
Tema 5

Subsistemas Combinacionales

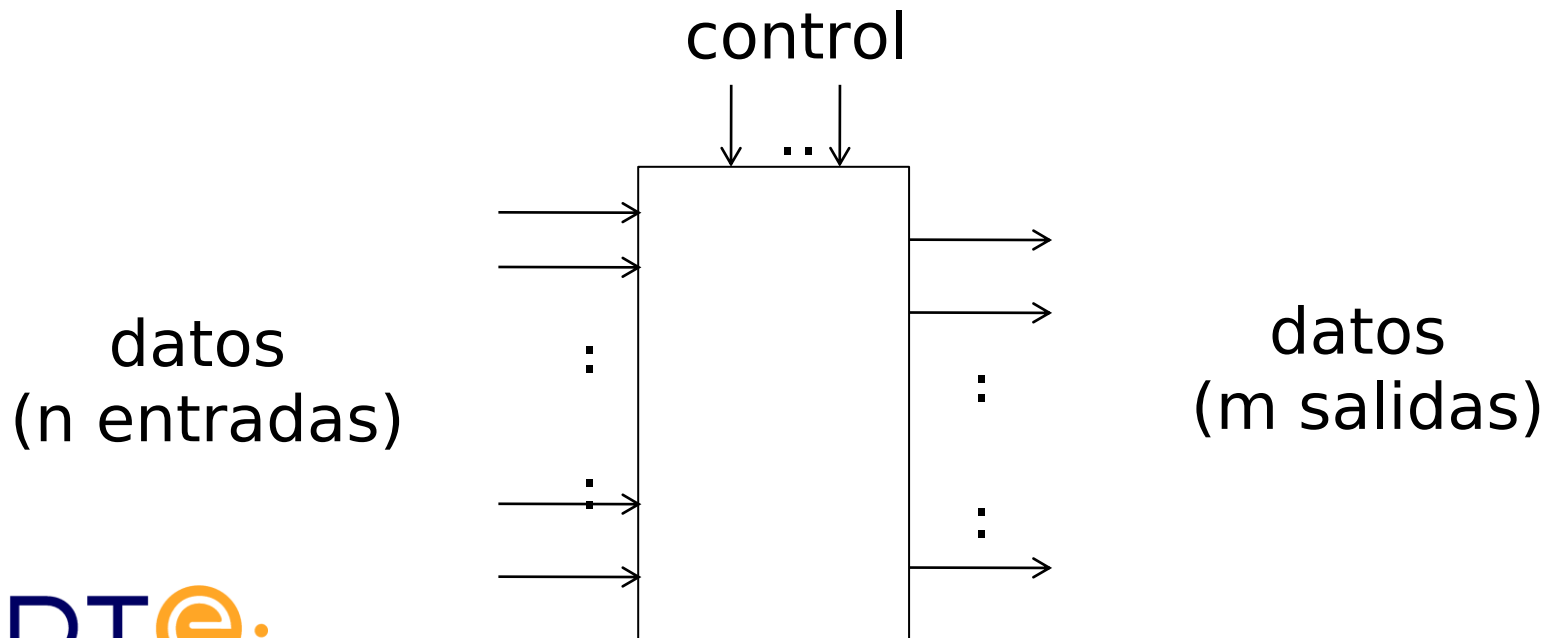
ATENCIÓN:
Este material NO son apuntes
de la asignatura, sino material
de apoyo para las clases de aula

Índice

- **Subsistemas combinacionales**
- **Subsistemas de propósito específico:**
 - Decodificadores, codificadores, convertidores de código, Codificadores de prioridad, comparadores, demultiplexores
- **Subsistemas de propósito general:**
 - Multiplexores
 - (Dispositivos programables, fuera del alcance de este tema)

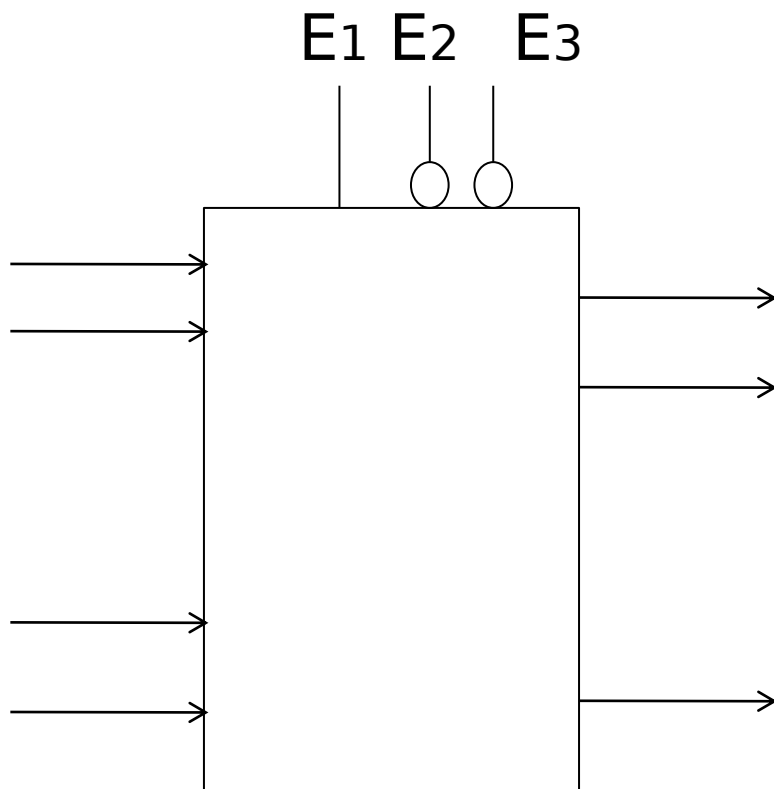
1. Subsistemas combinacionales

- Facilitan el proceso de diseño, pues **realizan funciones complejas** habituales que requieren un gran número de puertas (100 a 1000; CI MSI/LSI)
- “muchas” entradas (n) y “muchas salidas” (m)
- Dos tipos de terminales: **datos** y **control**



Subsistemas Combinacionales (entradas y salidas)

- Señales de control: condicionan el funcionamiento del subsistema (habilitan, inhiben, etc)
- Niveles de activación: en alto o en bajo



Ejemplo:

- ▣ E1, ENABLE activo en alto
- ▣ E2 y E3, ENABLES activos en bajo
- ▣ Activado, si y solo si:

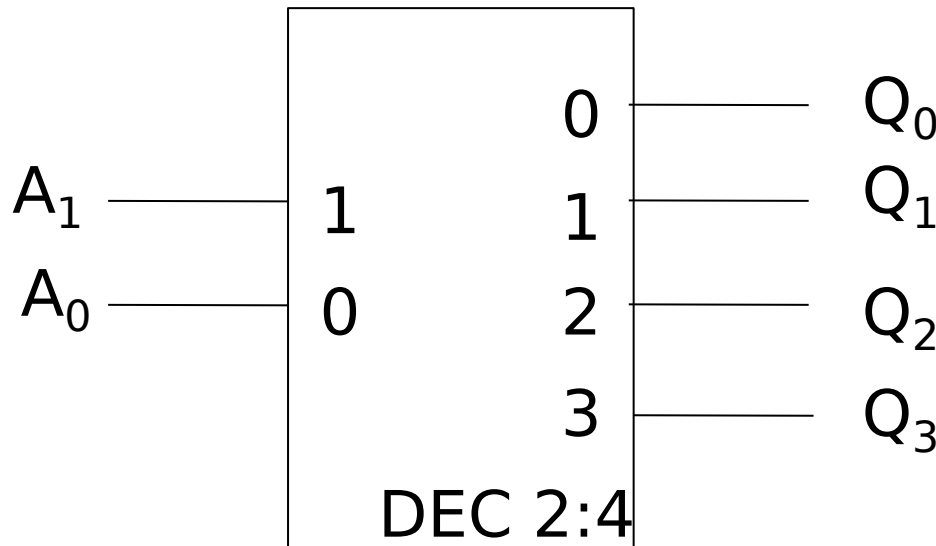
$$E1=1, E2=E3=0$$

Subsistemas de propósito específico:

2.1 Decodificadores

- Decodificadores: “decodifican” un dato de entrada
 - n entradas (en un determinado código) y m salidas (código “one-hot”, sólo una salida activa en cada momento)

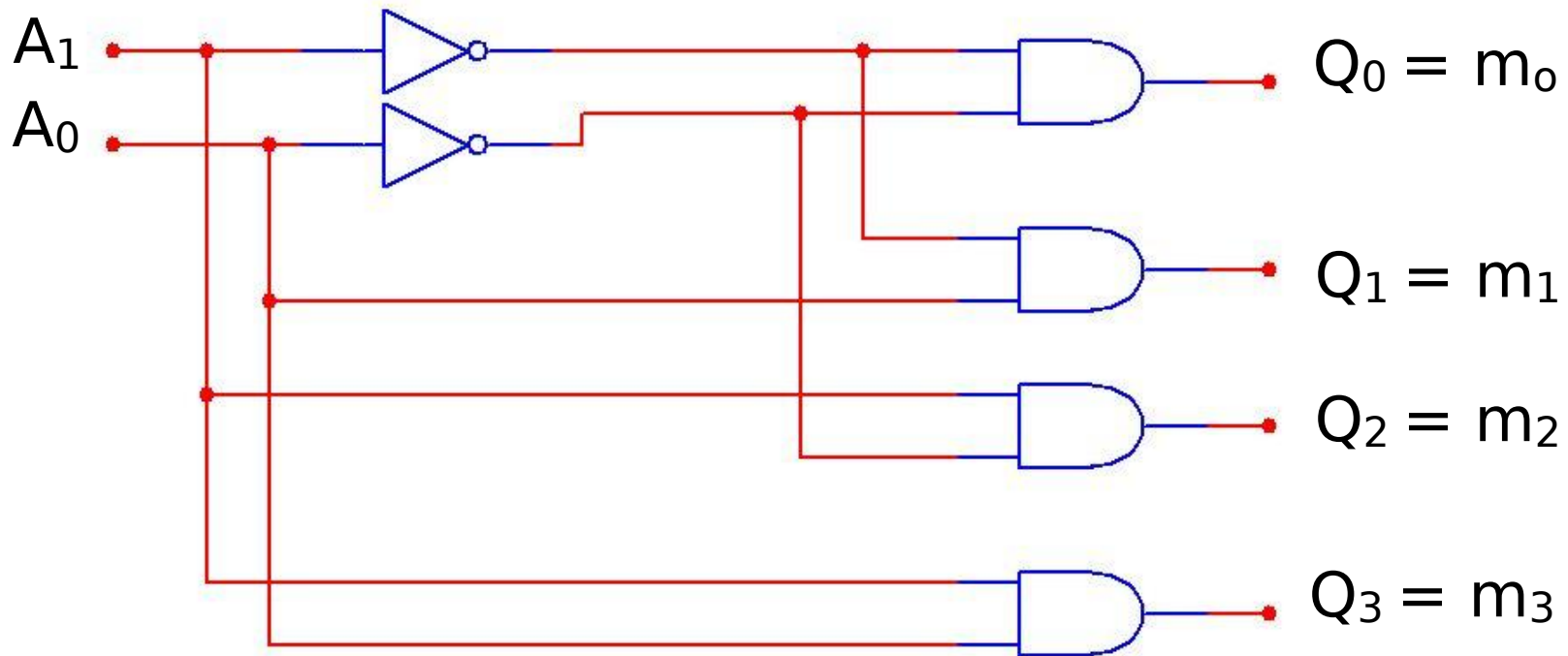
Ejemplo: DEC 2:4 con salidas activas en alto



A1	A0	Q0	Q1	Q2	Q3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

Decodificadores (diseño interno)

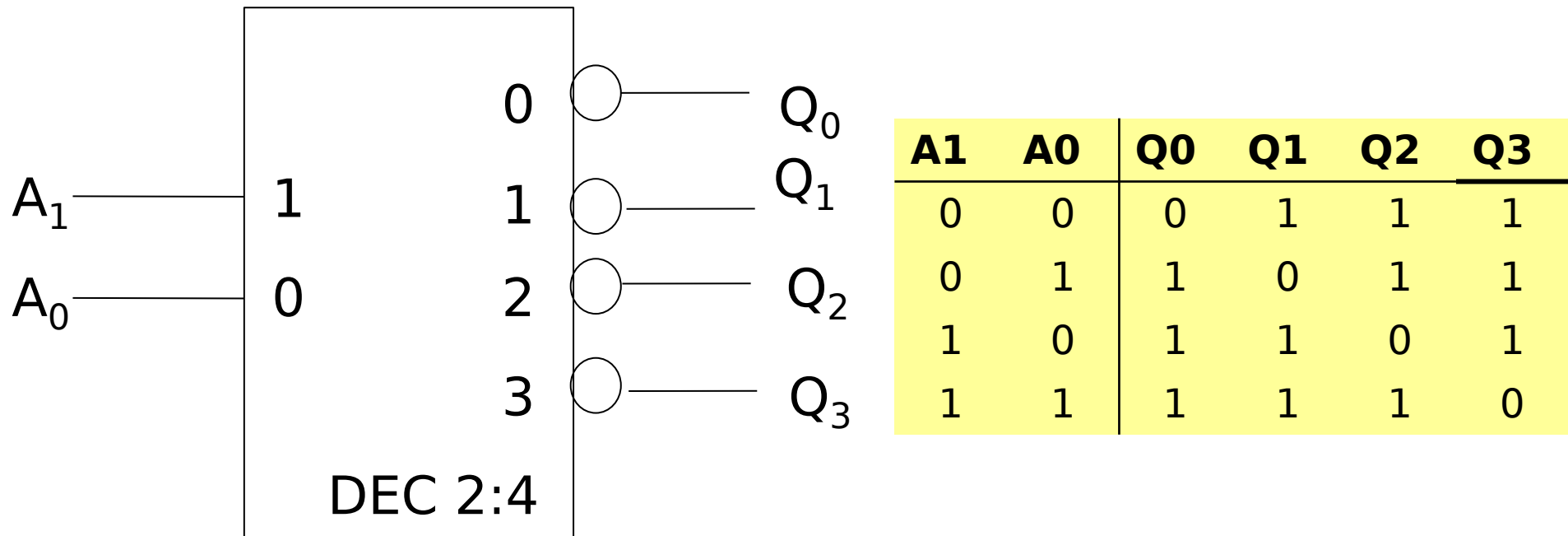
Ejemplo: *DEC 2:4 con salidas activas en alto*



(a partir de su tabla de verdad)

Decodificadores (funciones de salida)

Ejemplo: DEC 2:4 con salidas activas en bajo

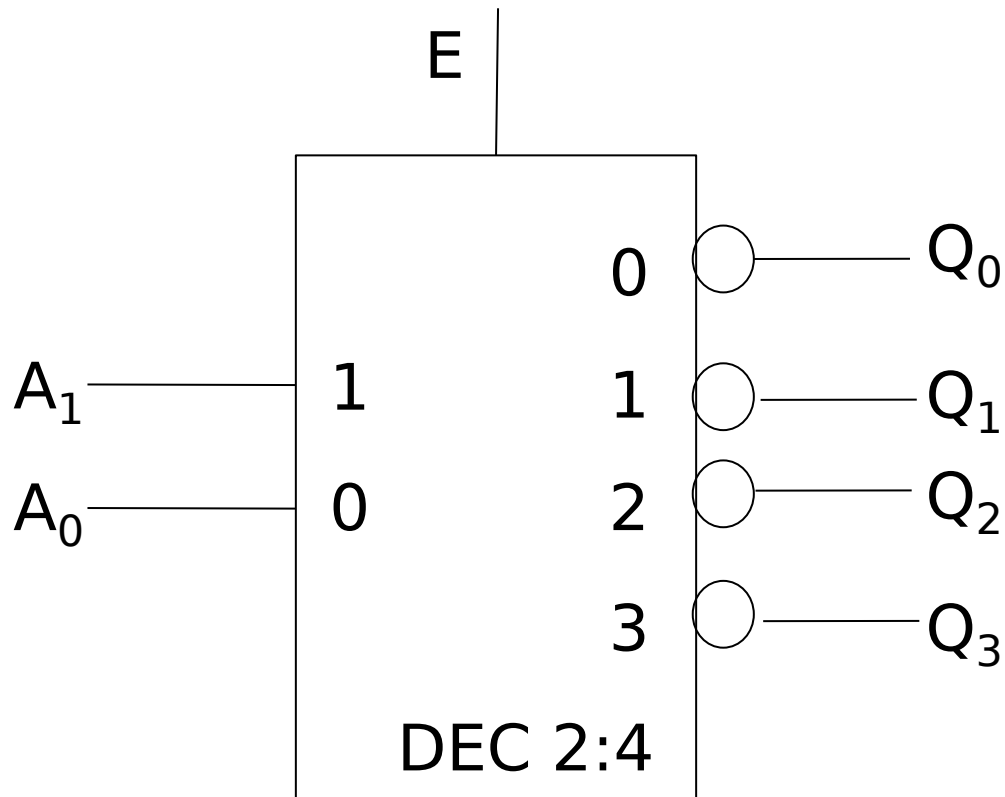


- DEC (**salidas en alto**): genera **mintérminos** ($Q_i = m_i$)
- DEC (**salidas en bajo**): genera **Maxtérminos** ($Q_i = M_i$)

