1.1. Diseñar un circuito multiplexor con prioridad de 4 bits. El circuito tiene 4 entradas de datos (I3-I0), 4 entradas de selección (S3-S0) y dos salidas Z y G. Cuando una o más de las entradas S están a 1, Z toma el valor de la entrada Ii, siendo i es el índice más alto de las entradas Si que están a 1; si todas las entradas S3-S0 están a 0, entonces Z toma el valor 0. La salida G se fija a 1 si al menos alguna entrada Si está a 1, en caso contrario se fija a 0.

a) Mostrar en una tabla el comportamiento lógico del circuito. Encontrar las ecuaciones lógicas de la salidas Z y G expresándolas en dos niveles y en forma factorizada.

Función	S3	S2	S1	S0	Z	G
$S3\ I3$	1	X	X	X	I3	1
$\overline{S3}\ S2\ I2$	0	1	X	X	I2	1
$\overline{S3}\ \overline{S2}\ S1\ I1$	0	0	1	X	I1	1
$\overline{S3}\ \overline{S2}\ \overline{S1}\ S0\ I0$	0	0	0	1	I0	1
$\overline{S3}\ \overline{S2}\ \overline{S1}\ \overline{S0}$ • 0	0	0	0	0	0	0

$$G = S3 + S2 + S1 + S0$$

$$Z = S3\ I3 + \overline{S3}\ S2\ I2 + \overline{S3}\ \overline{S2}\ S1\ I1 + \overline{S3}\ \overline{S2}\ \overline{S1}\ S0\ I0$$

$$Z = S3\ I3 + \overline{S3}\ (S2\ I2 + \overline{S2}\ S1\ I1 + \overline{S2}\ \overline{S1}\ S0\ I0)$$

$$Z = S3\ I3 + \overline{S3}\ [S2\ I2 + \overline{S2}\ (S1\ I1 + \overline{S1}\ S0\ I0)]$$

b) Implementar la expresión factorizada de Z utilizando multiplexores de 2 entradas.

$Z = S3\ I3 + \overline{S3}\ [S2\ I2 + \overline{S2}\ (S1\ I1 + \overline{S1}\ S0\ I0)]$

2-Inp MUX

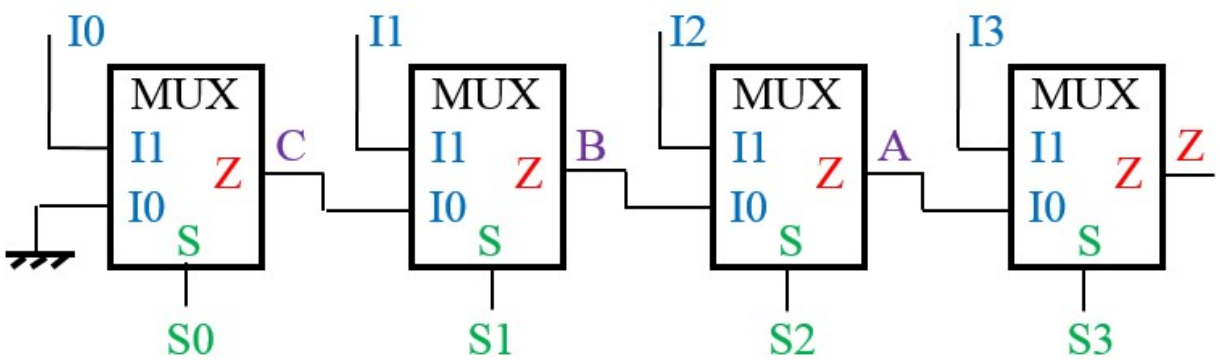
$Z = S\ I1 + \overline{S}\ I0$

$Z = S3\ I3 + \overline{S3}\ A$

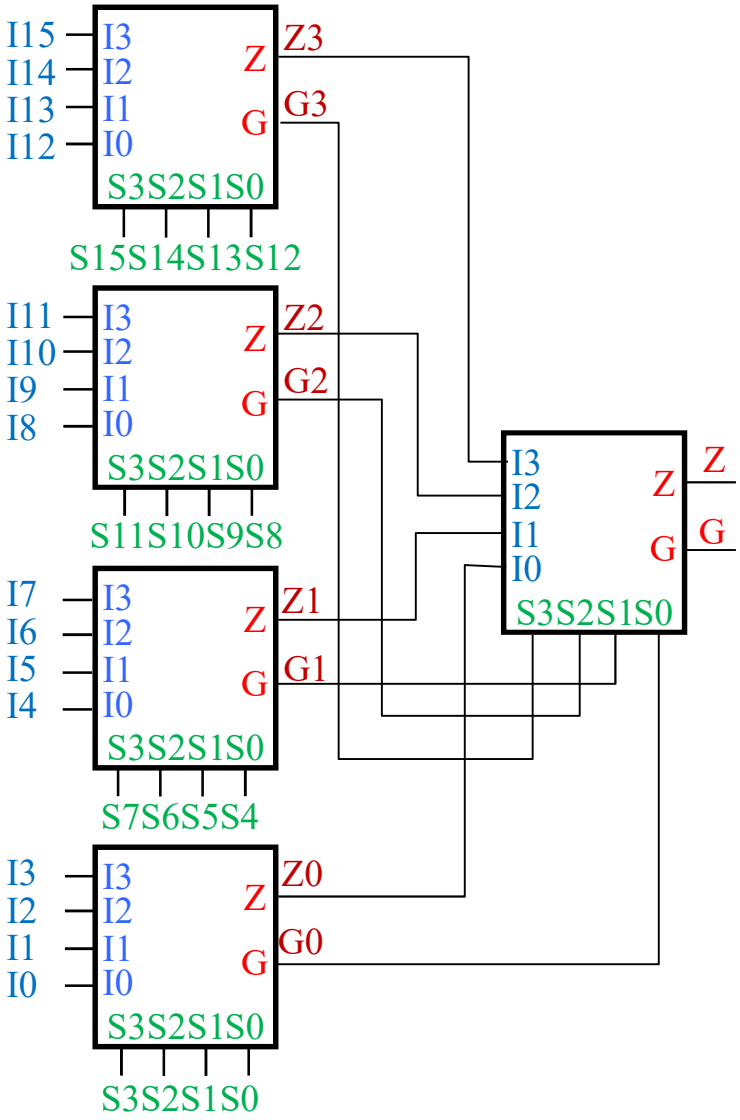
$A = S2\ I2 + \overline{S2}\ (S1\ I1 + \overline{S1}\ S0\ I0) = S2\ I2 + \overline{S2}\ B$

$B = S1\ I1 + \overline{S1}\ S0\ I0 = S1\ I1 + \overline{S1}\ C$

$C = S0\ I0 = S0\ I0 + \overline{S0} \bullet 0$



c) Diseñar un multiplexor con prioridad de 16 bits en base a los multiplexores con prioridad de 4 bits diseñados.



Si hay algún 1 en S15-S12 => G3 = 1
=> Z = Z3 (uno de I15-I12), G = 1

Si no:
Si hay algún 1 en S11-S8 => G2 = 1
=> Z = Z2 (uno de I11-I8), G = 1

Si no:
Si hay algún 1 en S7-S4 => G1 = 1
=> Z = Z1 (uno de I7-I4), G = 1

Si no:
Si hay algún 1 en S3-S0 => G0 = 1
=> Z = Z0 (uno de I3-I0), G = 1

Si no:
G3-0 = 0 => Z = 0, G = 0

d) Realizar una descripción VHDL del multiplexor con prioridad.

```
library ieee;
use ieee.std_logic_1164.all;

entity muxconpri4 is
port( S, I: in std_logic_vector(3 downto 0);
      Z, G: out std_logic );
end muxconpri4;

architecture uno of muxconpri4 is
begin
process(S, I)
begin
if ( S(3) = '1' ) then
   Z <= I(3); G <= '1';
elsif ( S(2) = '1' ) then
   Z <= I(2); G <= '1';
elsif ( S(1) = '1' ) then
   Z <= I(1); G <= '1';
elsif ( S(0) = '1' ) then
   Z <= I(0); G <= '1';
else Z <= '0'; G <= '0';
end if;
end process;
end uno;
```

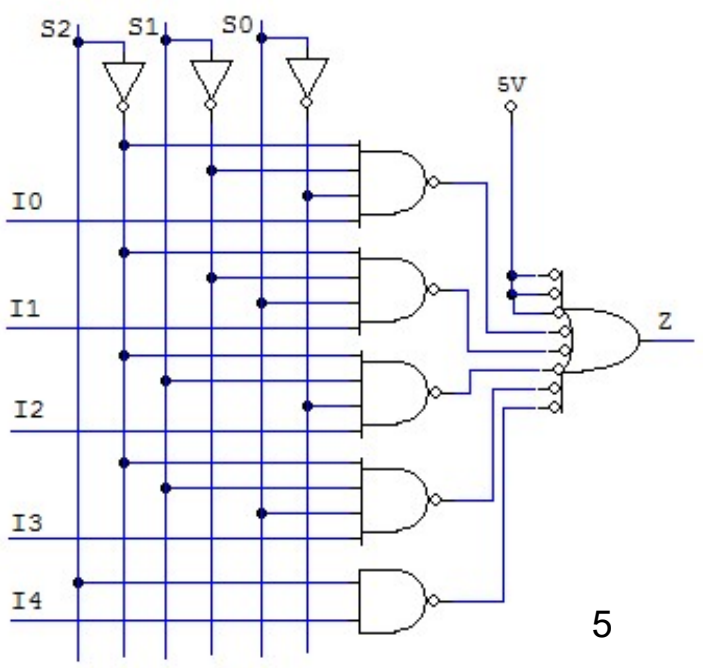
```
library ieee;
use ieee.std_logic_1164.all;

entity muxconprigen is
generic (N: integer := 4);
port( S, I: in std_logic_vector(N-1 downto 0);
      Z, G: out std_logic );
end muxconprigen;

architecture uno of muxconprigen is
begin
process(S, I)
begin
Z <= '0'; G <= '0';
for j in N-1 downto 0 loop
   if ( S(j) = '1' ) then
      Z <= I(j); G <= '1';
      exit; -- Finaliza el lazo si hay 1 en S
   end if;
end loop;
end process;
end uno;
```

2.1. Construir un multiplexor de 5 entradas a) utilizando puertas lógicas.

Función	S2	S1	S0	Z
$\overline{S2}\ \overline{S1}\ \overline{S0}\ I0$	0	0	0	I0
$\overline{S2}\ \overline{S1}\ S0\ I1$	0	0	1	I1
$\overline{S2}\ S1\ \overline{S0}\ I2$	0	1	0	I2
$\overline{S2}\ S1\ S0\ I3$	0	1	1	I3
$S2\ \overline{S1}\ \overline{S0}\ I4$	1	0	0	I4

Función	S2	S1	S0	Z
$\overline{S2}\ \overline{S1}\ \overline{S0}\ I0$	0	0	0	I0
$\overline{S2}\ \overline{S1}\ S0\ I1$	0	0	1	I1
$\overline{S2}\ S1\ \overline{S0}\ I2$	0	1	0	I2
$\overline{S2}\ S1\ S0\ I3$	0	1	1	I3
$S2\ I4$	1	X	X	I4

$$Z = \overline{S2}\ \overline{S1}\ \overline{S0}\ I0 + \overline{S2}\ \overline{S1}\ S0\ I1 + \overline{S2}\ S1\ \overline{S0}\ I2 + \overline{S2}\ S1\ S0\ I3 + S2\ I4$$



2.1. Construir un multiplexor de 5 entradas
b) utilizando multiplexores de dos entradas.

$$Z = \overline{S2}\,\overline{S1}\,\overline{S0}\,I0 + \overline{S2}\,\overline{S1}\,S0\,I1 + \overline{S2}\,S1\,\overline{S0} + \overline{S2}\,S1\,S0\,I3 + S2\,I4$$

$$Z = S2\,I4 + \overline{S2}\,[S1\,(S0\,I3 + \overline{S0}\,I2) + \overline{S1}\,(S0\,I1 + \overline{S0}\,I0)]$$

$$Z = S2\,I4 + \overline{S2}\,A$$

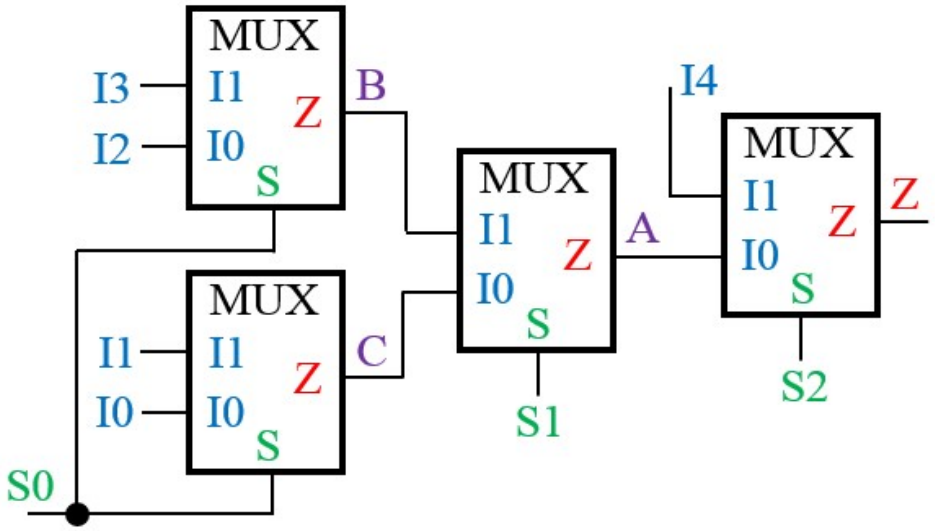
$$A = S1\,B + \overline{S1}\,C$$

$$B = S0\,I3 + \overline{S0}\,I2$$

$$C = S0\,I1 + \overline{S0}\,I0$$

2-Inp MUX

$$Z = S\,I1 + \overline{S}\,I0$$



2.2. Un circuito de “desplazamiento en barril” (“barrel-shifter”) mueve los datos de entrada de forma que aparezcan en la salida girados el número de posiciones marcados por las señales de control. Construir utilizando multiplexores un “barrel-shifter” de 4 bits de entrada (a3a2a1a0) y 4 bits de salida (z3z2z1z0) con 4 posibles desplazamientos (dos señales de control c1c0):

(c1c0) = 0 => (z3z2z1z0) = (a3a2a1a0),

(c1c0) = 1 => (z3z2z1z0) = (a2a1a0a3),

(c1c0) = 2 => (z3z2z1z0) = (a1a0a3a2),

(c1c0) = 3 => (z3z2z1z0) = (a0a3a2a1).

Realizar la descripción VHDL de este circuito.

S1	S0	Z	C1	C0	Z3	Z2	Z1	Z0
0	0	I0	0	0	A3	A2	A1	A0
0	1	I1	0	1	A2	A1	A0	A3
1	0	I2	1	0	A1	A0	A3	A2
1	1	I3	1	1	A0	A3	A2	A1

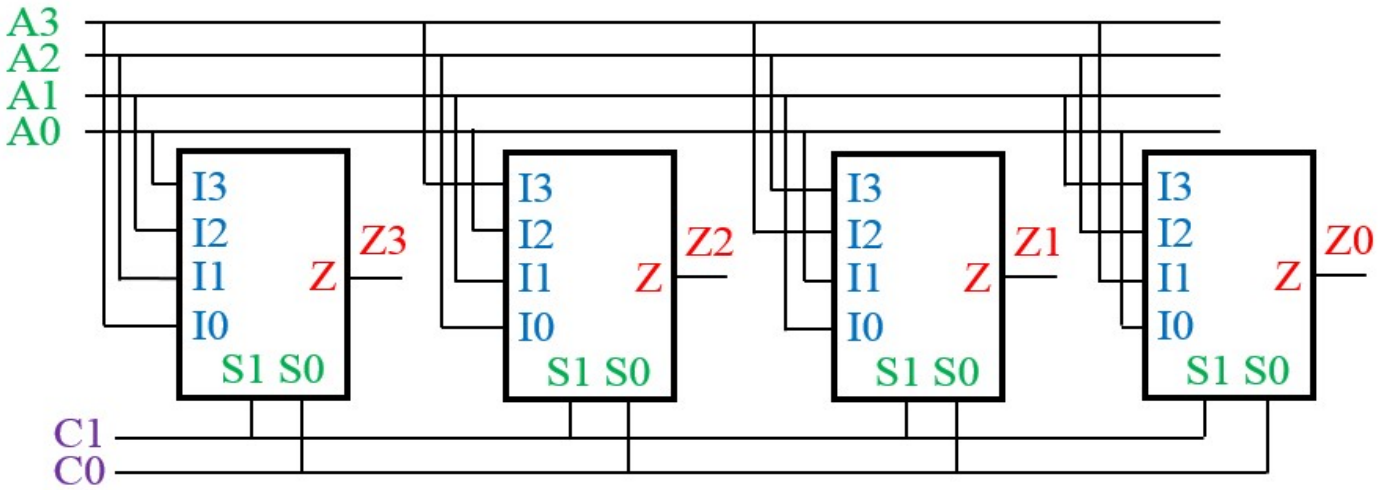
4-Inp Mux. Barrel-Shifter

Se necesitan cuatro 4-Inp Muxs. Uno por cada salida Z.

S1	S0	Z	C1	C0	Z3	Z2	Z1	Z0
0	0	I0	0	0	A3	A2	A1	A0
0	1	I1	0	1	A2	A1	A0	A3
1	0	I2	1	0	A1	A0	A3	A2
1	1	I3	1	1	A0	A3	A2	A1

4-Inp Mux. Barrel-Shifter

```
library ieee;
use ieee.std_logic_1164.all;

entity barrel is
port(A: in std_logic_vector(3 downto 0);
     C: in std_logic_vector(1 downto 0);
     Z: out std_logic_vector(3 downto 0) );
end barrel;

architecture uno of barrel is
begin
process(A,C)
begin
case C is
   when "00" => Z <= A;
   when "01" => Z <= A(2 downto 0) & A(3);
   when "10" => Z <= A(1) & A(0) & A(3) & A(2);
   when "11" => Z <= A(0) & A(3 downto 1);
end case;
end process;
end uno;
```

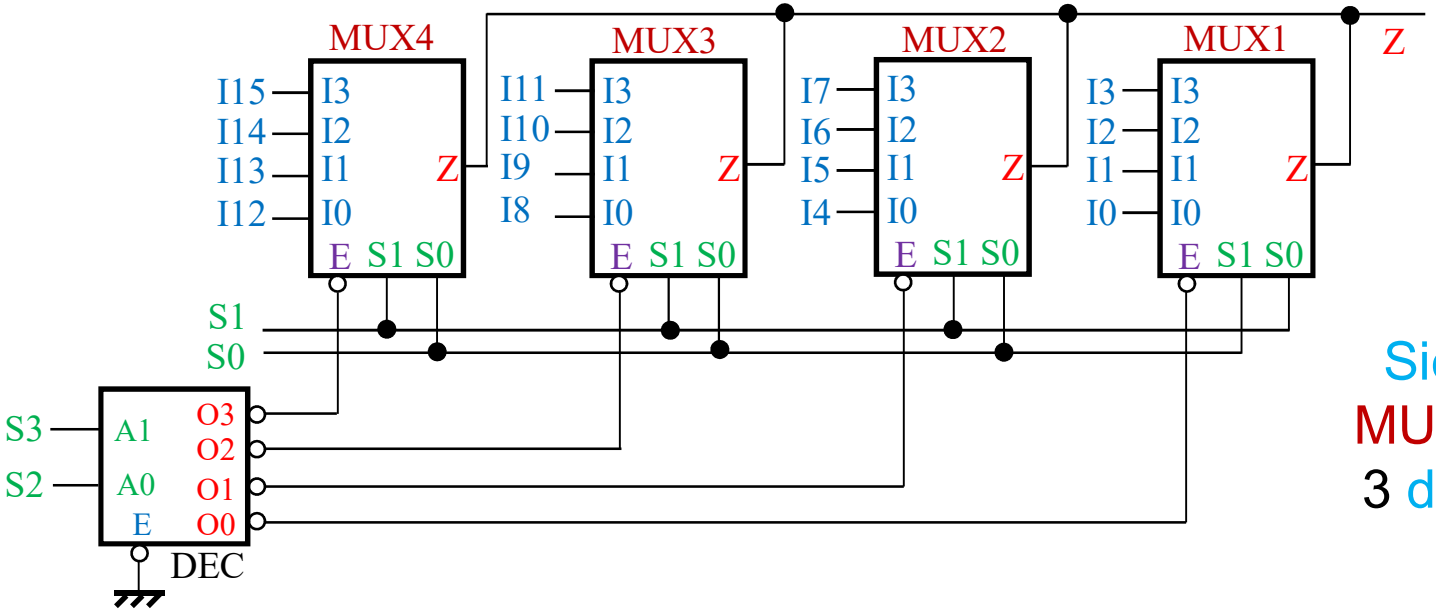


3.1. Diseñar un multiplexor de 16 entradas utilizando 4 multiplexores triestado de 4 entradas con habilitador (el circuito deshabilitado queda en alta impedancia) y un decodificador 2 a 4.

16 entradas de datos I15-0
4 entradas de selección S3-0

⇨ Cuatro 4-Inp MUX

4 entradas de datos I3-0
2 entradas de selección S1-0
1 Enable



Siempre hay 1 MUX habilitado y 3 deshabilitados (a Hi-Z)

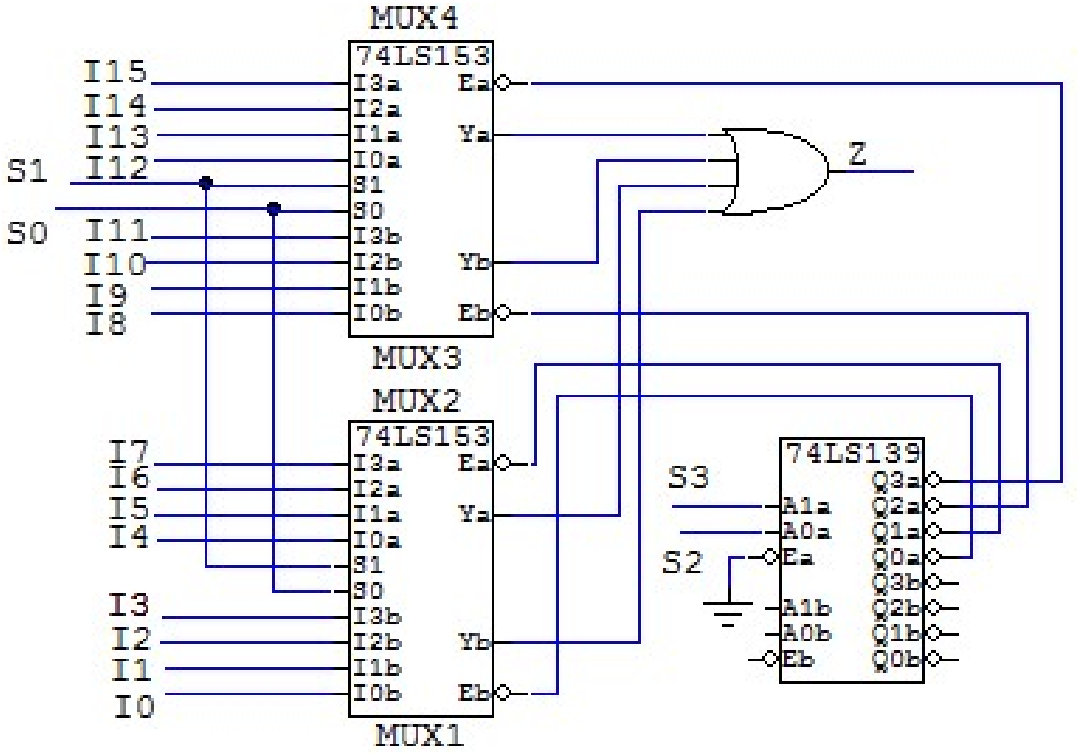
S3	S2	MUX1	MUX2	MUX3	MUX4	Z (F(S1,S0))
0	0	ON	OFF	OFF	OFF	I3,I2,I1,I0
0	1	OFF	ON	OFF	OFF	I7,I6,I5,I4
1	0	OFF	OFF	ON	OFF	I11,I10,I9,I8
1	1	OFF	OFF	OFF	ON	I15,I14,I13,I12

3.1. Diseñar un multiplexor de 16 entradas.
b) Indicar como debería diseñarse el circuito con dos chips 74'153, un chip 74'139 y una puerta lógica.

16 entradas de datos I15-0
4 entradas de selección S3-0

⇨ Cuatro 4-Inp MUX

4 entradas de datos I3-0
2 entradas de selección S1-0
1 Enable



Siempre hay 1 MUX habilitado y 3 deshabilitados (a 0).
Las 4 salidas de los 4-Inp Muxs se unen con una OR.

Z = Y4 + Y3 + Y2 + Y1

Si 3 Ys están a 0
Z = Yi (i+1 = S3S2)

3.2. Se quiere diseñar un decodificador de 40 direcciones de 0 a 39 utilizando decodificadores binarios (2 a 4, 3 a 8, 4 a 16, etc). Indicar cuál es el número mínimo de decodificadores binarios que hay que utilizar y realizar el diseño del decodificador utilizando los decodificadores binarios y las puertas lógicas que sean necesarias (un inversor).

40 direcciones (de 0 a 39) ⇨ $39 = (1\ 0\ 0\ 1\ 1\ 1)_2$

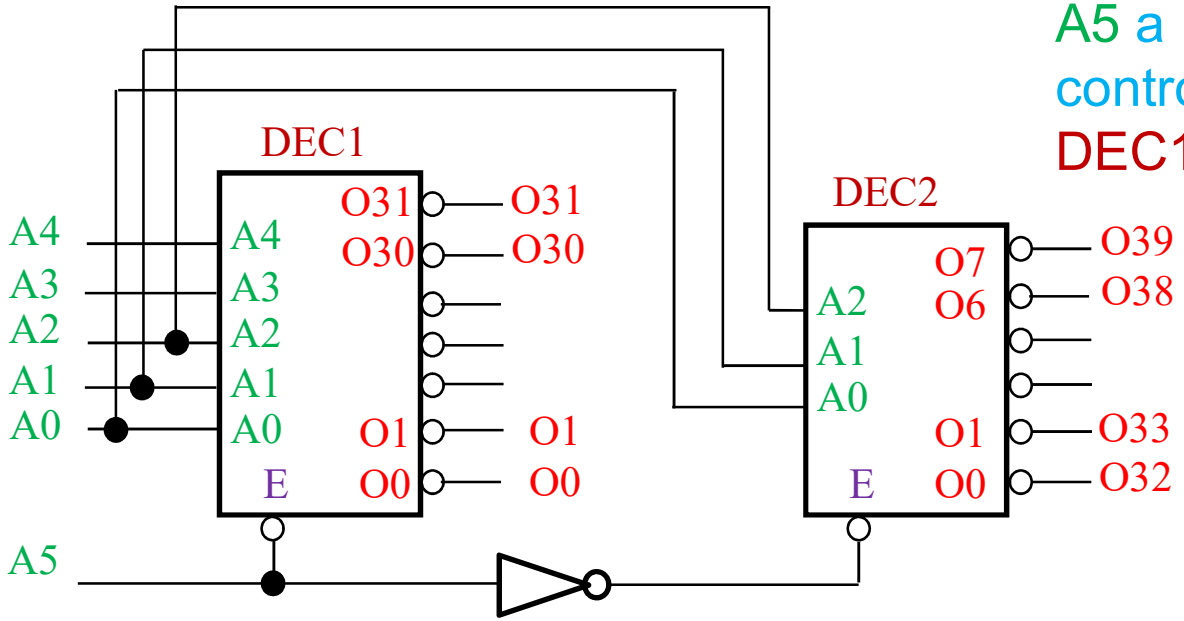
6 bits de entrada (A5-A0)

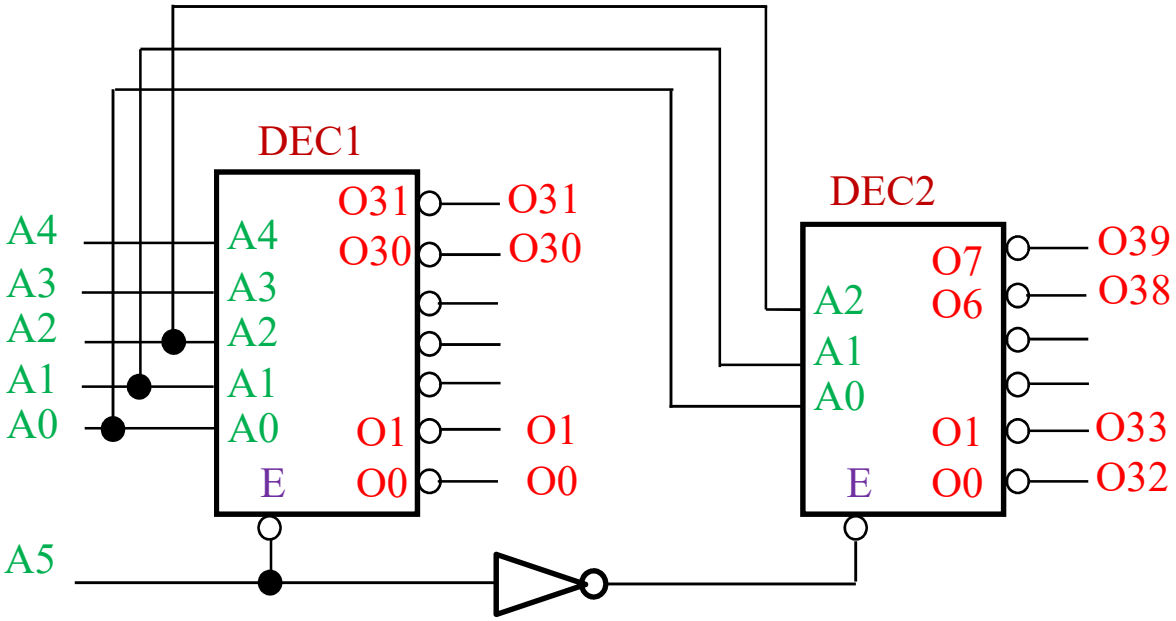
40 = 32 + 8 ⇨ 2 DECs: 5 a 32, 3 a 8

DEC1 DEC2

A5 a 0, direcciones 0 a 31 controladas por A4-A0
DEC1 ON, DEC2 OFF

A5 a 1, direcciones 32 a 39 controladas por A2-A0
DEC1 OFF, DEC2 ON





Ejemplos:

32	16	8	4	2	1				
A5	**A4**	**A3**	**A2**	**A1**	**A0**	**DEC1**	**DEC2**	**O31-0**	**O39-32**
0	0	0	0	0	0	ON	OFF	O0 a 1, resto a 0	salidas a 0
0	0	0	1	1	0	ON	OFF	O6 a 1, resto a 0	salidas a 0
0	1	0	1	0	0	ON	OFF	O20 a 1, resto a 0	salidas a 0
0	1	1	1	1	1	ON	OFF	O31 a 1, resto a 0	salidas a 0
1	X	X	0	0	0	OFF	ON	salidas a 0	O32 a 1, resto a 0
1	X	X	0	1	1	OFF	ON	salidas a 0	O35 a 1, resto a 0
1	X	X	1	1	1	OFF	ON	salidas a 0	O39 a 1, resto a 0

3.3. Diseñar un circuito decodificador del código de Hamming capaz de recuperar un error simple en un código (M0M1M2M3) con bits de paridad (P0P1P2) par utilizando puertas EXOR (para determinar F2F1F0 la dirección del bit erróneo, y para complementar dicho bit) y un 3 a 8 DEC (para indicar el bit erróneo en función de F2F1F0).

