	S_0	Y
0	0	E_0
0	1	E_1
1	0	E_2
1	1	E_3

- ***Función lógica de la salida.***
  - Se describe el funcionamiento de la salida Y como suma de productos.
  - Y es 1 si la entrada de datos seleccionada es 1. Por tanto, habrá 4 términos producto:
    - Cada entrada de datos por el valor de S_1 y S_0 que la selecciona.

$$\mathbf{Y = \overline{S}_1\,\overline{S}_0\,E_0 + \overline{S}_1\,S_0\,E_1 + S_1\,\overline{S}_0\,E_2 + S_1\,S_0\,E_3}$$

Multiplexores

- ***Estructura interna de un multiplexor de 2^n a 1.***
  - ***Un multiplexor de 2^n a 1*** *se construye mediante un* ***decodificador modificado de n a 2^n*** *y* ***una puerta OR de 2^n entradas***.
    - ✓ *Las puertas AND del decodificador tienen una entrada adicional a la que se conecta la entrada de datos correspondiente.*
    - ✓ *Las salidas de las puertas AND se conectan a la puerta OR que genera la salida Y del multiplexor.*
- *Como un* ***multiplexor*** *de* ***2^n a 1*** *realiza la suma de los productos de los* ***2^n*** *minterms de las entradas de control de selección junto con la correspondiente entrada de datos, éste podrá* ***implementar una función lógica de n variables de entrada***.
- *Una función lógica de n variables de entrada se puede implementar mediante un multiplexor de n-1 entradas de control de selección:* ***mux de 2^{n-1} a 1,*** *conectando las entradas de datos en función de una de las variables de entrada de la función lógica, como se verá posteriormente.*

Multiplexores

- ***Metodología de implementación de una función lógica de n variables mediante un mux de 2^{n-1} a 1.***
  1. Determinar el tamaño del mux.
  2. Expresar la función lógica mediante su tabla de verdad.
  3. Se asignan las n-1 variables de mayor peso de la función lógica a las entradas de control de selección por orden de peso.
  4. La función lógica se obtiene en la salida del mux, Y = F.
  5. Determinar los valores de las entradas de datos del mux:
     - Cada entrada de datos se selecciona para dos minterms.
     - Si la función lógica tiene el mismo valor para ambos minterms, la entrada de datos se conecta a ese valor.
     - Si la función lógica tiene un valor diferente para ambos minterms, la entrada de datos dependerá de la variable libre.
       - Se compara el valor de la función lógica con el de la variable libre para ambos minterms.
       - Hay dos casos:
         - Tienen el mismo valor. Entrada de datos es igual a la variable libre.
         - Tienen distinto valor. Entrada de datos es igual a la variable libre complementada.

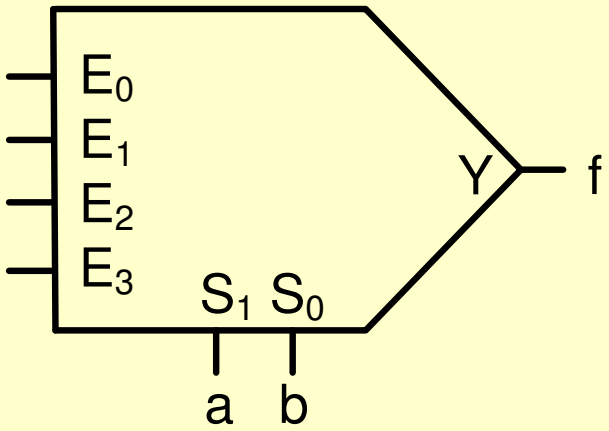
Multiplexores

- ***Ejemplo. Implementación de la función lógica***

  $$f(a, b, c) = \Sigma m(3, 4, 6, 7)$$

1. Como n = 3, se necesita un mux de 2^{3-1} a 1, es decir de 4 a 1.
2. Se expresa la función lógica mediante su tabla de verdad.
3. Se asignan las n-1 variables de mayor peso de la función lógica a las entradas de control de selección por orden de peso.
4. Se asigna la variable de salida de la función a la del mux.

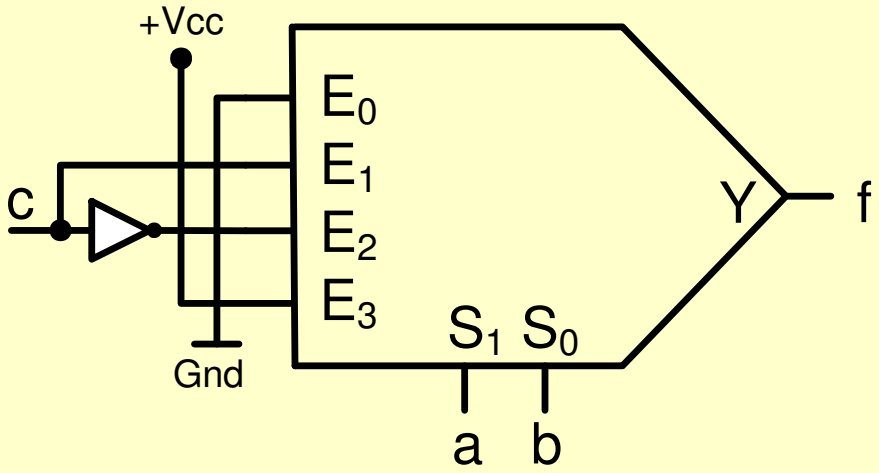
a	b	c	f
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1



Multiplexores

5. Se determina el valor de las entradas de datos del mux.

a	b	c	f	E_i
0	0	0	0	$E_0 = 0$
0	0	1	0	
0	1	0	0	$E_1 = c$
0	1	1	1	
1	0	0	1	$E_2 = \overline{c}$
1	0	1	0	
1	1	0	1	$E_3 = 1$
1	1	1	1	



Conclusiones:

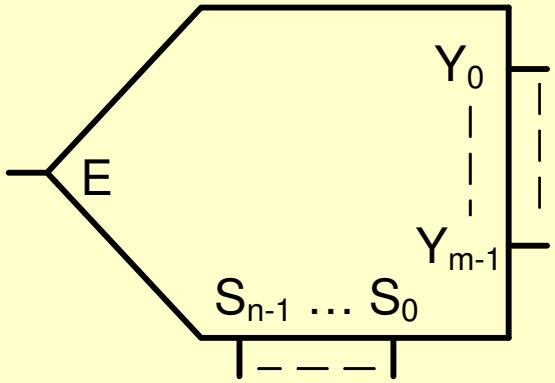
Se sigue usando el mux como selector de datos.