Decodificadores (funciones de salida)

Ejemplo:

DEC 2:4 con salidas activas en bajo (con enable en alto)



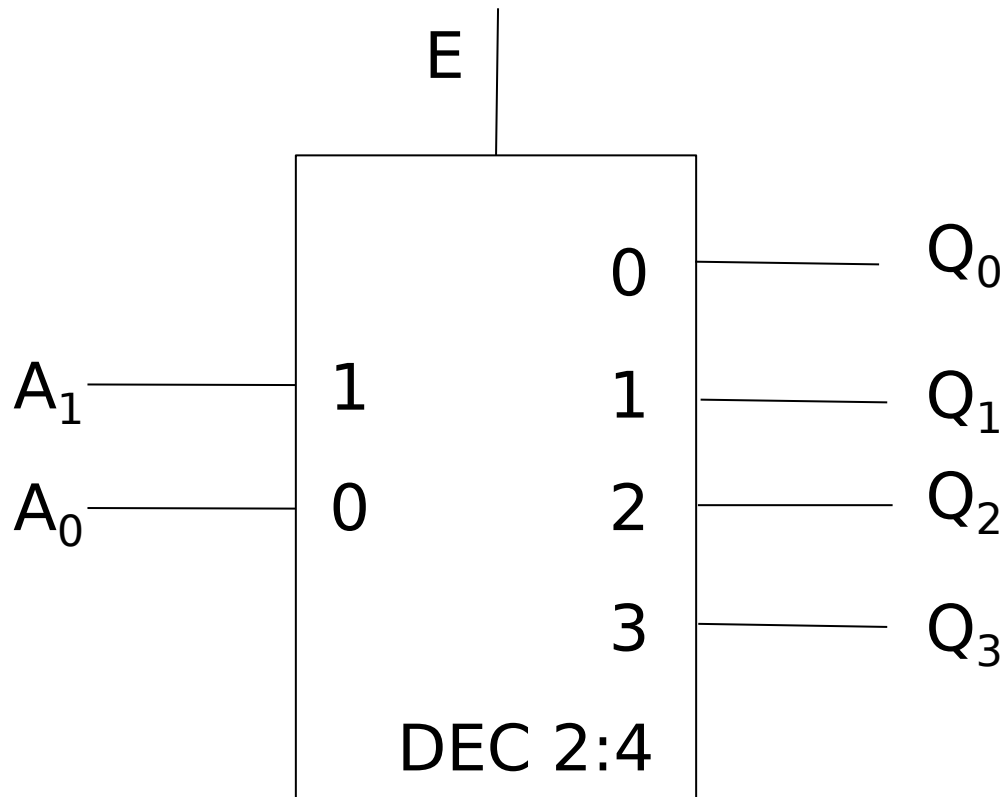
E	A1	A0	Q0	Q1	Q2	Q3
0	X	X	1	1	1	1
1	0	0	0	1	1	1
1	0	1	1	0	1	1
1	1	0	1	1	0	1
1	1	1	1	1	1	0

$$Q_i = M_i + \overline{E} \quad \star$$

Decodificadores (funciones de salida)

Ejemplo:

DEC 2:4 con salidas activas en alto (con enable en alto)



E	A1	A0	Q0	Q1	Q2	Q3
0	X	X	0	0	0	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	0	1

$$Q_i = m_i E$$

Decodificador (descripción Verilog)

Ej: Descripción Verilog “funcional” de un

DEC 2:4 con salidas activas en alto (con enable en alto)

```
// Decodificador 2:4 con salidas en alto y  habilitación en alto  
// descripción verilog “funcional”
```

```
module decodificador_2_a_4_df_v(E, A0, A1, Q0, Q1, Q2, Q3);  
    input E, A0, A1;  
    output Q0, Q1, Q2, Q3;
```

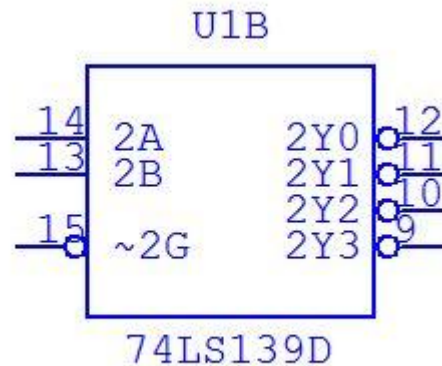
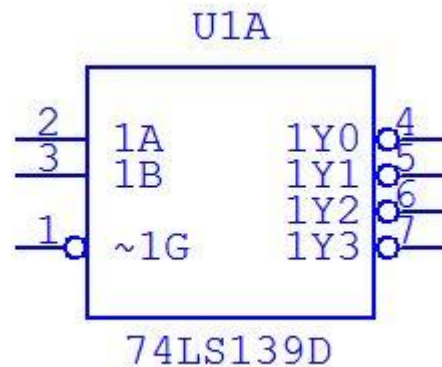
```
    assign Q0= E & ~A1 & ~A0;  
    assign Q1= E & ~A1 & A0;  
    assign Q2= E & A1 & ~A0;  
    assign Q3= E & A1 & A0;
```

```
endmodule
```

Decodificadores comerciales

74-139 (DUAL 2 TO 4 DECODER/DEMULTIPLEXER)

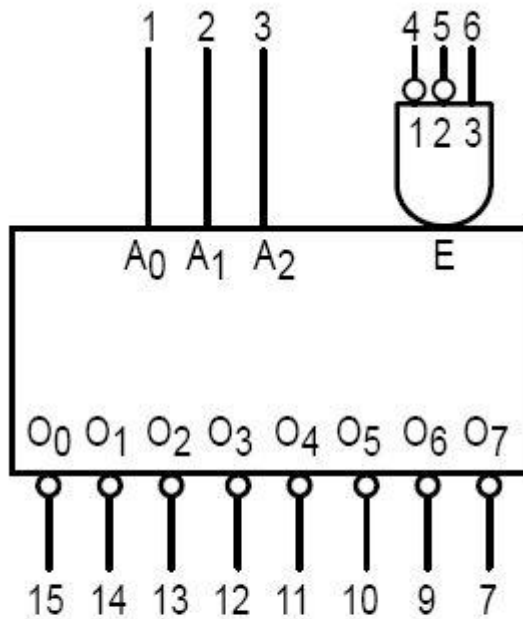
Contiene 2 DEC 2:4 con salidas activas en bajo (con ENABLE en bajo)



Decodificadores comerciales, ejemplos

74-138 (3 TO 8 DECODER/DEMULTIPLEXER)

DEC 3:8 con salidas activas en bajo (3 entradas ENABLE)



V_{CC} = PIN 16
GND = PIN 8

TRUTH TABLE

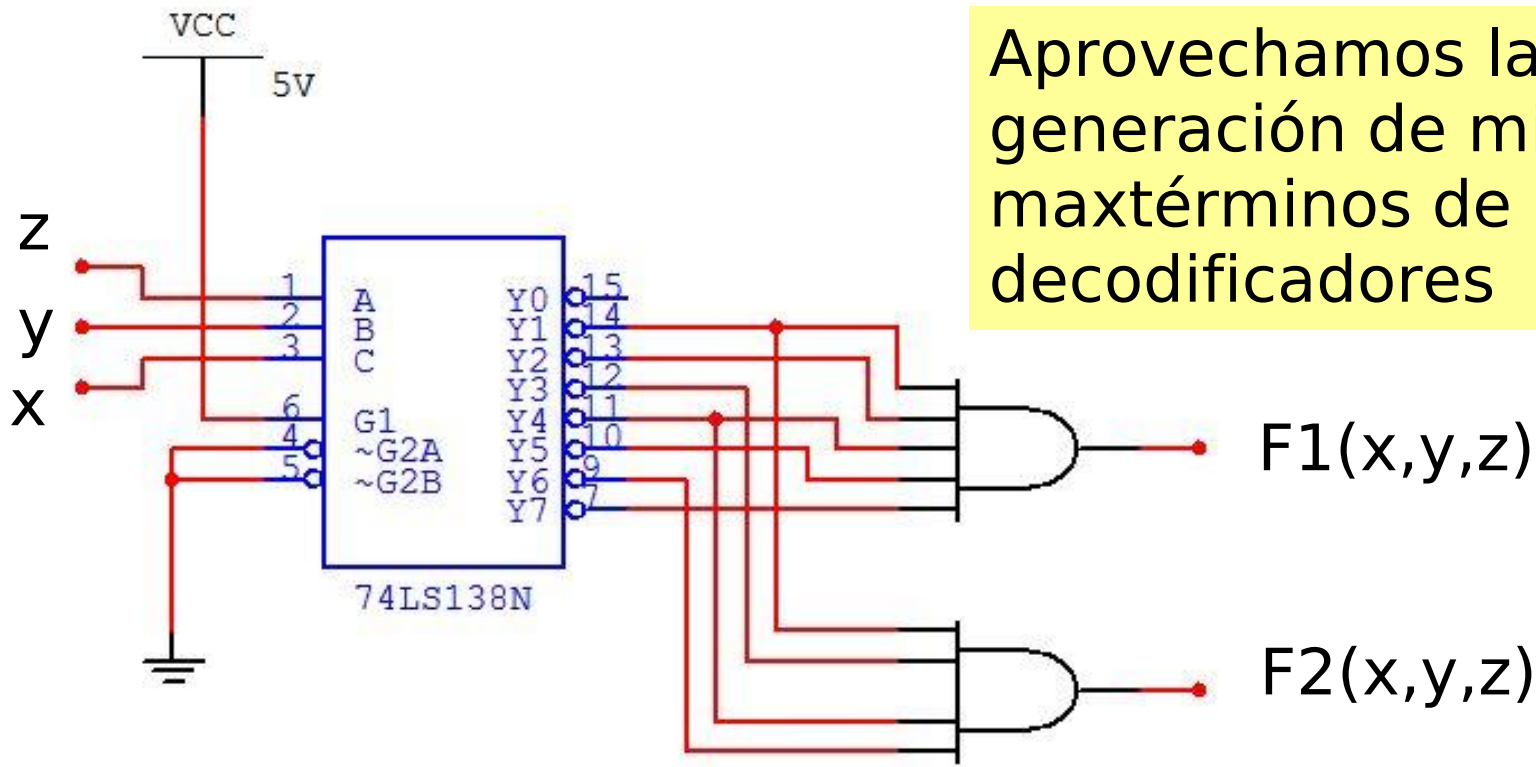
INPUTS						OUTPUTS							
E ₁	E ₂	E ₃	A ₀	A ₁	A ₂	O ₀	O ₁	O ₂	O ₃	O ₄	O ₅	O ₆	O ₇
H	X	X	X	X	X	H	H	H	H	H	H	H	H
X	H	X	X	X	X	H	H	H	H	H	H	H	H
X	X	L	X	X	X	H	H	H	H	H	H	H	H
L	L	H	L	L	L	L	H	H	H	H	H	H	H
L	L	H	H	L	L	H	L	H	H	H	H	H	H
L	L	H	L	H	L	H	H	L	H	H	H	H	H
L	L	H	H	H	L	H	H	H	L	H	H	H	H
L	L	H	L	L	H	H	H	H	H	L	H	H	H
L	L	H	H	L	H	H	H	H	H	H	L	H	H
L	L	H	L	H	H	H	H	H	H	H	H	L	H
L	L	H	H	H	H	H	H	H	H	H	H	H	L

H = HIGH Voltage Level
L = LOW Voltage Level
X = Don't Care

Diseño con decodificadores y puertas

Ejemplo: Realice la siguiente función multisalida con decodificadores y puertas:

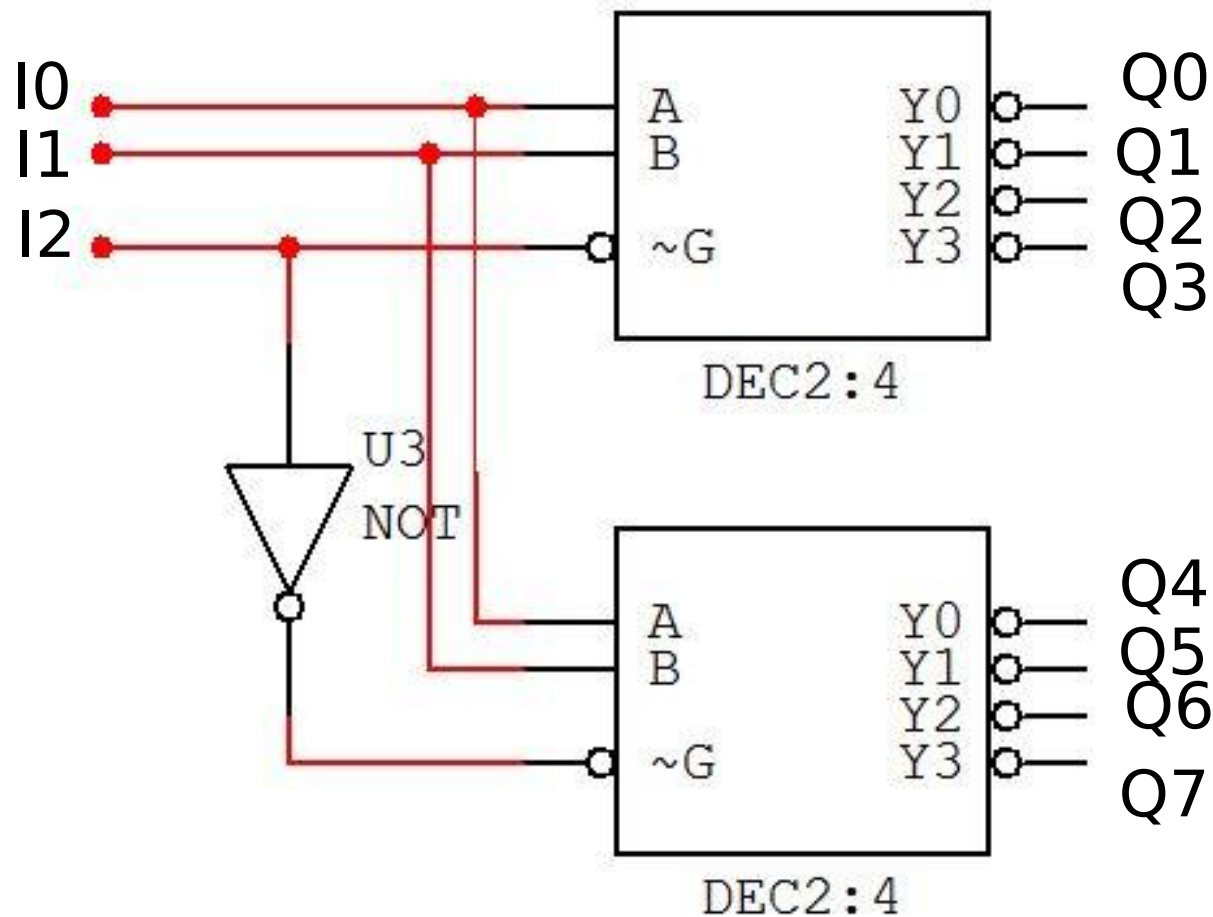
$$F1(x,y,z) = \Sigma(0,3,6) \quad F2(x,y,z) = \Pi(1,3,4,6)$$



Aprovechamos la capacidad de generación de mintérminos o maxtérminos de los decodificadores