1 P0	2 P1	3 M0	4 P2	5 M1	6 M2	7 M3	
X		X		X		X	F0
	X	X			X	X	F1
			X	X	X	X	F2

Fi es 1 si el nº de 1s es impar => EXOR

$F0 = P0 \oplus M0 \oplus M1 \oplus M3$

$F1 = P1 \oplus M0 \oplus M2 \oplus M3$

$F2 = P2 \oplus M1 \oplus M2 \oplus M3$

(F2F1F0) indica la columna errónea => (O7 O6 O5 O4 O3 O2 O1 O0)

DEC 3 a 8
Inp => Fs
Outs => Os

⇦

Oi es 1 (0) si hay (no hay) error en la columna i.
O0 = 1 => No hay ningún error.

Oi	Ii	Zi
0	0	0
0	1	1
1	0	1
1	1	0

Si Oi es 0, Zi mantiene Ii
Si Oi es 1, Zi cambia Ii

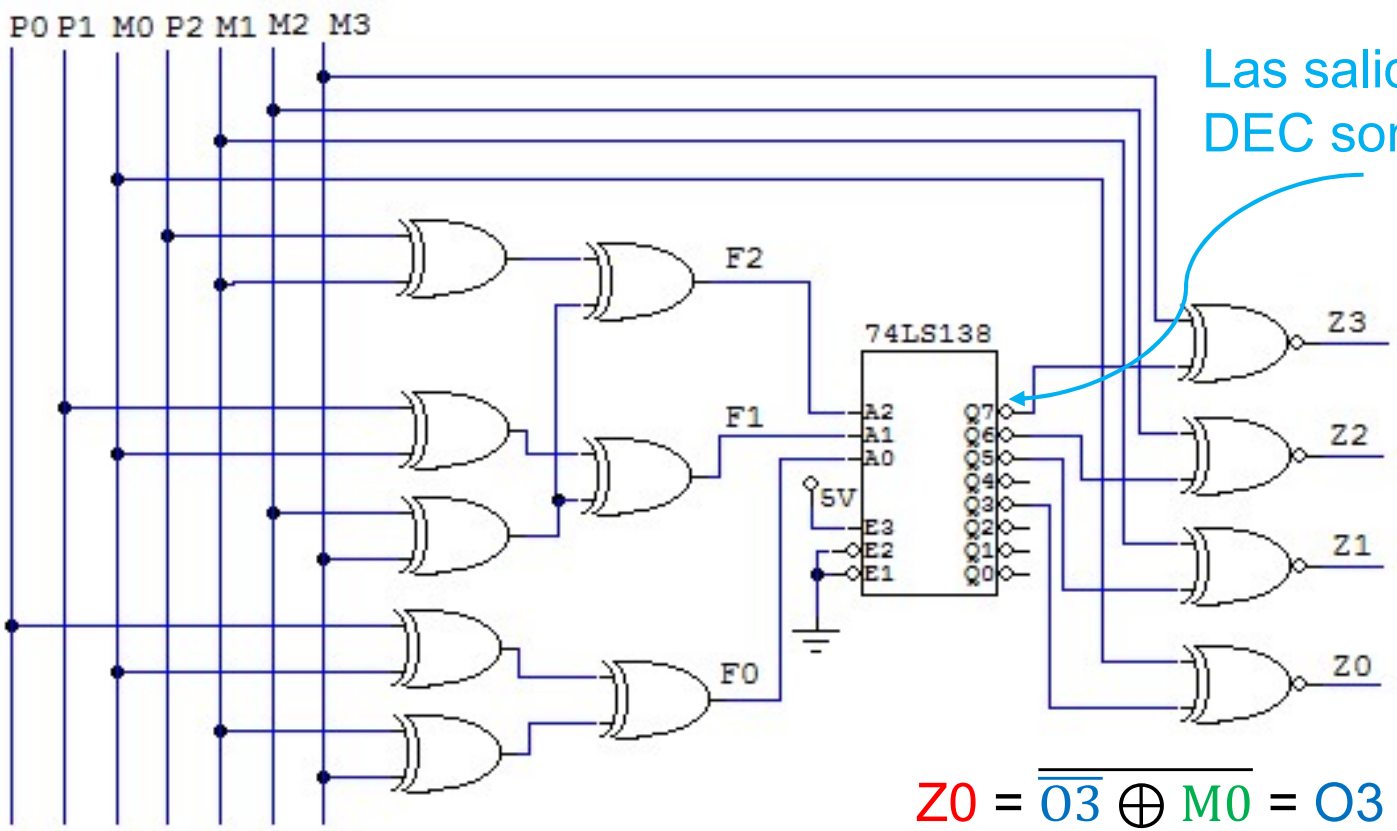
$Z0 = O3 \oplus M0$

$Z1 = O5 \oplus M1$

$Z2 = O6 \oplus M2$

$Z3 = O7 \oplus M3$

3.3. Diseñar un circuito decodificador del código de Hamming capaz de recuperar un error simple en un código (M0M1M2M3) con bits de paridad (P0P1P2) par utilizando puertas EXOR (para determinar F2F1F0 la dirección del bit erróneo, y para complementar dicho bit) y un 3 a 8 DEC (para indicar el bit erróneo en función de F2F1F0).



Las salidas del DEC son .L

$F0 = P0 \oplus M0 \oplus M1 \oplus M3$

$F1 = P1 \oplus M0 \oplus M2 \oplus M3$

$F2 = P2 \oplus M1 \oplus M2 \oplus M3$

$Z0 = \overline{\overline{O3} \oplus M0} = O3 \oplus M0$

$Z1 = \overline{\overline{O5} \oplus M1} = O5 \oplus M1$

$Z2 = \overline{\overline{O6} \oplus M2} = O6 \oplus M2$

$Z3 = \overline{\overline{O7} \oplus M3} = O7 \oplus M3$

4.1. Realizar un multiplicador para números A (a1a0) y B (b1b0) de 2 bits en complemento-2, utilizando decodificadores 74LS138 y puertas NAND de 4 entradas (74LS20) o de 8 entradas (74LS30). El resultado P (p3p2p1p0) es de 4 bits en complemento-2.

Pmax = (-2) * (-2) = +4 Pmin = (-2) * 1 = -2

		-2	1	-2	1			-8	4	2	1
A	**B**	**A1**	**A0**	**B1**	**B0**	**DEC**	**P**	**P3**	**P2**	**P1**	**P0**
0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	1	1	0	0	0	0	0
0	-2	0	0	1	0	2	0	0	0	0	0
0	-1	0	0	1	1	3	0	0	0	0	0
1	0	0	1	0	0	4	0	0	0	0	0
1	1	0	1	0	1	5	1	0	0	0	1
1	-2	0	1	1	0	6	-2	1	1	1	0
1	-1	0	1	1	1	7	-1	1	1	1	1
-2	0	1	0	0	0	8	0	0	0	0	0
-2	1	1	0	0	1	9	-2	1	1	1	0
-2	-2	1	0	1	0	10	4	0	1	0	0
-2	-1	1	0	1	1	11	2	0	0	1	0
-1	0	1	1	0	0	12	0	0	0	0	0
-1	1	1	1	0	1	13	-1	1	1	1	1
-1	-2	1	1	1	0	14	2	0	0	1	0
-1	-1	1	1	1	1	15	1	0	0	0	1

P = 4, necesita 4 bits

P = (0 1 0 0), con pesos (-8 4 2 1)

Pi = Fi (A1, A0, B1, B0)

P3 = ∑ (6,7,9,13)

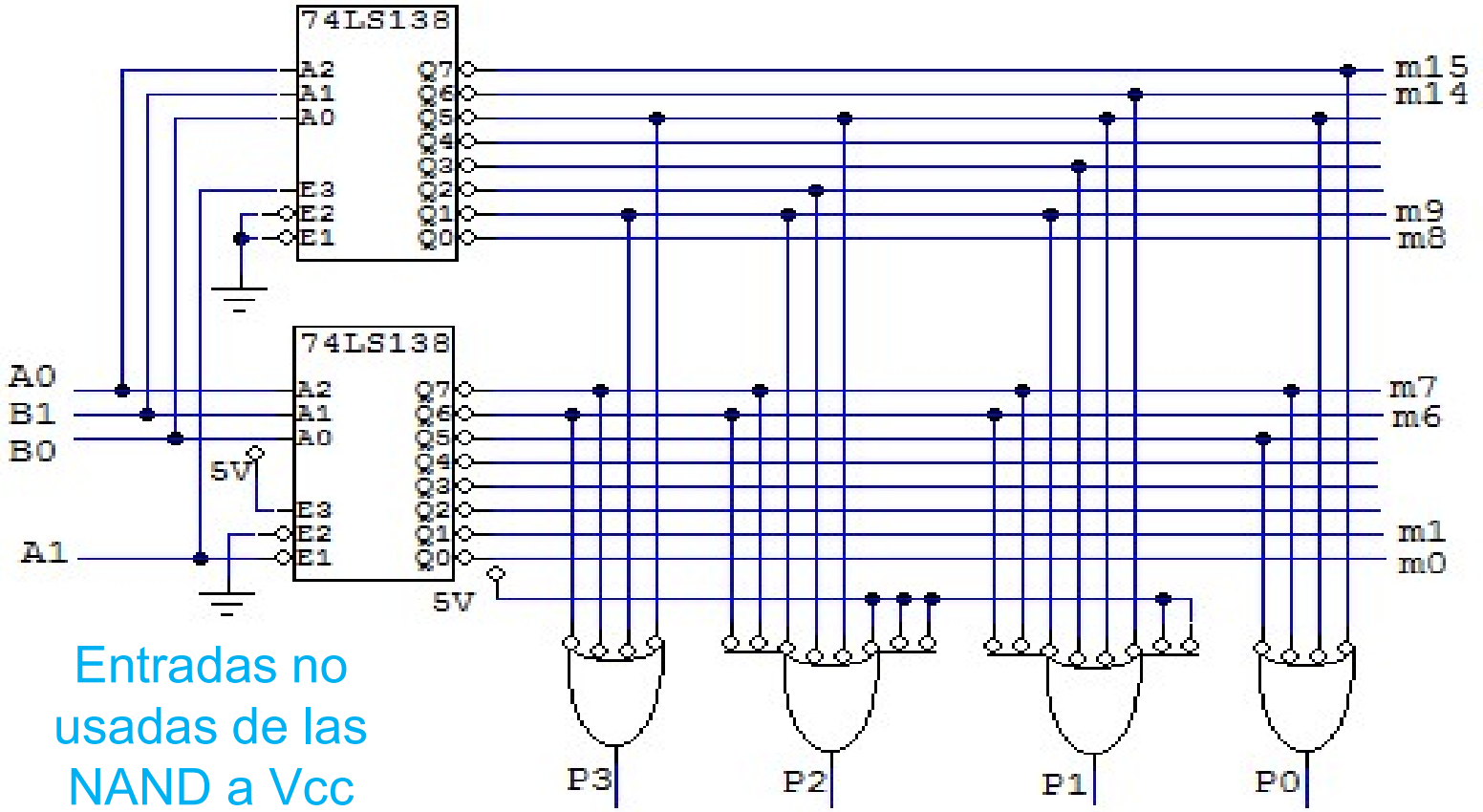
P2 = ∑ (6,7,9,10,13)

P1 = ∑ (6,7,9,11,13,14)

P0 = ∑ (5,7,13,15)

P3 = ∑ (6,7,9,13)
P2 = ∑ (6,7,9,10,13)
P1 = ∑ (6,7,9,11,13,14)
P0 = ∑ (5,7,13,15)

DEC 4 a 16 + NAND



Entradas no usadas de las NAND a Vcc

4.2. Diseñar un circuito que realice simultáneamente:

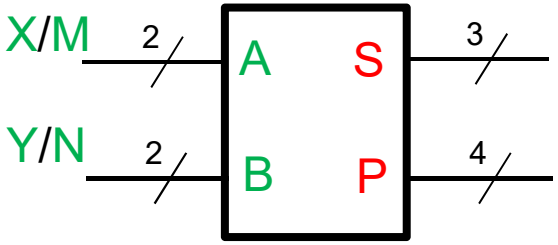
a) La suma de dos números positivos X (X1.H, X0.H) e Y (Y1.H Y0.H) de dos bits.

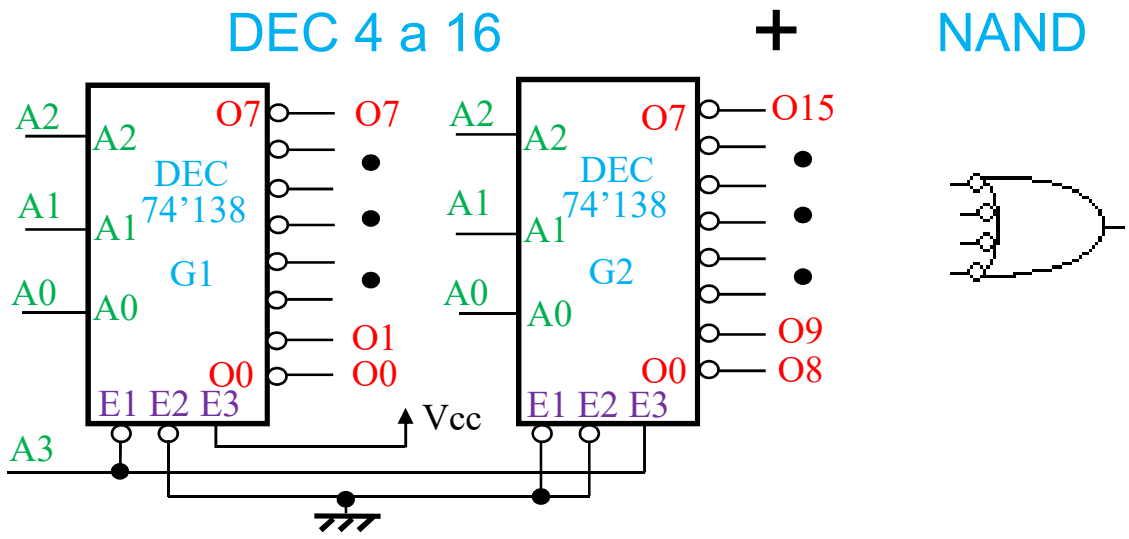
b) El producto de dos números positivos M (M1.H, M0.L) e N (N1.L N0.H) de dos bits.

El circuito debe tener sólo cuatro bits de entrada y dos grupos diferenciados de salidas para cada una de las operaciones, todas las salidas deben ser de polaridad positiva. El diseño debe realizarse utilizando el menor número de decodificadores 74'138, y el menor número de puertas NAND (suponer que es posible cualquier número de entradas en las puertas).

Smax = 3 + 3 = 6 => 3 bits (S2S1S0)

Pmax = 3 * 3 = 9 => 4 bits (P3P2P1P0)





4.2. Diseñar un circuito que realice:

a) La suma de dos números positivos X (X1.H, X0.H) e Y (Y1.H Y0.H) de dos bits.

		2	1	2	1		4	2	1	8	4	2	1	
X	**Y**	**X1**	**X0**	**Y1**	**Y0**	**S**	**S2**	**S1**	**S0**	**A1**	**A0**	**B1**	**B0**	**DECS**
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	1	1	0	0	1	0	0	0	1	1
0	2	0	0	1	0	2	0	1	0	0	0	1	0	2
0	3	0	0	1	1	3	0	1	1	0	0	1	1	3
1	0	0	1	0	0	1	0	0	1	0	1	0	0	4
1	1	0	1	0	1	2	0	1	0	0	1	0	1	5
1	2	0	1	1	0	3	0	1	1	0	1	1	0	6
1	3	0	1	1	1	4	1	0	0	0	1	1	1	7
2	0	1	0	0	0	2	0	1	0	1	0	0	0	8
2	1	1	0	0	1	3	0	1	1	1	0	0	1	9
2	2	1	0	1	0	4	1	0	0	1	0	1	0	10
2	3	1	0	1	1	5	1	0	1	1	0	1	1	11
3	0	1	1	0	0	3	0	1	1	1	1	0	0	12
3	1	1	1	0	1	4	1	0	0	1	1	0	1	13
3	2	1	1	1	0	5	1	0	1	1	1	1	0	14
3	3	1	1	1	1	6	1	1	0	1	1	1	1	15

S2 = FS2(A1, A0, B1, B0) = ∑(7, 10, 11, 13, 14, 15)

S1 = FS1(A1, A0, B1, B0) = ∑(2, 3, 5, 6, 8, 9, 12, 15)

S0 = FS0(A1, A0, B1, B0) = ∑(1, 3, 4, 6, 9, 11, 12, 14)

4.2. Diseñar un circuito que realice simultáneamente:

b) El producto de dos números positivos M (M1.H, M0.L) e N (N1.L N0.H) de dos bits.

Compl. la columna ← .L .L

		2	1	2	1		8	4	2	1	8	4	2	1	
M	**N**	**M1**	**M0**	**N1**	**N0**	**P**	**P3**	**P2**	**P1**	**P0**	**A1**	**A0**	**B1**	**B0**	**DECP**
0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	6
0	1	0	0	0	1	0	0	0	0	0	0	1	1	1	7
0	2	0	0	1	0	0	0	0	0	0	0	1	0	0	4
0	3	0	0	1	1	0	0	0	0	0	0	1	0	1	5
1	0	0	1	0	0	0	0	0	0	0	0	0	1	0	2
1	1	0	1	0	1	1	0	0	0	1	0	0	1	1	3
1	2	0	1	1	0	2	0	0	1	0	0	0	0	0	0
1	3	0	1	1	1	3	0	0	1	1	0	0	0	1	1
2	0	1	0	0	0	0	0	0	0	0	1	1	1	0	14
2	1	1	0	0	1	2	0	0	1	0	1	1	1	1	15
2	2	1	0	1	0	4	0	1	0	0	1	1	0	0	12
2	3	1	0	1	1	6	0	1	1	0	1	1	0	1	13
3	0	1	1	0	0	0	0	0	0	0	1	0	1	0	10
3	1	1	1	0	1	3	0	0	1	1	1	0	1	1	11
3	2	1	1	1	0	6	0	1	1	0	1	0	0	0	8
3	3	1	1	1	1	9	1	0	0	1	1	0	0	1	9

P3 = FP3(A1, A0, B1, B0) = ∑(9)

P2 = FP2(A1, A0, B1, B0) = ∑(8, 12, 13)

P1 = FP1(A1, A0, B1, B0) = ∑(0, 1, 8, 11, 13, 15)

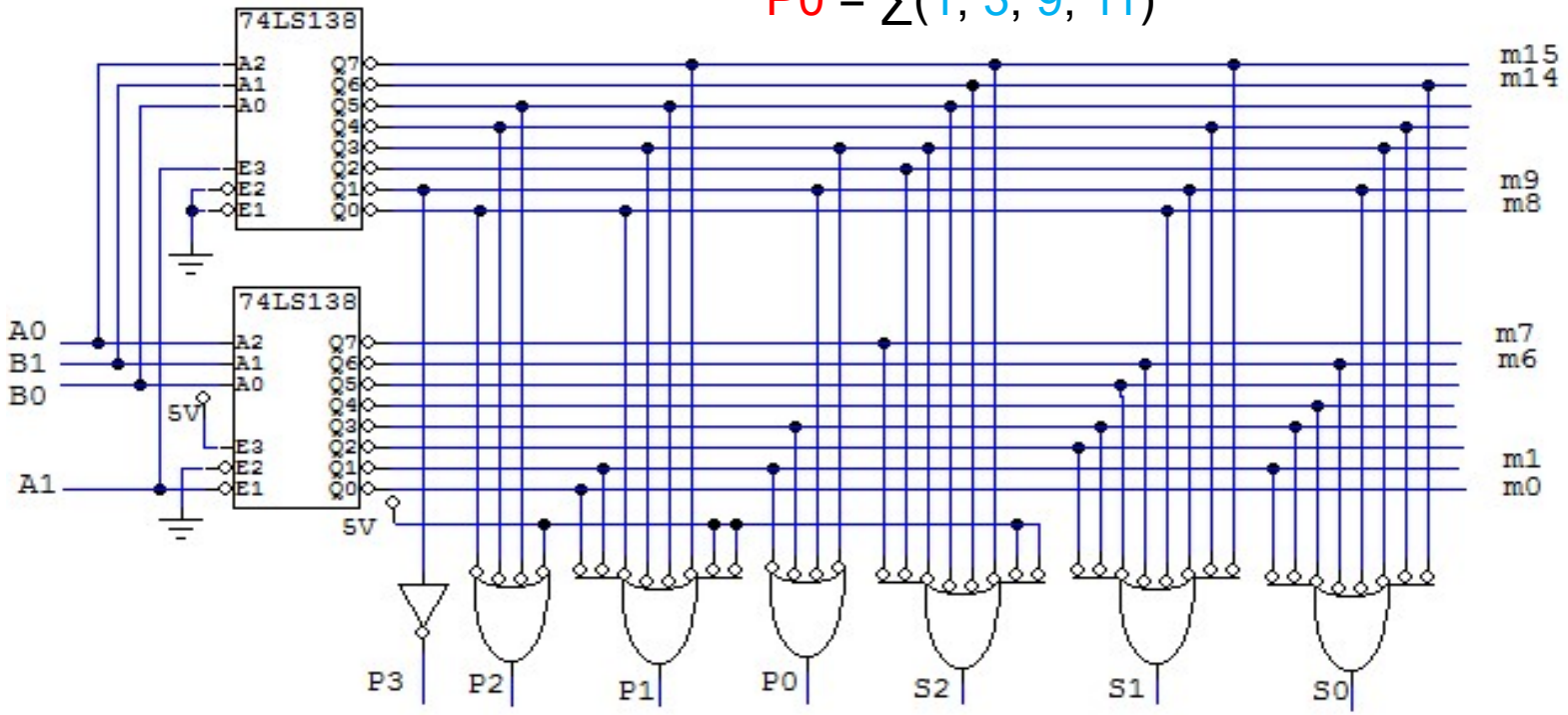
P0 = FP0(A1, A0, B1, B0) = ∑(1, 3, 9, 11)

4.2. Diseñar un circuito que realice simultáneamente la suma y el producto de dos números positivos A y B de dos bits.

El circuito debe tener sólo cuatro bits de entrada y dos grupos diferenciados de salidas para cada una de las operaciones, todas las salidas deben ser de polaridad positiva. El diseño debe realizarse utilizando el menor número de decodificadores 74'138, y el menor número de puertas NAND (suponer que es posible cualquier número de entradas en las puertas).

$S2 = \sum(7, 10, 11, 13, 14, 15)$
$S1 = \sum(2, 3, 5, 6, 8, 9, 12, 15)$
$S0 = \sum(1, 3, 4, 6, 9, 11, 12, 14)$

$P3 = \sum(9)$
$P2 = \sum(8, 12, 13)$
$P1 = \sum(0, 1, 8, 11, 13, 15)$
$P0 = \sum(1, 3, 9, 11)$



5.1. Implementar las siguientes funciones utilizando el menor número de decodificadores 3 a 8, y 2 a 4.

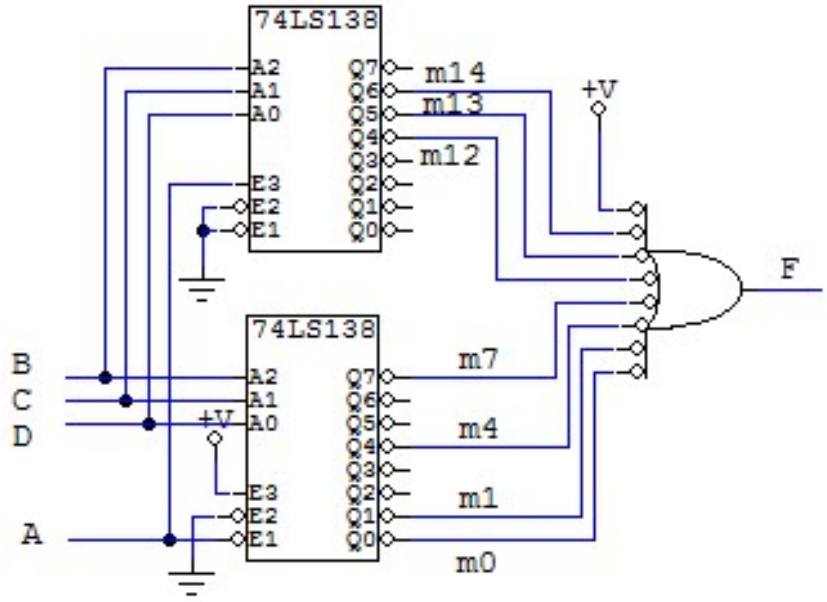
a) F(A,B,C,D) = ∑(0,1,4,7,12,13,14) con A.H, B.H, C.H y D.H.

No hay entradas redundantes

AB \ CD	00 (D)	01	11 (B)	10
00	1	1	0	0
01	1	0	1	0
11	1	1	0	1 (A)
10	0	0	0	0

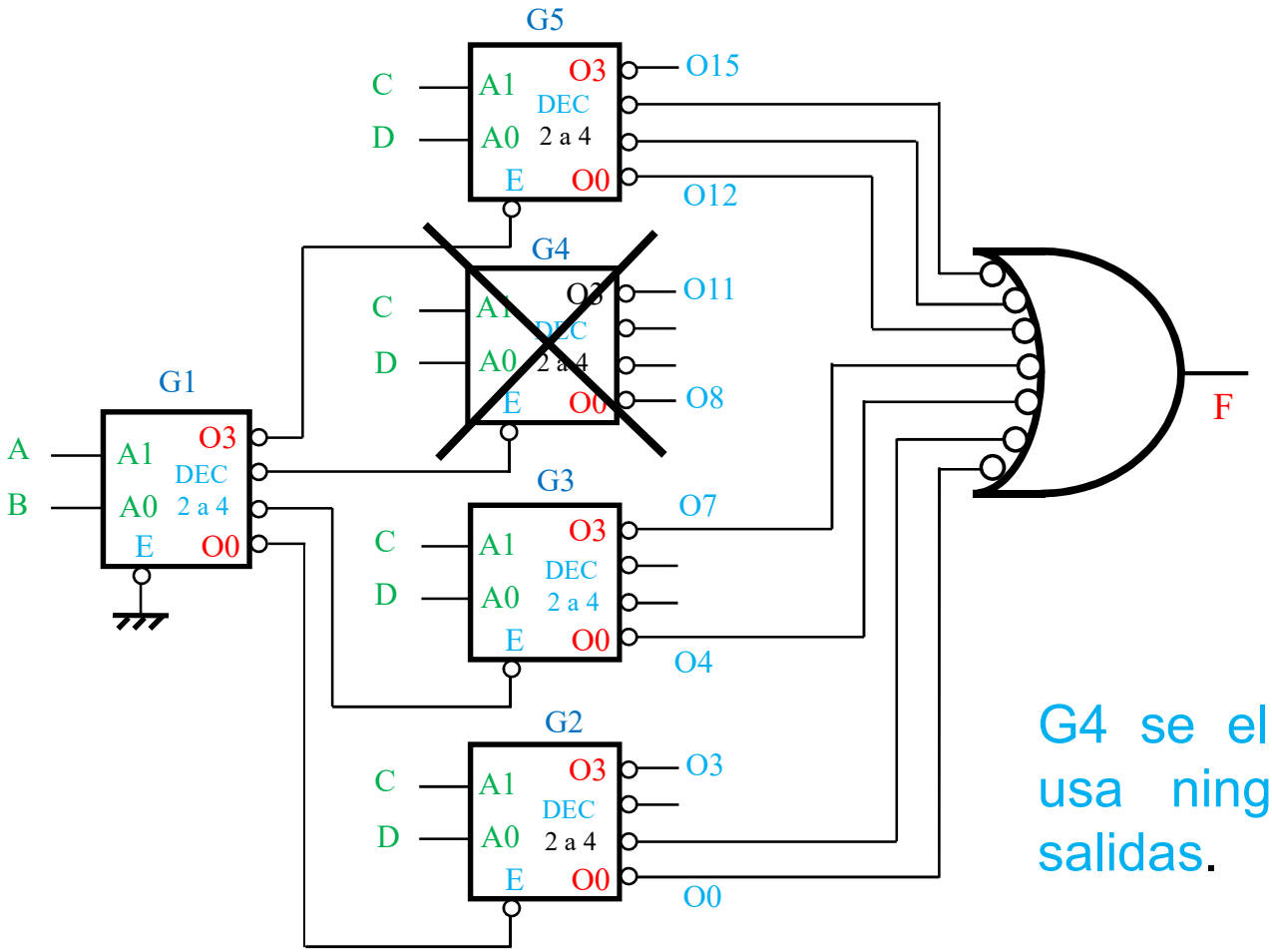
C

Utilizo dos decodificadores 3 a 8 (como 1 DEC 4 a 16) y puertas NAND.



5.1. Implementar las siguientes funciones utilizando el menor número de decodificadores 3 a 8, y 2 a 4.

a) F(A,B,C,D) = ∑(0,1,4,7,12,13,14) con A.H, B.H, C.H y D.H.