Debe poner en su salida el valor de la función lógica para cada uno de sus minters.

Se puede realizar otra asignación de variables de la función a la entradas de control de selección, pero la realizada simplifica el proceso

Demultiplexores

- Un demultiplexor realiza la función contraria del multiplexor.
- *Un demultiplexor es un bloque combinacional que tiene* ***n entradas de control de selección*** *(S_{n-1}, ..., S_0),* ***una única entrada de datos*** *E y* ***m salidas de datos*** *(Y_0, ..., Y_{m-1}).*
- *Lo común es que $m = 2^n$. Se denomina demultiplexor binario.*
- ***Representación a nivel de bloque.***



- ***Especificación.***
  - 1 a m ó 1 a 2^n
- ***Función.***
  - Un demultiplexor conecta su única entrada de datos con una y solo una de las salidas de datos, y en concreto, aquella cuyo subíndice coincide con el equivalente decimal del número binario aplicado en las entradas de control de selección.

Demultiplexores

- Un decodificador con entrada de habilitación funciona como un demultiplexor.
  - La entrada de habilitación del decodificador es la entrada de datos del demultiplexor.
  - Las entradas del decodificador son las entradas de control de selección del demultiplexor
  - Las salidas del decodificador son las salidas de datos del demultiplexor.

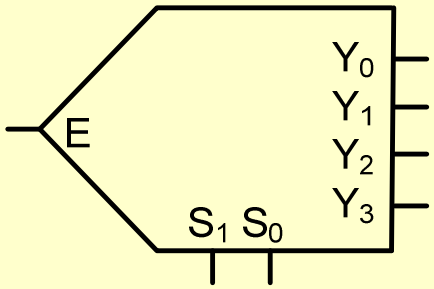


Tabla de verdad de un decod. 2 a 4 con entrada de habilitación

E / EN	S_1 / E_1	S_0 / E_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	1	1	0	0	0	0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0

S_1	S_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	E
0	1	0	0	E	0
1	0	0	E	0	0
1	1	E	0	0	0

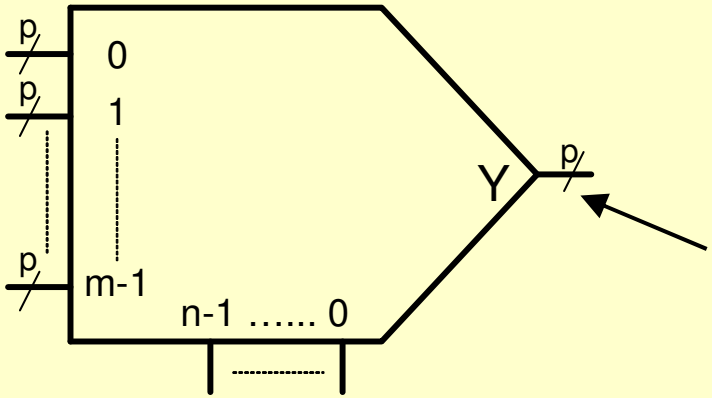
- *Se deduce fácilmente que la tabla de verdad corresponde a un demultiplexor de 1 a 4.*

- ***Funciones lógicas.***

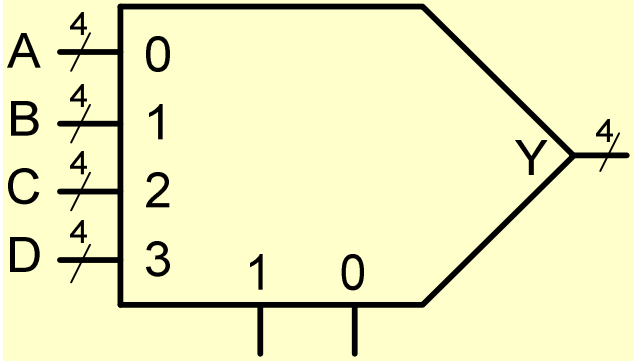
$$Y_0 = E\,\overline{S}_1\,\overline{S}_0$$
$$Y_1 = E\,\overline{S}_1\,S_0$$
$$Y_2 = E\,S_1\,\overline{S}_0$$
$$Y_3 = E\,S_1\,S_0$$

Asociación de multiplexores y demultiplexores.
Ampliación del número de bits. Multiplexores y demultiplexores

- ***Representación genérica de un mux de 2^n a 1 de p bits.***



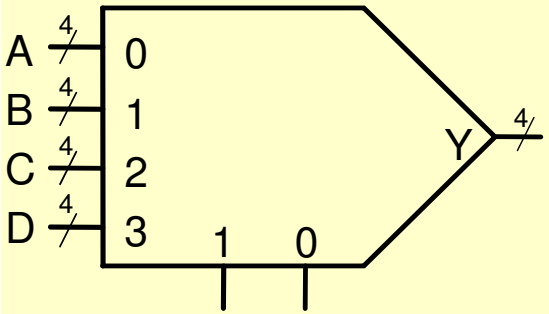
- ***Multiplexor de 2^n a 1 de p bits.***
  - *El número de entradas de datos no cambia.*
  - *Se aumenta el número de bits de la información que se conecta a sus entradas y salida de datos.*
    - *De 1 bit se aumenta a p bits.*
    - *Selecciona 1 palabra de p bits entre 2^n palabras de p bits.*
  - *Esta formado por p mux de 2^n a 1.*
  - La barra inclinada y el número indica el número de bits del mux, es decir el número de mux de los que consta.