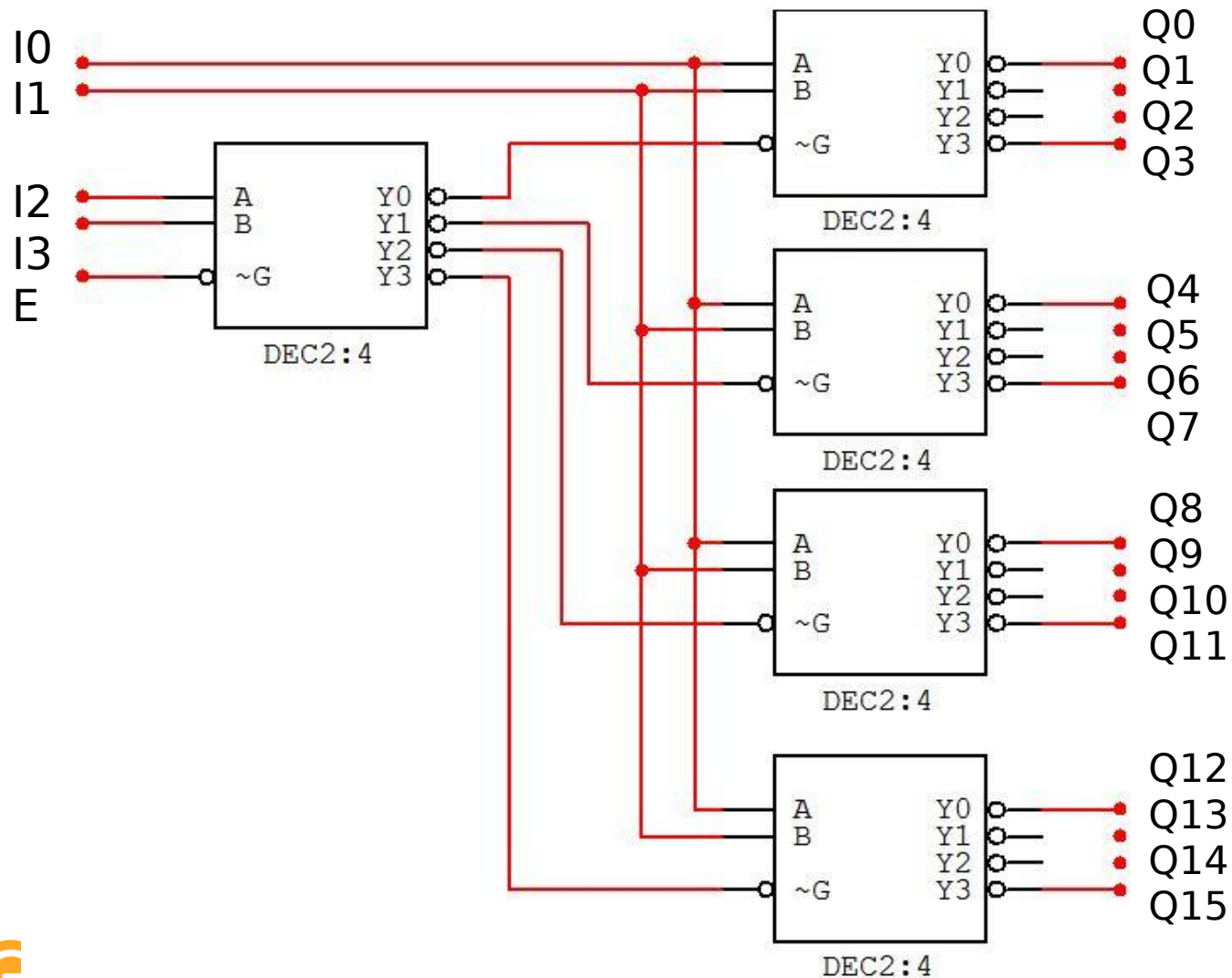
Asociación de decodificadores

Ejemplo: Realizar un DEC 3:8 a partir de DECs 2:4



Asociación de decodificadores (i)

Ejemplo: Realizar un DEC 4:16 a partir de DECs 2:4

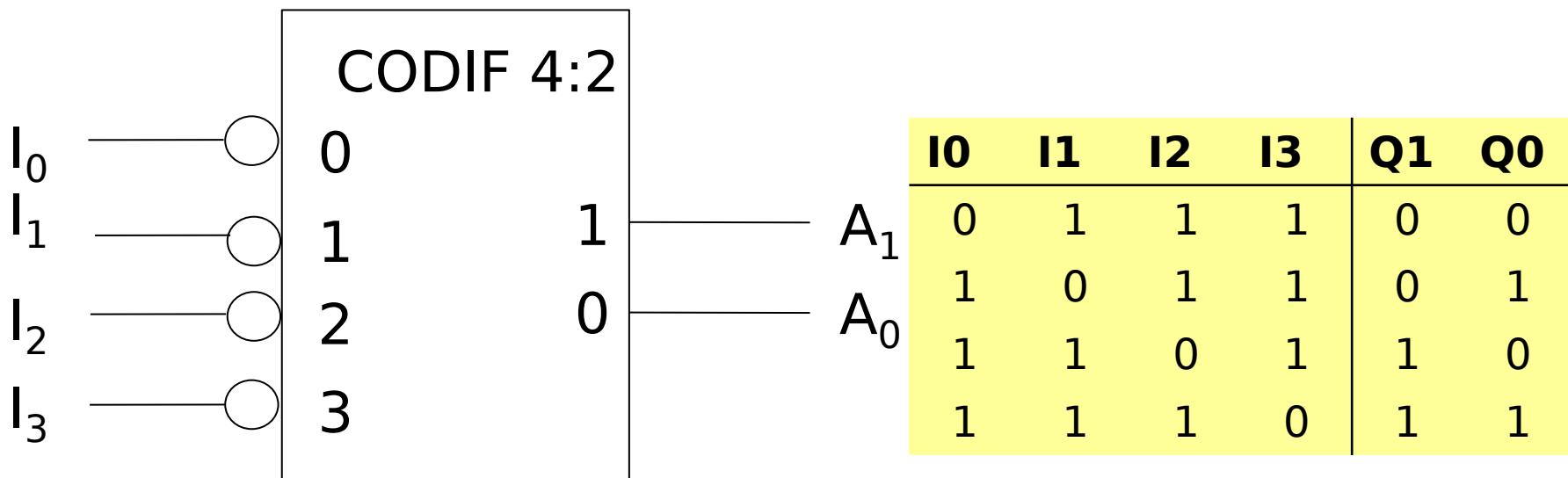


Subsistemas de propósito específico:

2.2 Codificadores

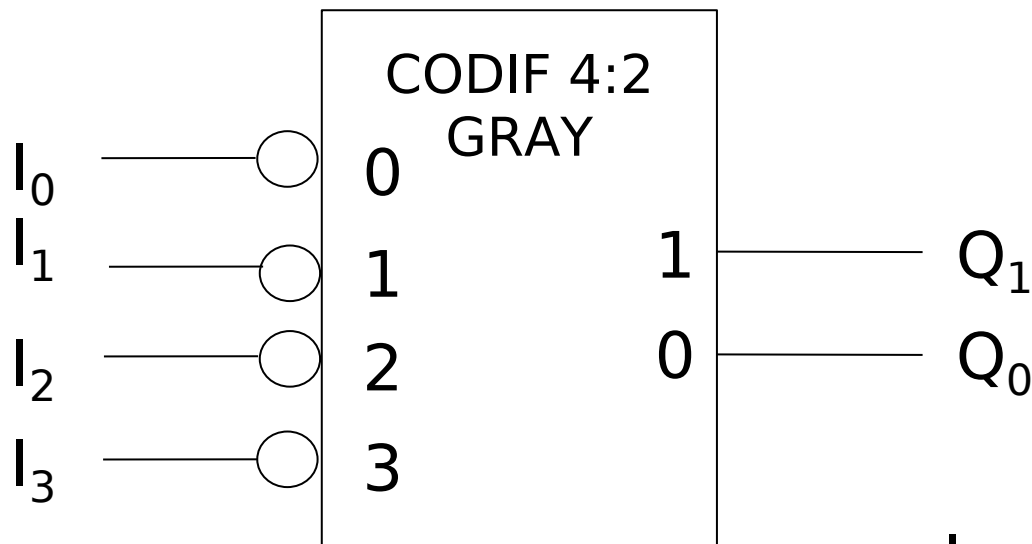
- Realizan la función inversa a los decodificadores: admiten sólo una entrada activada, “codificando” su valor en un código concreto

Ejemplo: CODIF 4:2 con entradas activas en bajo y salidas activas en alto (binario natural)



Diseño de codificadores (ii)

Ejemplo: Realizar un CODIF 4:2 con entradas activas en bajo y salidas activas en alto (GRAY)

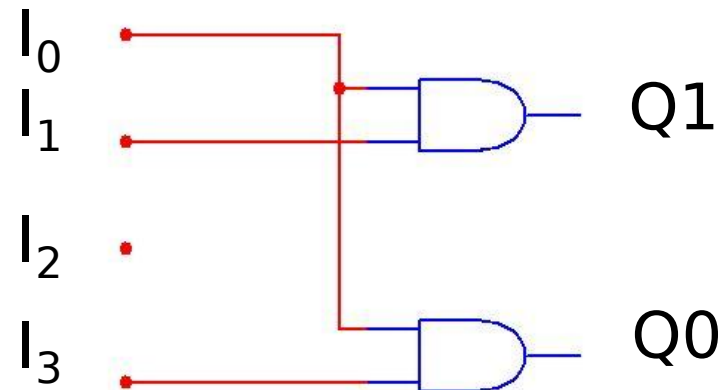


I0	I1	I2	I3	Q1	Q0
0	1	1	1	0	0
1	0	1	1	0	1
1	1	0	1	1	1
1	1	1	0	1	0

Simplificando k-mapa:

$$Q1 = I0 * I1$$

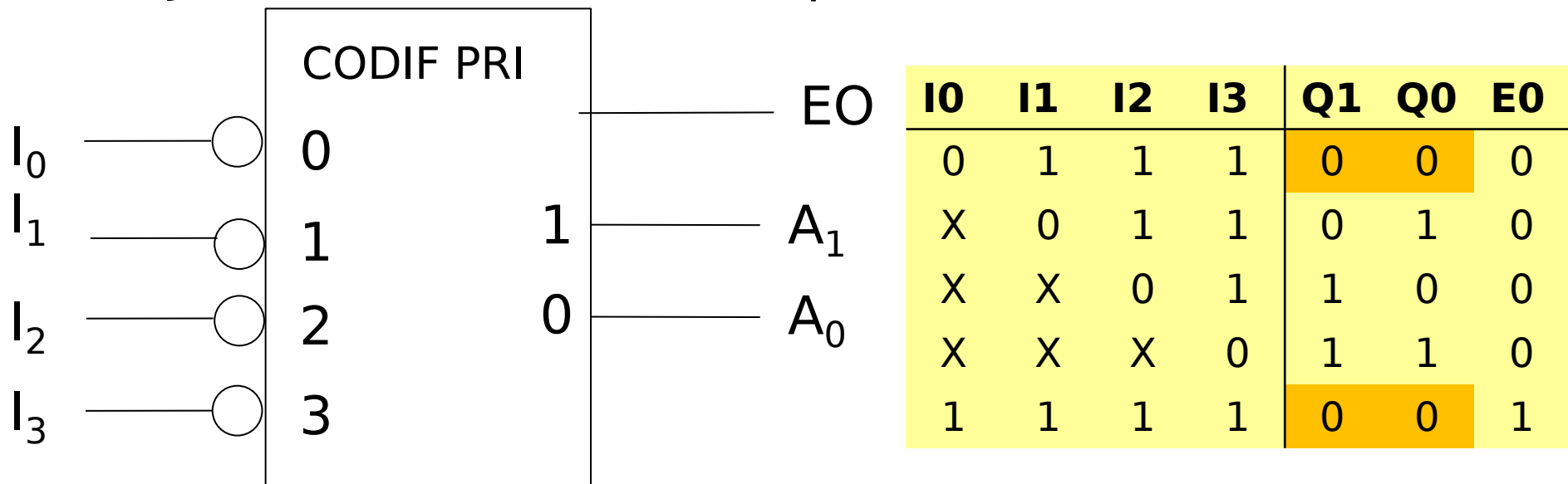
$$Q0 = I0 * I3$$



Subsistemas de propósito específico:

2.3 Codificadores de prioridad

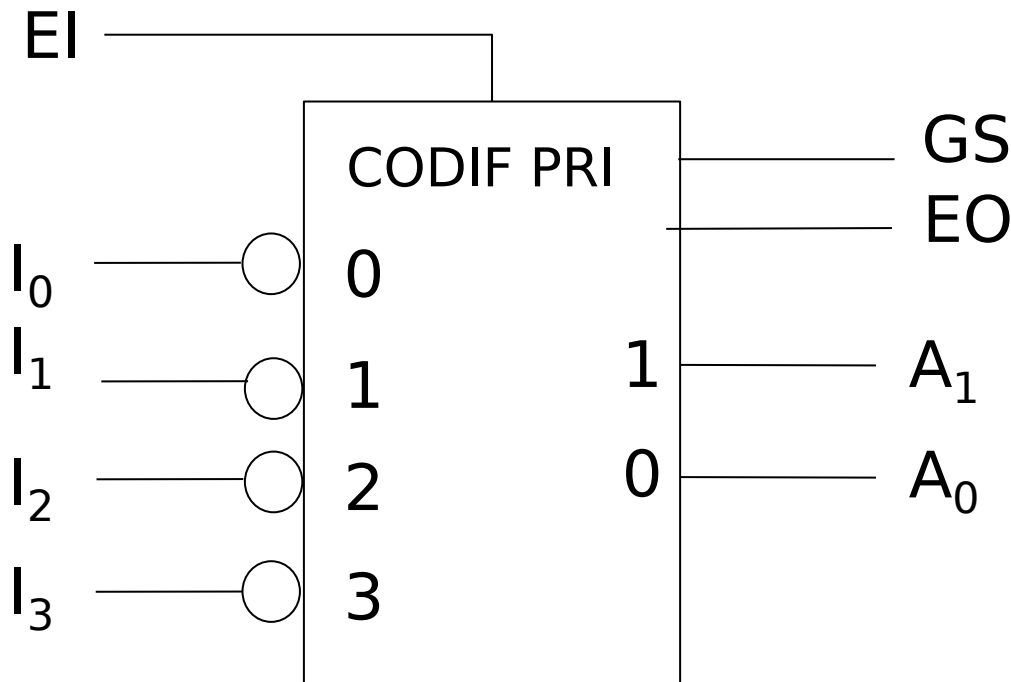
- Son codificadores que admiten más de una entrada activada en cada momento. Generan a la salida el código de la entrada más prioritaria (mayor peso)
- *EJEMPLO: Codificador de prioridad 4:2(bin. natural)*



EO: ENABLE OUTPUT (se activa cuando el codificador está activado pero no hay ninguna entrada activada)

Asociación de codif. de prioridad (i)

- Requieren entrada de ENABLE (EI) y GS (Group select, para saber si el codificador está activado