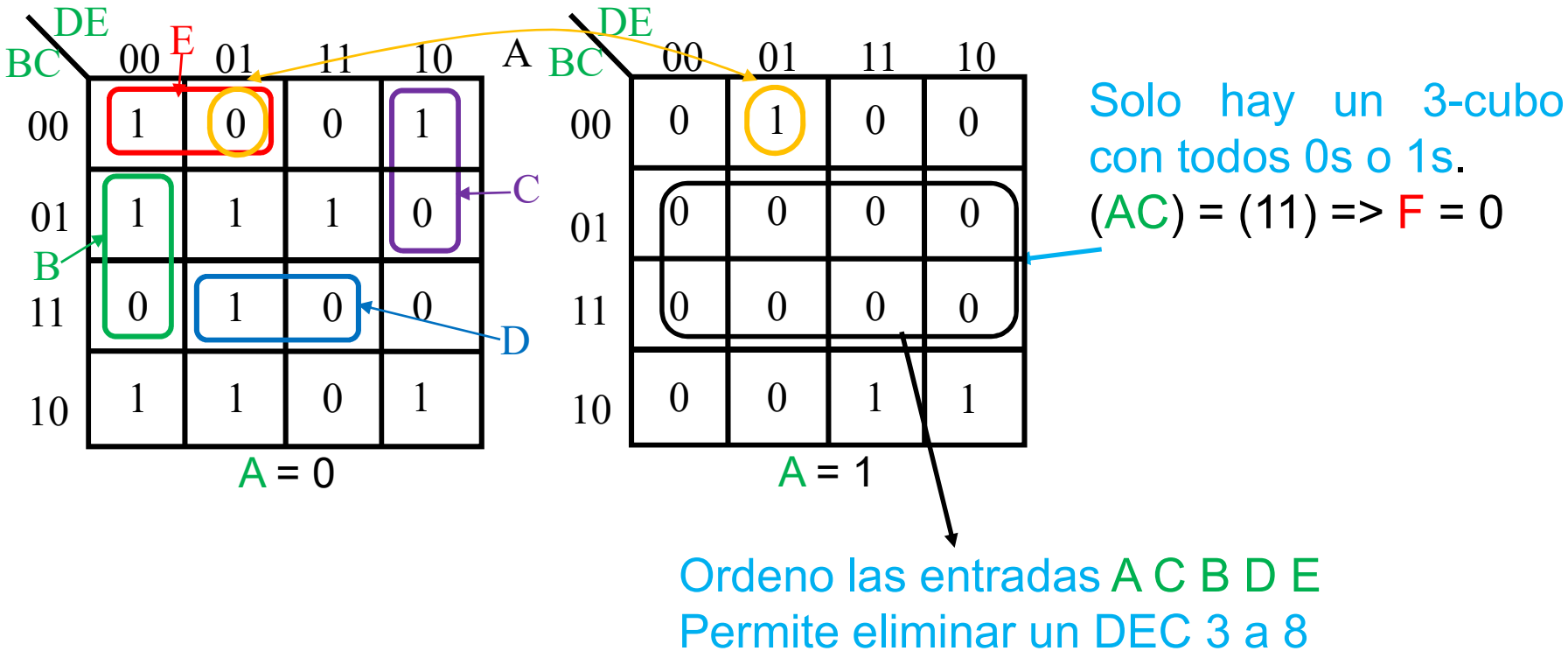


G4 se elimina. No se usa ninguna de sus salidas.

5.1. Implementar las siguientes funciones utilizando el menor número de decodificadores 3 a 8, y 2 a 4.

b) F(A,B,C,D,E) = ∑(0,2,4,5,7,8,9,10,13,17,26,27) con A.H, B.L, C.L, D.L y E.H.

No hay entradas redundantes



5.1. Implementar las siguientes funciones utilizando el menor número de decodificadores 3 a 8, y 2 a 4.

b) F(A,B,C,D,E) = ∑(0,2,4,5,7,8,9,10,13,17,26,27) con A.H, B.L, C.L, D.L y E.H.

Ordeno las entradas A C B D E. Permite eliminar un DEC 3 a 8.

Complemento B, C y D, ya que son .L

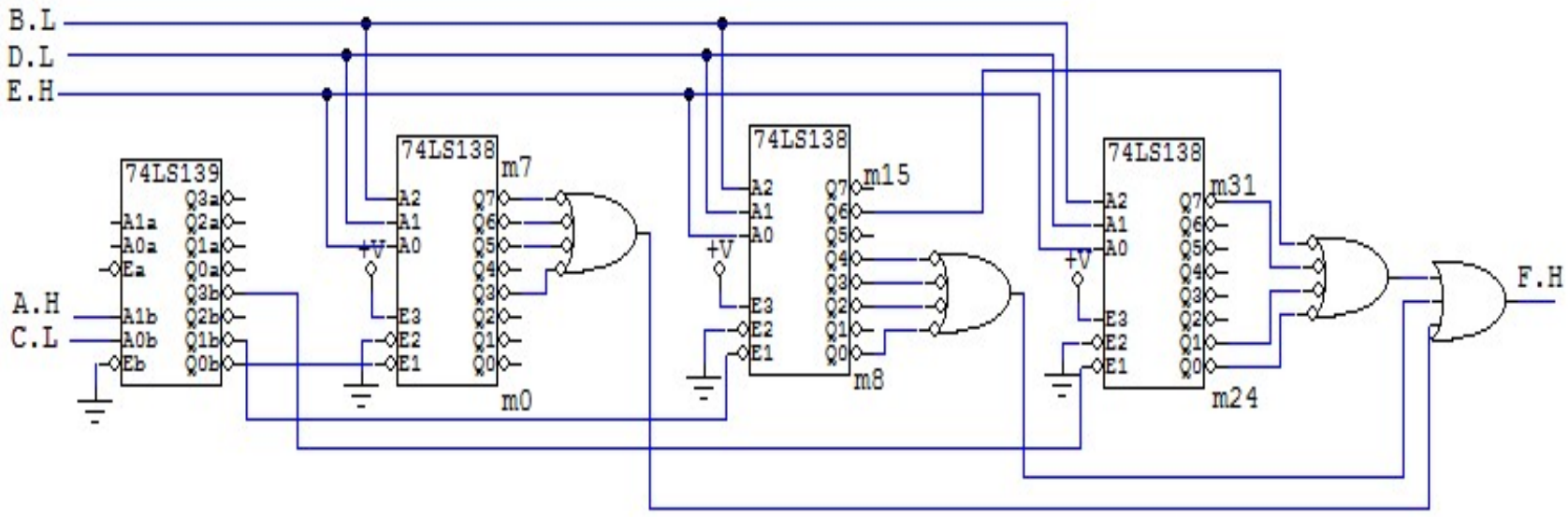
	16	8	4	2	1	16	8	4	2	1	
DEC	**A**	**B**	**C**	**D**	**E**	**A**	**C**	**B**	**D**	**E**	**DEC**
0	0	0	0	0	0	0	1	1	1	0	14
2	0	0	0	1	0	0	1	1	0	0	12
4	0	0	1	0	0	0	0	1	1	0	6
5	0	0	1	0	1	0	0	1	1	1	7
7	0	0	1	1	1	0	0	1	0	1	5
8	0	1	0	0	0	0	1	0	1	0	10
9	0	1	0	0	1	0	1	0	1	1	11
10	0	1	0	1	0	0	1	0	0	0	8
13	0	1	1	0	1	0	0	0	1	1	3
17	1	0	0	0	1	1	1	1	1	1	31
26	1	1	0	1	0	1	1	0	0	0	24
27	1	1	0	1	1	1	1	0	0	1	25

F(A, B, C, D, E) = ∑(3, 5, 6, 7, 8, 10, 11, 12, 14, 24, 25, 31)

5.1. Implementar las siguientes funciones utilizando el menor número de decodificadores 3 a 8, y 2 a 4.

b) F(A,B,C,D,E) = ∑(0,2,4,5,7,8,9,10,13,17,26,27) con A.H, B.L, C.L, D.L y E.H.

F(A, B, C, D, E) = ∑(3, 5, 6, 7, 8, 10, 11, 12, 14, 24, 25, 31)



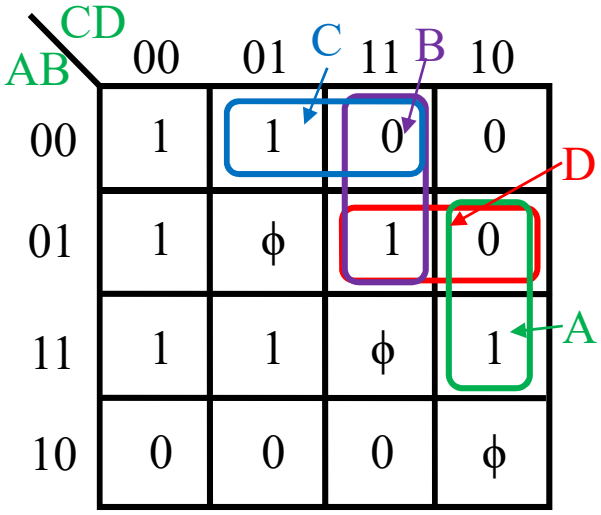
1 DEC 2 a 4

3 DECs 3 a 8

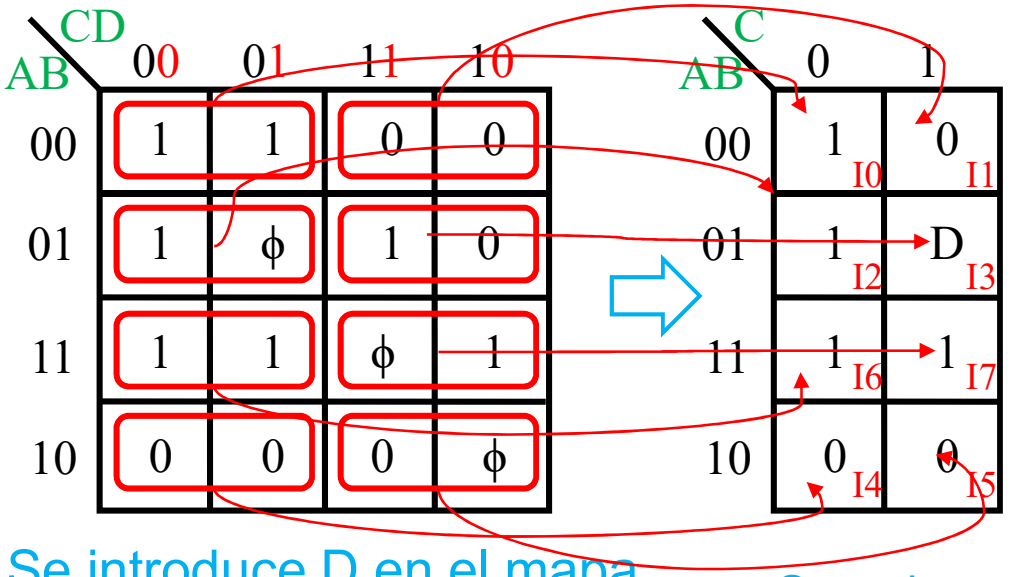
El DEC 3 a 8 para m16-m23 no es necesario, no se usa ninguna de sus salidas

5.2. Implementar las siguientes funciones lógicas utilizando un único multiplexor, lo más pequeño posible.

a) F(A,B,C,D) = ∑(0,1,4,7,12,13,14) + ∑ø(5,10,15) con A.H, B.H, C.H, D.H y F.H.



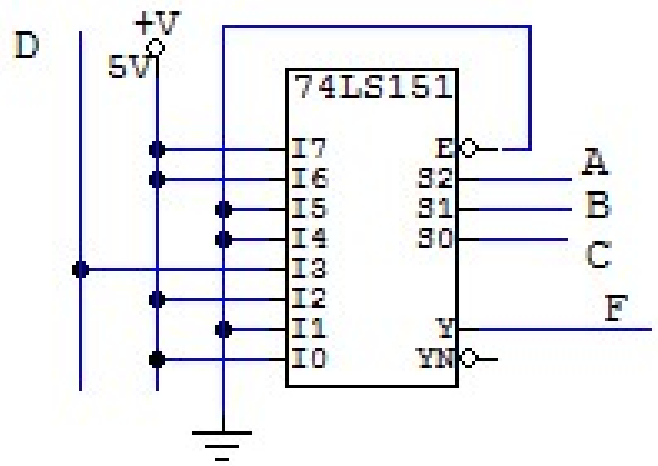
No hay entradas redundantes



Se introduce D en el mapa

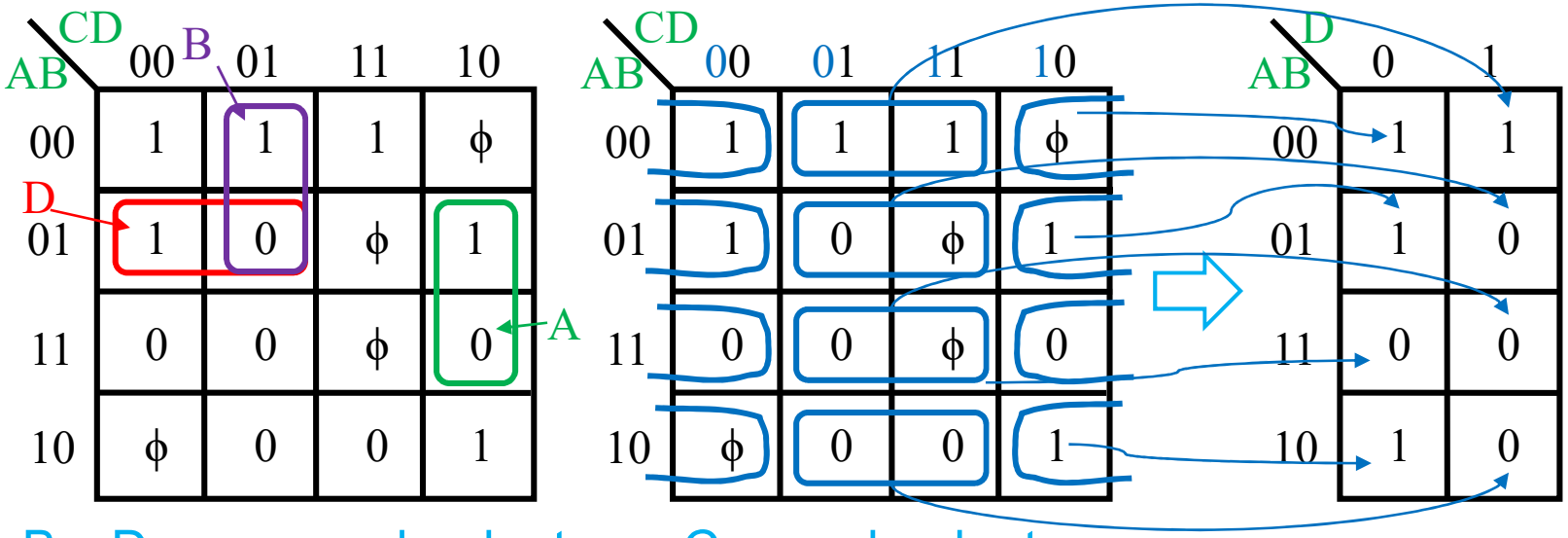
Se ordena según A B C

Todas las señales .H
No hay complementaciones por cambio de polaridad



5.2. Implementar las siguientes funciones lógicas utilizando un único multiplexor, lo más pequeño posible.

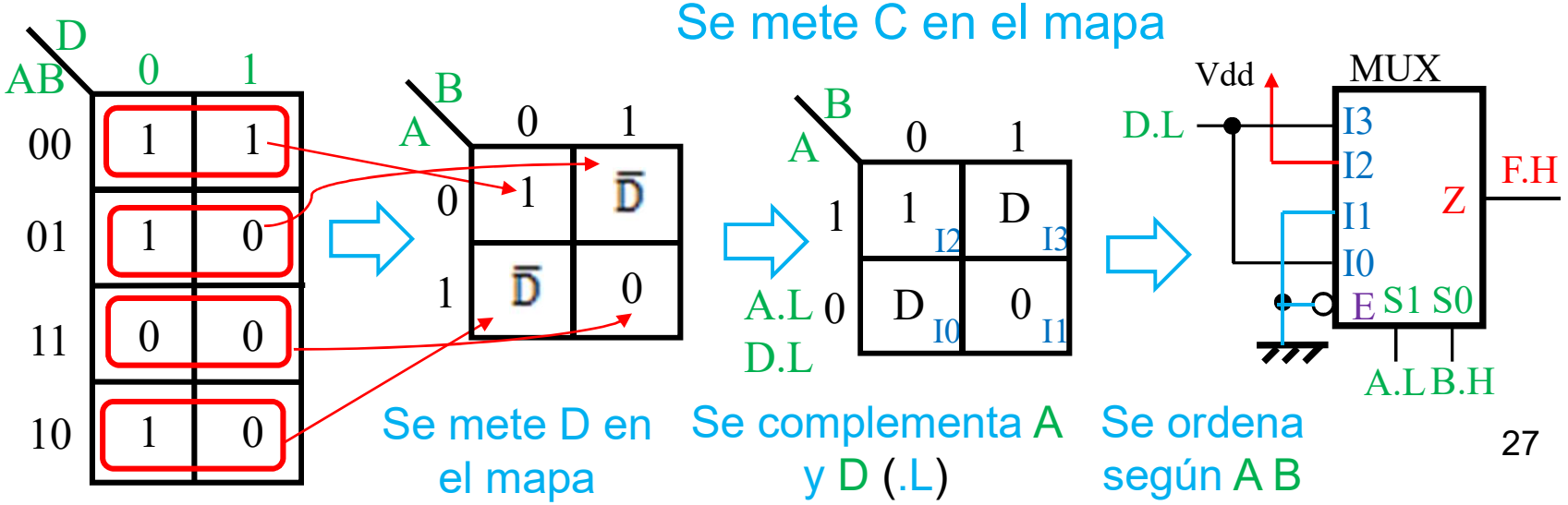
b) F(A,B,C,D) = ∑(0, 1, 3, 4, 6, 10) + ∑ø(2, 7, 8, 15) para A.L, B.H, C.H, D.L, F.H.



A, B y D no son redundantes

C es redundante.

Se mete C en el mapa



Se mete D en el mapa

Se complementa A y D (.L)

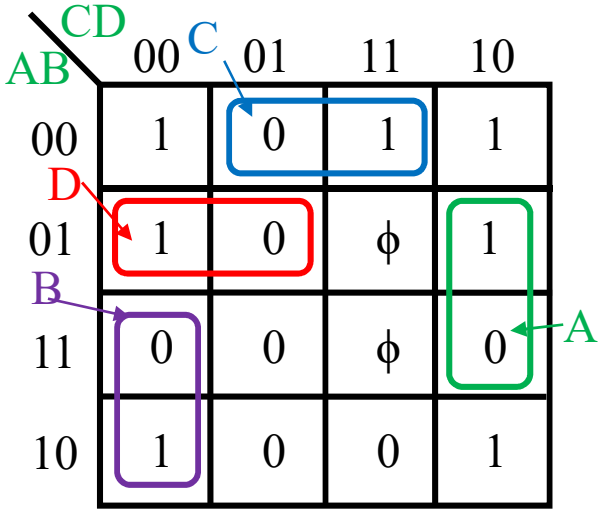
Se ordena según A B

5.3. Implementar utilizando un multiplexor de cuatro entradas y el menor número de multiplexores de dos entradas la función lógica.

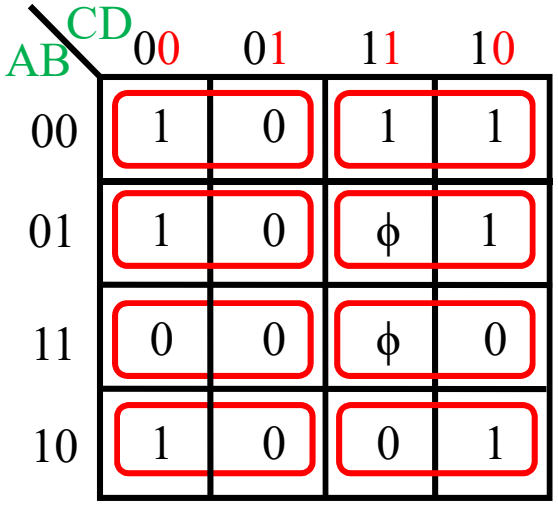
F.H = F(A, B, C, D) = ∏(1, 5, 9, 11, 12, 13, 14) • ∏ø(7, 15),

para entradas A.H, B.L, C.H y D.L.

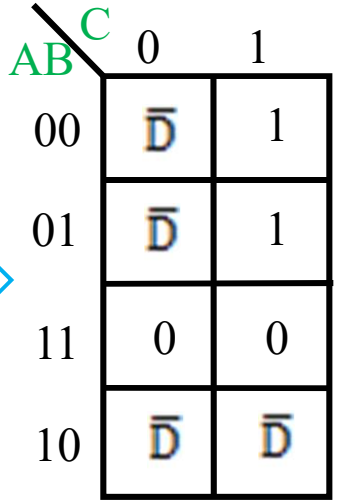
Prueba estándar. Se mete D en el mapa



No hay entradas redundantes



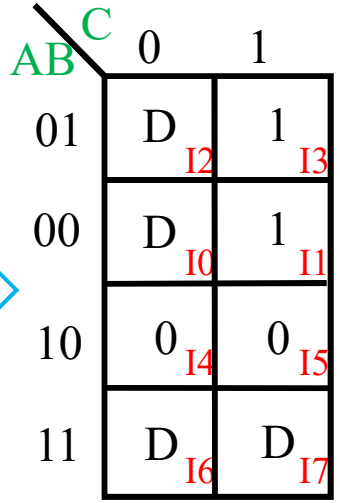
Se mete D en el mapa



Se complementan B (.L) y D (.L)

Se ordena según A B C

B.L
D.L



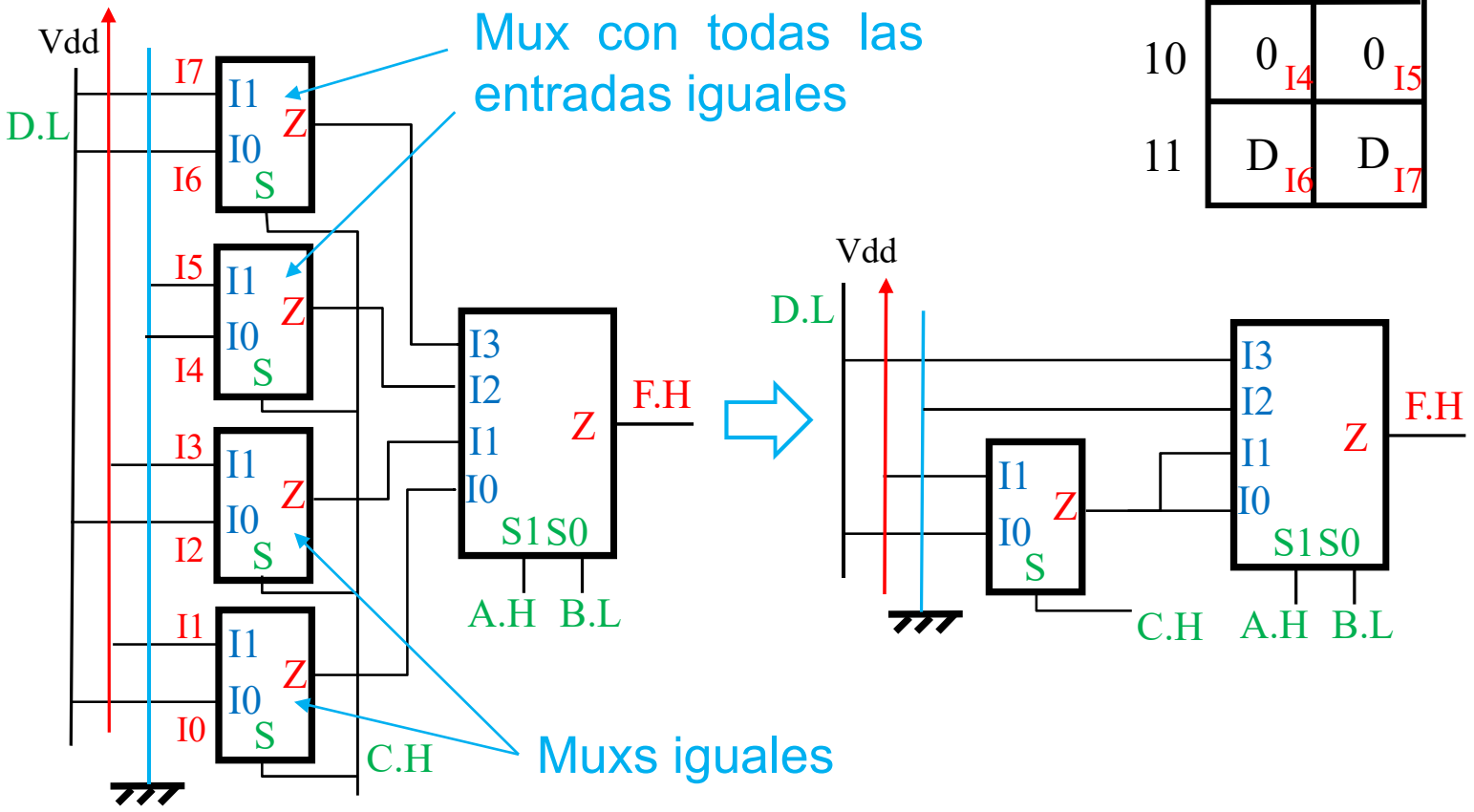
5.3. Implementar utilizando un multiplexor de cuatro entradas y el menor número de multiplexores de dos entradas la función lógica.

F.H = F(A, B, C, D) = ∏(1, 5, 9, 11, 12, 13, 14) • ∏ø(7, 15),

para entradas A.H, B.L, C.H y D.L.

AB \ C	0	1
01	D I2	1 I3
00	D I0	1 I1
10	0 I4	0 I5
11	D I6	D I7

8-Inp Mux hecho con un 4-Inp Mux y cuatro 2-Inp Muxs



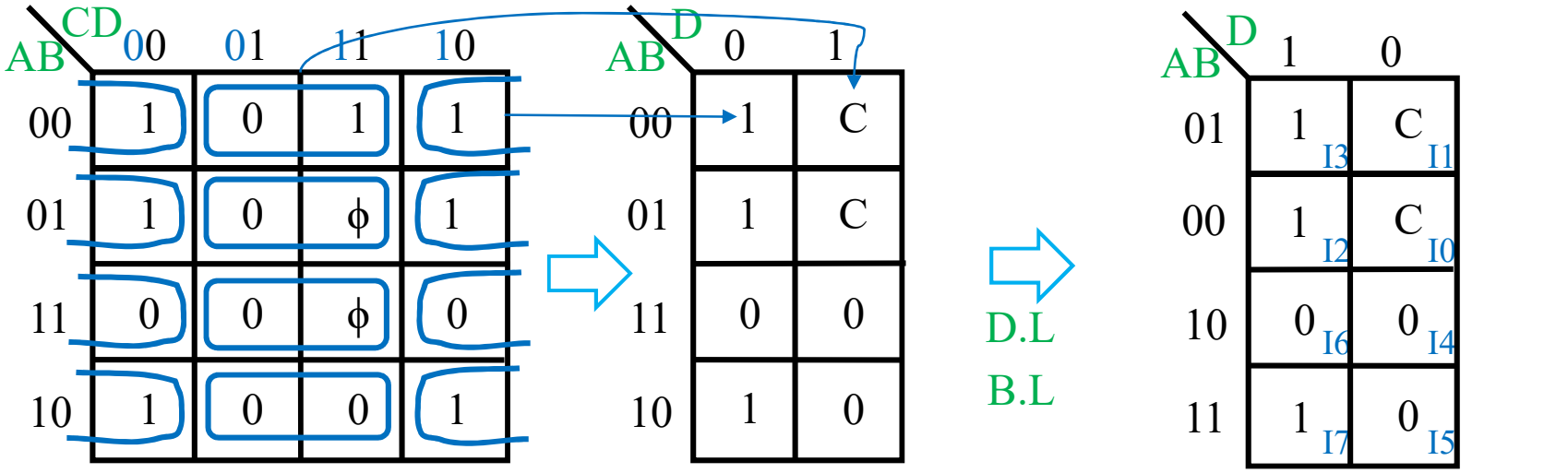
5.3. Implementar utilizando un multiplexor de cuatro entradas y el menor número de multiplexores de dos entradas la función lógica.

F.H = F(A, B, C, D) = ∏(1, 5, 9, 11, 12, 13, 14) • ∏ø(7, 15),

para entradas A.H, B.L, C.H y D.L.

Mejor prueba: se mete C en el mapa; ϕ 7 a 1, ϕ 15 a 0

Orden de las entradas de selección: A, D y B



Se mete C en el mapa

Se complementan D y B (.L)

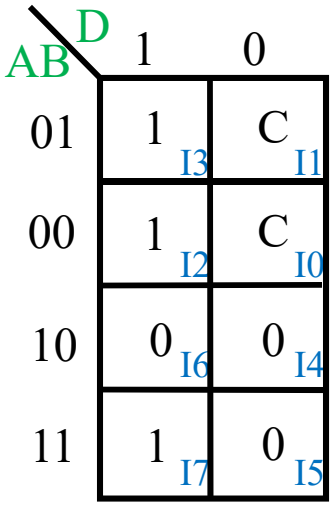
Se ordena según A D B

MUX4 MUX2

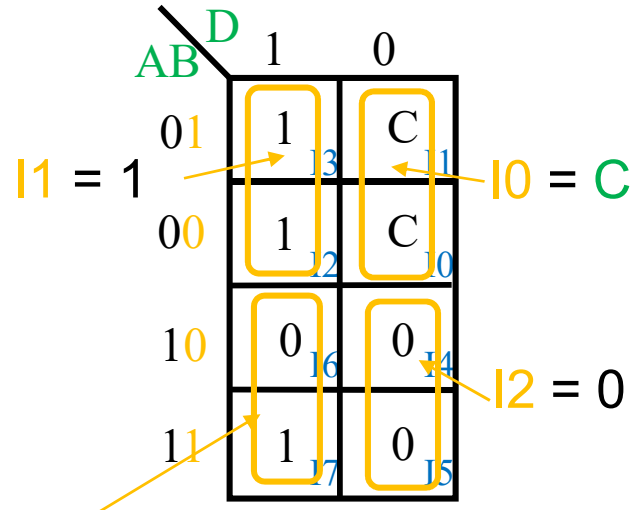
5.3. Implementar utilizando un multiplexor de cuatro entradas y el menor número de multiplexores de dos entradas la función lógica.

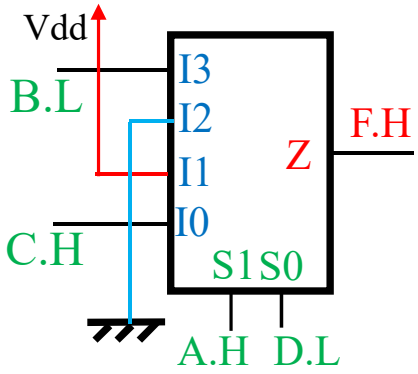
F.H = F(A, B, C, D) = ∏(1, 5, 9, 11, 12, 13, 14) • ∏ø(7, 15),

para entradas A.H, B.L, C.H y D.L.



Se ordena según B (MUX2) para valores fijos de A y D (MUX4)





I3 = B

Si B es 0 Z es 0

Si B es 1 Z es 1

Z = B

6.1. Obtener las expresiones lógicas minimizadas que permiten encontrar cuál de 7 líneas de entrada A1, A4, A6, A8, A9, A13, A14 está puesta a valor lógico 1, dando como resultado su correspondiente codificación binaria: por ejemplo, A8 daría como resultado 8 en la salida (codificado en binario).
a) Solo puede estar una línea a valor lógico 1.

A1	A4	A6	A8	A9	A13	A14	Z3	Z2	Z1	Z0
1	0	0	0	0	0	0	0	0	0	1
0	1	0	0	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	1	0
0	0	0	1	0	0	0	1	0	0	0
0	0	0	0	1	0	0	1	0	0	1
0	0	0	0	0	1	0	1	1	0	1
0	0	0	0	0	0	1	1	1	1	0

Z3 = A8 + A9 + A13 + A14
Z2 = A4 + A6 + A13 + A14
Z1 = A6 + A14
Z0 = A1 + A9 + A13

El resto de combinaciones en la tabla serían ϕs en las salidas

6.1. Obtener las expresiones lógicas minimizadas que permiten encontrar cuál de 7 líneas de entrada A1, A4, A6, A8, A9, A13, A14 está puesta a valor lógico 1, dando como resultado su correspondiente codificación binaria: por ejemplo, A8 daría como resultado 8 en la salida (codificado en binario).

b) Varias líneas de entrada están simultáneamente a 1, pero la salida tomará el valor binario de la línea de índice más bajo.