- ***Multiplexor de 4 a 1 de 4 bits.***
  - Permite seleccionar un número binario de 4 bits entre 4 posibles.
  - Consideremos los números A, B, C y D de 4 bits
- *Esta formado por 4 mux de 4 a 1.*

S_1	S_0	Y
0	0	A
0	1	B
1	0	C
1	1	D

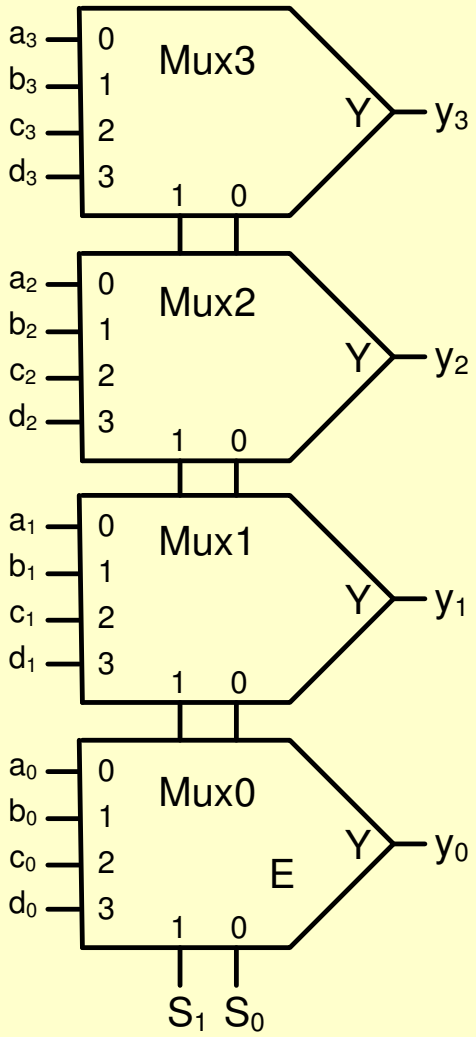
Asociación de multiplexores y demultiplexores.
Ampliación del número de bits. Multiplexores y demultiplexores



S_1	S_0	Y	y_3	y_2	y_1	y_0
0	0	A	a_3	a_2	a_1	a_0
0	1	B	b_3	b_2	b_1	b_0
1	0	C	c_3	c_2	c_1	c_0
1	1	D	d_3	d_2	d_1	d_0

Mux3 Mux2 Mux1 Mux0

- ***La salida del mux es de 4 bits***
  - *Se necesitan 4 mux de 4 a 1*
- ***Cada mux genera una de las 4 salidas de datos***
- ***A las entradas de datos de cada mux se conecta los bits de mismo peso de los 4 números, y cuyo peso coincide con el de la salida que genera.***



Asociación de multiplexores y demultiplexores.
Ampliación del tamaño de los multiplexores

- ***Ampliación del tamaño usando multiplexores del mismo tamaño.***
  - *Si n_1 es el número de entradas de control de selección de los mux que se van a usar y n_2 el número de entradas de control de selección del mux que se va a implementar, se debe cumplir que:*

$$p = \frac{n_2}{n_1} = \text{número entero}$$

  - *Se conectan los mux necesarios en una estructura de árbol de **p niveles** , en la que todos los mux del mismo nivel están controlados por las mismas señales de control de selección.*
  - ***Ejemplo. Mux de 8 a 1 mediante mux de 2 a 1.***

$$\left.\begin{array}{ll} 2 \text{ a } 1 = 2^1 \text{ a } 1 & \Rightarrow n_1 = 1 \\ 8 \text{ a } 1 = 2^3 \text{ a } 1 & \Rightarrow n_2 = 3 \end{array}\right\} \Rightarrow p = \frac{n_2}{n_1} = \frac{3}{1} = 3$$

    - Consta de 3 niveles de mux de 2 a 1.

Asociación de multiplexores y demultiplexores.
Ampliación del tamaño de los multiplexores

- *Se comienza poniendo los mux desde las entradas hacia la única salida.*
- *La salida de datos de los mux de un nivel se conectan a las entradas de datos de los mux del siguiente nivel.*