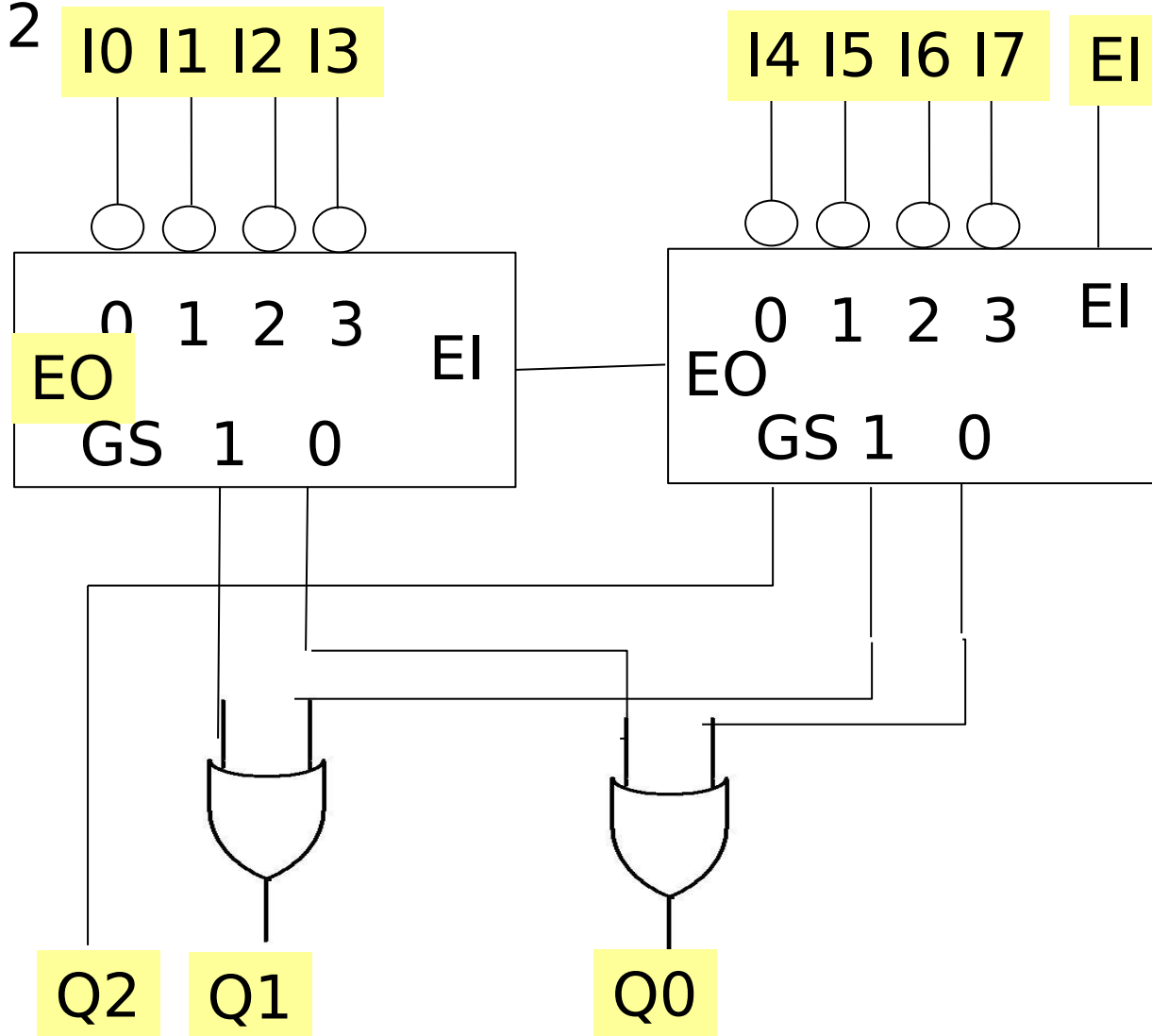


EI	I0	I1	I2	I3	Q1	Q0	GS	E0
0	X	X	X	X	0	0	0	0
1	0	1	1	1	0	0	1	0
1	X	0	1	1	0	1	1	0
1	X	X	0	1	1	0	1	0
1	X	X	X	0	1	1	1	0
1	1	1	1	1	0	0	0	1

GS: se activa cuando el codificador está activado y hay alguna entrada activada

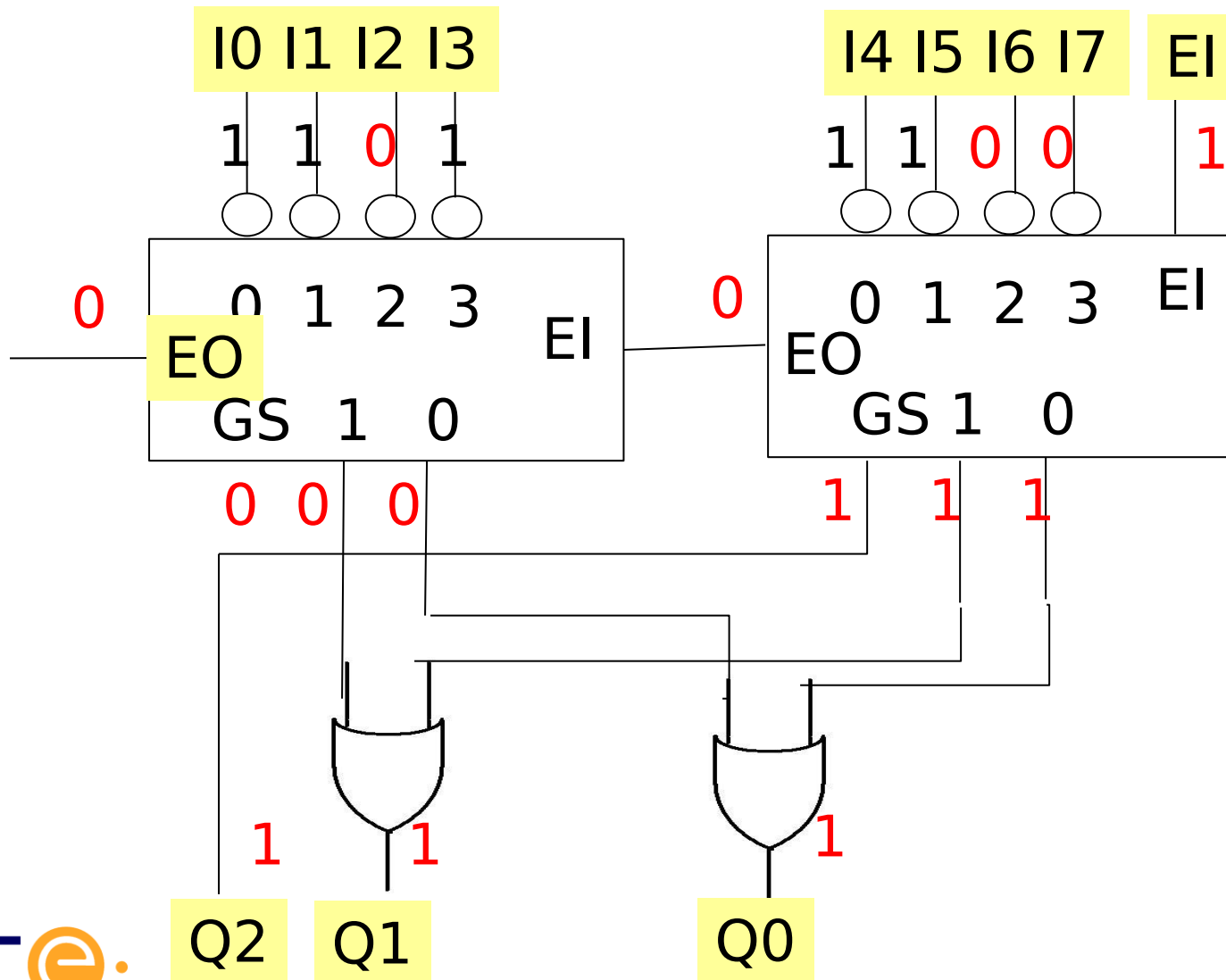
Asociación de codif. de prioridad (ii)

EJEMPLO: Realizar un CODIF PRI 8:3 a partir de CODIF PRI 4:2



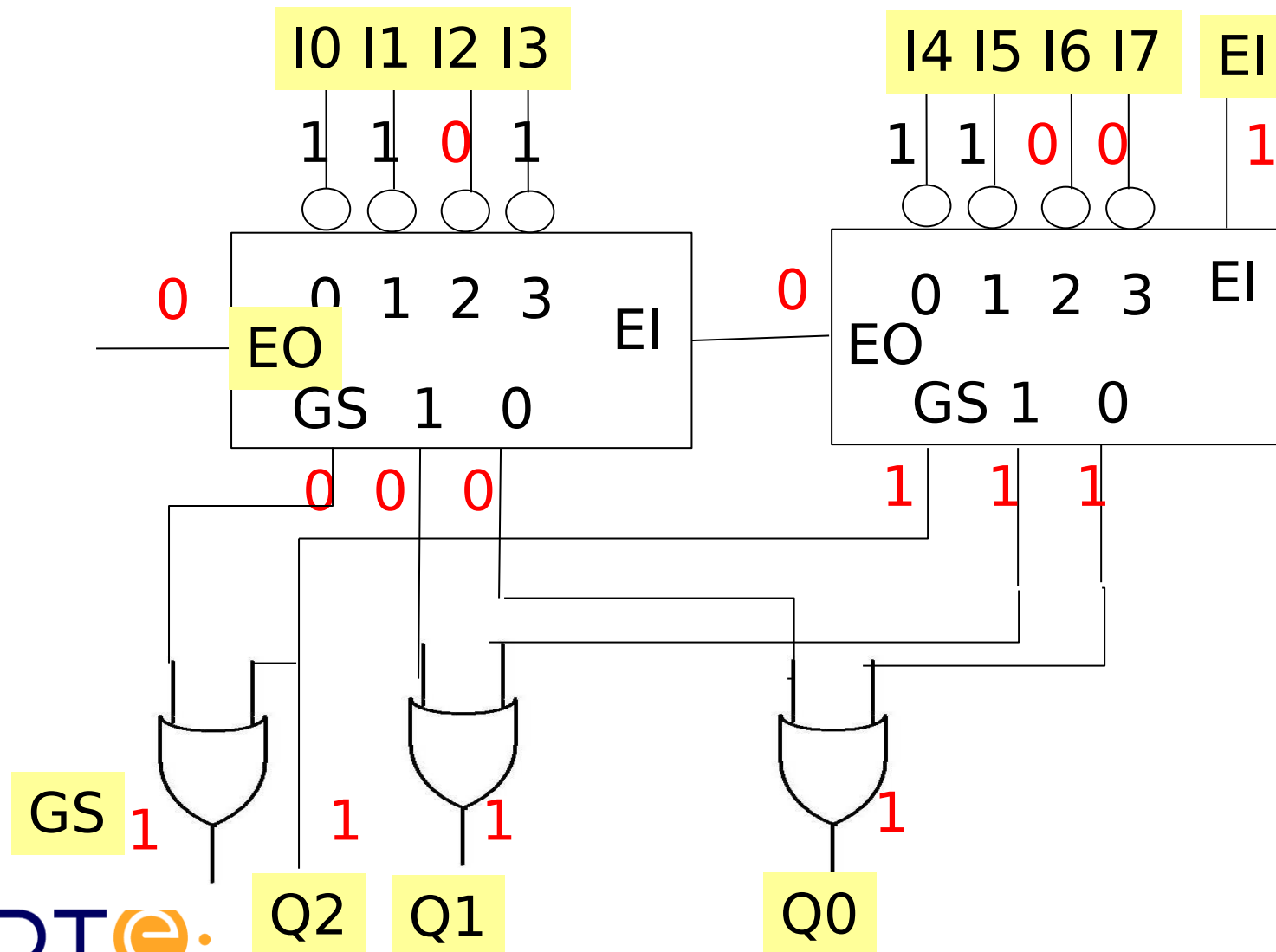
Asociación de codif. de prioridad (iii)

EJEMPLO de funcionamiento



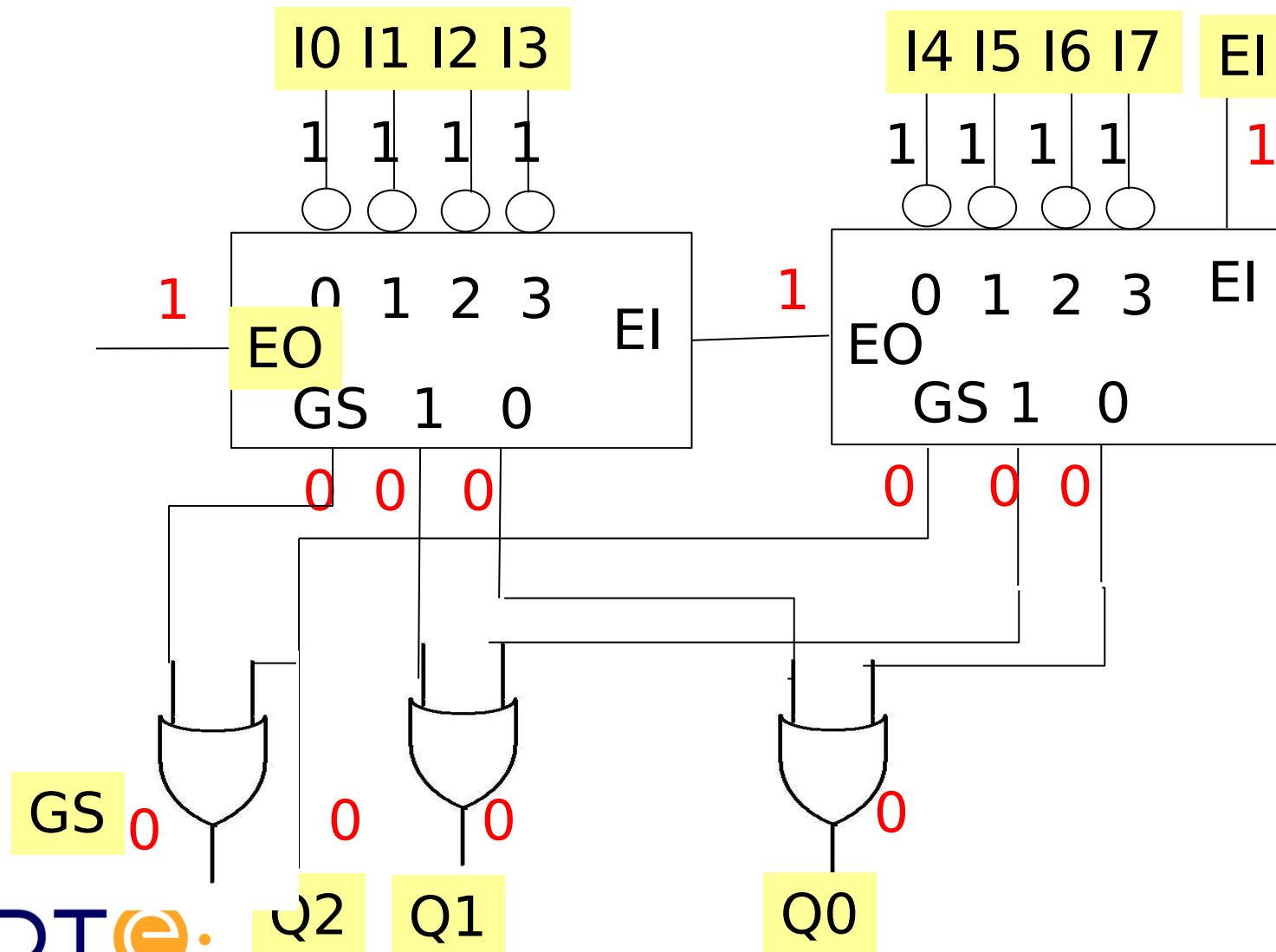
Asociación de codif. de prioridad (iv)

EJEMPLO de funcionamiento



Asociación de codif. de prioridad (v)

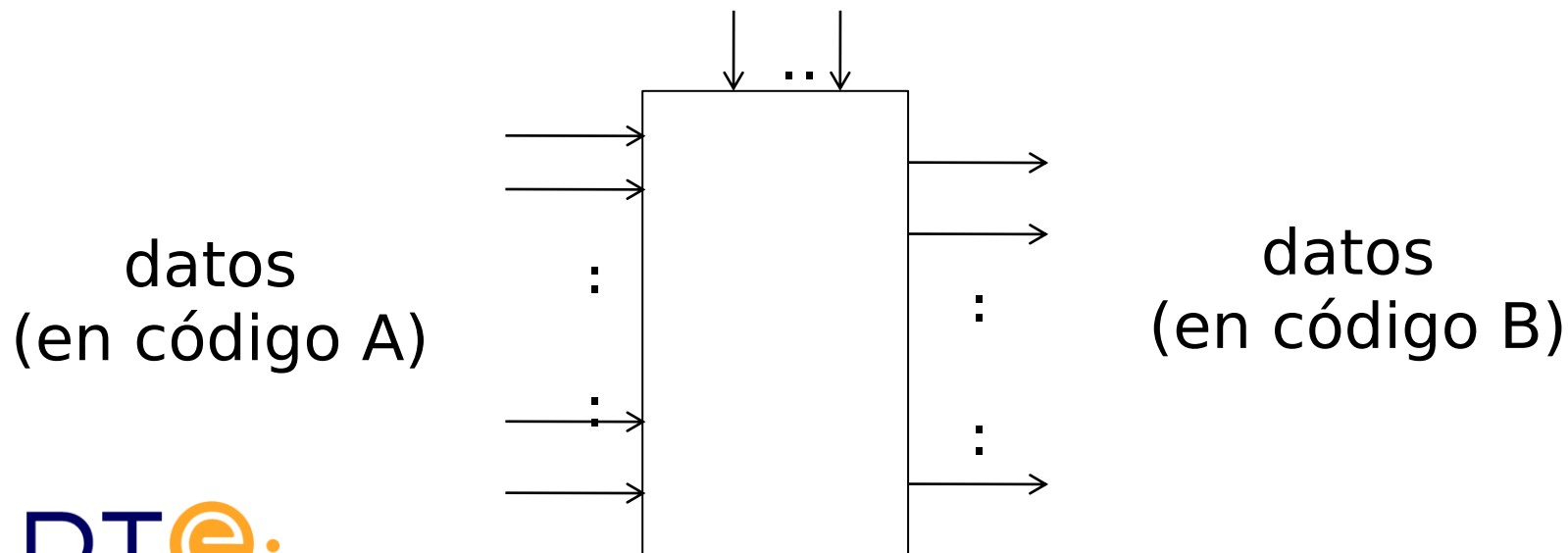
EJEMPLO de funcionamiento



Subsistemas de propósito específico:

2.3 Convertidores de código

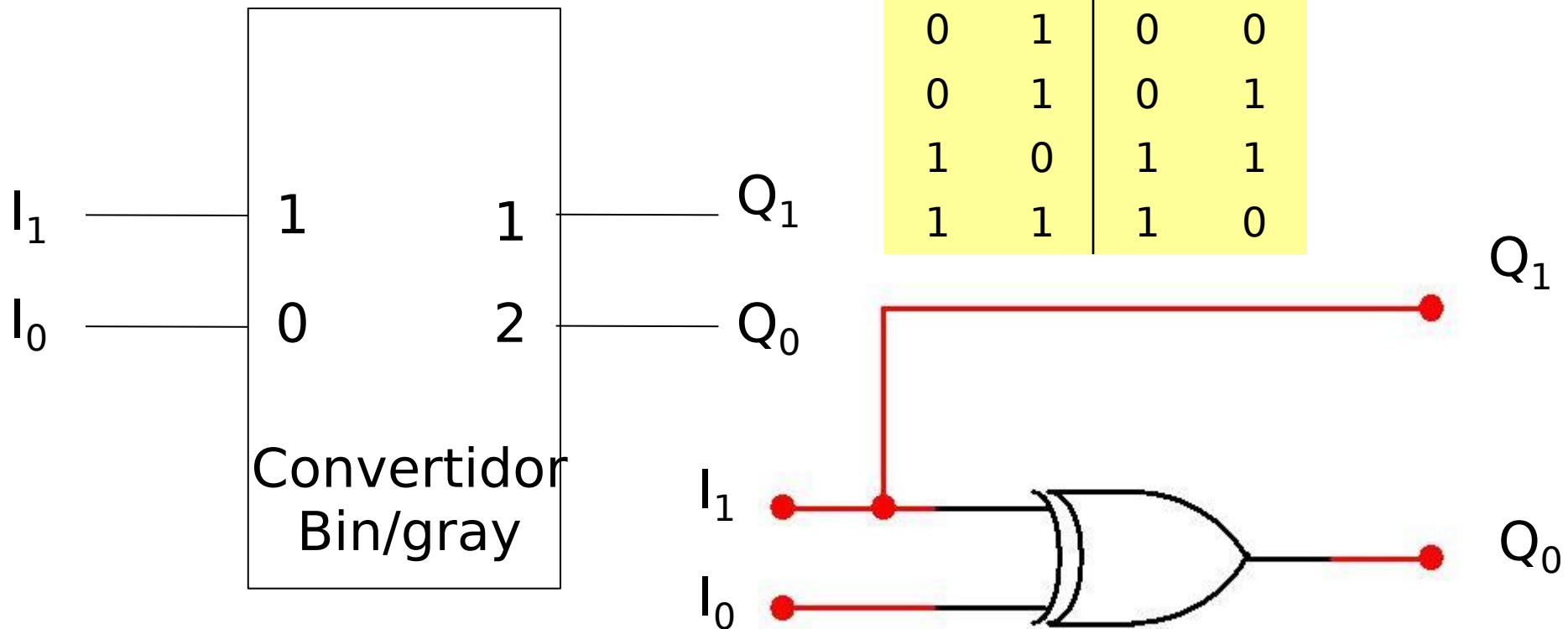
- Realizan un cambio en el código de representación de la información; por ejemplo:
 - *Binario/gray*
 - *Gray/binario*
 - *BCD/7 segmentos*



Diseño de convertidores de código

- Opción 1: a partir de su tabla de verdad

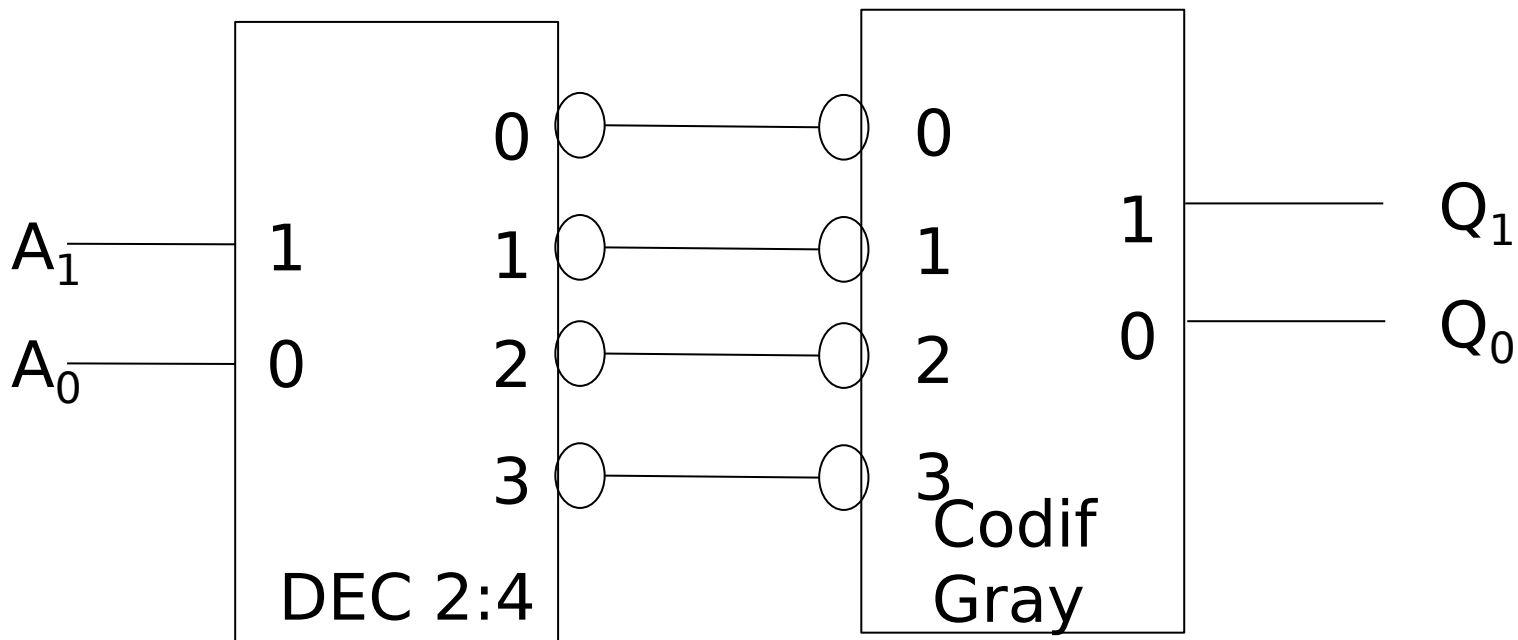
EJEMPLO: Realizar un convertidor de binario natural a gray de dos bits



Diseño de convertidores de código

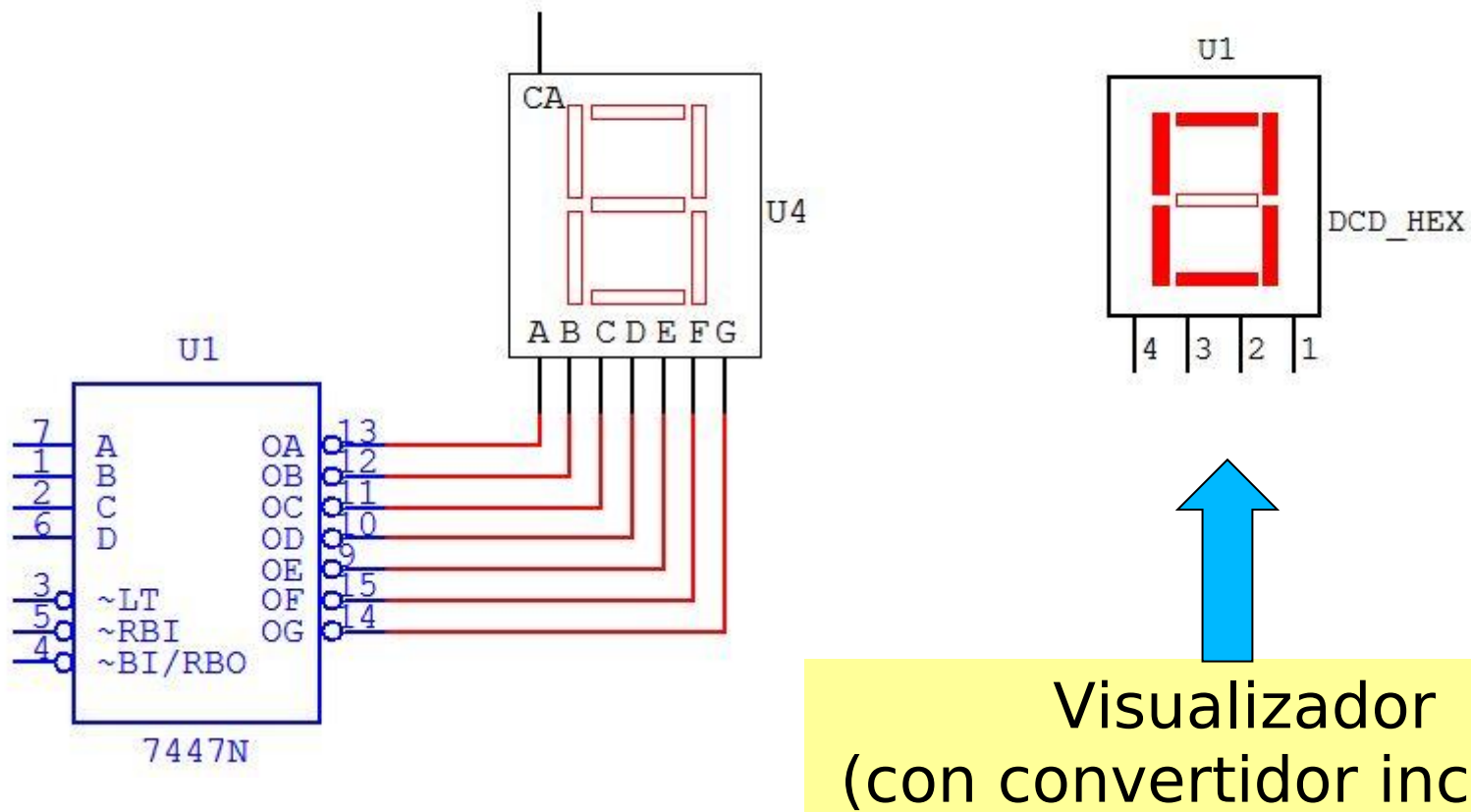
- Opción 2: Asociando DEC-CODIF (con los códigos adecuados)

EJEMPLO: Realizar un convertidor de binario natural a gray de dos bits



Convertidores de código comerciales

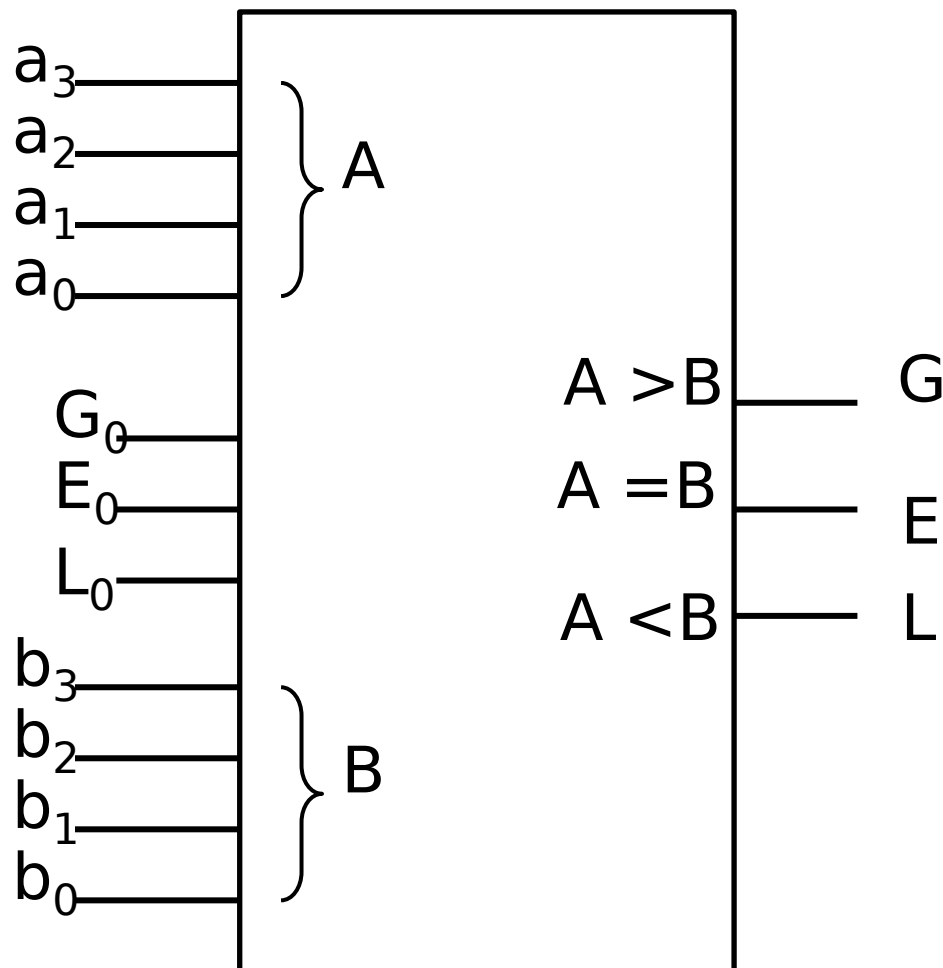
- 74-47 Convertidor BCD a 7 segmentos



Subsistemas de propósito específico:

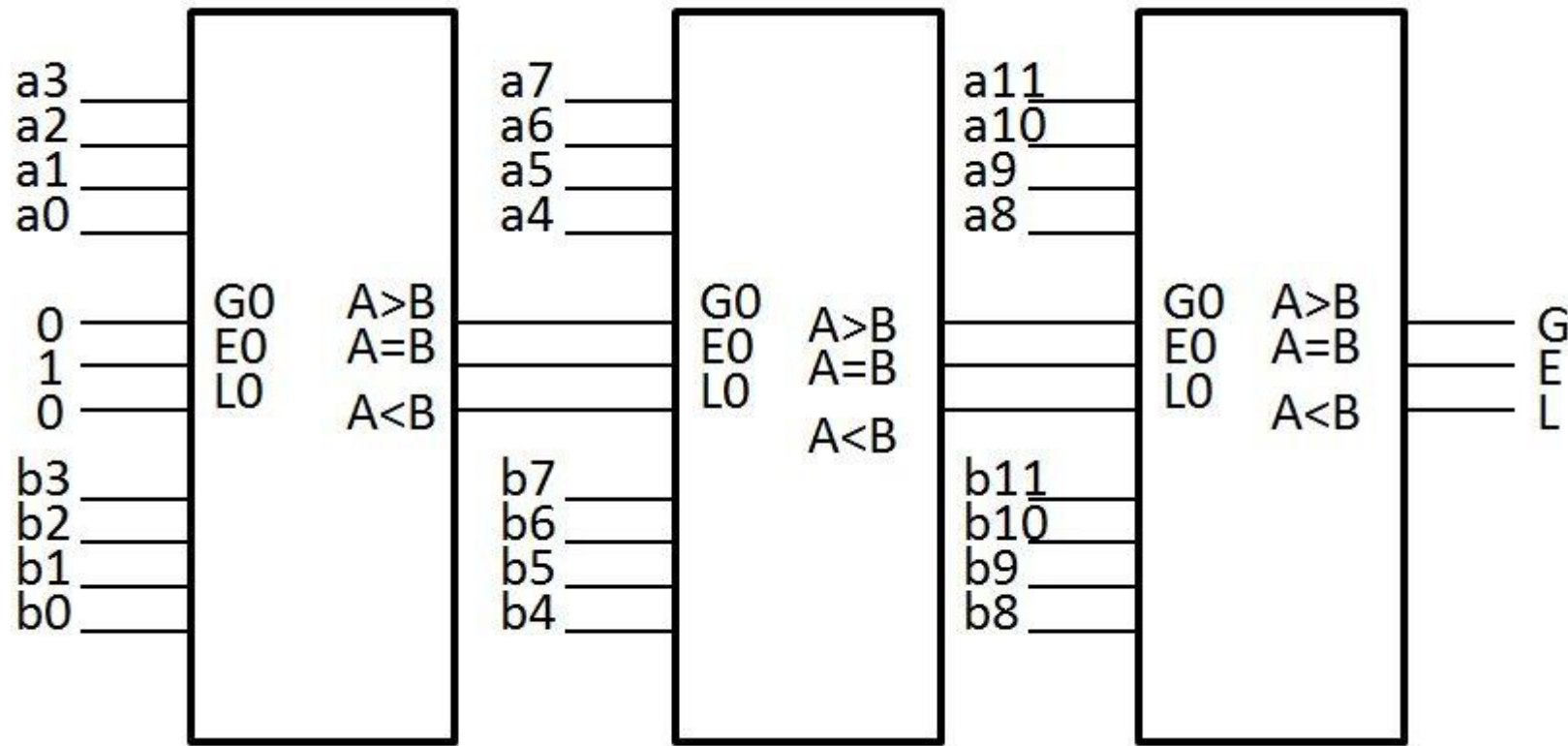
2.4 Comparadores de magnitud

- Comparan las magnitudes de dos números de n bits



A	B	G	E	L
$A > B$		1	0	0
$A = B$		G_0	E_0	L_0
$A < B$		0	0	1

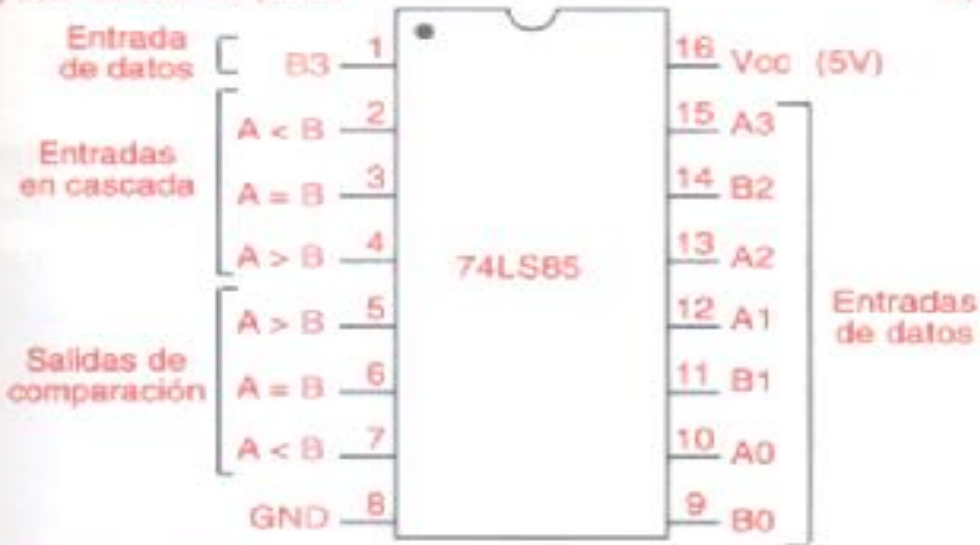
Conexión en cascada de comparadores



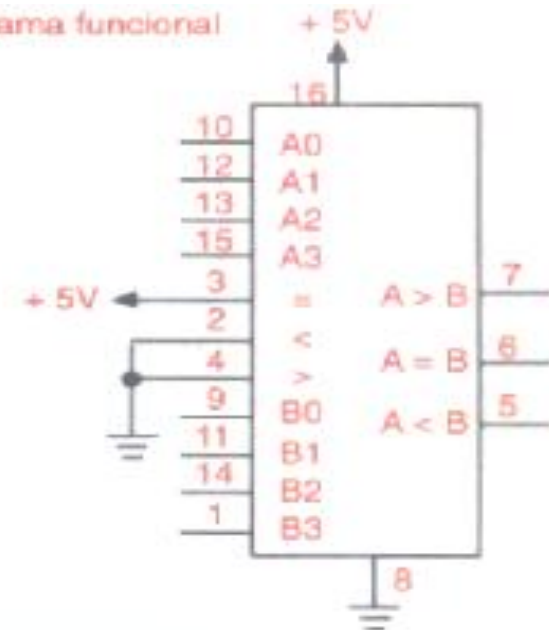
Conexión en cascada para formar un comparador de números de 12 bits

CI 74LS85 (Comparador de 4 bits)

a) Distribución de pines



b) Diagrama funcional



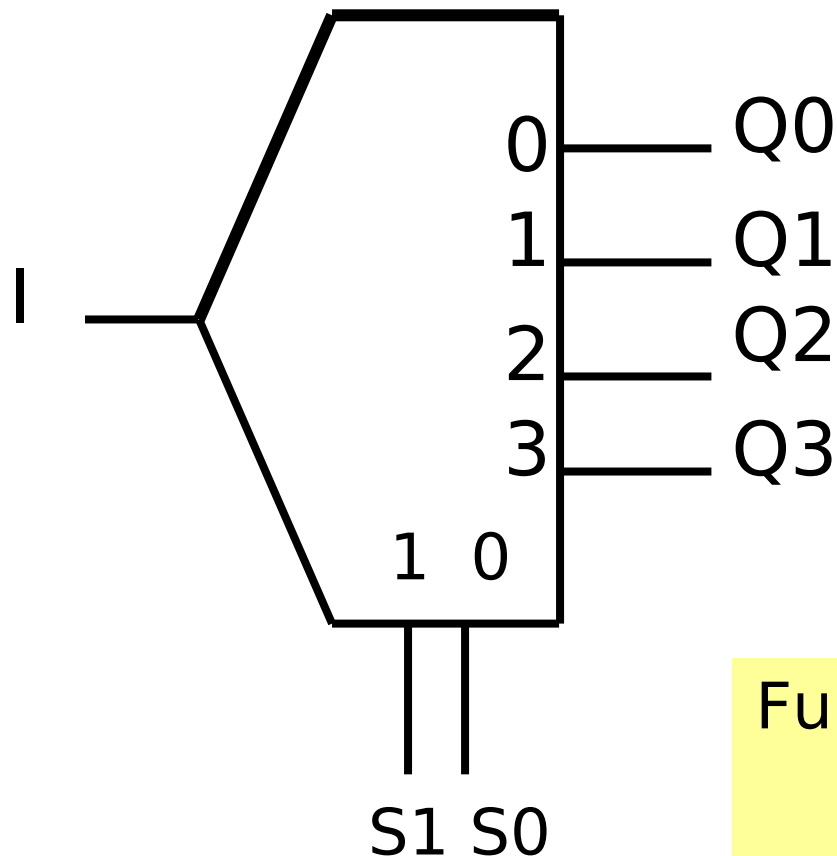
c) Tabla de verdad

Entradas de comparación				Entradas en cascada			Salidas		
A3,B3	A2,B2	A1,B1	A0,B0	A > B	A < B	A = B	A > B	A < B	A = B
A3 > B3	X	X	X	X	X	X	1	0	0
A3 < B3	X	X	X	X	X	X	0	1	0
A3 = B3	A2 > B2	X	X	X	X	X	1	0	0
A3 = B3	A2 < B2	X	X	X	X	X	0	1	0
A3 = B3	A2 = B2	A1 > B1	X	X	X	X	1	0	0
A3 = B3	A2 = B2	A1 < B1	X	X	X	X	0	1	0
A3 = B3	A2 = B2	A1 = B1	A0 > B0	X	X	X	1	0	0
A3 = B3	A2 = B2	A1 = B1	A0 < B0	X	X	X	0	1	0
A3 = B3	A2 = B2	A1 = B1	A0 = B0	1	0	0	1	0	0
A3 = B3	A2 = B2	A1 = B1	A0 > B0	0	1	0	0	1	0
A3 = B3	A2 = B2	A1 = B1	A0 < B0	0	0	1	0	0	1
A3 = B3	A2 = B2	A1 = B1	A0 = B0	X	X	1	0	0	1
A3 = B3	A2 = B2	A1 = B1	A0 > B0	1	1	0	0	0	0
A3 = B3	A2 = B2	A1 = B1	A0 < B0	0	0	0	1	1	0

Subsistemas de propósito específico:

2.5 Demultiplexores

Permiten llevar la información de una línea de entrada, a alguno de los 2^n canales de salida, usando n líneas de selección



DEMUX 1:4

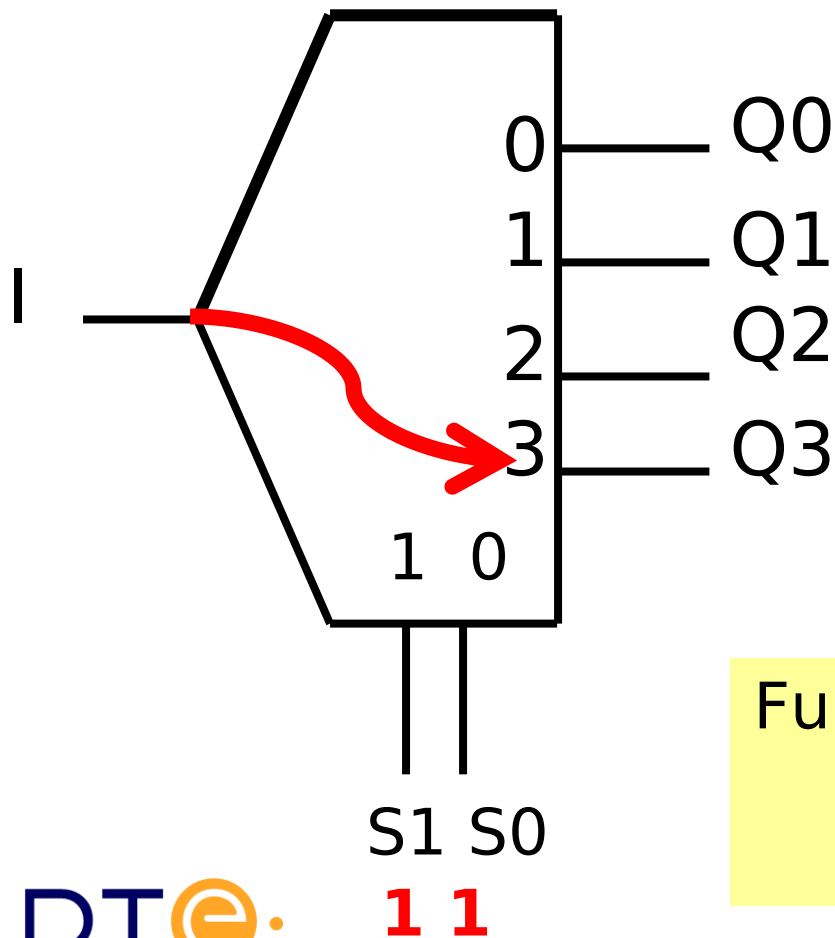
I	S1	S0	Q0	Q1	Q2	Q3
0	X	X	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

Funcionalmente equivalente a un
Decodificador
(DEC2:4/DEMUX1:4)

Subsistemas de propósito específico:

2.5 Demultiplexores

Permiten llevar la información de una línea de entrada, a alguno de los 2^n canales de salida, usando n líneas de selección



DEMUX 1:4

I	S1	S0	Q0	Q1	Q2	Q3
0	X	X	0	0	0	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	0	1

Funcionalmente equivalente a un
Decodificador
(DEC2:4/DEMUX1:4)

Demultiplexor (descripción verilog)

Ej: Descripción Verilog “funcional”

DEMUX 1:4

```
// Demultiplexor 1:4 con salidas en alto
// descripción verilog “funcional”

module demultiplexor_1a4_df_v(I, S0, S1, Q0, Q1, Q2, Q3);
    input I, S0, S1;
    output Q0, Q1, Q2, Q3;

    assign Q0= I & ~S1 & ~S0;
    assign Q1= I & ~S1 & S0;
    assign Q2= I & S1 & ~S0;
    assign Q3= I & S1 & S0;

endmodule
```

Demultiplexor (descripción verilog) (ii)

Ej: Descripción Verilog “funcional” (usando variables multidimensionales)

DEMUX 1:4

```
// Demultiplexor 1:4 con salidas en alto
// descripción verilog “flujo de datos”

module demultiplexor_1a4_df_v(I, S, Q);
    input I, [1:0]S;
    output [3:0]Q;

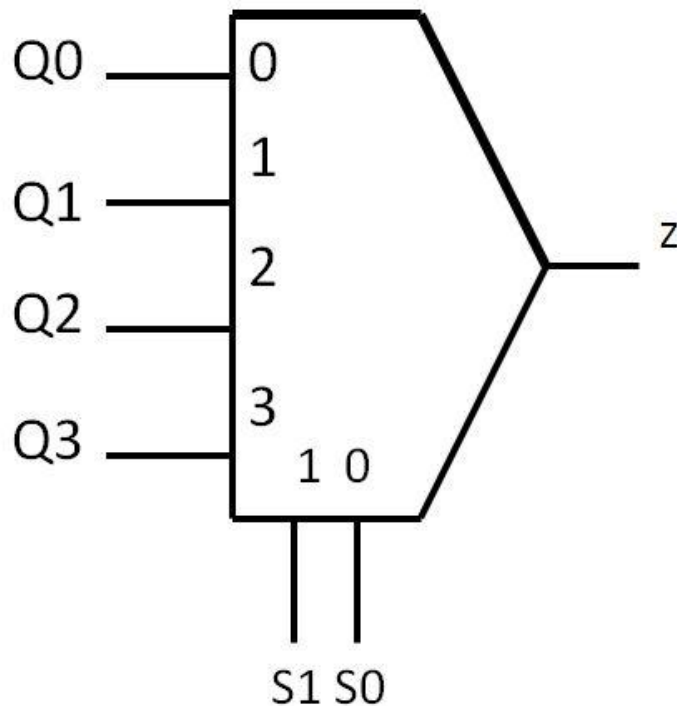
    assign Q[0]= I & ~S[1] & ~S[0];
    assign Q[1]= I & ~S[1] & S[0];
    assign Q[2]= I & S[1] & ~S[0];
    assign Q[3]= I & S[1] & S[0];

endmodule
```

Subsistemas de propósito general:

3.1 Multiplexor MUX $2^n:1$ (MUX-n)

*Dispone de 2^n canales de entrada, 1 línea de salida
entradas de selección de canal, que permiten elegir
el dato de entrada que estará presente en la salida*



S1	S0	Z
0	0	Q0
0	1	Q1
1	0	Q2
1	1	Q3

$$Z = \bar{S}_1 \bar{S}_0 Q_0 + \bar{S}_1 S_0 Q_1 + S_1 \bar{S}_0 Q_2 + S_1 S_0 Q_3$$

Multiplexor (descripción Verilog)