Función	A1	A4	A6	A8	A9	A13	A14	Z3	Z2	Z1	Z0
A1	1	X	X	X	X	X	X	0	0	0	1
$\overline{A1}$ A4	0	1	X	X	X	X	X	0	1	0	0
$\overline{A1}$ $\overline{A4}$ A6	0	0	1	X	X	X	X	0	1	1	0
$\overline{A1}$ $\overline{A4}$ $\overline{A6}$ A8	0	0	0	1	X	X	X	1	0	0	0
$\overline{A1}$ $\overline{A4}$ $\overline{A6}$ $\overline{A8}$ A9	0	0	0	0	1	X	X	1	0	0	1
$\overline{A1}$ $\overline{A4}$ $\overline{A6}$ $\overline{A8}$ $\overline{A9}$ A13	0	0	0	0	0	1	X	1	1	0	1
$\overline{A1}$ $\overline{A4}$ $\overline{A6}$ $\overline{A8}$ $\overline{A9}$ $\overline{A13}$ A14	0	0	0	0	0	0	1	1	1	1	0
$\overline{A1}$ $\overline{A4}$ $\overline{A6}$ $\overline{A8}$ $\overline{A9}$ $\overline{A13}$ $\overline{A14}$	0	0	0	0	0	0	0	0	0	0	0

$Z3 = \overline{A1}\ \overline{A4}\ \overline{A6}\ A8 + \overline{A1}\ \overline{A4}\ \overline{A6}\ \overline{A8}\ A9 + \overline{A1}\ \overline{A4}\ \overline{A6}\ \overline{A8}\ \overline{A9}\ A13 + \overline{A1}\ \overline{A4}\ \overline{A6}\ \overline{A8}\ \overline{A9}\ \overline{A13}\ A14$

$Z2 = \overline{A1}\ A4 + \overline{A1}\ \overline{A4}\ A6 + \overline{A1}\ \overline{A4}\ \overline{A6}\ \overline{A8}\ \overline{A9}\ A13 + \overline{A1}\ \overline{A4}\ \overline{A6}\ \overline{A8}\ \overline{A9}\ \overline{A13}\ A14$

$Z1 = \overline{A1}\ \overline{A4}\ A6 + \overline{A1}\ \overline{A4}\ \overline{A6}\ \overline{A8}\ \overline{A9}\ \overline{A13}\ A14$

$Z0 = A1 + \overline{A1}\ \overline{A4}\ \overline{A6}\ \overline{A8}\ A9 + \overline{A1}\ \overline{A4}\ \overline{A6}\ \overline{A8}\ \overline{A9}\ A13$

$Z3 = \overline{A1}\ \overline{A4}\ \overline{A6}\ A8 + \overline{A1}\ \overline{A4}\ \overline{A6}\ \overline{A8}\ A9 + \overline{A1}\ \overline{A4}\ \overline{A6}\ \overline{A8}\ \overline{A9}\ A13 + \overline{A1}\ \overline{A4}\ \overline{A6}\ \overline{A8}\ \overline{A9}\ \overline{A13}\ A14 =$

$= \overline{A1}\ \overline{A4}\ \overline{A6}\ (A8 + \overline{A8}\ A9 + \overline{A8}\ \overline{A9}\ A13 + \overline{A8}\ \overline{A9}\ \overline{A13}\ A14) =$

$= \overline{A1}\ \overline{A4}\ \overline{A6}\ (A8 + A9 + \overline{A9}\ A13 + \overline{A9}\ \overline{A13}\ A14) =$

$= \overline{A1}\ \overline{A4}\ \overline{A6}\ (A8 + A9 + A13 + \overline{A13}\ A14) =$

$= \overline{A1}\ \overline{A4}\ \overline{A6}\ (A8 + A9 + A13 + A14\ )$

$Z2 = \overline{A1}\ A4 + \overline{A1}\ \overline{A4}\ A6 + \overline{A1}\ \overline{A4}\ \overline{A6}\ \overline{A8}\ \overline{A9}\ A13 + \overline{A1}\ \overline{A4}\ \overline{A6}\ \overline{A8}\ \overline{A9}\ \overline{A13}\ A14 =$

$= \overline{A1}\ (A4 + \overline{A4}\ A6 + \overline{A4}\ \overline{A6}\ \overline{A8}\ \overline{A9}\ A13 + \overline{A4}\ \overline{A6}\ \overline{A8}\ \overline{A9}\ \overline{A13}\ A14) =$

$= \overline{A1}\ (A4 + A6 + \overline{A6}\ \overline{A8}\ \overline{A9}\ A13 + \overline{A6}\ \overline{A8}\ \overline{A9}\ \overline{A13}\ A14) =$

$= \overline{A1}\ (A4 + A6 + \overline{A8}\ \overline{A9}\ A13 + \overline{A8}\ \overline{A9}\ \overline{A13}\ A14) =$

$= \overline{A1}\ [A4 + A6 + \overline{A8}\ \overline{A9}\ (A13 + \overline{A13}\ A14)] =$

$= \overline{A1}\ [A4 + A6 + \overline{A8}\ \overline{A9}\ (A13 + A14)]$

$Z1 = \overline{A1}\ \overline{A4}\ A6 + \overline{A1}\ \overline{A4}\ \overline{A6}\ \overline{A8}\ \overline{A9}\ \overline{A13}\ A14 =$

$= \overline{A1}\ \overline{A4}\ (A6 + \overline{A6}\ \overline{A8}\ \overline{A9}\ \overline{A13}\ A14) =$

$= \overline{A1}\ \overline{A4}\ (A6 + \overline{A8}\ \overline{A9}\ \overline{A13}\ A14)$

$Z0 = A1 + \overline{A4}\ \overline{A6}\ \overline{A8}\ A9 + \overline{A4}\ \overline{A6}\ \overline{A8}\ \overline{A9}\ A13 =$

$= A1 + \overline{A4}\ \overline{A6}\ \overline{A8}\ (A9 + \overline{A9}\ A13) =$

$= A1 + \overline{A4}\ \overline{A6}\ \overline{A8}\ (A9 + A13)$

Utilizo la propiedad distributiva y el teorema de simplificación

6.2. Encontrar las ecuaciones lógicas que permiten definir un circuito codificador con prioridad baja de 8 bits de entrada (I7-I0) y salidas en código Gray (de más a menos significativas: A B C).

Función	I0	I1	I2	I3	I4	I5	I6	I7	A	B	C
$I0$	1	X	X	X	X	X	X	X	0	0	0
$\overline{I0}\ I1$	0	1	X	X	X	X	X	X	0	0	1
$\overline{I0}\ \overline{I1}\ I2$	0	0	1	X	X	X	X	X	0	1	1
$\overline{I0}\ \overline{I1}\ \overline{I2}\ I3$	0	0	0	1	X	X	X	X	0	1	0
$\overline{I0}\ \overline{I1}\ \overline{I2}\ \overline{I3}\ I4$	0	0	0	0	1	X	X	X	1	1	0
$\overline{I0}\ \overline{I1}\ \overline{I2}\ \overline{I3}\ \overline{I4}\ I5$	0	0	0	0	0	1	X	X	1	1	1
$\overline{I0}\ \overline{I1}\ \overline{I2}\ \overline{I3}\ \overline{I4}\ \overline{I5}\ I6$	0	0	0	0	0	0	1	X	1	0	1
$\overline{I0}\ \overline{I1}\ \overline{I2}\ \overline{I3}\ \overline{I4}\ \overline{I5}\ \overline{I6}\ I7$	0	0	0	0	0	0	0	1	1	0	0
$\overline{I0}\ \overline{I1}\ \overline{I2}\ \overline{I3}\ \overline{I4}\ \overline{I5}\ \overline{I6}\ \overline{I7}$	0	0	0	0	0	0	0	0	0	0	0

Código Gray de 0 a 7

$$A = \overline{I0}\ \overline{I1}\ \overline{I2}\ \overline{I3}\ I4 + \overline{I0}\ \overline{I1}\ \overline{I2}\ \overline{I3}\ \overline{I4}\ I5 + \overline{I0}\ \overline{I1}\ \overline{I2}\ \overline{I3}\ \overline{I4}\ \overline{I5}\ I6 + \overline{I0}\ \overline{I1}\ \overline{I2}\ \overline{I3}\ \overline{I4}\ \overline{I5}\ \overline{I6}\ I7$$

$$B = \overline{I0}\ \overline{I1}\ I2 + \overline{I0}\ \overline{I1}\ \overline{I2}\ I3 + \overline{I0}\ \overline{I1}\ \overline{I2}\ \overline{I3}\ I4 + \overline{I0}\ \overline{I1}\ \overline{I2}\ \overline{I3}\ \overline{I4}\ I5$$

$$C = \overline{I0}\ I1 + \overline{I0}\ \overline{I1}\ I2 + \overline{I0}\ \overline{I1}\ \overline{I2}\ \overline{I3}\ \overline{I4}\ I5 + \overline{I0}\ \overline{I1}\ \overline{I2}\ \overline{I3}\ \overline{I4}\ \overline{I5}\ I6$$

$$A = \bar{I0}\,\bar{I1}\,\bar{I2}\,\bar{I3}\,I4 + \bar{I0}\,\bar{I1}\,\bar{I2}\,\bar{I3}\,\bar{I4}\,I5 + \bar{I0}\,\bar{I1}\,\bar{I2}\,\bar{I3}\,\bar{I4}\,\bar{I5}\,I6 + \bar{I0}\,\bar{I1}\,\bar{I2}\,\bar{I3}\,\bar{I4}\,\bar{I5}\,\bar{I6}\,I7 =$$

$$= \bar{I0}\,\bar{I1}\,\bar{I2}\,\bar{I3}\,(I4 + \bar{I4}\,I5 + \bar{I4}\,\bar{I5}\,I6 + \bar{I4}\,\bar{I5}\,\bar{I6}\,I7) =$$

$$= \bar{I0}\,\bar{I1}\,\bar{I2}\,\bar{I3}\,(I4 + I5 + \bar{I5}\,I6 + \bar{I5}\,\bar{I6}\,I7)$$

$$= \bar{I0}\,\bar{I1}\,\bar{I2}\,\bar{I3}\,(I4 + I5 + I6 + \bar{I6}\,I7) =$$

$$= \bar{I0}\,\bar{I1}\,\bar{I2}\,\bar{I3}\,(I4 + I5 + I6 + \bar{I6}\,I7) = \bar{I0}\,\bar{I1}\,\bar{I2}\,\bar{I3}\,(I4 + I5 + I6 + I7)$$

$$B = \bar{I0}\,\bar{I1}\,I2 + \bar{I0}\,\bar{I1}\,\bar{I2}\,I3 + \bar{I0}\,\bar{I1}\,\bar{I2}\,\bar{I3}\,I4 + \bar{I0}\,\bar{I1}\,\bar{I2}\,\bar{I3}\,\bar{I4}\,I5 =$$

$$= \bar{I0}\,\bar{I1}\,(I2 + \bar{I2}\,I3 + \bar{I2}\,\bar{I3}\,I4 + \bar{I2}\,\bar{I3}\,\bar{I4}\,I5) =$$

$$= \bar{I0}\,\bar{I1}\,(I2 + I3 + \bar{I3}\,I4 + \bar{I3}\,\bar{I4}\,I5) =$$

$$= \bar{I0}\,\bar{I1}\,(I2 + I3 + I4 + \bar{I4}\,I5) =$$

$$= \bar{I0}\,\bar{I1}\,(I2 + I3 + I4 + I5)$$

Utilizo la propiedad distributiva y el teorema de simplificación

$$C = \bar{I0}\,I1 + \bar{I0}\,\bar{I1}\,I2 + \bar{I0}\,\bar{I1}\,\bar{I2}\,\bar{I3}\,\bar{I4}\,I5 + \bar{I0}\,\bar{I1}\,\bar{I2}\,\bar{I3}\,\bar{I4}\,\bar{I5}\,I6 =$$

$$= \bar{I0}\,(I1 + \bar{I1}\,I2 + \bar{I1}\,\bar{I2}\,\bar{I3}\,\bar{I4}\,I5 + \bar{I1}\,\bar{I2}\,\bar{I3}\,\bar{I4}\,\bar{I5}\,I6) =$$

$$= \bar{I0}\,(I1 + I2 + \bar{I2}\,\bar{I3}\,\bar{I4}\,I5 + \bar{I2}\,\bar{I3}\,\bar{I4}\,\bar{I5}\,I6) =$$

$$= \bar{I0}\,[I1 + I2 + \bar{I3}\,\bar{I4}\,I5 + \bar{I3}\,\bar{I4}\,\bar{I5}\,I6] =$$

$$= \bar{I0}\,[I1 + I2 + \bar{I3}\,\bar{I4}\,(I5 + \bar{I5}\,I6)] =$$

$$= \bar{I0}\,[I1 + I2 + \bar{I3}\,\bar{I4}\,(I5 + I6)]$$

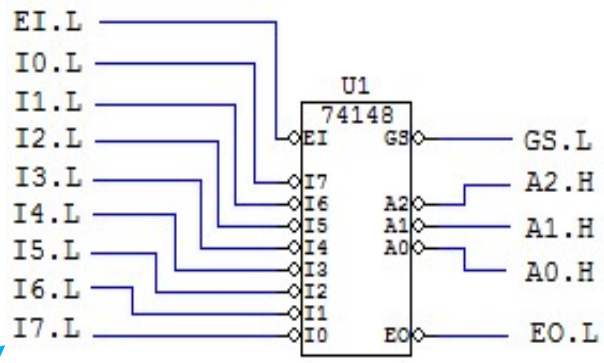
6.3. Construir un circuito codificador binario de 8 a 3 con prioridad baja tomando como base el circuito codificador 74LS148, y el menor número posible de puertas lógicas que sean necesarias. Se permite definir como mejor convenga la polaridad de las entradas y de las salidas.

74LS148

FUNCTION TABLE

INPUTS									OUTPUTS				
EI	0	1	2	3	4	5	6	7	A2	A1	A0	GS	EO
H	X	X	X	X	X	X	X	X	H	H	H	H	H
L	H	H	H	H	H	H	H	H	H	H	H	H	L
L	X	X	X	X	X	X	X	L	L	L	L	L	H
L	X	X	X	X	X	X	L	H	L	L	H	L	H
L	X	X	X	X	X	L	H	H	L	H	L	L	H
L	X	X	X	X	L	H	H	H	L	H	H	L	H
L	X	X	X	L	H	H	H	H	H	L	L	L	H
L	X	X	L	H	H	H	H	H	H	L	H	L	H
L	X	L	H	H	H	H	H	H	H	H	L	L	H
L	L	H	H	H	H	H	H	H	H	H	H	L	H

COD 8 a 3
HPRI
I y A son .L



EI	I7	I6	I5	I4	I3	I2	I1	I0	A2	A1	A0	GS	E0
0	X	X	X	X	X	X	X	X	1	1	1	0	0
1	0	0	0	0	0	0	0	0	1	1	1	0	1
1	X	X	X	X	X	X	X	1	0	0	0	1	0
1	X	X	X	X	X	X	1	0	0	0	1	1	0
1	X	X	X	X	X	1	0	0	0	1	1	1	0
1	X	X	X	X	1	0	0	0	0	1	0	1	0
1	X	X	X	1	0	0	0	0	1	1	0	1	0
1	X	X	1	0	0	0	0	0	1	1	1	1	0
1	X	1	0	0	0	0	0	0	1	0	1	1	0
1	1	0	0	0	0	0	0	0	1	0	0	1	0

Cambio en la tabla el índice en las entradas I, y la polaridad en las salidas A. El mismo circuito es un COD 8 a 3 de prioridad baja y salidas A2A1A0 .H

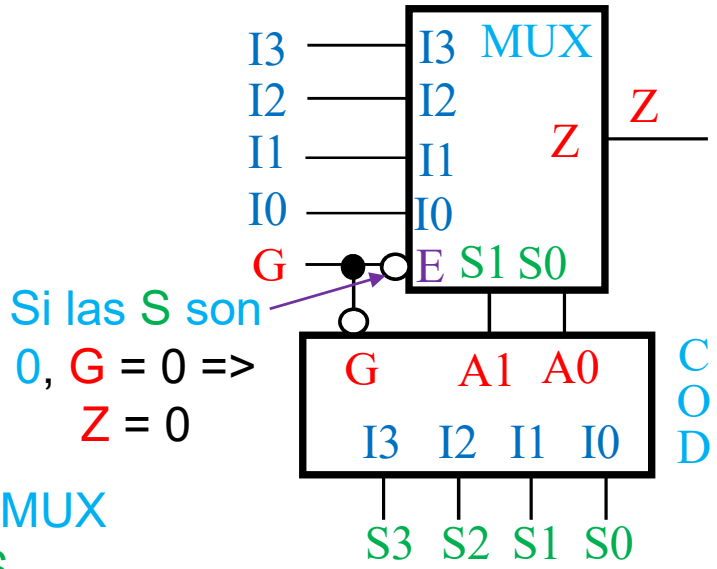
7.1. Diseñar un circuito multiplexor con prioridad de 4 bits. El circuito tiene 4 entradas de datos (I3-I0), 4 entradas de selección (S3-S0) y dos salidas Z y G. Cuando una o más de las entradas S están a 1, Z toma el valor de la entrada Ii, siendo i es el índice más alto de las entradas Si que están a 1; si todas las entradas S3-S0 están a 0, entonces Z toma el valor 0. La salida G se fija a 1 si al menos alguna entrada Si está a 1, en caso contrario se fija a 0. Utilizar en el diseño circuitos MSI convencionales: un 74LS148 (8 a 3 HPRI COD) y un circuito 74LS153 (4-input MUX).

Tabla del problema: incluye señales A intermedias

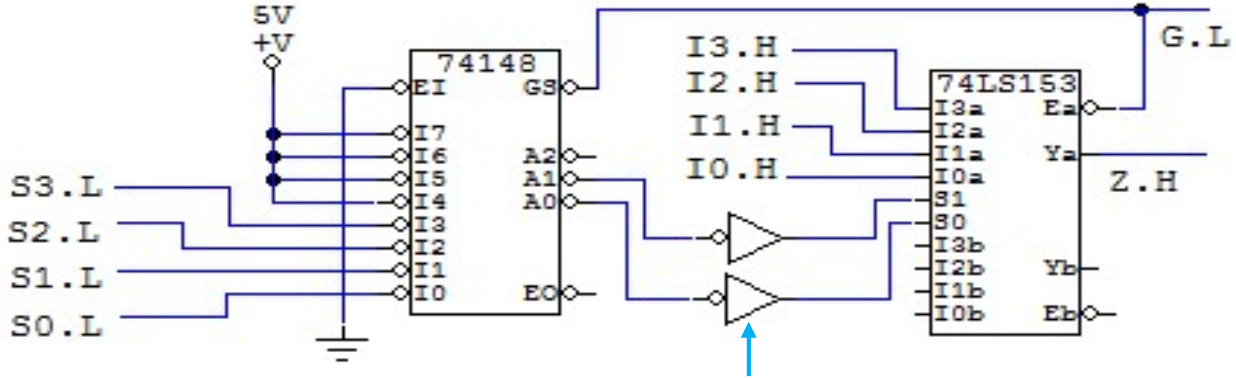
S3	S2	S1	S0	A1	A0	Z	G
1	X	X	X	1	1	I3	1
0	1	X	X	1	0	I2	1
0	0	1	X	0	1	I1	1
0	0	0	1	0	0	I0	1
0	0	0	0	0	0	0	0

HPRI COD 4 a 2
S => I
A => A
G => G

4-INP MUX
A => S
I3-0 => I3-0
G => E
Z => Z

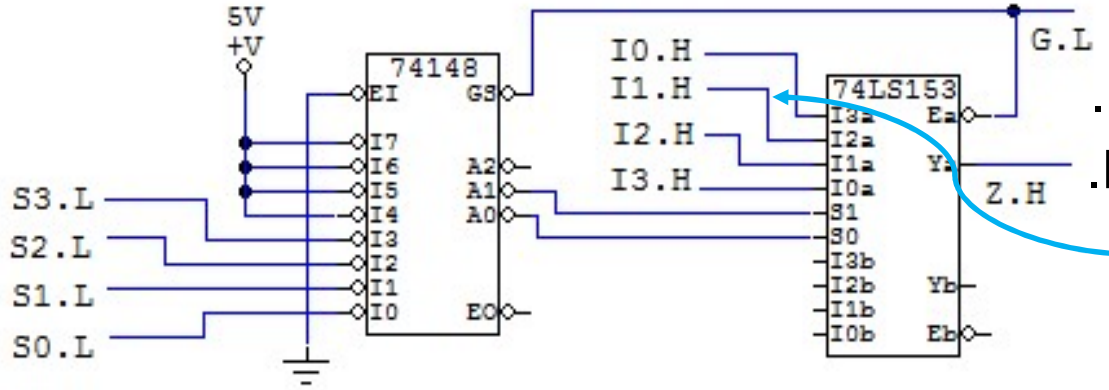


7.1. Diseñar un circuito multiplexor con prioridad de 4 bits. El circuito tiene 4 entradas de datos (I3-I0), 4 entradas de selección (S3-S0) y dos salidas Z y G. Utilizar en el diseño circuitos MSI convencionales: un 74LS148 (8 a 3 HPRI COD) y un circuito 74LS153 (4-input MUX).



148 es un HPRI COD 8 a 3
Fijo I7-I3 a 0 (H en .L). A2 no se usa

HPRI COD 8 a 3: salidas A .L
MUX: entradas S .H
Uso NOT



.L
.H

Sin NOTs
A1A0 = 00 = HH => Z = I0
S1S0 = HH = 11 => Za = I3a
Conectar I0 a I3a
Igualmente
I1 a I2a, I2 a I1a, I3 a I0a

7.2. Realizar un circuito conversor del código BCD con pesos (8, 7, -2, -4) al código NBCD (8, 4, 2, 1) usando únicamente circuitos 74'138 (decodificador 3 a 8) y 74'147 (codificador con prioridad alta 10 a 4).

Tabla del problema: incluye señales B intermedias

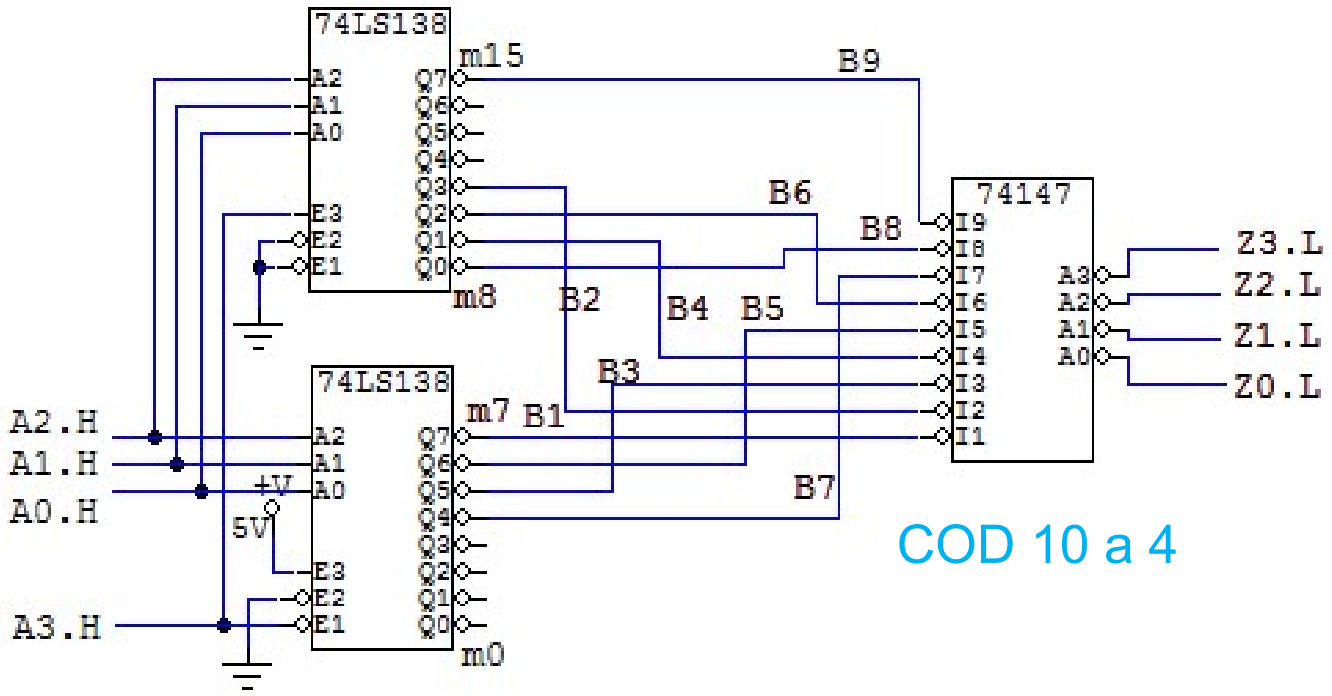
	8	7	-2	-4												8	4	2	1
D	**A3**	**A2**	**A1**	**A0**	**O**	**B9**	**B8**	**B7**	**B6**	**B5**	**B4**	**B3**	**B2**	**B1**	**B0**	**A3**	**A2**	**A1**	**A0**
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
1	0	1	1	1	7	0	0	0	0	0	0	0	0	1	0	0	0	0	1
2	1	0	1	1	11	0	0	0	0	0	0	0	1	0	0	0	0	1	0
3	0	1	0	1	5	0	0	0	0	0	0	1	0	0	0	0	0	1	1
4	1	0	0	1	9	0	0	0	0	0	1	0	0	0	0	0	1	0	0
5	0	1	1	0	6	0	0	0	0	1	0	0	0	0	0	0	1	0	1
6	1	0	1	0	10	0	0	0	1	0	0	0	0	0	0	0	1	1	0
7	0	1	0	0	4	0	0	1	0	0	0	0	0	0	0	0	1	1	1
8	1	0	0	0	8	0	1	0	0	0	0	0	0	0	0	1	0	0	0
9	1	1	1	1	15	1	0	0	0	0	0	0	0	0	0	1	0	0	1

DECOD 4 a 10
(no binario)
A => A, B => O
Construir con DEC 3 a 8
74'138

Solo un 1 en las entradas.
El COD podría ser sin prioridad, solo con puertas OR.

COD HPRI 10 a 4
(binario)
B => I, A => A
74'147

7.2. Realizar un circuito conversor del código BCD con pesos (8, 7, -2, -4) al código NBCD (8, 4, 2, 1) usando únicamente circuitos 74'138 (decodificador 3 a 8) y 74'147 (codificador con prioridad alta 10 a 4).



7.3. Se quiere realizar un circuito de 8 entradas (I7-I0) y 8 salidas (O7-O0), tal que la salida muestra la entrada pero eliminando todos los unos menos el más significativo. Por ejemplo, si I = “01101101”, O = “01000000”; si I = “00010110”, O = “00010000”, etc. Si todos los bits de la entrada son 0, los de la salida también: I = “00000000”, O = “00000000”.

a) Realizar un código VHDL para la descripción del problema.

```
library ieee;
use ieee.std_logic_1164.all;

entity problema7_3 is
port (I: in std_logic_vector(7 downto 0);
      O: out std_logic_vector(7 downto 0));
end problema7_3;

architecture uno of problema7_3 is
begin
process(I)
begin
if ( I(7) = '1') then O <= "10000000";
elsif ( I(6) = '1') then O <= "01000000";
elsif ( I(5) = '1') then O <= "00100000";
elsif ( I(4) = '1') then O <= "00010000";
elsif ( I(3) = '1') then O <= "00001000";
elsif ( I(2) = '1') then O <= "00000100";
elsif ( I(1) = '1') then O <= "00000010";
elsif ( I(0) = '1') then O <= "00000001";
else O <= "00000000";
end if;
end process;
end uno;
```

```
library ieee;
use ieee.std_logic_1164.all;

entity problema7_3gen is
generic(N: integer := 8);
port (I: in std_logic_vector(N-1 downto 0);
      O: out std_logic_vector(N-1 downto 0));
end problema7_3gen;

architecture uno of problema7_3gen is
begin
process(I)
begin
O <= (others => '0');
for j in N-1 downto 0 loop
   if ( I(j) = '1' ) then
      O(j) <= '1';
      exit;
   end if;
end loop;
end process;
end uno;
```

b) Implementar el circuito con un circuito codificador 8 a 3 74LS148 y un circuito decodificador 3 a 8 74LS138. Suponer las entradas y salidas I7.L, … I0.L; O7.L, … O0.L en polaridad negativa.

Tabla del problema: incluye señales A y G/E intermedias

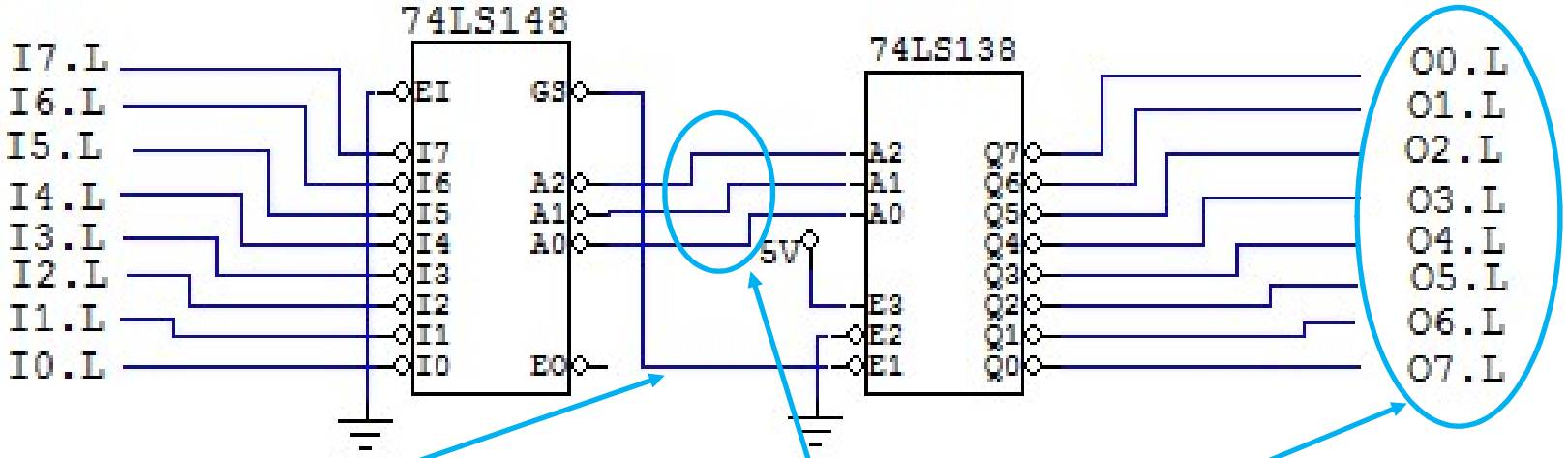
I7	I6	I5	I4	I3	I2	I1	I0	A2	A1	A0	G/E	O7	O6	O5	O4	O3	O2	O1	O0
1	X	X	X	X	X	X	X	1	1	1	0	1	0	0	0	0	0	0	0
0	1	X	X	X	X	X	X	1	1	0	0	0	1	0	0	0	0	0	0
0	0	1	X	X	X	X	X	1	0	1	0	0	0	1	0	0	0	0	0
0	0	0	1	X	X	X	X	1	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	1	X	X	X	0	1	1	0	0	0	0	0	1	0	0	0
0	0	0	0	0	1	X	X	0	1	0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	1	X	0	0	1	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

HPRI COD 8 a 3
I => I
A => A
G => G

DEC 3 a 8
A => A
O => O
G => E

Conecto la salida G del COD a la entrada E del DEC. Si las Is son 0, las Os son 0.

b) Implementar el circuito con un circuito codificador 8 a 3 74LS148 y un circuito decodificador 3 a 8 74LS138. Suponer las entradas y salidas I7.L, … I0.L; O7.L, … O0.L en polaridad negativa.