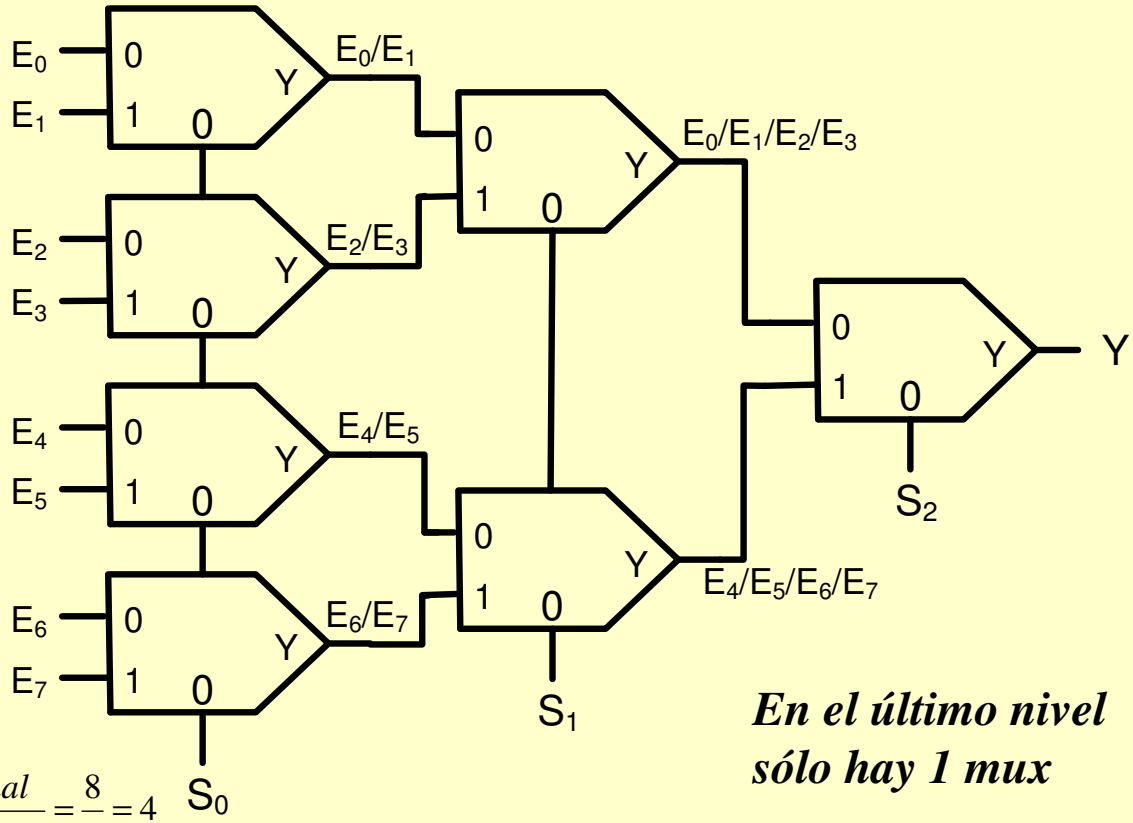


Tabla de verdad del mux de 8 a 1

S_2	S_1	S_0	Y
0	0	0	E_0
0	0	1	E_1
0	1	0	E_2
0	1	1	E_3
1	0	0	E_4
1	0	1	E_5
1	1	0	E_6
1	1	1	E_7

Nº de mux del 1º nivel

$$\frac{N^{o}\ entradas\ datos\ mux\ final}{N^{o}\ entradas\ datos\ mux\ usados} = \frac{8}{2} = 4$$

- *Se conectan la entrada de control de selección de menor peso a la entrada de control de todos los mux del 1º nivel.*

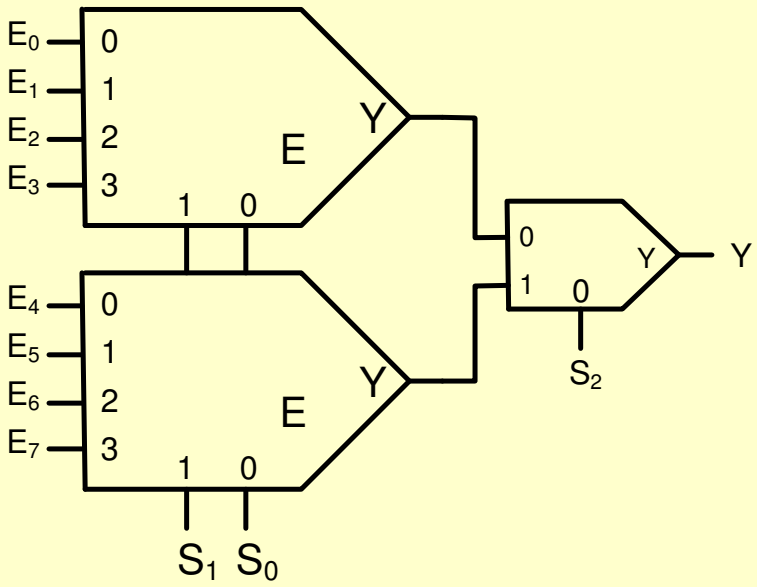
En el último nivel sólo hay 1 mux

En el 2º nivel se necesitan 4/2 = 2 mux

Asociación de multiplexores y demultiplexores.
Ampliación del tamaño de los multiplexores

- **Ampliación del tamaño usando el mínimo número de multiplexores.**
  - *La estructura de árbol tiene dos niveles.*
  - *El 1º nivel está compuesto de 2 mux.*
    - *El tamaño de los multiplexores será* ⇨ ***Nº entradas datos mux final / 2***
  - *El 2º nivel consta de 1 mux de 2 a 1.*
  - ***Ejemplo. Mux de 8 a 1.***
    - ***El 1º nivel está compuesto de 2 mux de 8/2 a 1, es decir 4 a 1.***

S_2	S_1	S_0	Y
0	0	0	E_0
0	0	1	E_1
0	1	0	E_2
0	1	1	E_3
1	0	0	E_4
1	0	1	E_5
1	1	0	E_6
1	1	1	E_7



Asociación de multiplexores y demultiplexores.
Ampliación del tamaño de los demultiplexores

- ***Ampliación del tamaño usando demultiplexores del mismo tamaño.***
  - *Si n_1 es el número de entradas de control de selección de los demux que se van a usar y n_2 el número de entradas de control de selección del demux que se va a implementar, se debe cumplir que:*

$$p = \frac{n_2}{n_1} = \text{número entero}$$

    - *Se conectan los demux necesarios en una estructura de árbol de **p niveles** , en la que todos los demux del mismo nivel están controlados por las mismas señales de control de selección.*
    - ***Ejemplo. Demux de 1 a 8 mediante demux de 1 a 2.***

$$1 \text{ a } 2 = 1 \text{ a } 2^1 \quad \Rightarrow n_1 = 1$$

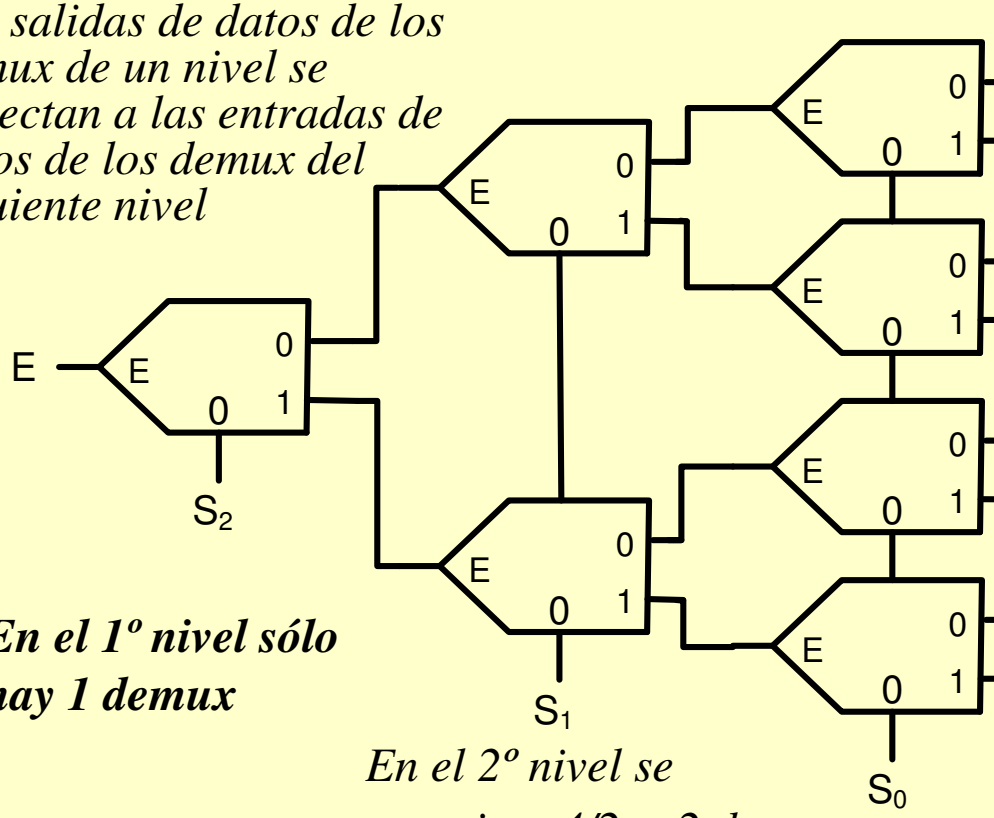
$$1 \text{ a } 8 = 1 \text{ a } 2^3 \quad \Rightarrow n_2 = 3$$