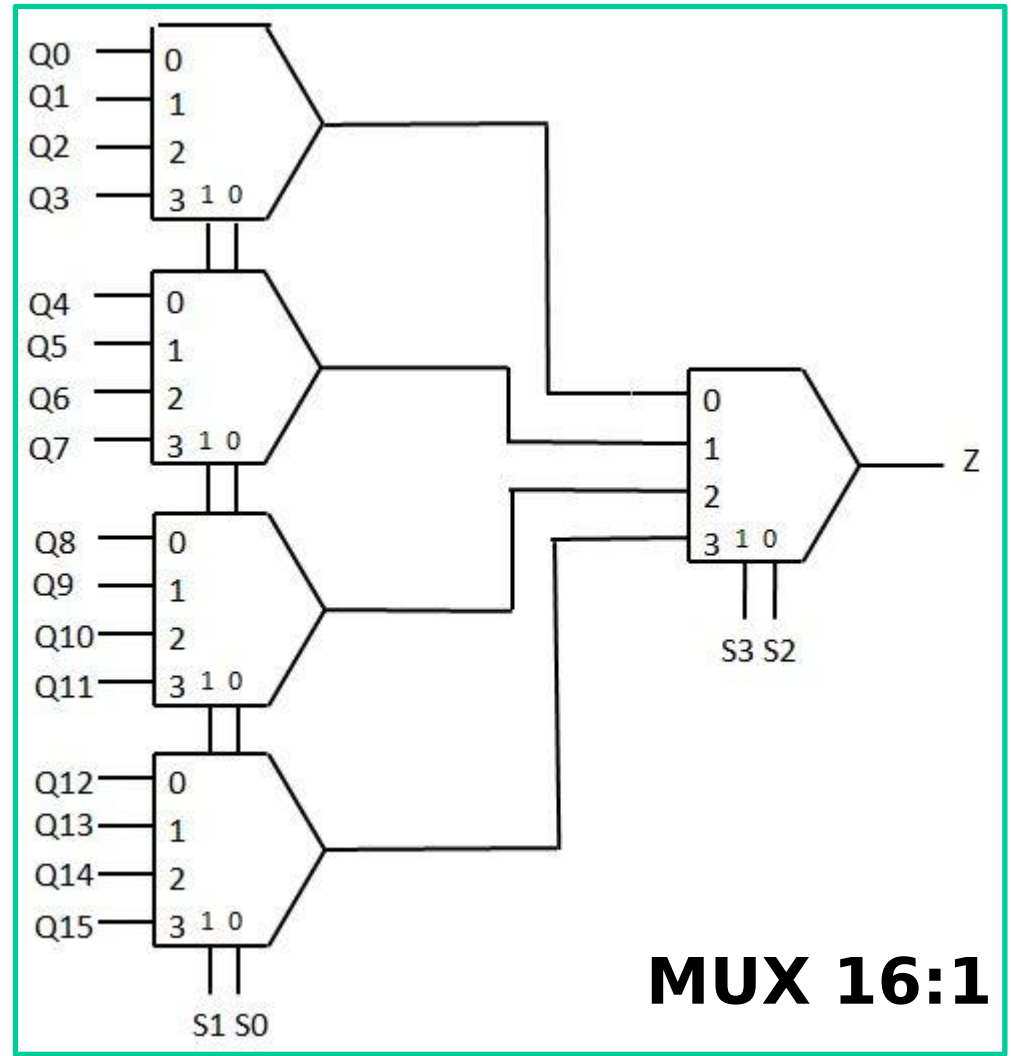
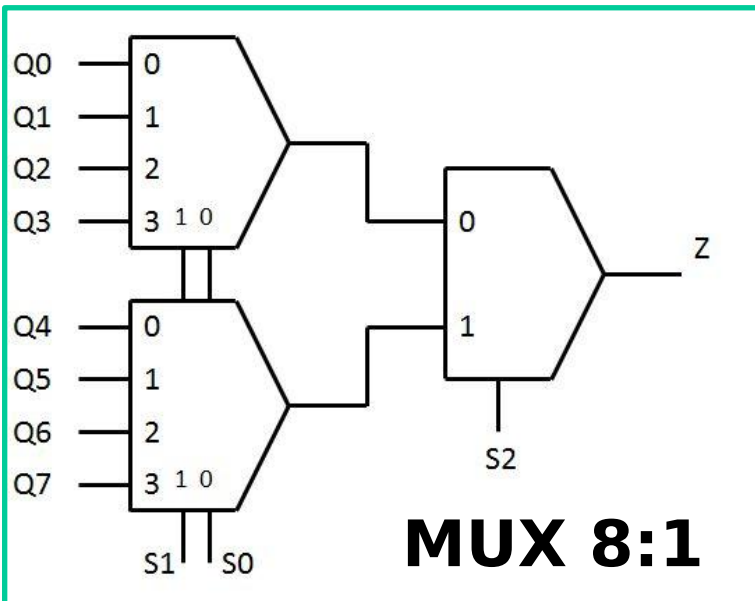
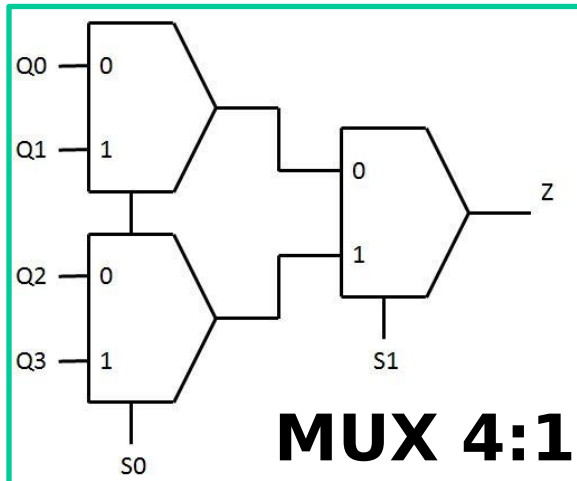
Ej: Descripción funcional Verilog MUX 4:1

```
// Multiplexor 4: con salidas en alto
// descripción verilog "funcional"

module multiplexor_4a1_df_v(F, S0, S1, Q0, Q1, Q2, Q3);
    input Q0, Q1, Q2, Q3;
    output F, S0, S1;

    assign z= (~S1 & ~S0 & Q0) | (~S1 & S0 & Q1) |
              (S1 & ~S0 & Q2) | (~S1 & ~S0 & Q3);
endmodule
```

Asociación de multiplexores

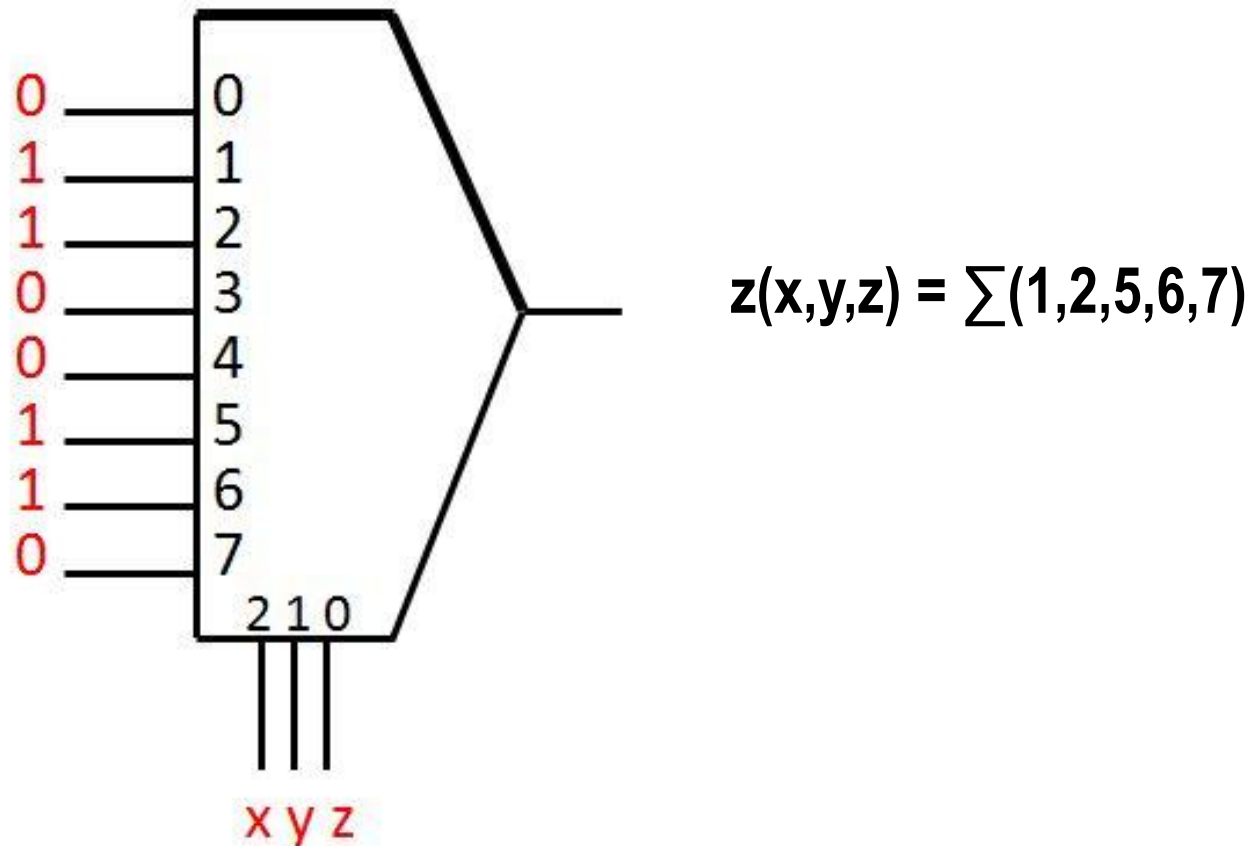


Realización de funciones de n variables con Multiplexores

MUX-n: permite implementar cualquier función de **n variables**

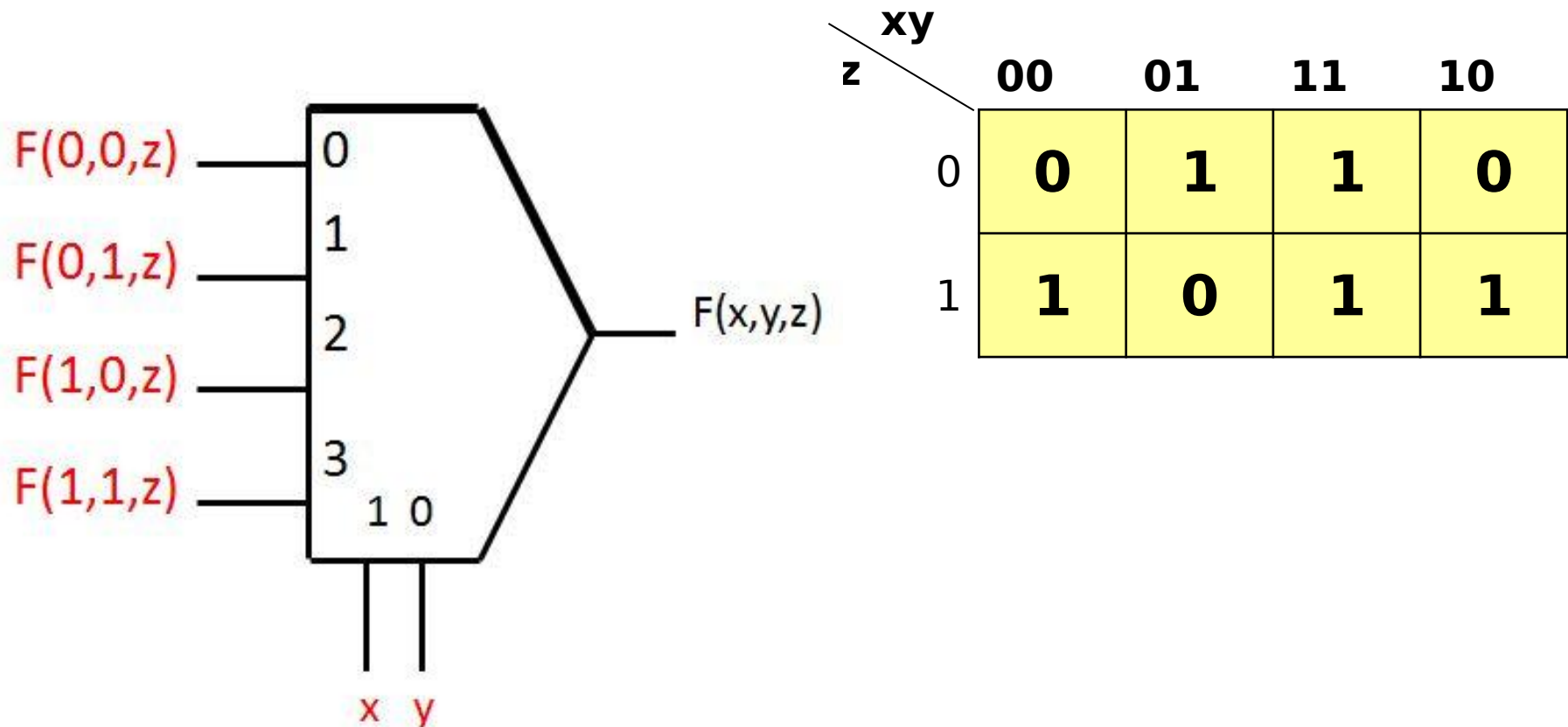
MUX-n + INVERSOR: permite implementar cualquier función de **n+1 variables**

Ejemplo 1:



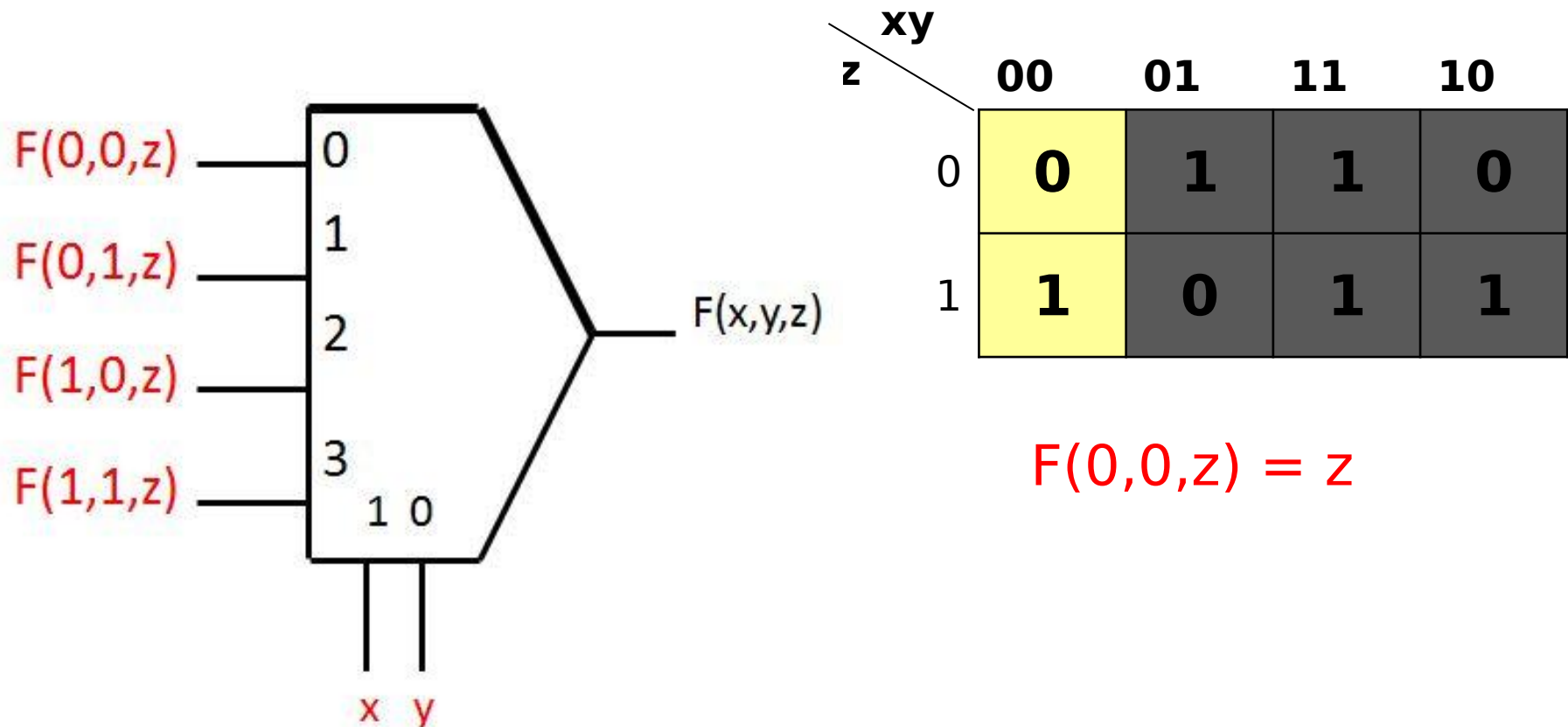
Realización de funciones de n variables con Multiplexores (ii)