GS (G) = 0 (H) =>
=> E = 0 (H) =>
=> Os = 0 (H)

Polaridad distinta
Solución 1: usar NOTs
Solución 2: cambiar la posición de las salidas.

.L COD A2A1A0 = 000 (HHH) => O0 = 1
.H DEC A2A1A0 = 111 (HHH) => O7(DEC) = 1
DEC O7(DEC) => O0. Igualmente:
O6(DEC) => O1; O5(DEC) => O2;
O4(DEC) => O3; O3(DEC) => O4;
O2(DEC) => O5; O1(DEC) => O6; O0(DEC) => O7

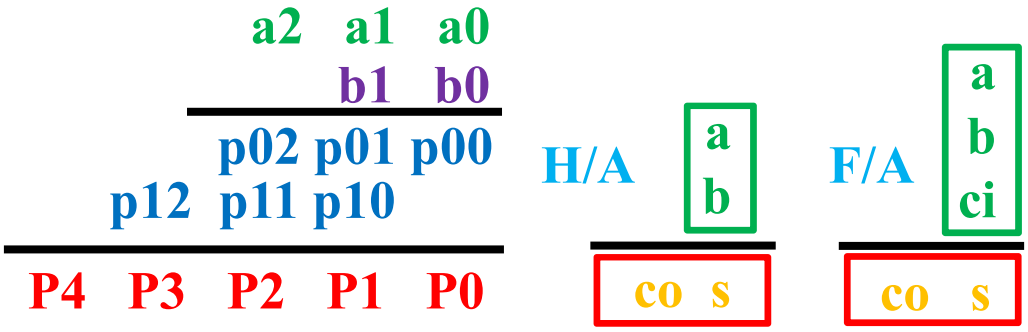
8.1. Diseñar utilizando únicamente semisumadores y sumadores completos un circuito digital que realice la multiplicación de un números binario de dos bits por otro de tres bits.

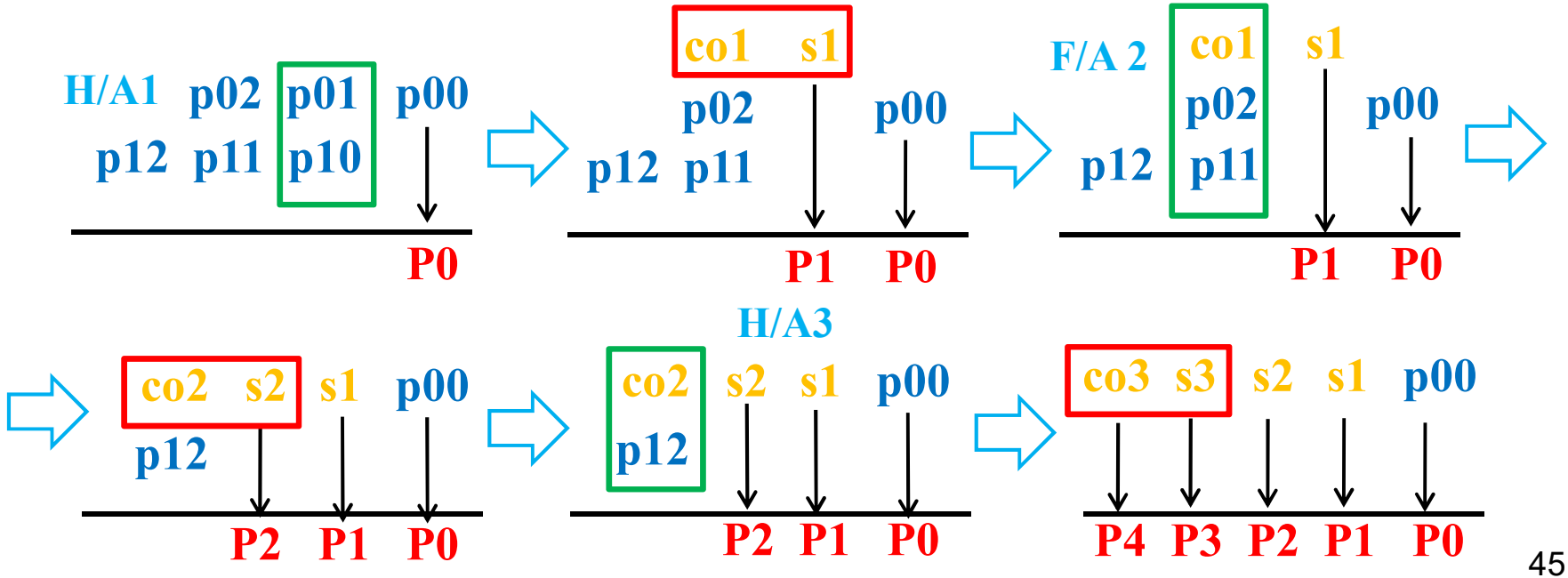
A (3 bits) * B (2 bits) = P (5 bits) Pmax = Amax * Bmax = 7 * 3 = 21

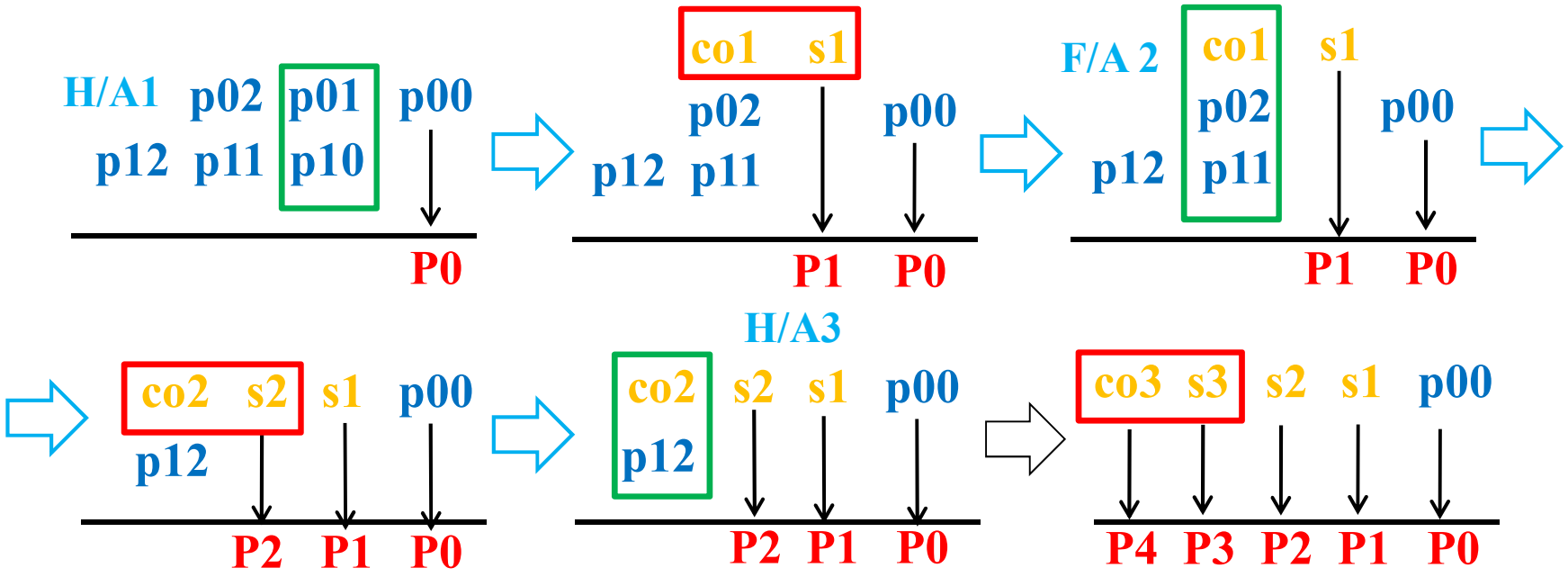
Bi	Aj	Pij
0	0	0
0	1	0
1	0	0
1	1	1

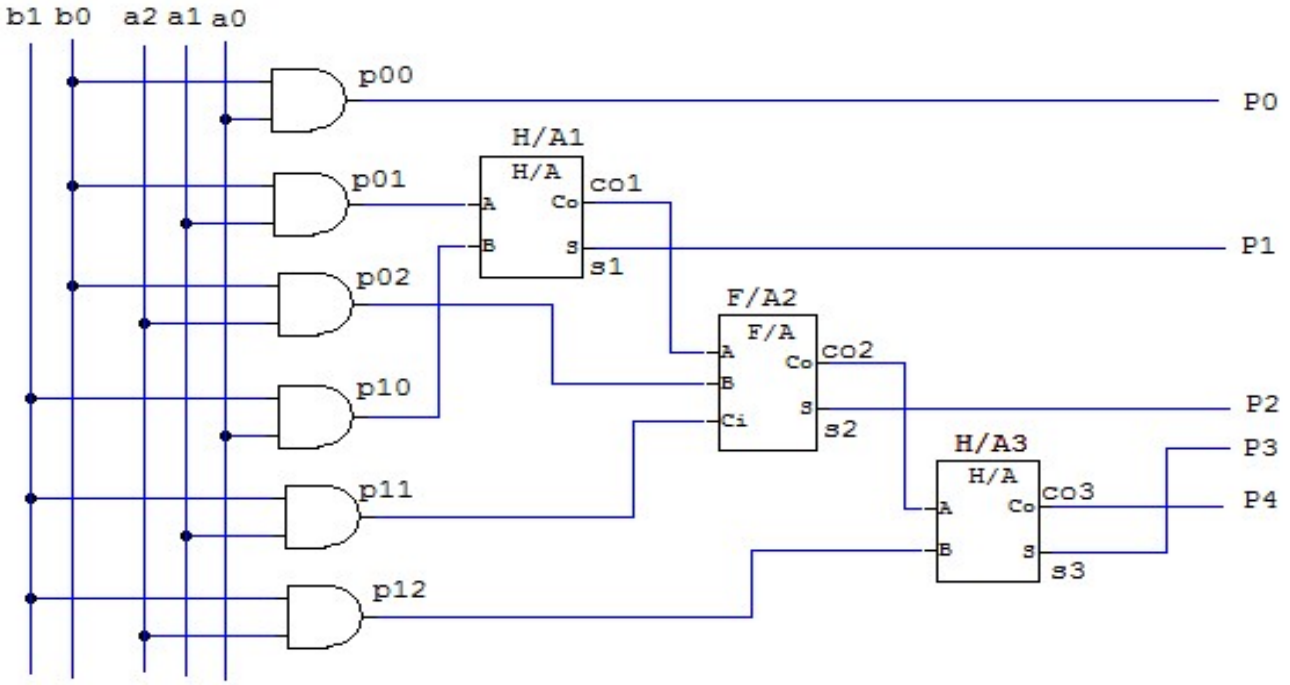
Multiplicador 1 bit: Pij = Bi * Aj

Puerta AND






H/A1
p02
p01
p00
p12
p11
p10
P0
co1
s1
p02
p00
p12
p11
P1
P0
F/A 2
co1
s1
p02
p00
p12
p11
P1
P0
co2
s2
s1
p00
p12
P2
P1
P0
H/A3
co2
s2
s1
p00
p12
P2
P1
P0
co3
s3
s2
s1
p00
P4
P3
P2
P1
P0


b1 b0 a2 a1 a0
p00
P0
H/A1
H/A
A
B
Co
S
co1
s1
p01
P1
p02
F/A2
F/A
A
B
Ci
Co
S
co2
s2
p10
P2
H/A3
H/A
A
B
Co
S
co3
s3
p11
P3
P4
p12

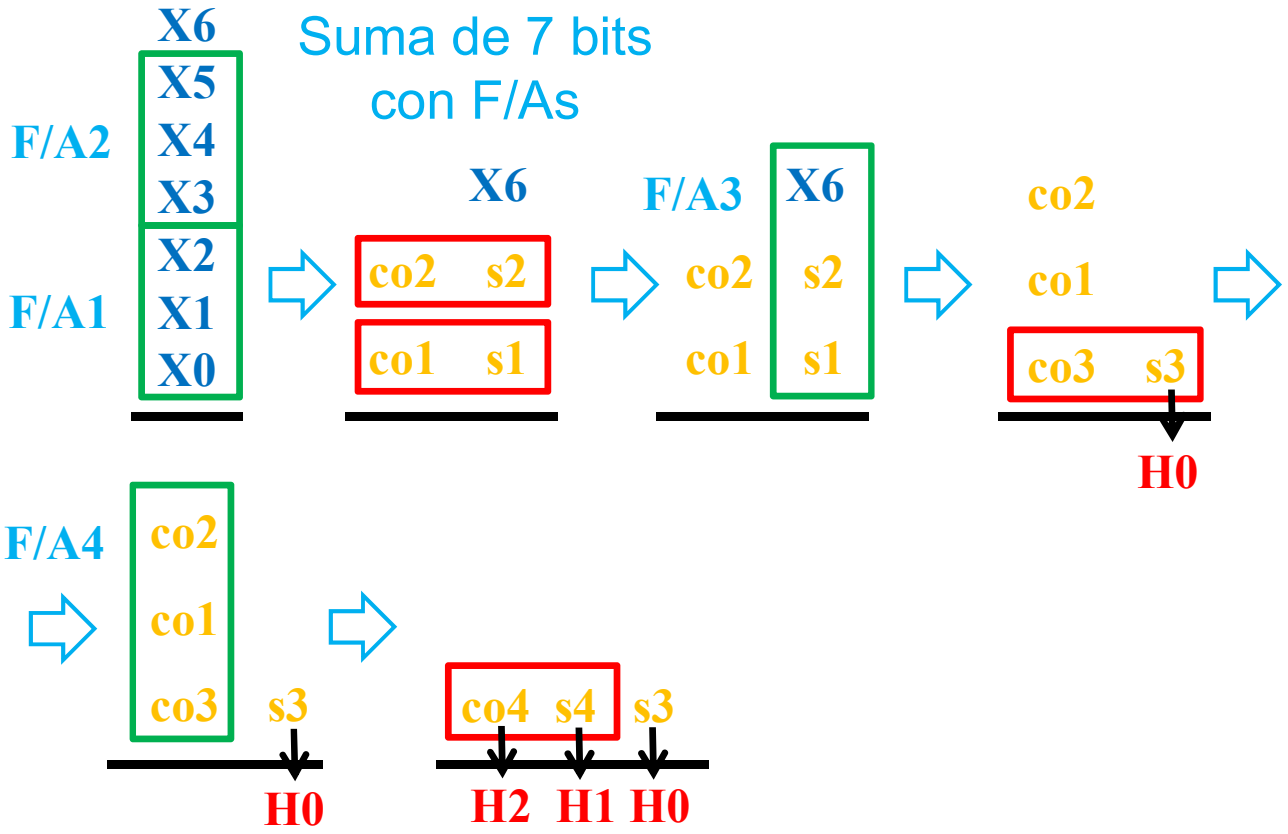
8.2. Diseñar un circuito que calcule la distancia de Hamming de dos palabras A y B de 7 bits usando el menor número posible de puertas lógicas y/o de dispositivos MSI.

A y B de 7 bits; DH valores entre 0 y 7 => DH es de 3 bits (H2 H1 H0)

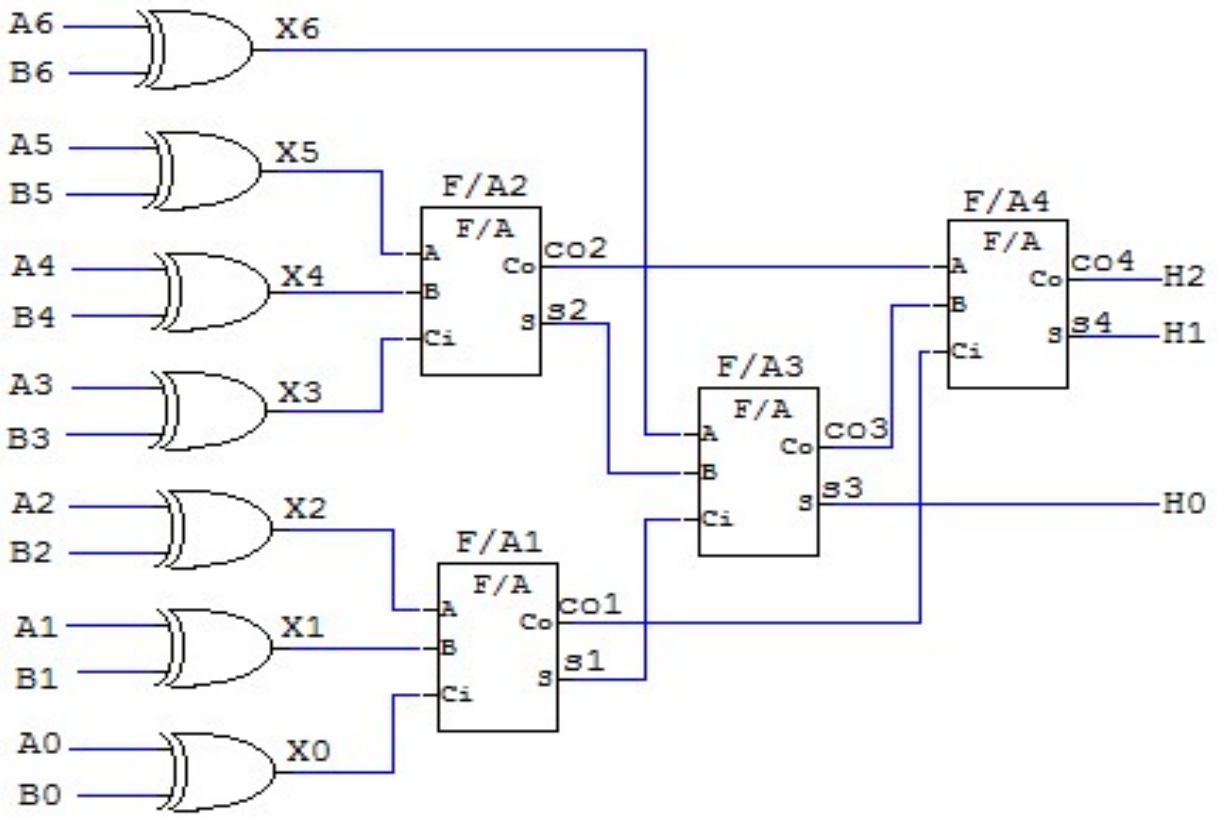
$DH = \sum_i (A_i \oplus Bi)$

A_i	B_i	$\neq$
0	0	0
0	1	1
1	0	1
1	1	0

$X_i = (A_i \neq B_i) = A_i \oplus B_i$

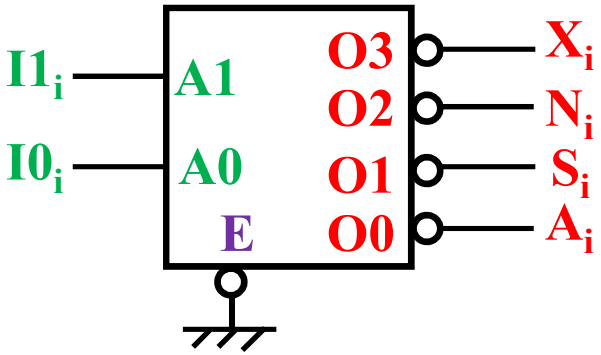


8.2. Diseñar un circuito que calcule la distancia de Hamming de dos palabras A y B de 7 bits usando el menor número posible de puertas lógicas y/o de dispositivos MSI.



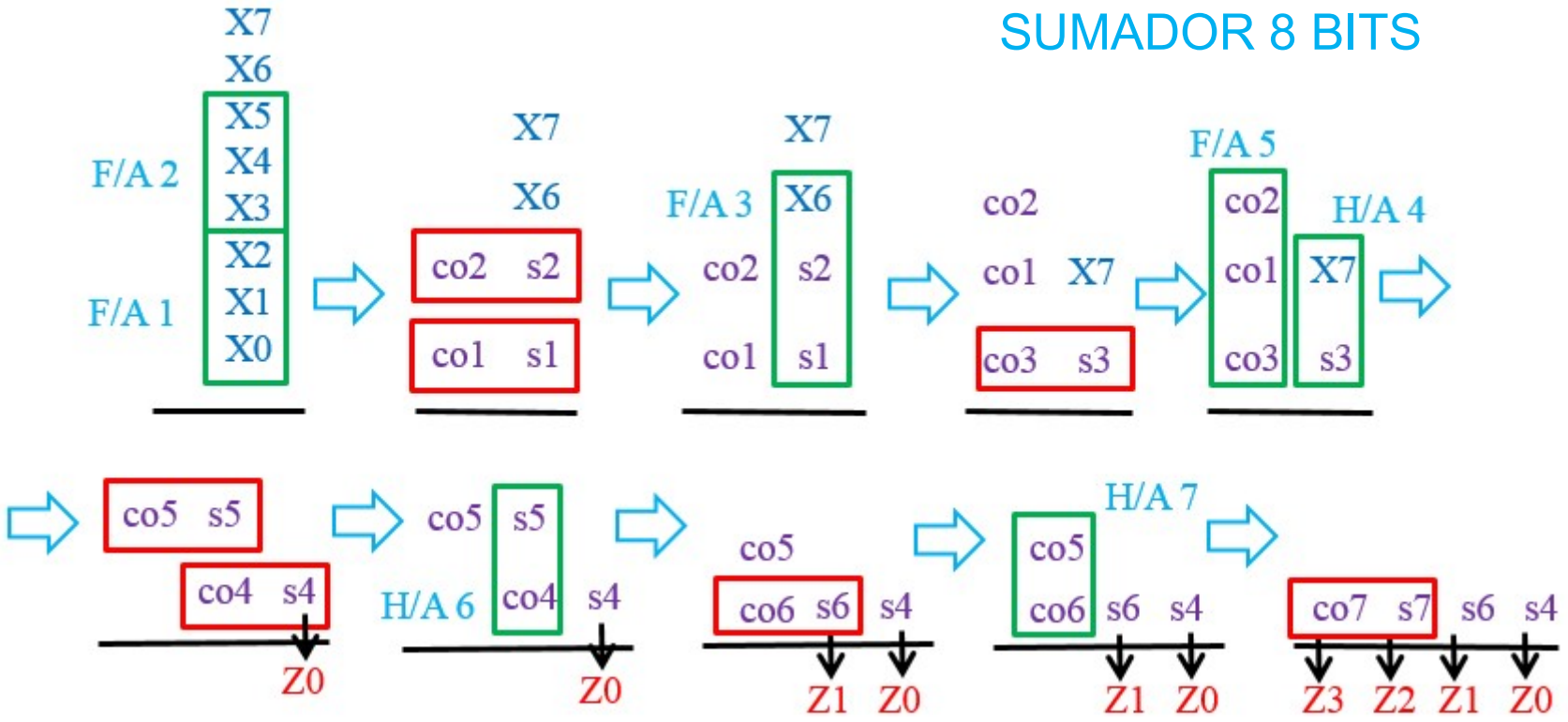
8.3. Diseñar utilizando elementos MSIs (sumadores y decodificadores) un circuito que calcule el resultado de una votación de siete votos. Cada voto aparece codificado mediante dos bits I1I0, de forma que la abstención se representa por 00, 'Si' por 01, 'No' por 10 y los votos nulos aparecen como 11. El resultado de la votación debe darse indicando el número de votos de cada tipo. Realizar lo mismo para una votación de ocho votos.

1 DEC por votante i

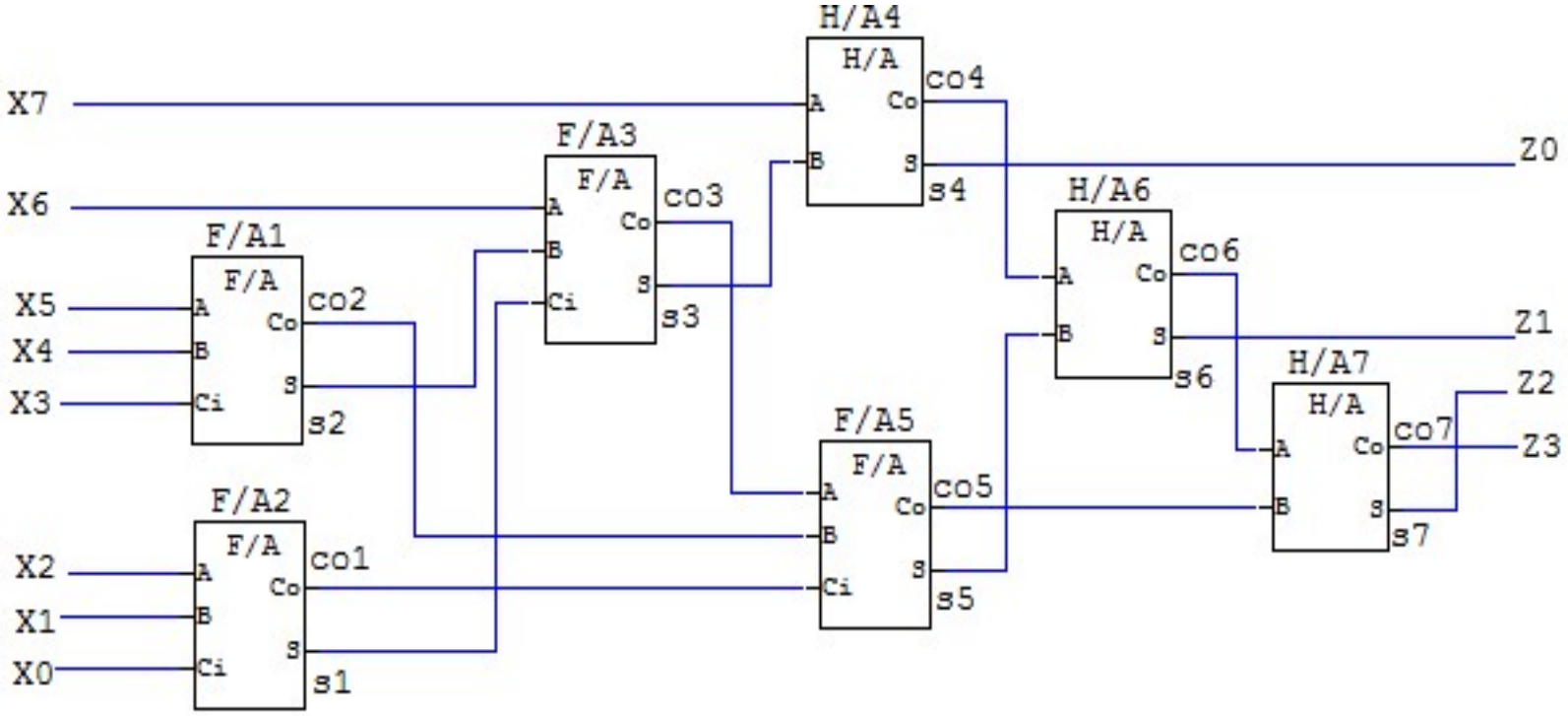


4 CIRCUITOS DE SUMA DE 7 bits. PROBLEMA ANTERIOR.

8.3. Diseñar utilizando elementos MSIs (sumadores y decodificadores) un circuito que calcule el resultado de una votación de siete votos. Cada voto aparece codificado mediante dos bits I1I0, de forma que la abstención se representa por 00, 'Si' por 01, 'No' por 10 y los votos nulos aparecen como 11. El resultado de la votación debe darse indicando el número de votos de cada tipo. Realizar lo mismo para una votación de ocho votos.



8.3. Diseñar utilizando elementos MSIs (sumadores y decodificadores) un circuito que calcule el resultado de una votación de siete votos. Cada voto aparece codificado mediante dos bits I1I0, de forma que la abstención se representa por 00, 'Si' por 01, 'No' por 10 y los votos nulos aparecen como 11. El resultado de la votación debe darse indicando el número de votos de cada tipo. Realizar lo mismo para una votación de ocho votos.

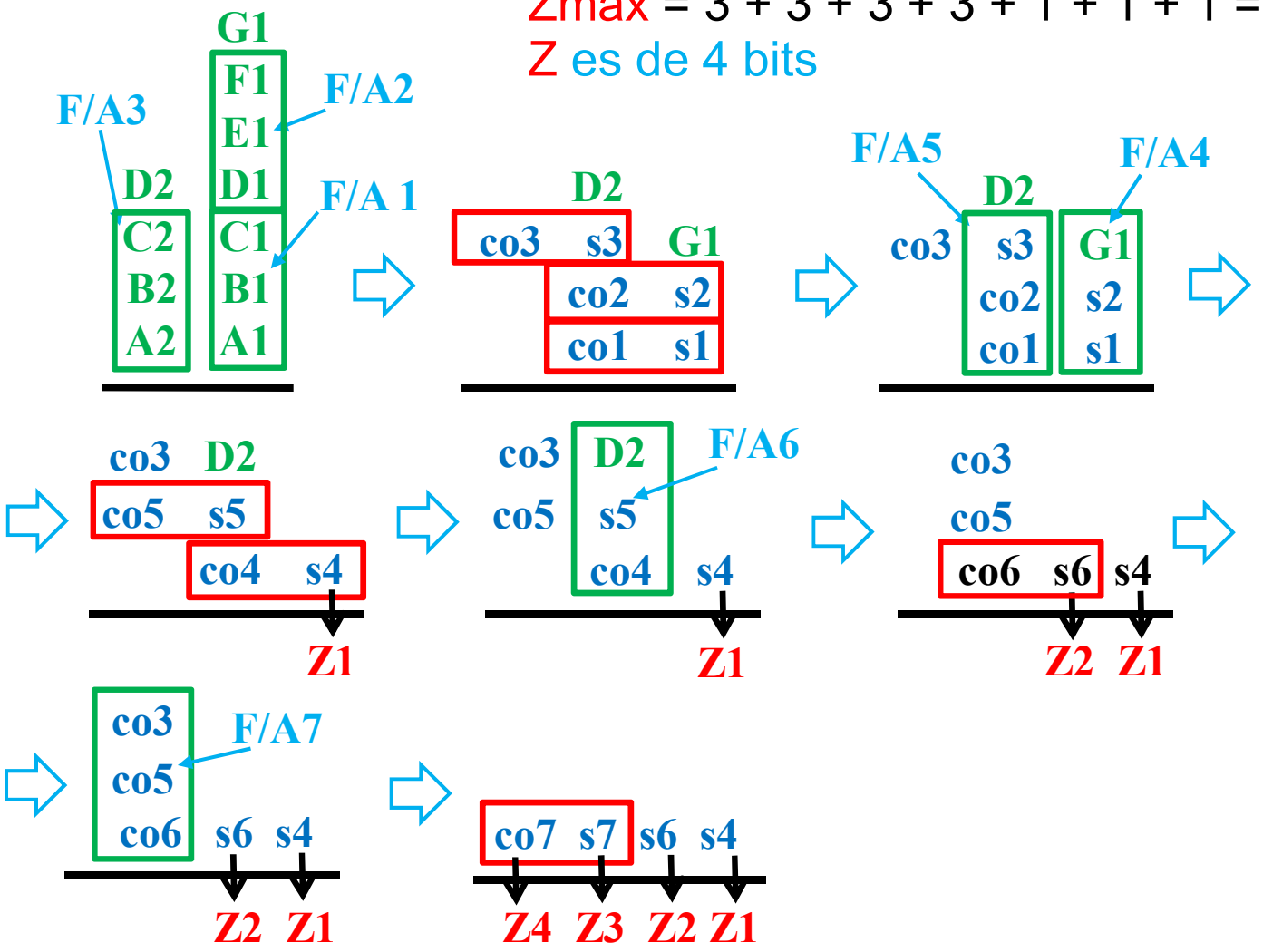


8.4. Realizar la suma de cuatro números de dos bits A (a2a1), B (b2b1), C (c2c1) y D (d2d1) y tres números de 1 bit, E (e1), F (f1) y G (g1) utilizando el menor número posible de sumadores completos ("full-adders").