$$\Rightarrow p = \frac{n_2}{n_1} = \frac{3}{1} = 3$$

      - Consta de 3 niveles de demux de 1 a 2.

Asociación de multiplexores y demultiplexores.
Ampliación del tamaño de los demultiplexores

- *Se comienza poniendo los demux desde la salidas hacia las entradas.*
- *Las salidas de datos de los demux de un nivel se conectan a las entradas de datos de los demux del siguiente nivel*



S_2	S_1	S_0	
0	0	0	Y_0
0	0	1	Y_1
0	1	0	Y_2
0	1	1	Y_3
1	0	0	Y_4
1	0	1	Y_5
1	1	0	Y_6
1	1	1	Y_7

En el 1º nivel sólo hay 1 demux

En el 2º nivel se necesitan 4/2 = 2 demux

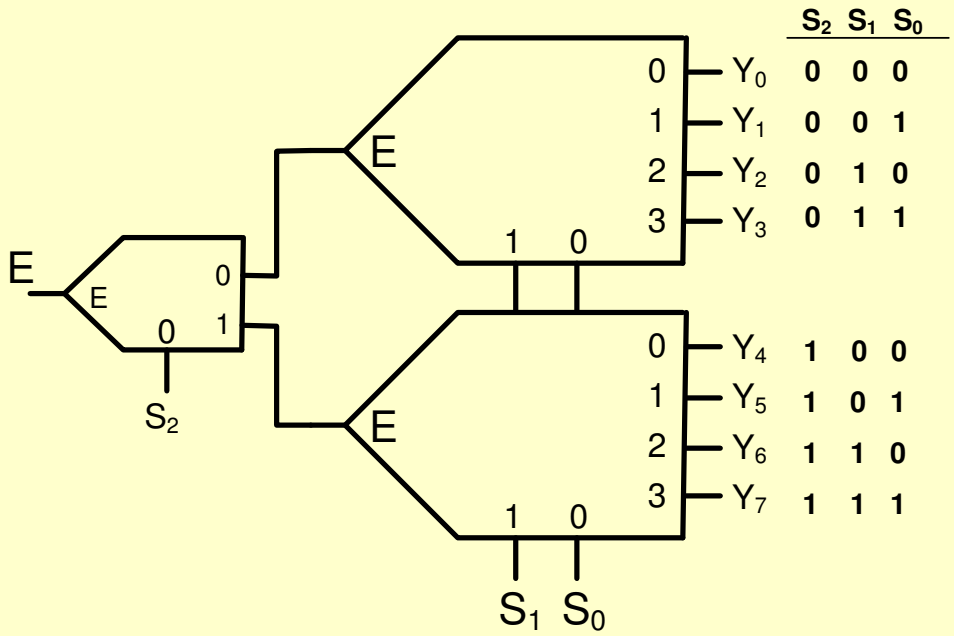
Nº demux del nivel de salida, que en este caso es el 3º

$$\frac{\text{Nº salidas demux final}}{\text{Nº salidas demux usados}} = \frac{8}{2} = 4$$

- *Se conectan la entrada de control de selección de menor peso a la entrada de todos los demux del nivel de salida.*

Asociación de multiplexores y demultiplexores.
Ampliación del tamaño de los demultiplexores

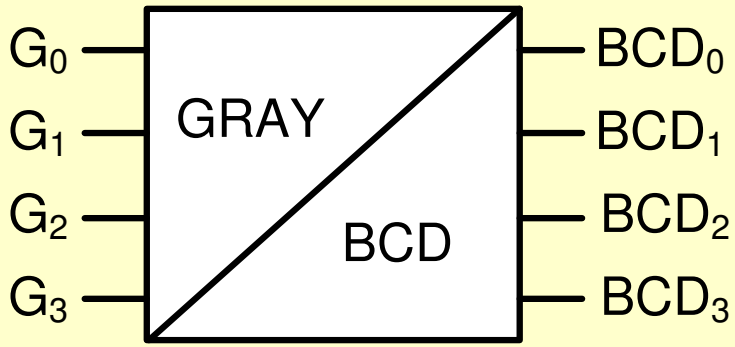
- **Ampliación del tamaño usando el mínimo número de demultiplexores.**
  - *La estructura de árbol tiene dos niveles.*
  - *El 2º nivel está compuesto de 2 demux.*
    - *El tamaño de los demultiplexores será* ⇨ **Nº salidas demux final / 2**
  - *El 1º nivel consta de 1 demux de 1 a 2.*
  - **Ejemplo. Demux de 1 a 8.**
    - **El 2º nivel está compuesto de 2 demux de 1 a 8/2, es decir 1 a 4.**



S_2	S_1	S_0	
0	0	0	Y_0
0	0	1	Y_1
0	1	0	Y_2
0	1	1	Y_3
1	0	0	Y_4
1	0	1	Y_5
1	1	0	Y_6
1	1	1	Y_7

Conversores de código

- ***Definición.***
  - Traduce una información representada en un código a otro código diferente.
  - Ejemplo, conversor de código Gray a BCD.
- ***Representación a nivel de bloque.***



- ***Metodología de diseño.***
  - *Metodología general de diseño de Sistemas Combinacionales.*
  - *Conectando en serie un decodificador y un codificador*, que tengan un código o sistema de representación común.