Ejemplo 2: Realizar $F(x,y,z)$ con MUX-2



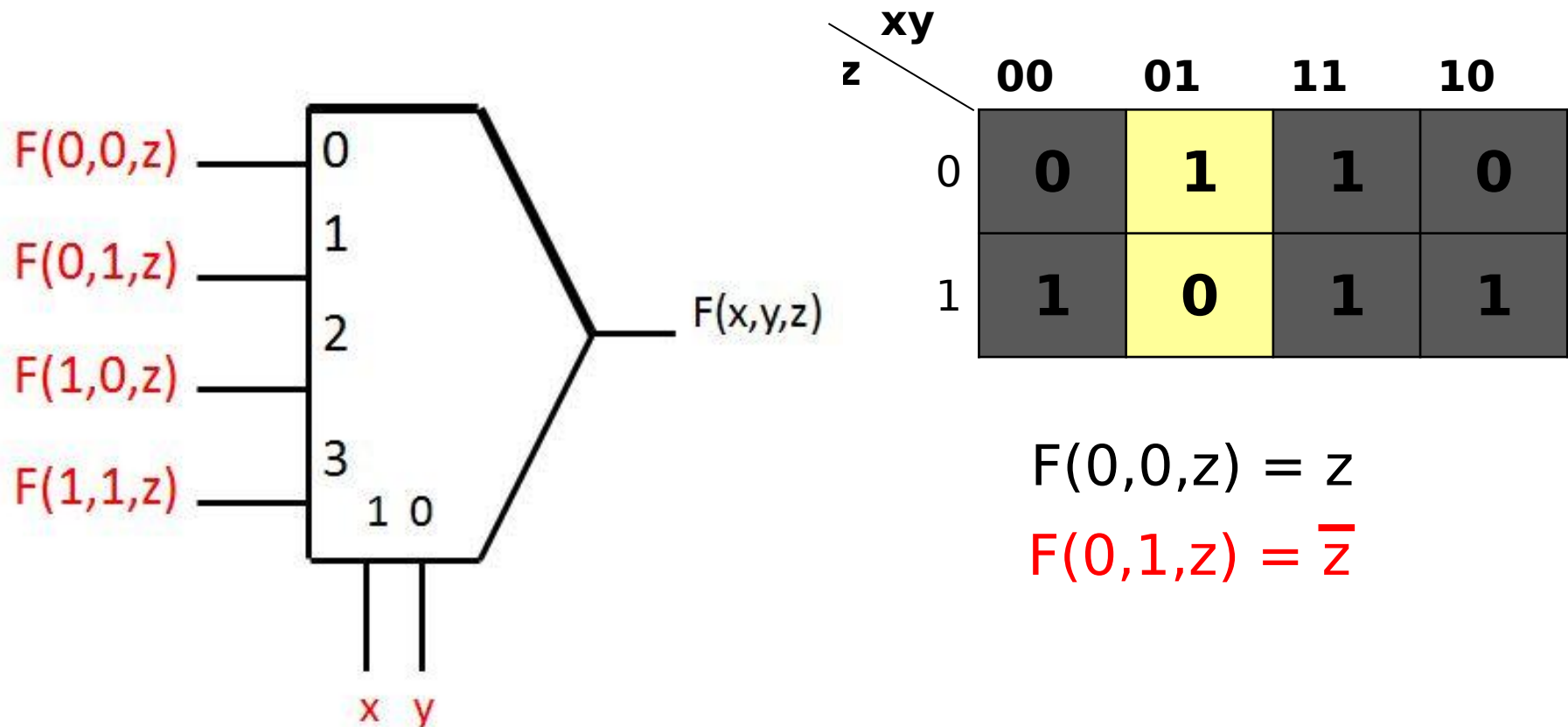
Realización de funciones de n variables con Multiplexores (ii)

Ejemplo 2: Realizar $F(x,y,z)$ con MUX-2



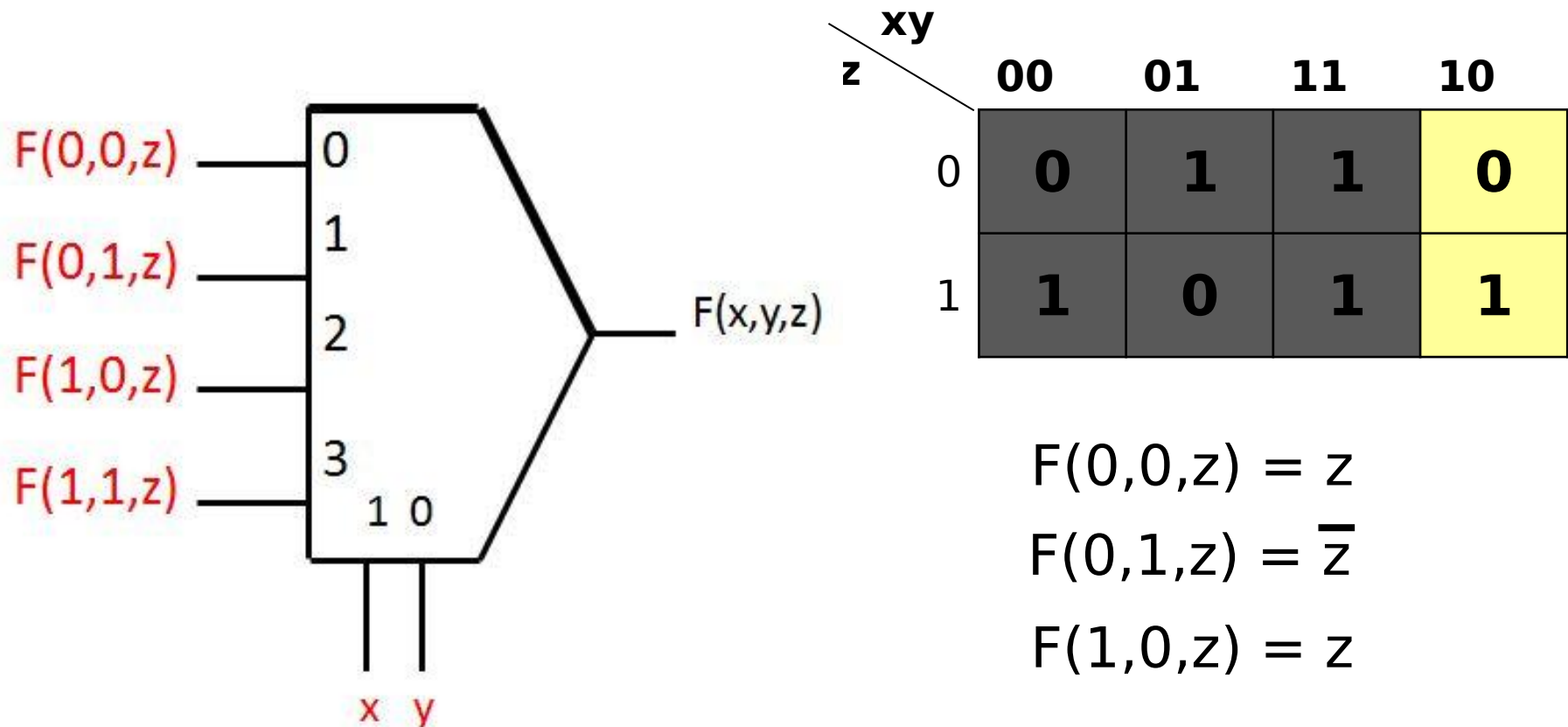
Realización de funciones de n variables con Multiplexores (ii)

Ejemplo 2: Realizar $F(x,y,z)$ con MUX-2



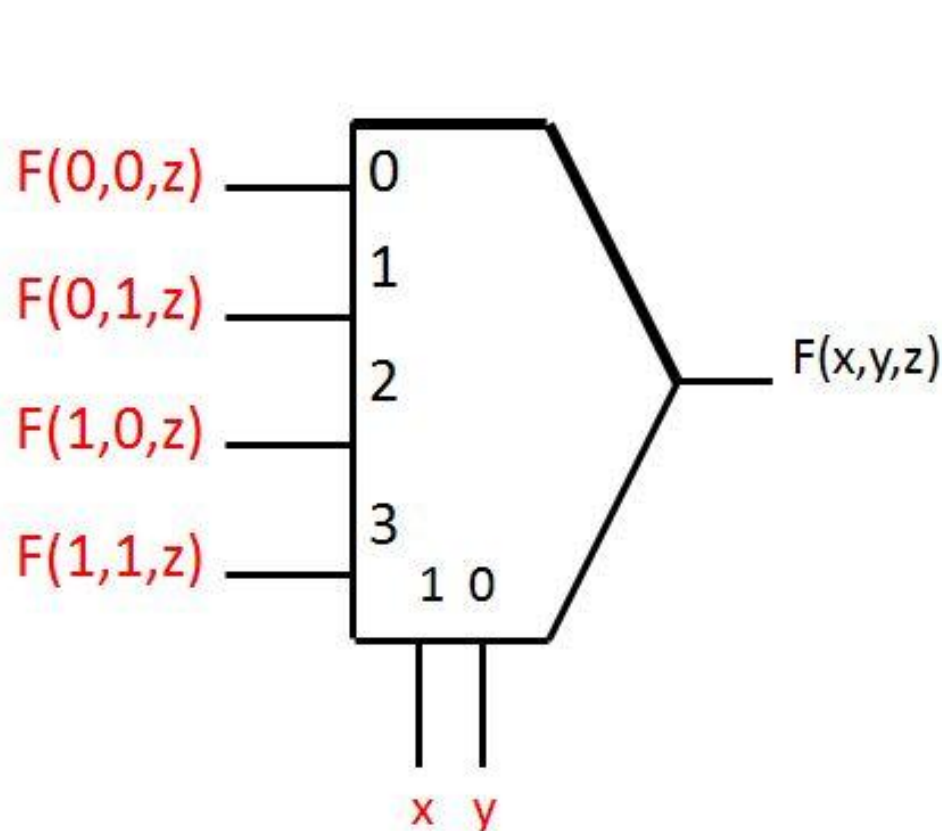
Realización de funciones de n variables con Multiplexores (ii)

Ejemplo 2: Realizar $F(x,y,z)$ con MUX-2



Realización de funciones de n variables con Multiplexores (ii)

Ejemplo 2: Realizar $F(x,y,z)$ con MUX-2



		xy			
		00	01	11	10
z	0	0	1	1	0
	1	1	0	1	1

$$F(0,0,z) = z$$

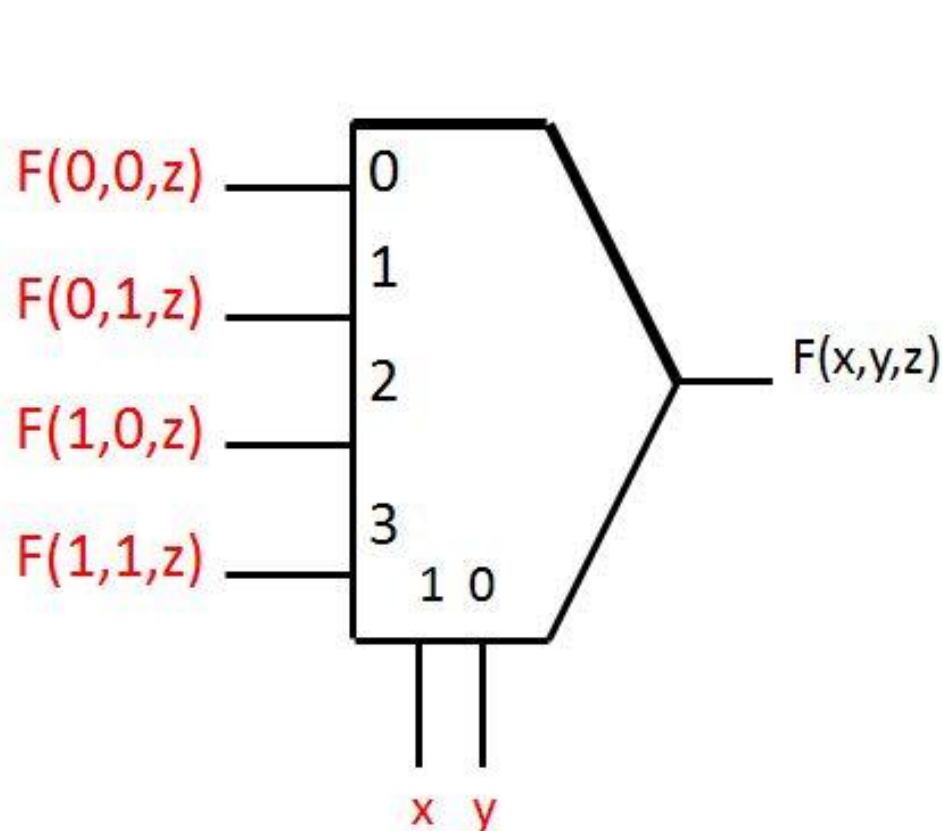
$$F(0,1,z) = \bar{z}$$

$$F(1,0,z) = z$$

$$F(1,1,z) = 1$$

Realización de funciones de n variables con Multiplexores (ii)

Ejemplo 2: Realizar $F(x,y,z)$ con MUX-2



		xy			
		00	01	11	10
z	0	0	1	1	0
	1	1	0	1	1

$$F(0,0,z) = z$$

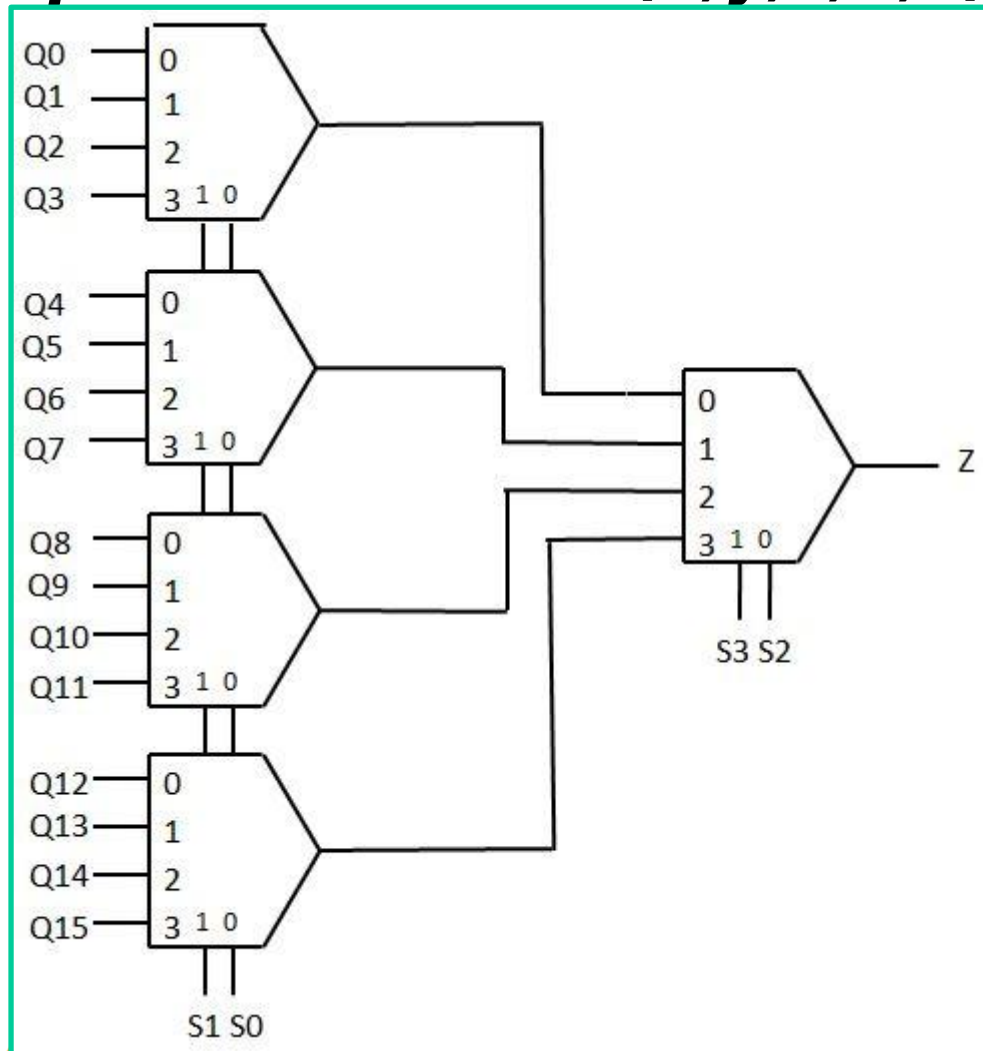
$$F(0,1,z) = \bar{z}$$

$$F(1,0,z) = z$$

$$F(1,1,z) = 1$$

Realización de funciones de n variables con Multiplexores (iii)

Ejemplo 3: Realizar $F(x,y,z,u,v)$ con MUX-2



5 variables requiere:
MUX-4 (+ inversor)

Asociamos MUX-2
para conseguir MUX-4

Realización de funciones de n variables con Multiplexores (iii)

Ejemplo 3: Realizar $F(x,y,z,u,v)$ con MUX-2

xyz uv		000	001	011	010	110	111	101	100
00	1	1	1	1	0	1	1	1	1
01	1	1	1	0	0	1	1	1	0
11	1	1	0	1	0	1	1	0	1
10	1	1	0	0	0	1	1	0	0

$$F(0,0,z,u,v) = 1$$

$$F(0,1,z,u,v) = F(1,0,z,u,v)$$

$$F(1,1,z,u,v) = z$$

Realización de funciones de n variables con Multiplexores (iv)

Ejemplo 3: (cont.)