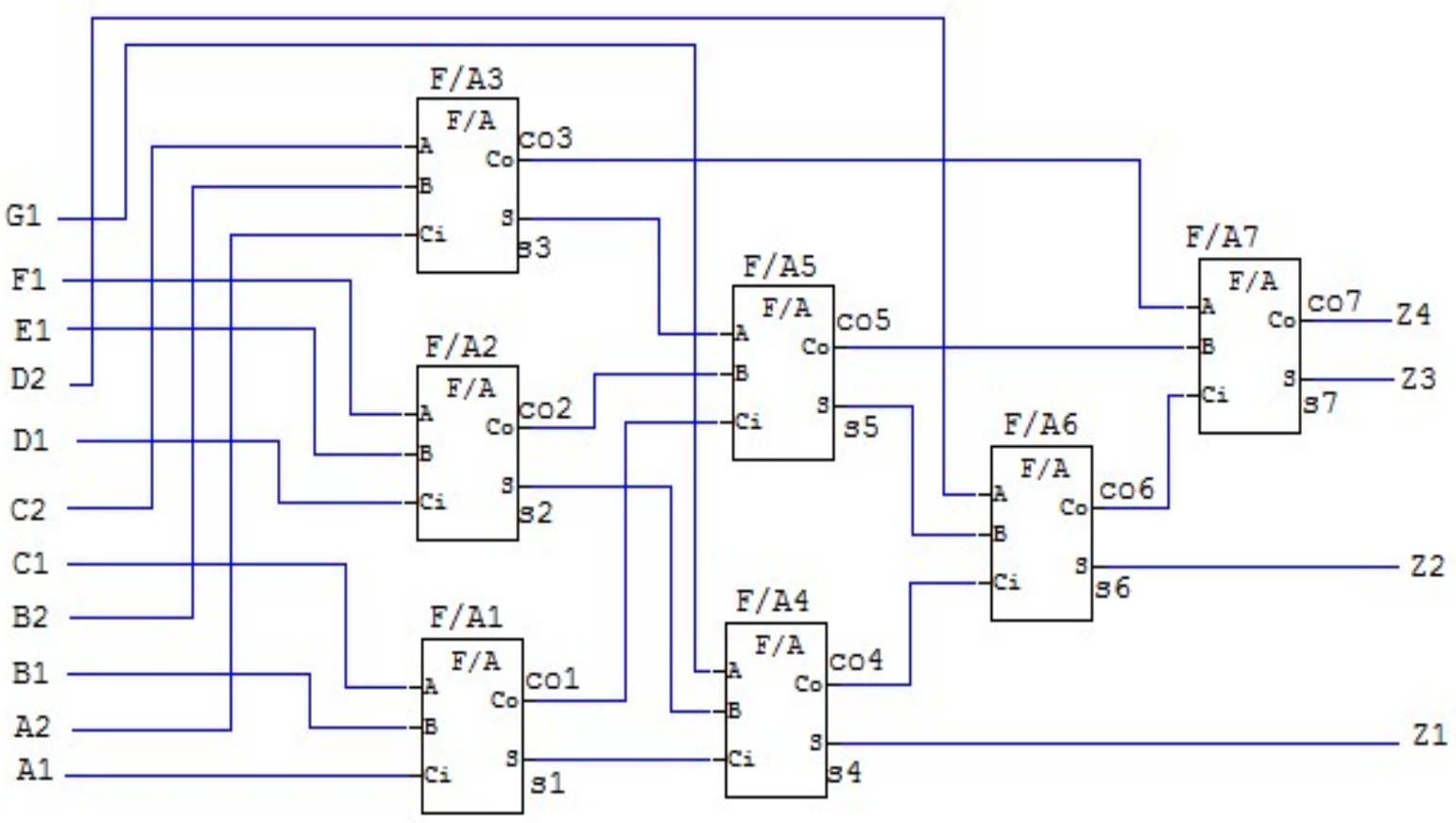
$Z = A + B + C + D + E + F + G$

$Z_{max} = 3 + 3 + 3 + 3 + 1 + 1 + 1 = 15$

Z es de 4 bits



8.4. Realizar la suma de cuatro números de dos bits A (a2a1), B (b2b1), C (c2c1) y D (d2d1) y tres números de 1 bit, E (e1), F (f1) y G (g1) utilizando el menor número posible de sumadores completos ("full-adders").



9.1. Diseñar un circuito que realice la operación aritmética:

$O = 5X + 2Y + Z$

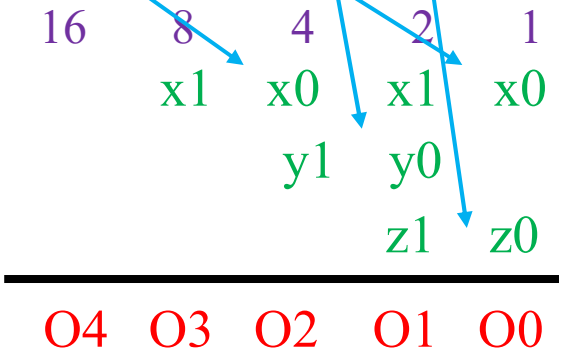
para operandos X (x1x0), Y (y1y0) y Z (z1z0) de dos bits, utilizando el menor número posible de semisumadores de dos bits de operandos de entradas A (a1a0) y B (b1b0).

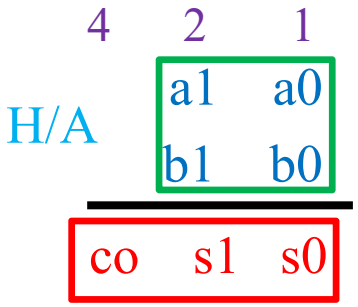
$O_{max} = 5 * 3 + 2 * 3 + 3 = 24$ O es de 5 bits

$O = 5X + 2Y + Z = 4X + X + 2Y + Z = 2^2 X + X + 2^1 Y + Z$

$O = (X00) + X + (Y0) + Z$

$2^N X = X\,\underbrace{0\ldots0}_{N}$

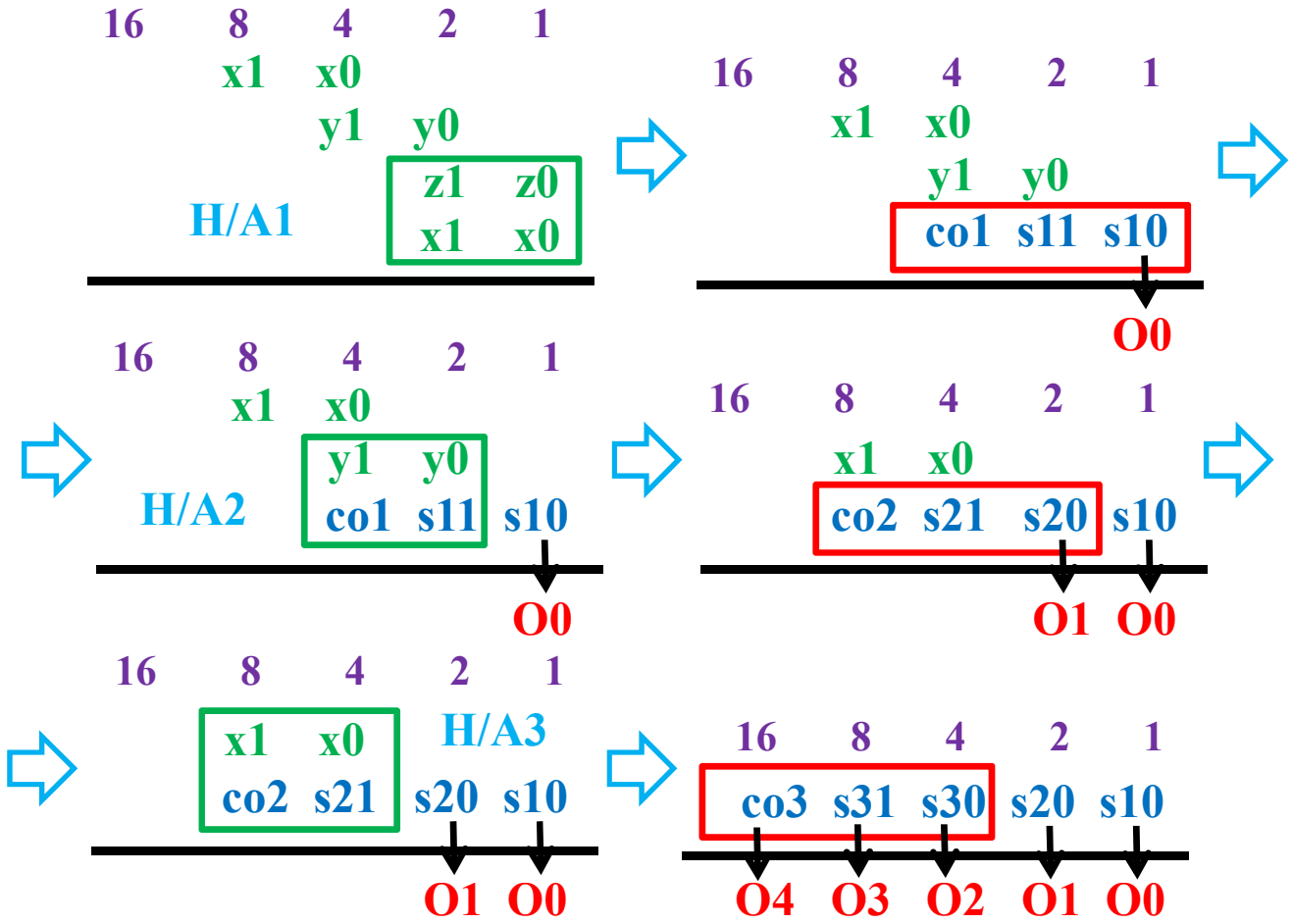




9.1. Diseñar un circuito que realice la operación aritmética:

$O = 5X + 2Y + Z$

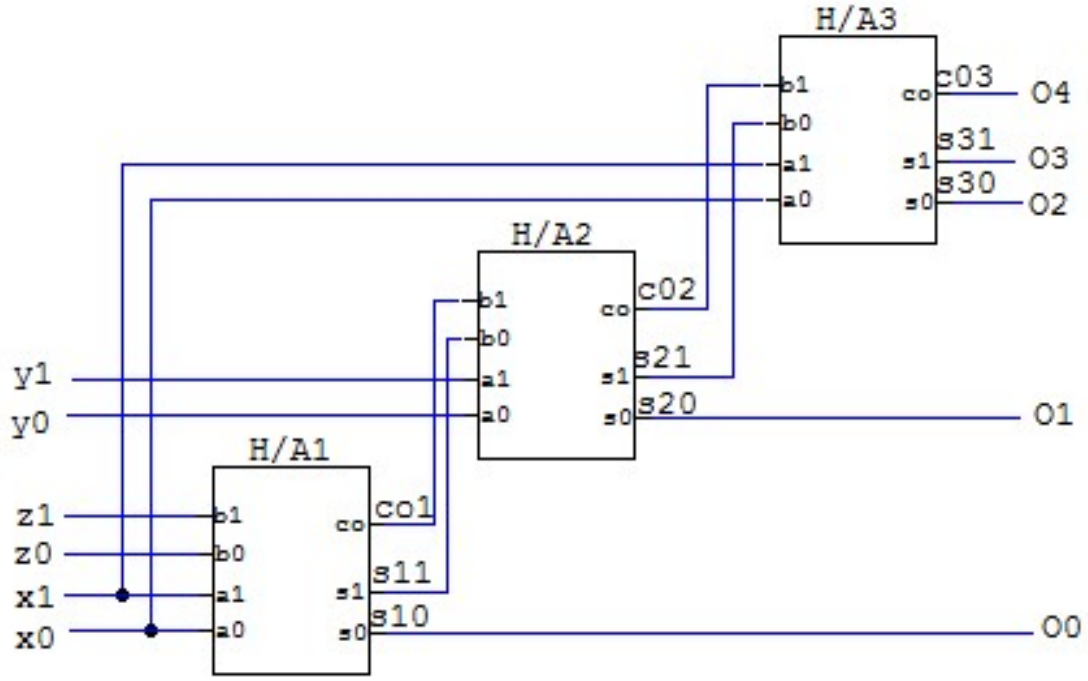
para operandos X (x1x0), Y (y1y0) y Z (z1z0) de dos bits, utilizando el menor número posible de semisumadores de dos bits de operandos de entradas A (a1a0) y B (b1b0).



9.1. Diseñar un circuito que realice la operación aritmética:

$O = 5X + 2Y + Z$

para operandos X (x1x0), Y (y1y0) y Z (z1z0) de dos bits, utilizando el menor número posible de semisumadores de dos bits de operandos de entradas A (a1a0) y B (b1b0).



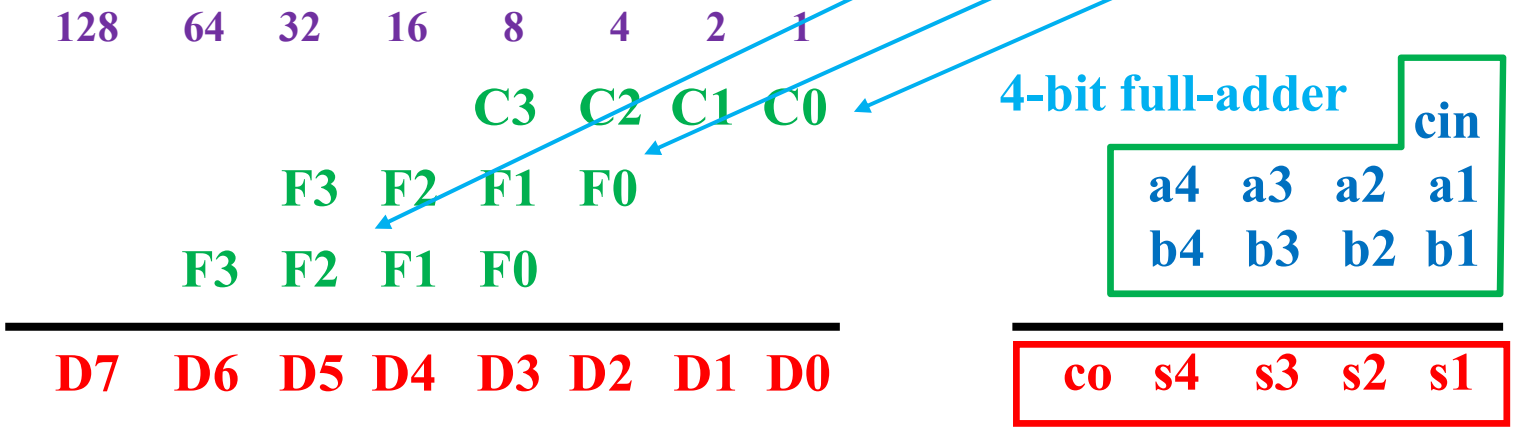
9.2. Un sistema digital accede a los elementos de una matriz de 12*12 (144 elementos) que están almacenados en memoria. Para acceder a un elemento de la matriz el sistema utiliza la posición de filas F y la posición de columnas C, ambas de 4 bits (F3-F0, C3-C0) en código binario con valores entre 0 y 11. Sin embargo, la memoria utilizada sólo tiene un bus de direcciones D.

Diseñar un circuito que genere la dirección D (8 bits D7-D0, valores entre 0 y 143) del elemento de la matriz en memoria de la forma D = 12 • F + C. Utilizar el menor número posible de sumadores (preferentemente 74LS83) para realizar la implementación.

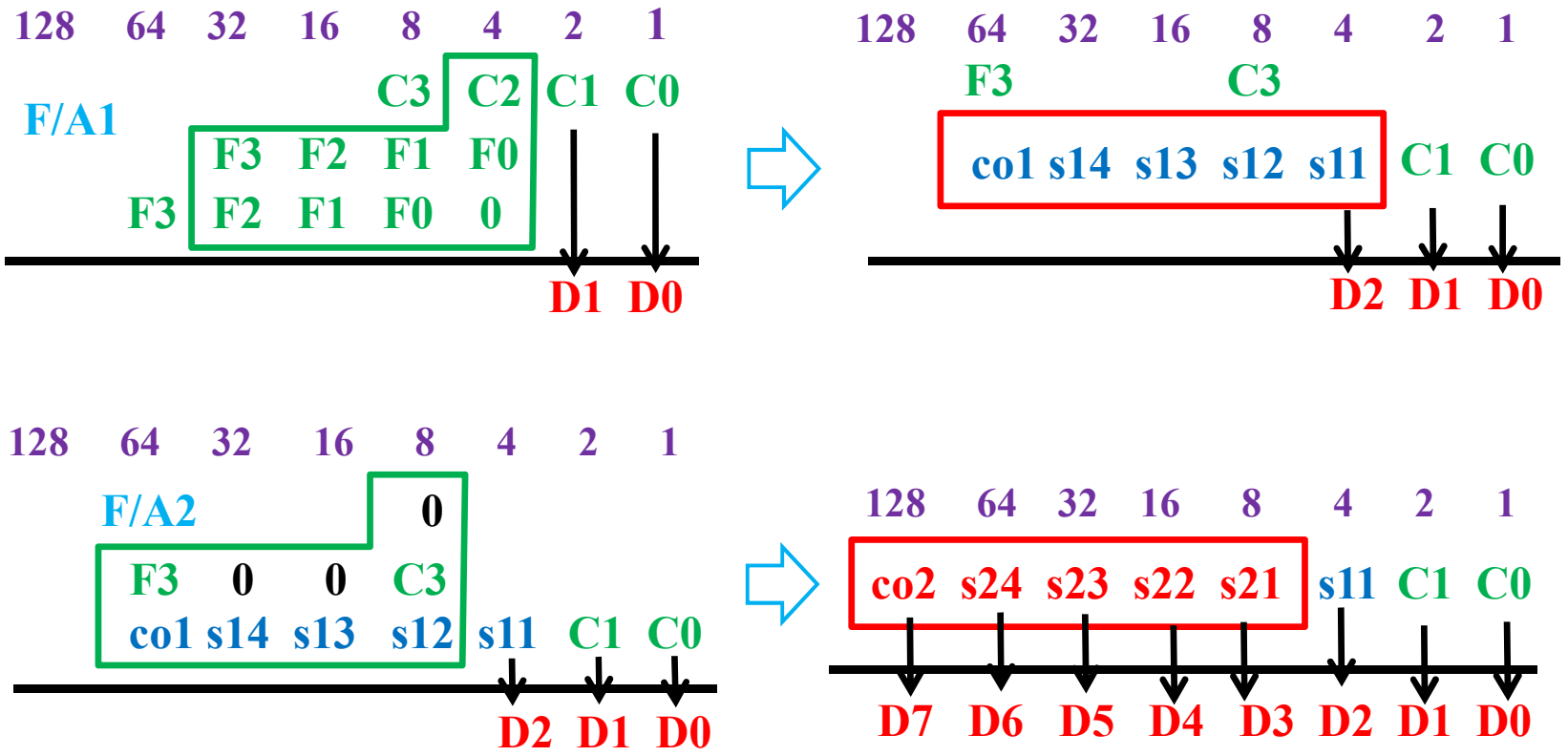
$D = 12\,F + C = 8\,F + 4\,F + C = 2^3\,F + 2^2\,F + C = (F000) + (F00) + C$

$2^N\,X = X\,0\ldots0$ (N)

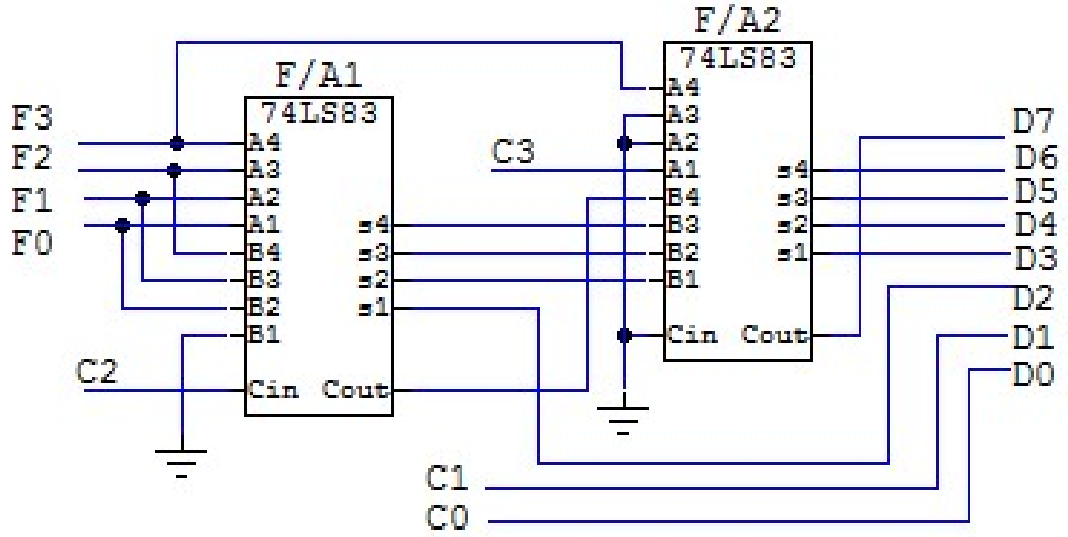
Dmax = 143 (8 bits)



9.2. Diseñar un circuito que genere la dirección D (8 bits D7-D0, valores entre 0 y 143) del elemento de la matriz en memoria de la forma D = 12 • F + C. Utilizar el menor número posible de sumadores (preferentemente 74LS83) para realizar la implementación.



9.2. Diseñar un circuito que genere la dirección D (8 bits D7-D0, valores entre 0 y 143) del elemento de la matriz en memoria de la forma D = 12 • F + C. Utilizar el menor número posible de sumadores (preferentemente 74LS83) para realizar la implementación.



9.3. Realizar el diseño de un circuito que sume dos dígitos NBCD, dando el resultado en código NBCD, utilizando puertas lógicas cuando sea necesario. Indicar como puede utilizarse este circuito para sumar números NBCD de más de un dígito. Ayuda: Hay que sumar 6

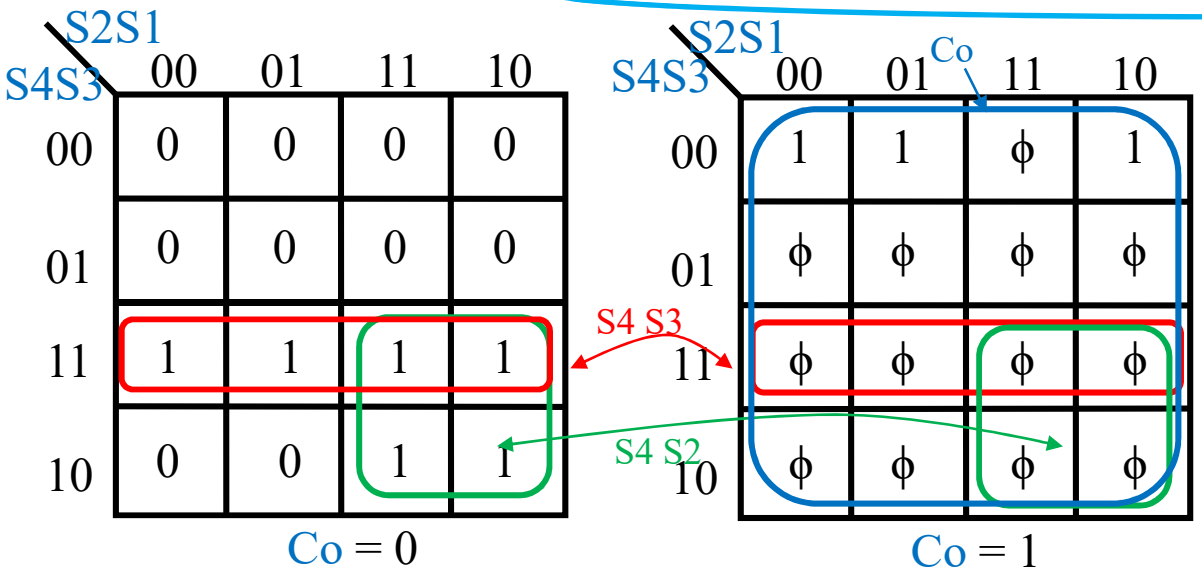
Entradas A y B [0, 9] 4 bits

Salida D = A + B [0, 18] (NBCD)
Dos dígitos NBCD D2 [0, 1], D1 [0, 9]

Sumador 1

S = A + B [0, 18] (binario, 5 bits CoS4S3S2S1); de S genero Sx (5 bits)
S < 10 => Sx = S + 0; D2 = 0 (o Sx[5]), D1 = S[4:1] = Sx[4:1]
S ≥ 10 => Sx = S + 6; D2 = 1 (o Sx[5]), D1 = Sx[4:1]
Z = (S ≥ 10) => Sx = S + (0ZZ0) (0ZZ0) es 0 si Z es 0, 6 si Z es 1

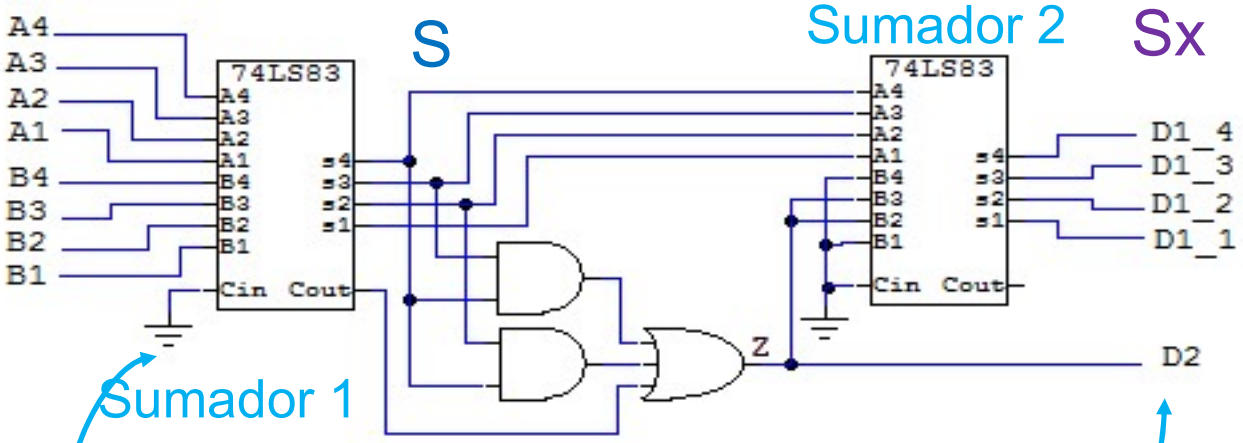
Sumador 2



D2 = Z (o Sx[5])
D1 = Sx[4:1]

Z = (S ≥ 10) = Co + S4 S3 + S4 S2

9.3. Realizar el diseño de un circuito que sume dos dígitos NBCD, dando el resultado en código NBCD, utilizando puertas lógicas cuando sea necesario. Ayuda: Hay que sumar 6



Indicar como puede utilizarse este circuito para sumar números NBCD de más de un dígito.

1 dígito BCD Ai + Bi = (D2i D1i)

C = A + B varios dígitos BCD

A0 + B0 = (D20 D10)

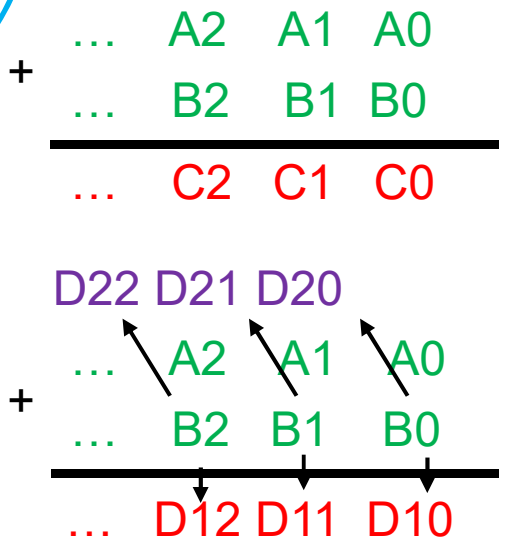
A1 + B1 + D20 = (D21 D11)

A2 + B2 + D21 = (D22 D12)

Ai + Bi + D2(i-1) [0, 19]

La función Z no cambia

Se conecta D2 del bit i al Cin del bit i+1



10.1. Realizar el diseño de un comparador de dos números A y B de cuatro bits tomando como base el sumador de números binarios de cuatro bits 74'83, utilizando puertas lógicas cuando sea necesario. El circuito debe generar tres salidas: O1 (A = B), O2 (A > B), O3 (A < B).