Conversores de código.
Diseño mediante la metodología general de diseño de sistemas combinacionales.

- Para realizar un convertidor de código utilizaremos una tabla de verdad donde las variables de entrada son los bits del código a convertir, y las de salida los bits del código convertido.
  - Los valores de entrada son las palabras del código a convertir.
  - En las variables de salida se pone la palabra del código convertido para cada palabra del código a convertir.

G_3 G_2 G_1 G_0	BCD_3	BCD_2	BCD_1	BCD_0	Decimal
0 0 0 0	0	0	0	0	0
0 0 0 1	0	0	0	1	1
0 0 1 1	0	0	1	0	2
0 0 1 0	0	0	1	1	3
0 1 1 0	0	1	0	0	4
0 1 1 1	0	1	0	1	5
0 1 0 1	0	1	1	0	6
0 1 0 0	0	1	1	1	7
1 1 0 0	1	0	0	0	8
1 1 0 1	1	0	0	1	9

G_3G_2 \ G_1G_0	00	01	11	10
00				
01				
11	1	1	-	-
10	-	-	-	-

$$BCD_3 = G_3$$

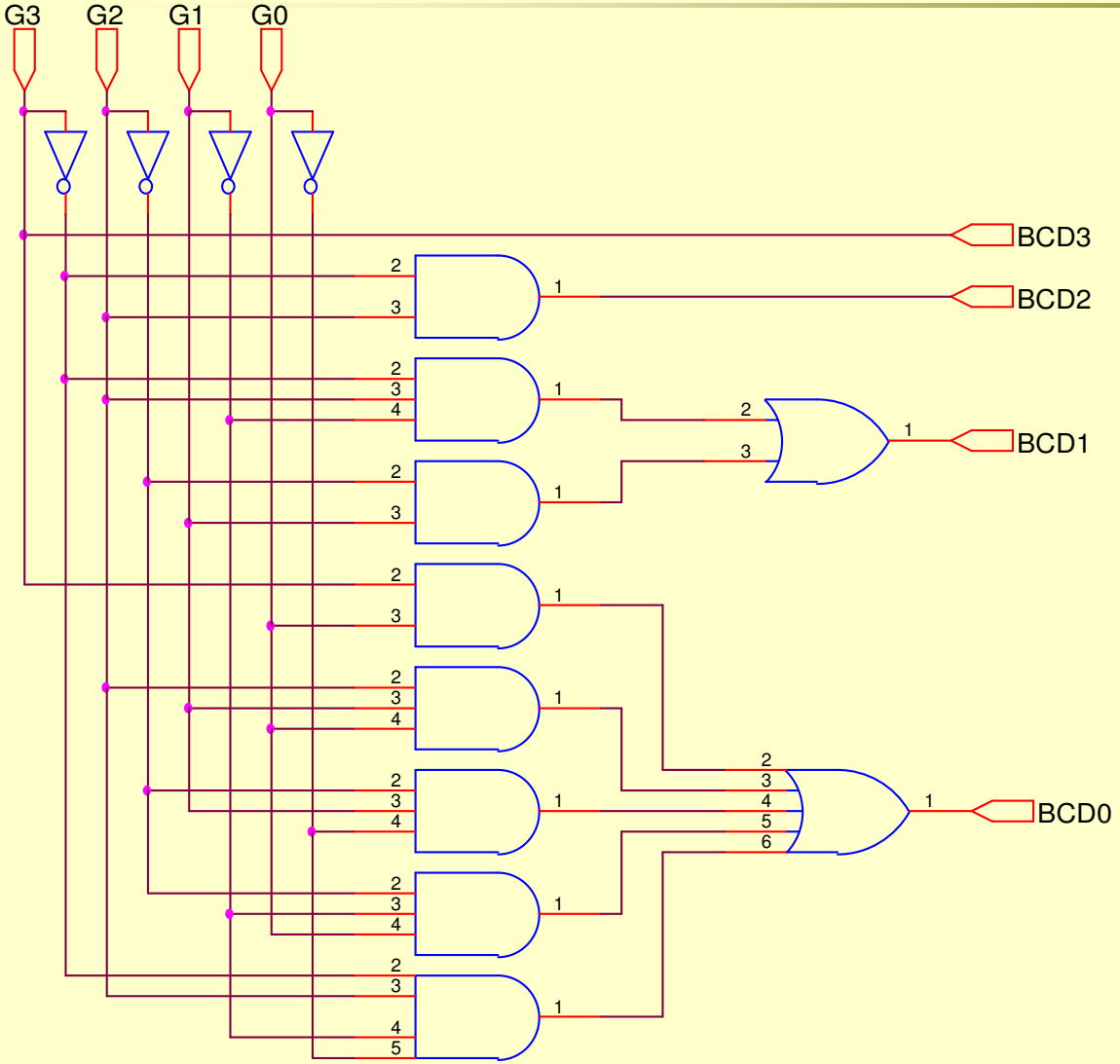
Conversores de código.
Diseño mediante la metodología general de diseño de sistemas combinacionales.

G_3G_2 \ G_1G_0	00	01	11	10
00		1		1
01	1		1	
11		1	-	-
10	-	-	-	-

$$BCD_0 = G_3 G_0 + G_2 G_1 G_0 + \overline{G}_2 G_1 \overline{G}_0 + \overline{G}_2 \overline{G}_1 G_0 + \overline{G}_3 G_2 \overline{G}_1 \overline{G}_0$$

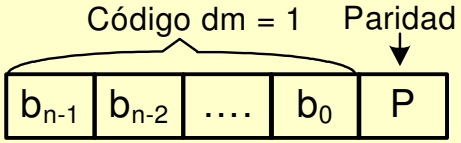
Conversores de código.