Realizo la comparación en base a una resta:

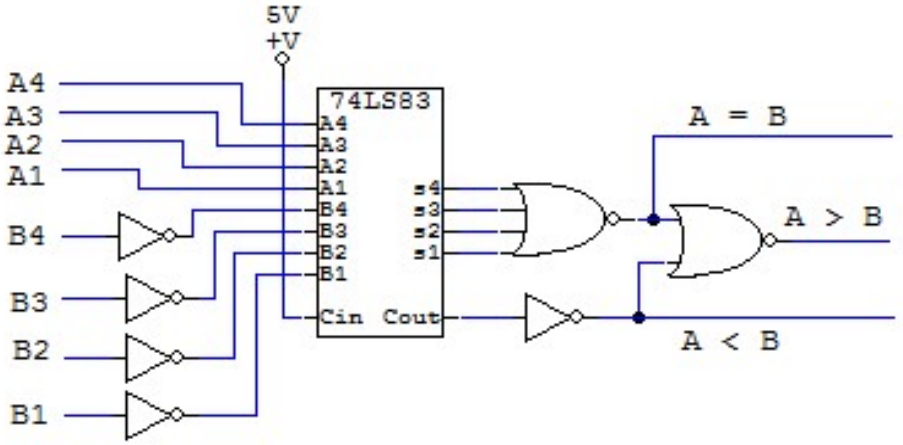
$A - B = A + (-B) = A + (B)_{2,c} = A + \overline{B} + 1$ -- Para el circuito

$A - B = A + (-B) = A + (2^N - B) = 2^N + (A - B)$ -- Para el cálculo

- Si $A < B \Rightarrow 2^N + (A - B) < 2^N \Rightarrow Cout = 0; S \neq 0$
- Si $A = B \Rightarrow 2^N + (A - B) = 2^N \Rightarrow Cout = 1; S = 0$
- Si $A > B \Rightarrow 2^N + (A - B) > 2^N \Rightarrow Cout = 1; S \neq 0$

- $A < B$ si $Cout = 0 \Rightarrow (A < B) = \overline{Cout}$
- $A = B$ si $S = 0 \Rightarrow (A = B) = \overline{S4}\,\overline{S3}\,\overline{S2}\,\overline{S1} = \overline{S4 + S3 + S2 + S1}$
- $A > B$ si $Cout = 1$ y $S \neq 0 \Rightarrow$

$(A > B) = Cout\,(S4 + S3 + S2 + S1) = Cout \cdot \overline{(A = B)} = \overline{\overline{Cout} + (A = B)}$



10.2. Diseñar un circuito que realice la operación aritmética Z = A + 1 cuando A es igual B, y la operación aritmética Z = (A - B) - 1 cuando A es mayor que B, donde A y B son números binarios de cuatro bits, siendo siempre A mayor o igual que B. Implementar el circuito utilizando como base el sumador 74'83 y otros elementos MSI y puertas lógicas.

Supongo entradas A y B, y salida Z de 4 bits

$A > B \Rightarrow Z = A - B - 1 = A + (B)_{2,c} + 1 = A + \overline{B} + 1 - 1 = A + \overline{B} + 0$

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

Se requiere un comparador para A > B, A = B ($(A > B) = (\overline{A = B})$)

y un sumador Zsum = Asum PLUS Bsum PLUS Cin

	C	Op.	Asum	Bsum	Cin
$A > B$	0	$A - B - 1$	A	$\overline{B}$	0
$A = B$	1	$A + 1$	A	0	1

C = (A = B)

Asum = A

Cin = C

Bsum = $\overline{C}\,\overline{B} + C \cdot 0 = \overline{C}\,\overline{B} = \overline{C + B}$ (4 puertas NOR)

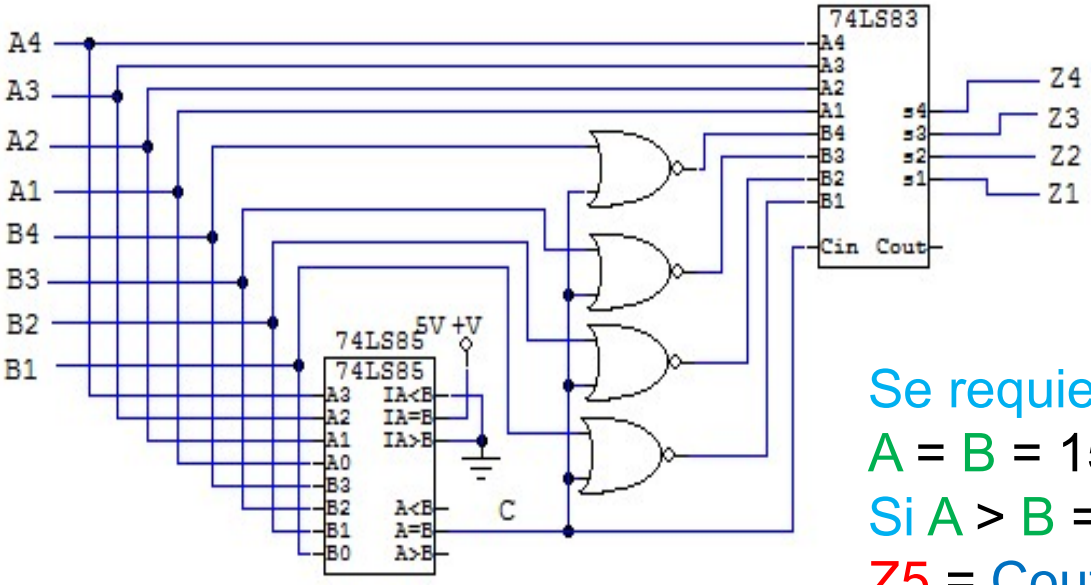
10.2. Diseñar un circuito que realice la operación aritmética Z = A + 1 cuando A es igual B, y la operación aritmética Z = (A - B) - 1 cuando A es mayor que B, donde A y B son números binarios de cuatro bits, siendo siempre A mayor o igual que B. Implementar el circuito utilizando como base el sumador 74'83 y otros elementos MSI y puertas lógicas.

C = (A = B)

Asum = A

Cin = C

Bsum = $\overline{C}\,\overline{B} + C \cdot 0 = \overline{C}\,\overline{B} = \overline{C + B}$ (4 puertas NOR)



Se requiere Z de 5 bits solo si A = B = 15 (A + 1 = 16 => Cout = 1).

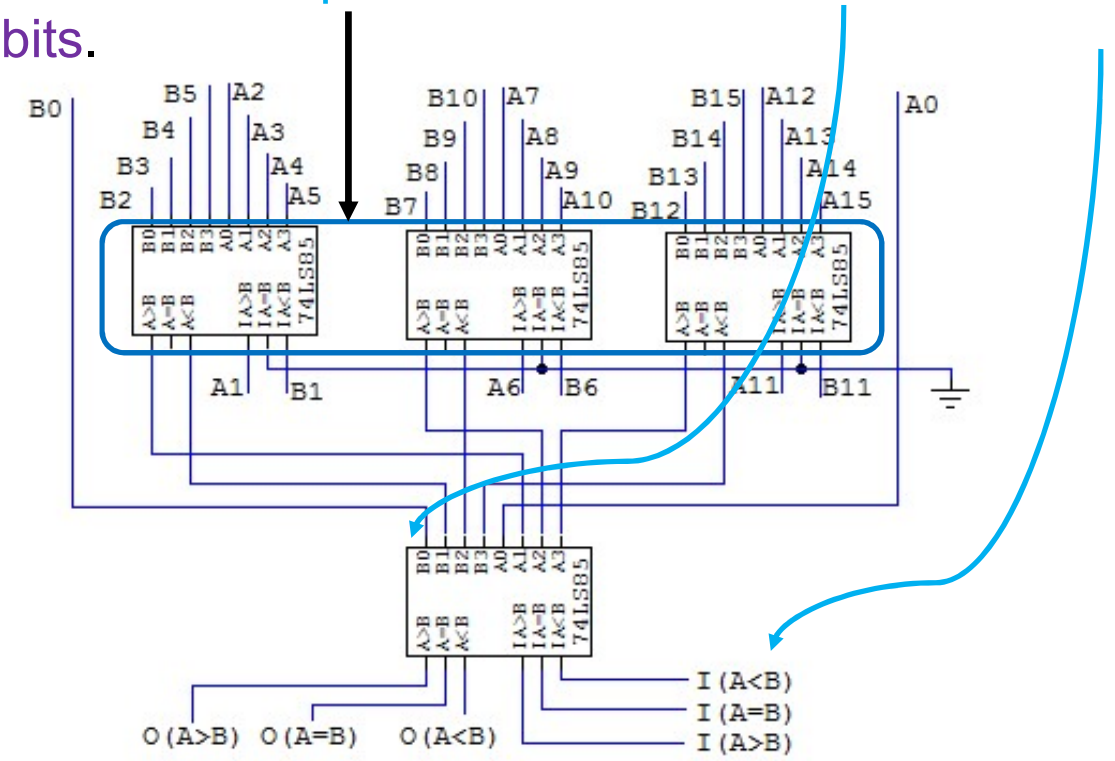
Si A > B => Cout = 0

Z5 = Cout · (A = B) => Usar 1 AND

10.3. Diseñar un circuito comparador COMP16 de números binarios de 16 bits utilizando únicamente cuatro comparadores 74'85 de cuatro bits y un máximo de dos niveles de lógica. El circuito debe tener dos entradas A y B de 16 bits, tres entradas de expansión I(A=B), I(A>B) y I(A<B), y tres salidas O(A=B), O(A>B) y O(A<B), como el circuito 74'85.

Dos niveles de lógica: disposición en paralelo. Las salidas > y < del primer nivel a las entradas A y B del segundo nivel.

Los comparadores del primer nivel pueden comparar 5 bits usando las entradas I> e I<. Uso 3 comparadores de 5 bits + 1 bit + entradas de expansión: 16 bits.



10.3. Diseñar un circuito comparador COMP16 de números binarios de 16 bits utilizando únicamente cuatro comparadores 74'85 de cuatro bits y un máximo de dos niveles de lógica. Indicar razonadamente el número máximo de bits de los números que se pueden comparar mediante un circuito formado por 3 COMP16 y 3 circuitos 74'85, y un máximo de tres niveles de comparadores 74'85 (sabiendo que en los circuitos COMP16 ya hay dos niveles).

Tres niveles de lógica: 5 conexiones en el tercer nivel. Uso 1 comparador de 4 bits 74’85.

5 conexiones:

- 3 COMP16 => 17 bits cada uno (16 de datos + 1 de expansión)
- 1 74’85 => 5 bits (4 de datos + 1 de expansión)
- 1 74’85 => 4 bits + entradas de expansión

Número de bits = 3 * 17 + 5 + 4 = 60 bits

11.1. Se quieren diseñar circuitos digitales que realicen la comparación de dos números binarios con signo X e Y. Se deben obtener tres salidas que indiquen cuando X = Y, X > Y o X < Y. Se debe utilizar en lo posible comparadores comerciales como el 74'85, y otros elementos lógicos cuando sea necesario. Indicar en cada caso el razonamiento o las ecuaciones lógicas que llevan al diseño final.

Recordar: los números positivos siempre son mayores que los negativos; entre números positivos el mayor es el de mayor valor absoluto (5 > 3), entre números negativos el mayor es el de menor valor absoluto (-3 > -5).

a) Suponer X (x3x2x1x0) e Y (y3y2y1y0) de 4 bits en complemento-2.

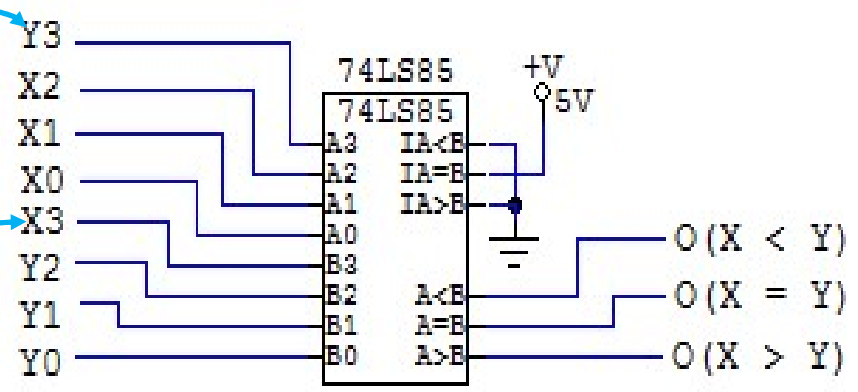
En el circuito 74'85 conecto X a A, e Y a B.

Bits 2-0: pesos (4 2 1) => operan como en un comparador convencional.

Bits 3: peso -8 => Bit a 1 (-8) < Bit a 0 (0). Opera de forma complementada a un comparador convencional.

Opción 1: usar puertas NOT en X3 e Y3 para compensar la complementación.

Opción 2: intercambiar X3 e Y3. No requiere puertas NOT.



b) Suponer X (Sx x3x2x1x0) e Y (Sy y3y2y1y0) de cinco bits en formato con bit de signo, donde Sx e Sy son los signos de X y de Y: 0 positivo, 1 negativo. Los otros bits contienen los módulos Mx y My de cada número en código binario. Para simplificar un poco el problema suponer que existe el +0 pero no existe el -0 (eso evita el problema de evaluar -0 = +0.

Igualdad. X = Y si tienen mismo signo y mismo módulo

$O(X=Y) = (Sx = Sy) \cdot (Mx = My) = \overline{Sx \oplus Sy} \cdot (Mx = My)$

* Si existiese -0 hay que añadir -0 = +0: ((Mx = My) y (Mx = 0))

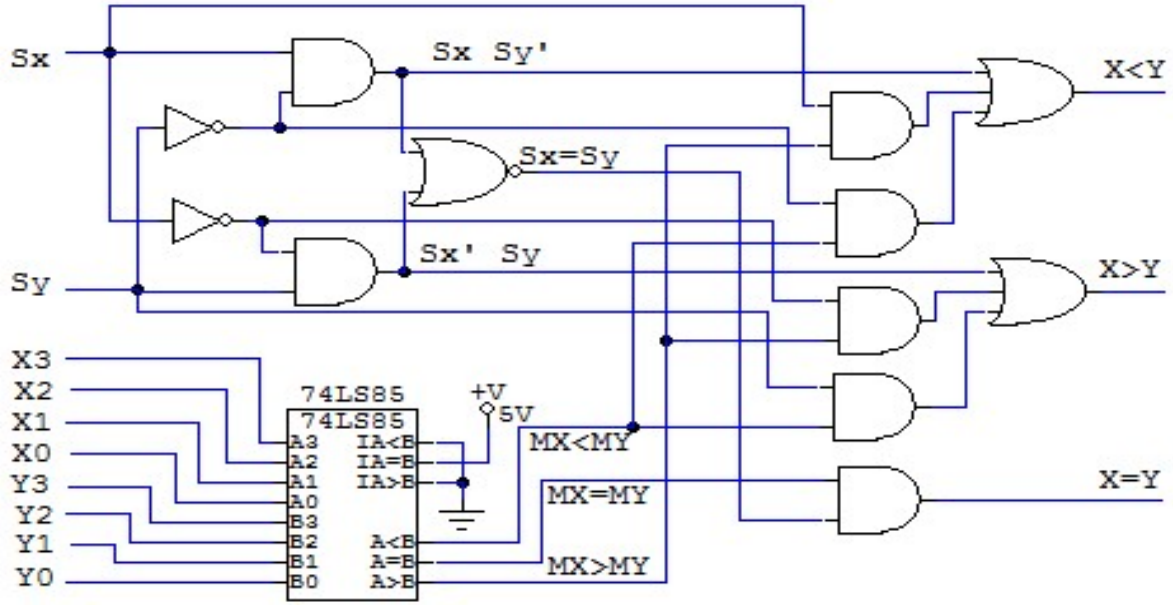
Mayor. X > Y si (X POS y Y NEG) o (X POS y Y POS y Mx > My) o
o (X NEG y Y NEG y Mx < My)

$$
\begin{aligned}
O(X>Y) &= \overline{\boldsymbol{Sx}} \cdot \boldsymbol{Sy} + \overline{\boldsymbol{Sx}} \cdot \overline{\boldsymbol{Sy}} \cdot (\boldsymbol{Mx} > \boldsymbol{My}) + Sx \cdot Sy \cdot (Mx < My) = \\
&= \overline{Sx} \cdot [\boldsymbol{Sy} + \overline{\boldsymbol{Sy}} \cdot (\boldsymbol{Mx} > \boldsymbol{My})] + Sx \cdot Sy \cdot (Mx < My) = \\
&= \overline{\boldsymbol{Sx}} \cdot [\boldsymbol{Sy} + (\boldsymbol{Mx} > \boldsymbol{My})] + Sx \cdot Sy \cdot (Mx < My) = \\
&= \overline{\boldsymbol{Sx}} \cdot \boldsymbol{Sy} + \overline{Sx} \cdot (Mx > My) + \boldsymbol{Sx} \cdot \boldsymbol{Sy} \cdot (\boldsymbol{Mx} < \boldsymbol{My}) = \\
&= \overline{Sx} \cdot (Mx > My) + Sy \cdot [\overline{\boldsymbol{Sx}} + \boldsymbol{Sx} \cdot (\boldsymbol{Mx} < \boldsymbol{My})] = \\
&= \overline{Sx} \cdot (Mx > My) + \boldsymbol{Sy} \cdot [\overline{\boldsymbol{Sx}} + (\boldsymbol{Mx} < \boldsymbol{My})] = \\
&= \overline{Sx} \cdot Sy + \overline{Sx} \cdot (Mx > My) + Sy \cdot (Mx < My)
\end{aligned}
$$

Menor. X < Y si (X NEG y Y POS) o (X POS y Y POS y Mx < My) o
o (X NEG y Y NEG y Mx > My)

$$O(X<Y) = \boldsymbol{Sx} \cdot \overline{\boldsymbol{Sy}} + \overline{\boldsymbol{Sx}} \cdot \overline{\boldsymbol{Sy}} \cdot (\boldsymbol{Mx} < \boldsymbol{My}) + Sx \cdot Sy \cdot (Mx > My) =$$
$$= \overline{Sy} \cdot [\boldsymbol{Sx} + \overline{\boldsymbol{Sx}} \cdot (\boldsymbol{Mx} < \boldsymbol{My})] + Sx \cdot Sy \cdot (Mx > My) =$$
$$= \overline{\boldsymbol{Sy}} \cdot [\boldsymbol{Sx} + (\boldsymbol{Mx} < \boldsymbol{My})] + Sx \cdot Sy \cdot (Mx > My) =$$
$$= \boldsymbol{Sx} \cdot \overline{\boldsymbol{Sy}} + \overline{Sy} \cdot (Mx < My) + \boldsymbol{Sx} \cdot \boldsymbol{Sy} \cdot (\boldsymbol{Mx} > \boldsymbol{My}) =$$
$$= \overline{Sy} \cdot (Mx < My) + Sx \cdot [\overline{\boldsymbol{Sy}} + \boldsymbol{Sy} \cdot (\boldsymbol{Mx} > \boldsymbol{My})] =$$
$$= \overline{Sy} \cdot (Mx < My) + \boldsymbol{Sx} \cdot [\overline{\boldsymbol{Sy}} + (\boldsymbol{Mx} > \boldsymbol{My})] =$$
$$= Sx \cdot \overline{Sy} + Sx \cdot (Mx > My) + \overline{Sy} \cdot (Mx < My)$$

(Mx = My), (Mx > My), (Mx < My) son las salidas =, >, < del 74’85



11.2. Indicar como se puede hacer un comparador de dos palabras de 4 bits X e Y que codifican números binarios sin signo cuando las entradas de datos están en lógica negativa, utilizando el comparador 74’85.

Señales X e Y (.L) se conectan a A y B (.H) del comparador 74’85

1ª opción: usar 8 puertas NOT para pasar X e Y a .H

Xi = Yi: (L = L), (H = H) son iguales .L o .H.

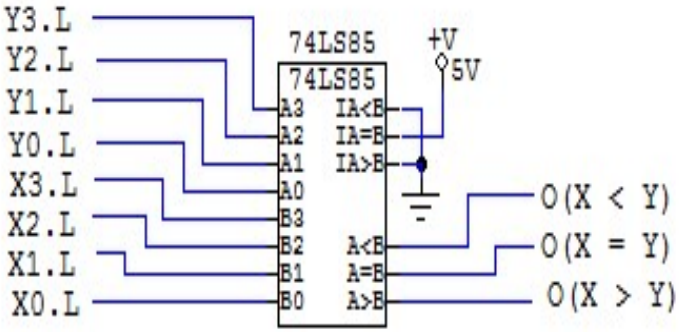
Xi > Yi. Xi a 1 (L) e Yi a 0 (H) => Ai a L (0) y Bi a H (1); Ai < Bi

Xi < Yi. Xi a 0 (H) e Yi a 1 (L) => Ai a H (1) y Bi a L (0); Ai > Bi

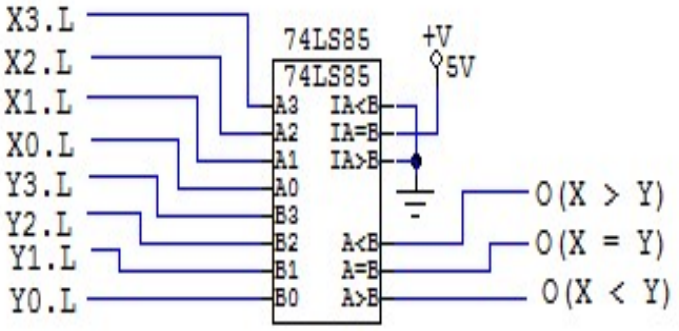
2ª opción: intercambiar X a B, e Y a A en el comparador 74’85.
Sin puertas NOT.

3ª opción: intercambiar las salidas > y < en el comparador 74’85.
Sin puertas NOT.

2ª opción



3ª opción



12.1. Dos tanques de agua A y B tienen cada uno de ellos 8 sensores (entradas A7-A0, B7-B0, el índice 0 es el nivel inferior, el índice 7 el nivel superior) que indican el nivel de agua de cada tanque. El sensor está activado cuando hay agua en el nivel y desactivado en caso contrario. Los tanques tienen unas esclusas que permiten introducir agua (salidas AI, BI) o sacar agua (salidas AO, BO).

Diseñar un circuito para automáticamente sacar agua del tanque (activar AO o BO) que más tiene cuando la suma de los niveles de los tanques pase de 11, o cargar agua en el tanque (activar AI o BI) que menos tiene cuando la suma de los niveles de los tanques sea menor de 5. Usar elementos MSI estándar y/o puertas lógicas.

Calcular el tiempo de propagación máximo del circuito.

I7	I6	I5	I4	I3	I2	I1	I0	SA2	SA1	SA0
1	1	1	1	1	1	1	1	1	1	1
0	1	1	1	1	1	1	1	1	1	0
0	0	1	1	1	1	1	1	1	0	1
0	0	0	1	1	1	1	1	1	0	0
0	0	0	0	1	1	1	1	0	1	1
0	0	0	0	0	1	1	1	0	1	0
0	0	0	0	0	0	1	1	0	0	1
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0	0	0	0

Valores posibles en los sensores del tanque

Nivel del tanque en binario

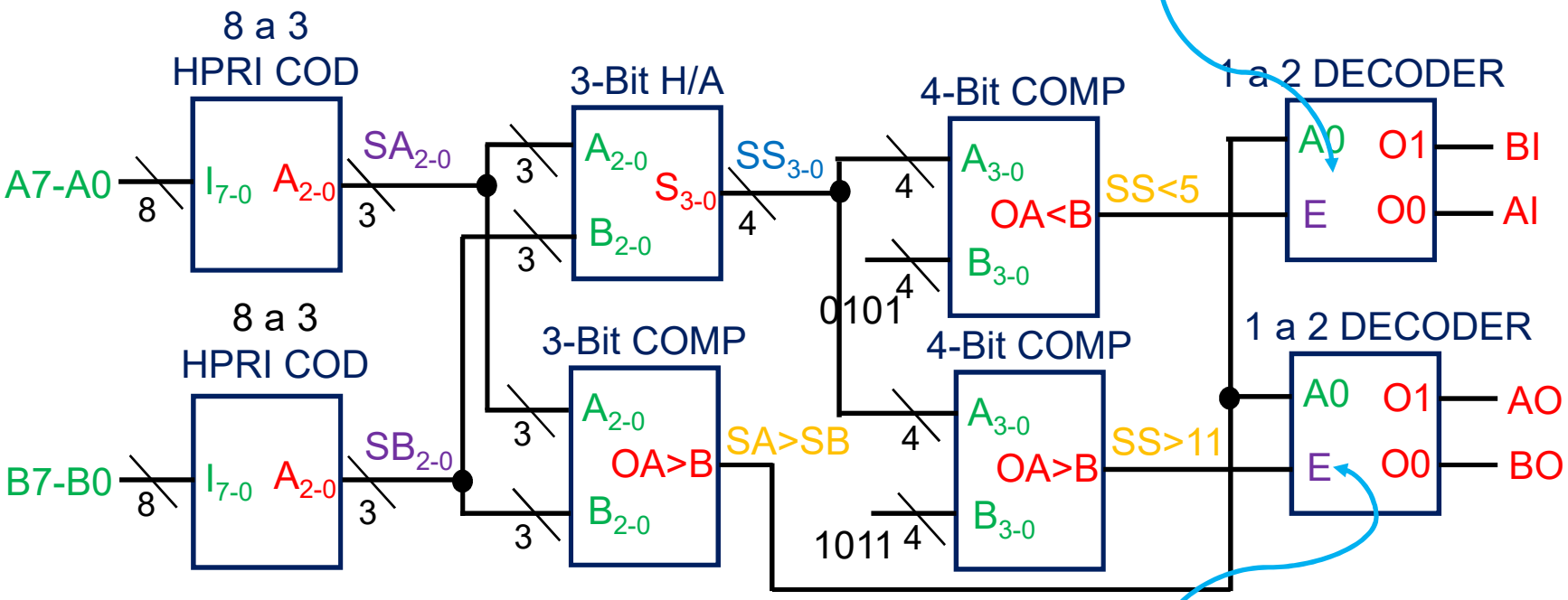
El tanque real estaría en vertical

Un HPRI COD 8 a 3 resuelve esta tabla

El circuito necesita los siguientes circuitos MSI:

- 2 HPRI COD 8 a 3 para A y B. Generan SA y SB de 3 bits ∈ [0, 7]
- 1 Sumador de 3 bits SA PLUS SB = SS de 4 bits ∈ [0, 14]
- 1 Comparador de 3 bits: SA > SB y/o SA < SB
- 2 Comparadores de 4 bits: SS > 11, SS < 5
  (o con puertas: (SS > 11) = SS3 SS2; (SS < 5) = $\overline{SS3}\ \overline{SS2} + \overline{SS3}\ \overline{SS1}\ \overline{SS0}$)
- 2 DECs 1 a 2 con habilitador: uno para AI, BI y otro para AO, BO

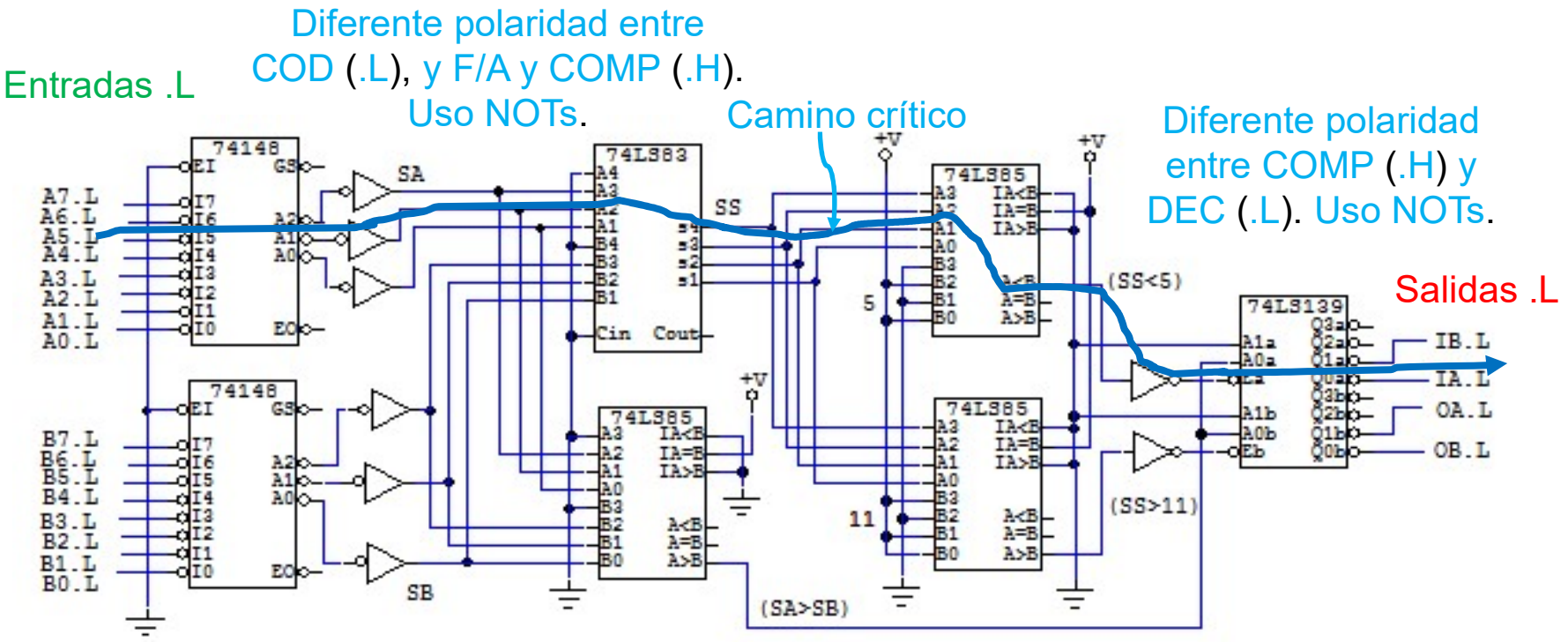
Si SS ≥ 5 (E = 0), AI = BI = 0;
Si SS < 5 (E = 1) => Si SA > SB (A0 = 1), BI (O1) = 1, si no (A0 = 0) AI (O0) = 1



Si SS ≤ 11 (E = 0), AO = BO= 0;
Si SS > 11 (E = 1) => Si SA > SB (A0 = 1), AO (O1) = 1, si no (A0 = 0), BO (O0) = 1

Utilizo circuitos de la familia 74:

- 2 HPRI COD 8 a 3 74LS148
- 1 Sumador de 4 bits 74LS83 con Cin = A4 = B4 = 0
- 1 Comparador de 3 bits 74LS85 con A4 = B4 = 0
- 2 Comparador de 4 bits 74LS85
- 2 DECs 1 a 2 con habilitador 74LS139 (DEC 2 a 4, con A1 a 0)

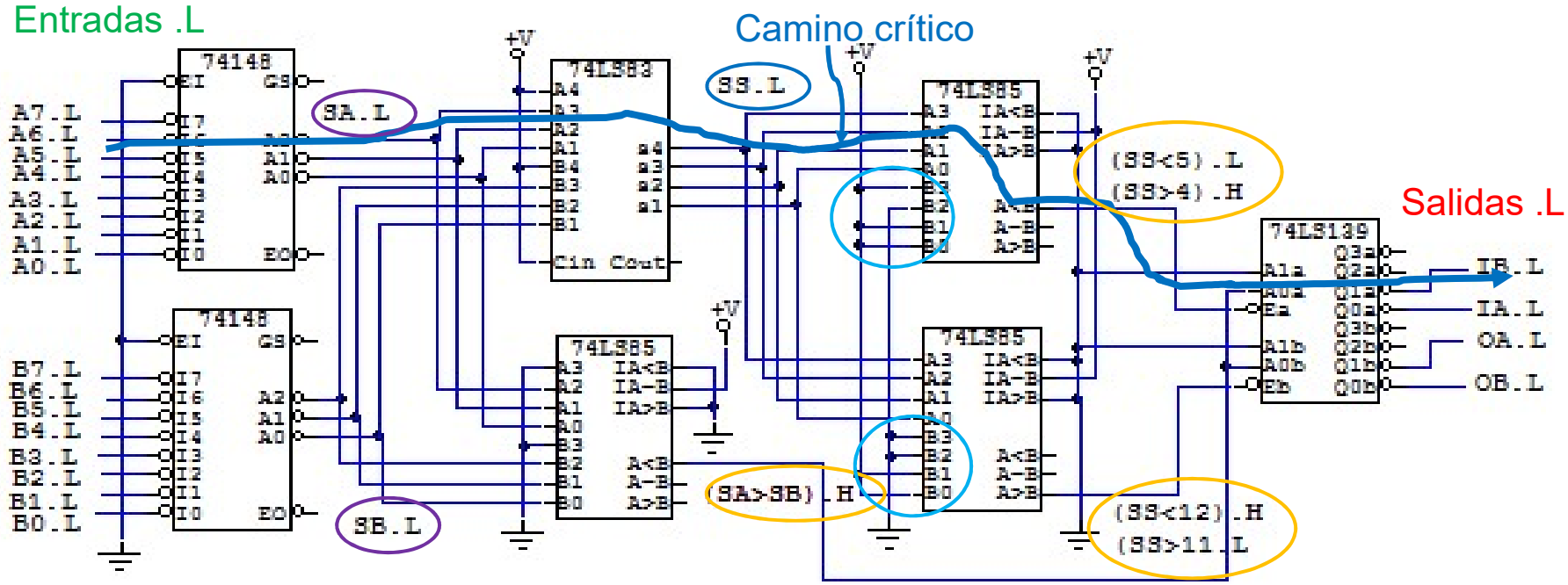


$Tp = Tp_{148}(I\text{-}A) + Tp_{04}(NOT) + Tp_{83}(A\text{-}S) + Tp_{85}(A\text{-}<) + Tp_{04}(NOT) + Tp_{139}(E\text{-}O)$

$Tp = 36\ ns + 15\ ns + 24\ ns + 36\ ns + 15\ ns + 32\ ns = 158\ ns$

Optimización del circuito (elimino las puertas NOT):

- Sumador 74LS83 opera en .L. Elimino NOTs. Cin = A4 = B4 = 0 (H)
- Comparadores 74LS85: entradas .L => intercambio salidas > y <
- Eb.L = (SS > 11).L =($\overline{SS \leq 11}$). L = ($\overline{SS < 12}$). L = (SS < 12).H. Elimino NOT
  Ea.L = (SS < 5).L = ($\overline{SS \geq 5}$). L = ($\overline{SS > 4}$). L = (SS > 4).H . Elimino NOT
  (12).L = (LLHH); (4).L = (HLHH)



$Tp = Tp_{148}(I\text{-}A) + Tp_{83}(A\text{-}S) + Tp_{85}(A\text{-}<) + Tp_{139}(E\text{-}O)$

Tp = 36 ns + 24 ns + 36 ns + 32 ns = 128 ns

12.2. Diseñar un circuito que dado como entrada un número binario positivo A de 3 bits realice la operación Z = 3*A si la entrada de control C está a 0 o la operación Z = 5*A si la entrada de control C está a 1, usando 1 sumador completo 74LS83 y el menor número de puertas lógicas u otros circuitos MSI (74LS157).