Diseño mediante la metodología general de diseño de sistemas combinacionales.



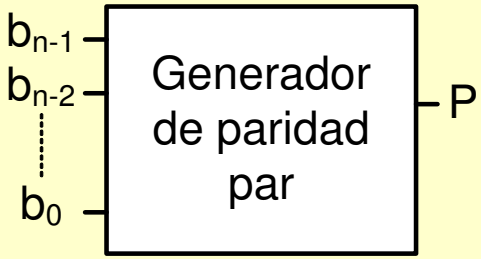
Generadores y comprobadores de paridad.

- Los ***códigos de paridad*** se forman a partir de un código de distancia mínima unidad al que se le añade un bit adicional, denominado ***bit de paridad***.



- Se distinguen dos tipos de paridad:
  - Paridad par.
  - Paridad impar.

- ***Generador de paridad par***.
  - Es un circuito combinacional que tendrá como entradas los bits del código de distancia mínima unidad: $\mathbf{b_{n-1}, b_{n-2}, ..., b_0}$, y como salida el bit de paridad par **P**.
  - En la salida **P** indica el bit de paridad par correspondiente al valor que tengan los bit del código de entrada.

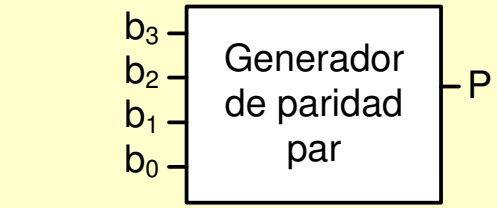


Generadores y comprobadores de paridad.
Generador de paridad par.

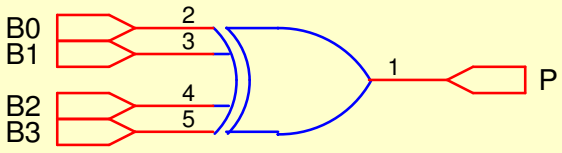
- El bit de paridad par P toma el valor adecuado para que el número total de bits a 1 del conjunto (b_{n-1}, b_{n-2}, ..., b_0, y P) sea par.
- *P será 1 si b_{n-1}, b_{n-2}, ..., b_0, tienen un número impar de bits a 1 y 0, en caso contrario. El 0 se considera par.*
- Por tanto, ***la función lógica que genera el bit de paridad par P es la XOR.***

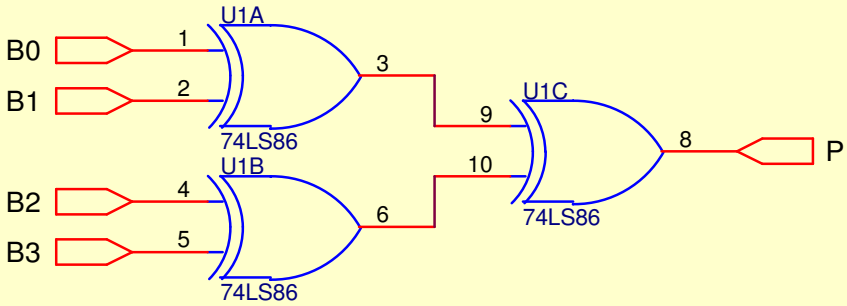
$$P = b_{n-1} \oplus b_{n-2} \oplus ... \oplus b_0$$

- ***Ejemplo. Generador de paridad par de 4 bits.***



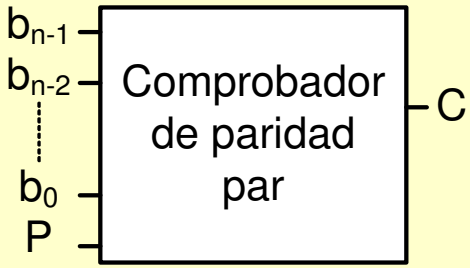
$$P = b_3 \oplus b_2 \oplus b_1 \oplus b_0$$



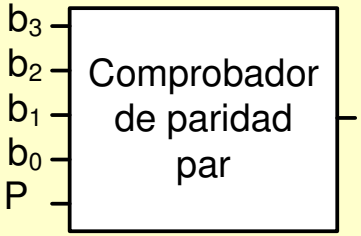


Generadores y comprobadores de paridad.
Comprobador de paridad par.

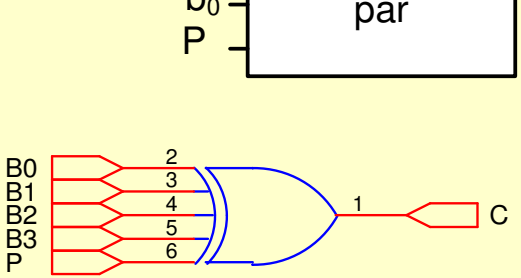
- ***Comprobador de paridad par.***
  - **Se genera la paridad par de todos los bits del código de paridad, b_{n-1}, b_{n-2}, ..., b_0, y P.**

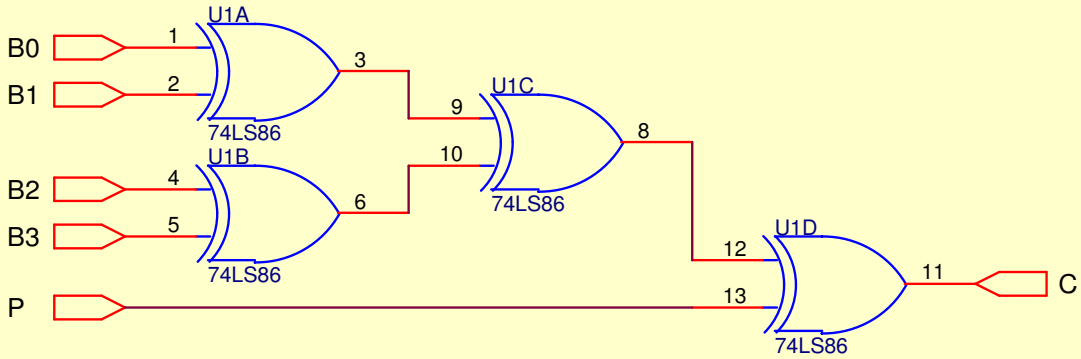


  - **Si C es 1** es porque el conjunto de bits tiene un nº impar de bits a 1, por lo que **la paridad es incorrecta**.
  - **Si C es 0** es porque el conjunto tiene un nº par de bits a 1, por lo que **la paridad es correcta**.
  - ***Ejemplo. Comprobador de paridad par de 4 bits.***



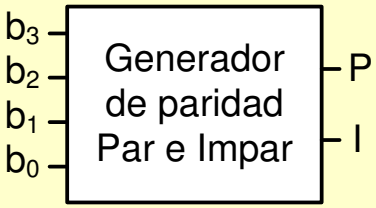
$$C = b_3 \oplus b_2 \oplus b_1 \oplus b_0 \oplus P$$





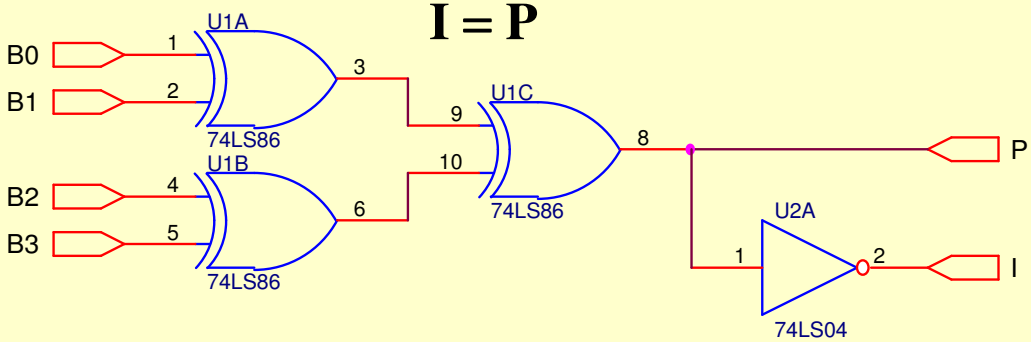
Generadores y comprobadores de paridad.
Generador de paridad impar.

- ***Generador de paridad impar.***
  - El bit de paridad impar **I** toma el valor adecuado para que el número total de bits a 1 del conjunto (b_{n-1}, b_{n-2}, ..., b_0, e I) sea impar.
  - ***I** será 1 si b_{n-1}, b_{n-2}, ..., b_0, tienen un número par de bits a 1 y 0, en caso contrario. El 0 se considera par.*
  - Cualquier *combinación binaria* de un código binario *tiene un nº par o impar de bits con valor 1*. Por tanto, **una es el complemento de la otra**.
  - Así, **los generadores y comprobadores de paridad impar se obtienen complementando la salida de los de paridad par**.
  - ***Ejemplo. Generador de paridad par e impar de 4 bits.***



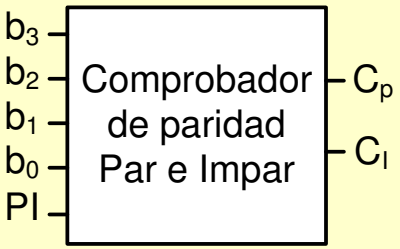
$$P = b_3 \oplus b_2 \oplus b_1 \oplus b_0$$

$$I = \overline{P}$$

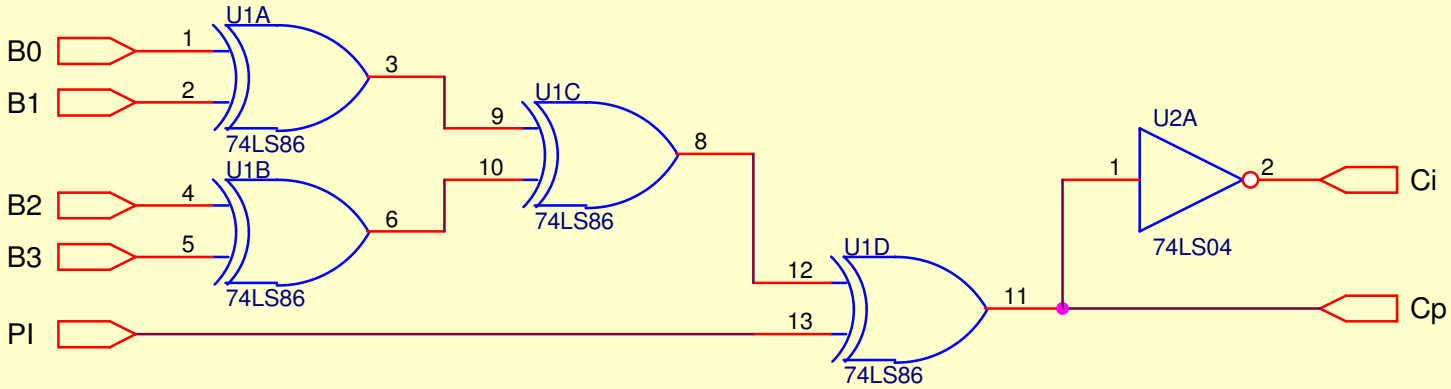


Generadores y comprobadores de paridad.
Comprobador de paridad impar.

- ***Comprobador de paridad impar***.
  - Consiste en un generador de paridad impar.
  - **Se obtiene a partir del comprobador de paridad par, complementado la salida**.
  - ***Ejemplo. Comprobador de paridad par e impar de 4 bits.***
    - ***PI*** es el bit de paridad par o impar.



$$C_P = b_3 \oplus b_2 \oplus b_1 \oplus b_0 \oplus PI$$
$$C_I = \overline{C_P}$$



Bibliografía detallada

Circuitos Combinacionales Lógicos

Las diapositivas se han confeccionado utilizando como fuente:

- "Diseño Lógico". A. Lloris, A. Prieto. Mc-Graw Hill. 1996.

Capítulos: 6.3 (párrafo primero), 6.3.1, 6.3.2, 6.4, 6.5(párrafo primero), 6.5.1