Calcular el tiempo de propagación máximo del circuito.

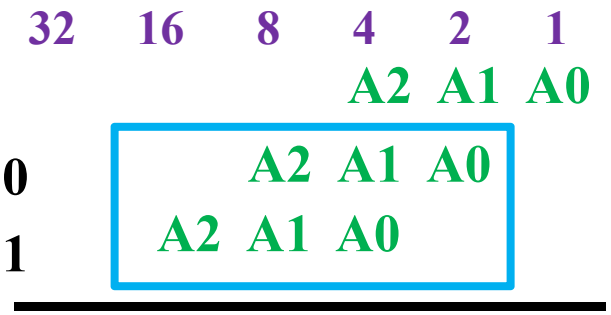
Amax = 7 (3 bits) => Zmax = 5 * 7 = 35 => Z es de 6 bits

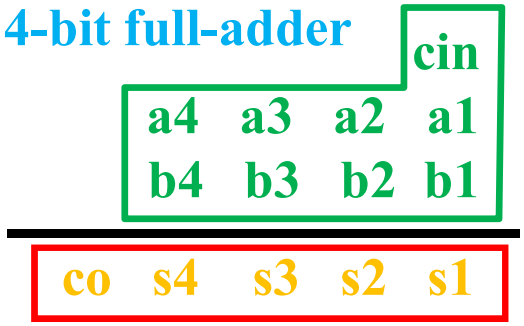
C = 0 => $Z = 3 * A = 2A + A = 2^1 * A + A = (A0) + A$

C = 1 => $Z = 5 * A = 4A + A = 2^2 * A + A = (A00) + A$

$2^N X = X\underbrace{0\ldots0}_{N}$

Defino M = $\bar{C}$ A + C (A0) como en un 2-Inp MUX





12.2. Diseñar un circuito que dado como entrada un número binario positivo A de 3 bits realice la operación Z = 3*A si la entrada de control C está a 0 o la operación Z = 5*A si la entrada de control C está a 1, usando 1 sumador completo 74LS83 y el menor número de puertas lógicas u otros circuitos MSI (74LS157).

Calcular el tiempo de propagación máximo del circuito.

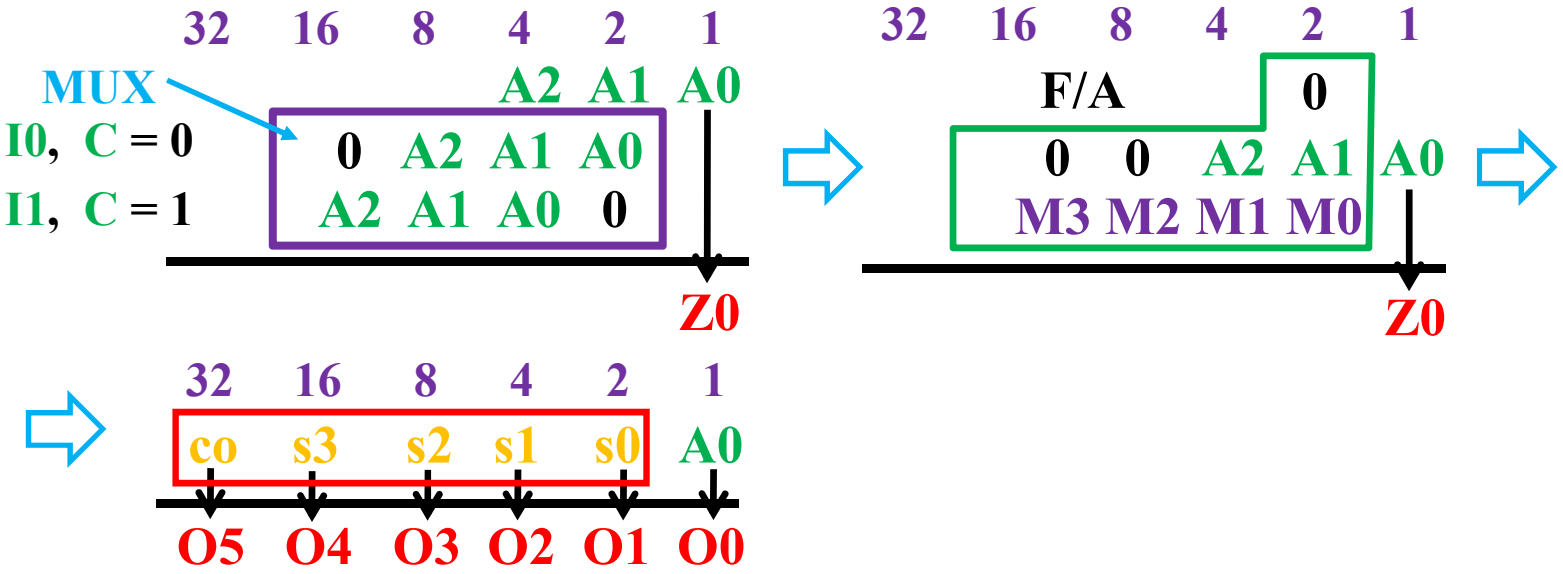
Amax = 7 (3 bits) => Zmax = 5 * 7 = 35 => Z es de 6 bits

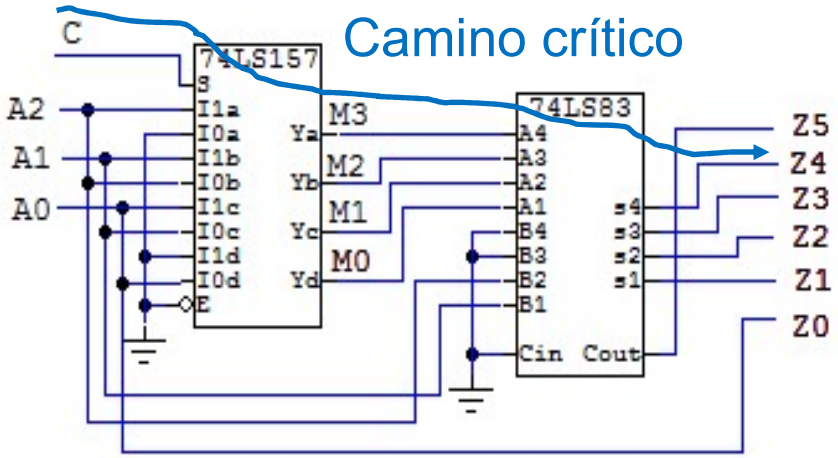
C = 0 => $Z = 3 * A = 2A + A = 2^1 * A + A = (A0) + A$

C = 1 => $Z = 5 * A = 4A + A = 2^2 * A + A = (A00) + A$

$2^N X = X\underbrace{0\ldots.0}_{N}$

Defino $M = \bar{C} A + C (A0)$ como en un 2-Inp MUX





74LS157: cuatro 2-Inp MUXs
I1 = (A2 A1 A0 0)
I0 = (0 A2 A1 A0)

74LS83: 4-Inp full-adder

AC CHARACTERISTICS (T_A = 25°C)

Symbol	Parameter	Limits Min	Limits Typ	Limits Max	Unit	Test Conditions	
t_{PLH} t_{PHL}	Propagation Delay Data to Output		9.0 9.0	14 14	ns	Figure 2	V_{CC} = 5.0 V C_L = 15 pF
t_{PLH} t_{PHL}	Propagation Delay Enable to Output		13 14	20 21	ns	Figure 1	
t_{PLH} t_{PHL}	Propagation Delay Select to Output		15 18	23 27	ns	Figure 2	

Tpmax = Tpmax(MUX) + Tpmax(F/A) =
= Tpmux(S-Z) + Tpf/a(A-S) =
= 27ns + 24ns =
= 51ns

AC CHARACTERISTICS (TA = 25°C)

Symbol	Parameter	Limits Min	Limits Typ	Limits Max	Unit
tPLH tPHL	Propagation Delay, C0 Input to any Σ Output		16 15	24 24	ns
tPLH tPHL	Propagation Delay, Any A or B Input to Σ Outputs		15 15	24 24	ns
tPLH tPHL	Propagation Delay, C0 Input to C4 Output		11 15	17 22	ns
tPLH tPHL	Propagation Delay, Any A or B Input to C4 Output		11 12	17 17	ns

13.1. Realizar la operación X^2 (X*X), siendo X un número binario sin signo de tres bits (X2X1X0). Utilizar el menor número posible de puertas AND, y semisumadores y sumadores completos de 1 bit. Para simplificar el circuito se recuerda que se puede utilizar la propiedad conmutativa, el teorema de la idempotencia, y que A PLUS A = 2*A = (A0).

$X \in [0, 7]$; $Z = X^2$; $Zmax = 7^2 = 49$ (6 bits)

* X^2 podría resolverse con mapas de Karnaugh para un problema de 3 entradas y 6 salidas.

Xij = Xi ● Xj
9 puertas AND

32	16	8	4	2	1
			X2	X1	X0
			X2	X1	X0
			X02	X01	X00
		X12	X11	X10	
	X22	X21	X20		

Simplificaciones

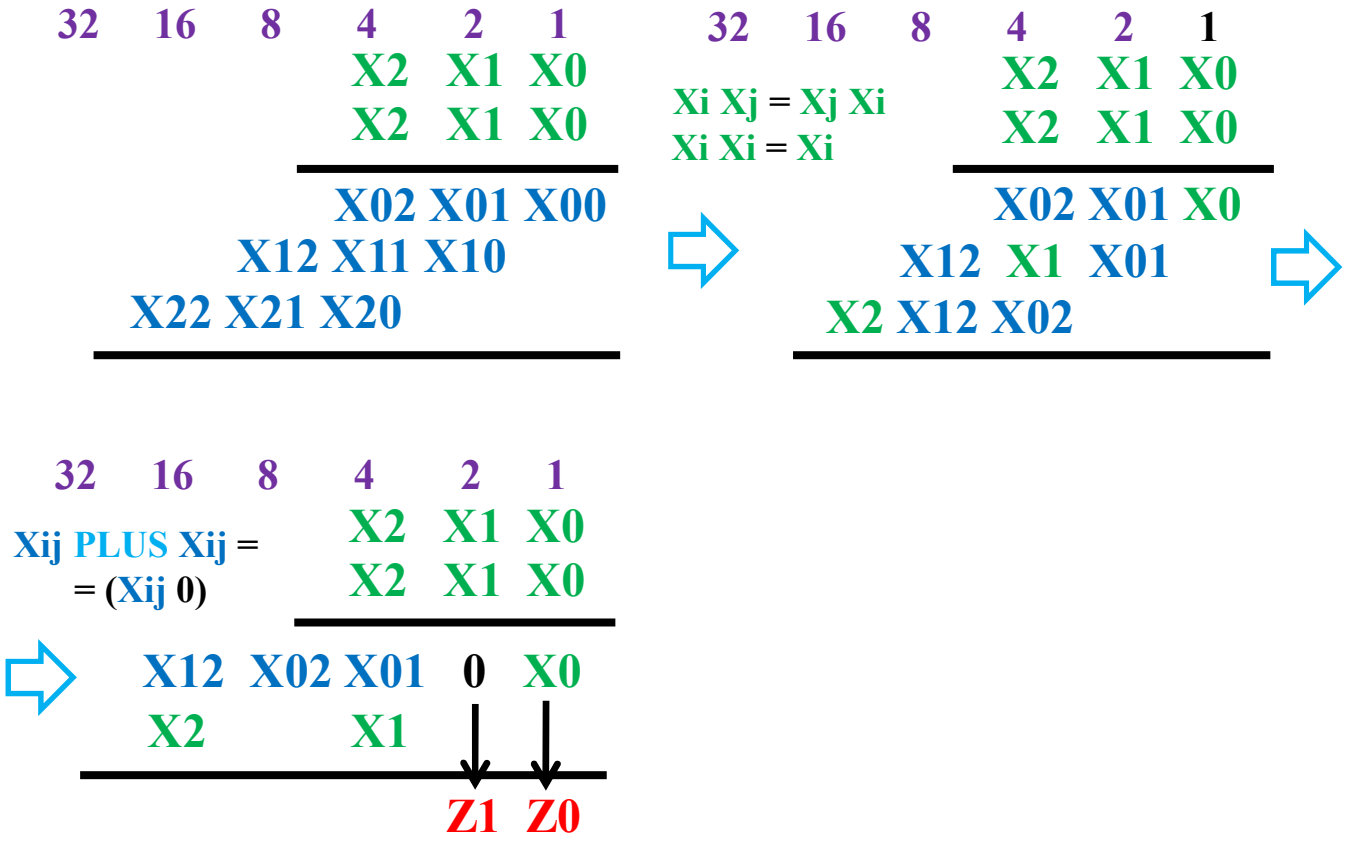
Conmutativa: Xij = Xji (Xi ● Xj = Xi ● Xj = Xij).

Idempotencia: Xii = Xi (Xi ● Xi = Xi).

Aritmética: A PLUS A = 2 * A = 2^1 * A = (A0)

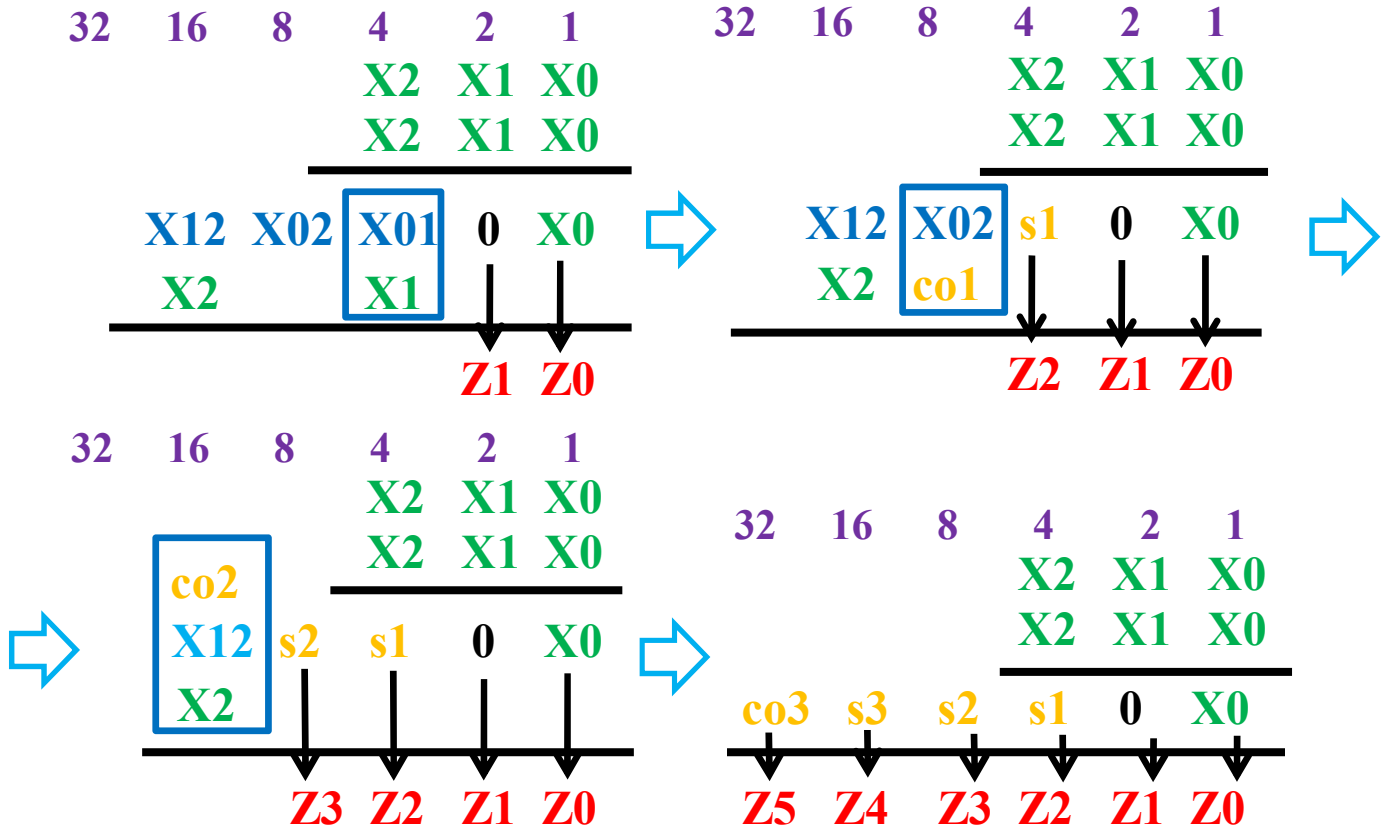
] 3 puertas AND

13.1. Realizar la operación X^2 (X*X), siendo X un número binario sin signo de tres bits (X2X1X0). Utilizar el menor número posible de puertas AND, y semisumadores y sumadores completos de 1 bit. Para simplificar el circuito se recuerda que se puede utilizar la propiedad conmutativa, el teorema de la idempotencia, y que A PLUS A = 2*A = (A0).

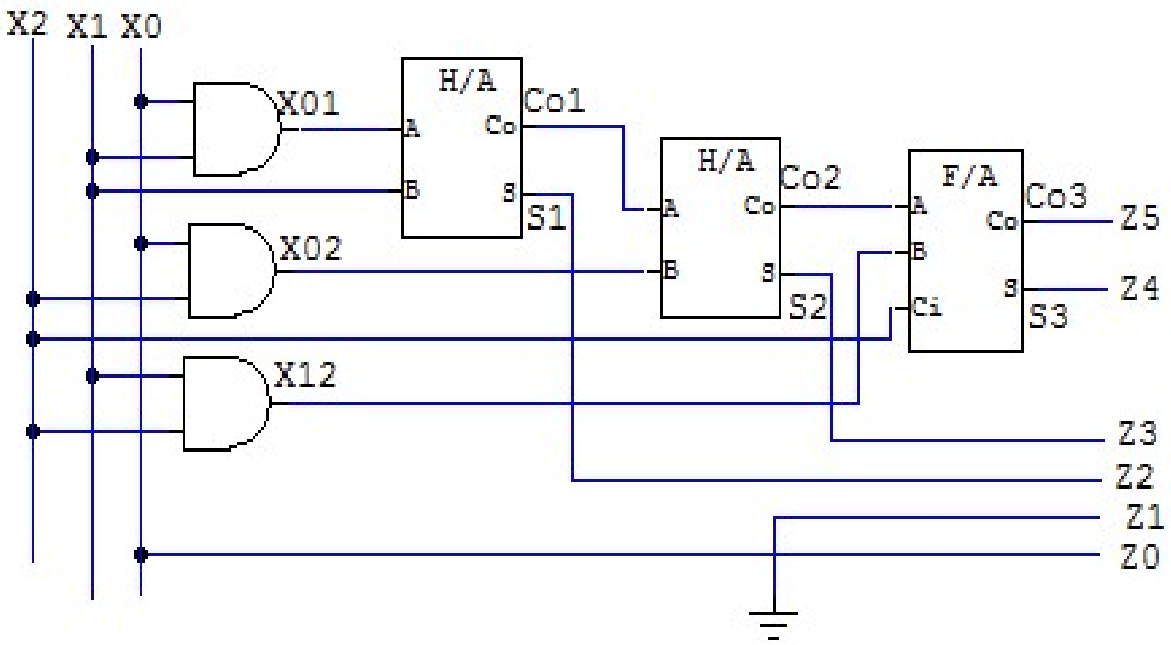


13.1. Realizar la operación X^2 (X*X), siendo X un número binario sin signo de tres bits (X2X1X0). Utilizar el menor número posible de puertas AND, y semisumadores y sumadores completos de 1 bit. Para simplificar el circuito se recuerda que se puede utilizar la propiedad conmutativa, el teorema de la idempotencia, y que A PLUS A = 2*A = (A0).

Se realizan las sumas con dos semisumadores y un sumador completo



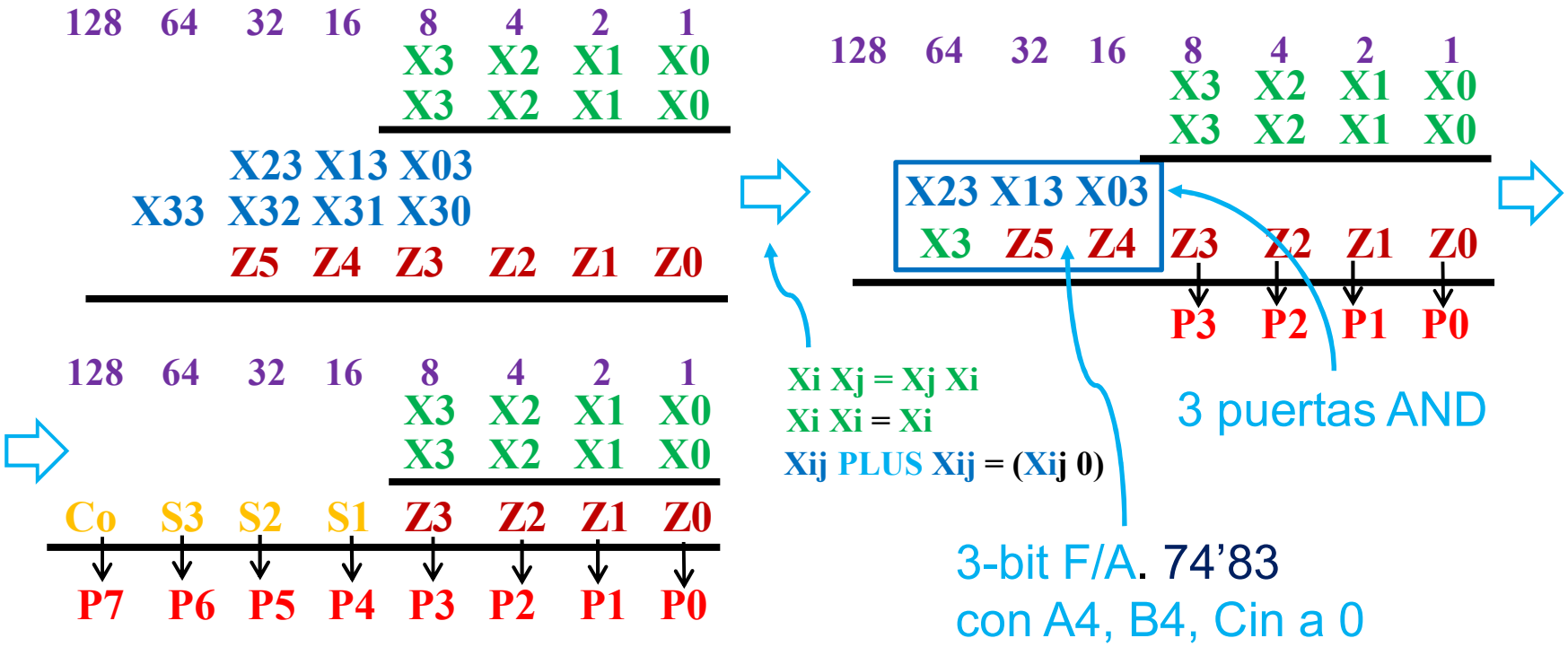
13.1. Realizar la operación X^2 (X*X), siendo X un número binario sin signo de tres bits (X2X1X0). Utilizar el menor número posible de puertas AND, y semisumadores y sumadores completos de 1 bit. Para simplificar el circuito se recuerda que se puede utilizar la propiedad conmutativa, el teorema de la idempotencia, y que A PLUS A = 2*A = (A0).



13.1. Realizar la operación X^2 (X*X), siendo X un número binario sin signo de tres bits (X2X1X0).

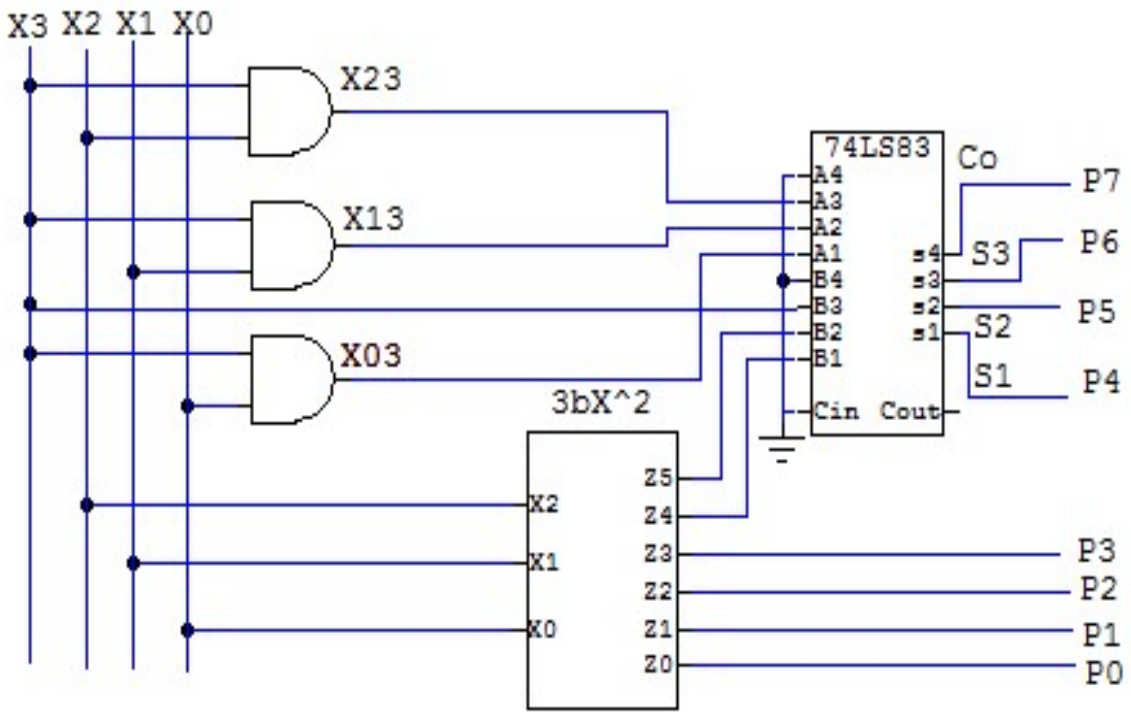
Diseñar un circuito que realice la operación X^2, siendo X de 4 bits (X3X2X1X0) usando el circuito X^2 para X de 3 bits diseñado anteriormente, el menor número de puertas AND y un único sumador 74LS83 (4-bit full-adder).

$X \in [0, 15]$; $P = X^2$; $P_{max} = 15^2 = 225$ (8 bits)



13.1. Realizar la operación X^2 (X*X), siendo X un número binario sin signo de tres bits (X2X1X0).

Diseñar un circuito que realice la operación X^2, siendo X de 4 bits (X3X2X1X0) usando el circuito X^2 para X de 3 bits diseñado anteriormente, el menor número de puertas AND y un único sumador 74LS83 (4-bit full-adder).



13.2. Se disponen de dos palabras de 3 bits A (Sa a1a0) y B (Sb b1b0) que representan números con signo en notación de bit de signo. Se quiere obtener la suma de A y B pero mostrando el resultado en complemento-2. La suma se hará convirtiendo los números A y B a complemento-2 usando decodificadores 74LS138 y puertas NAND de 4 entradas, y luego sumando los números convertidos usando un sumador completo de 4 bits como el 74LS83. Mostrar la implementación del circuito.

A, B ∈ [-3, +3]; SC = A + B ∈ [-6, +6] en c-a-2 (4 bits)

	+ (0) / - (1)	2	1		-4	2	1
N	**Sx**	**X1**	**X0**	**D**	**Y2**	**Y1**	**Y0**
0	0	0	0	0	0	0	0
1	0	0	1	1	0	0	1
2	0	1	0	2	0	1	0
3	0	1	1	3	0	1	1
-0	1	0	0	4	0	0	0
-1	1	0	1	5	1	1	1
-2	1	1	0	6	1	1	0
-3	1	1	1	7	1	0	1

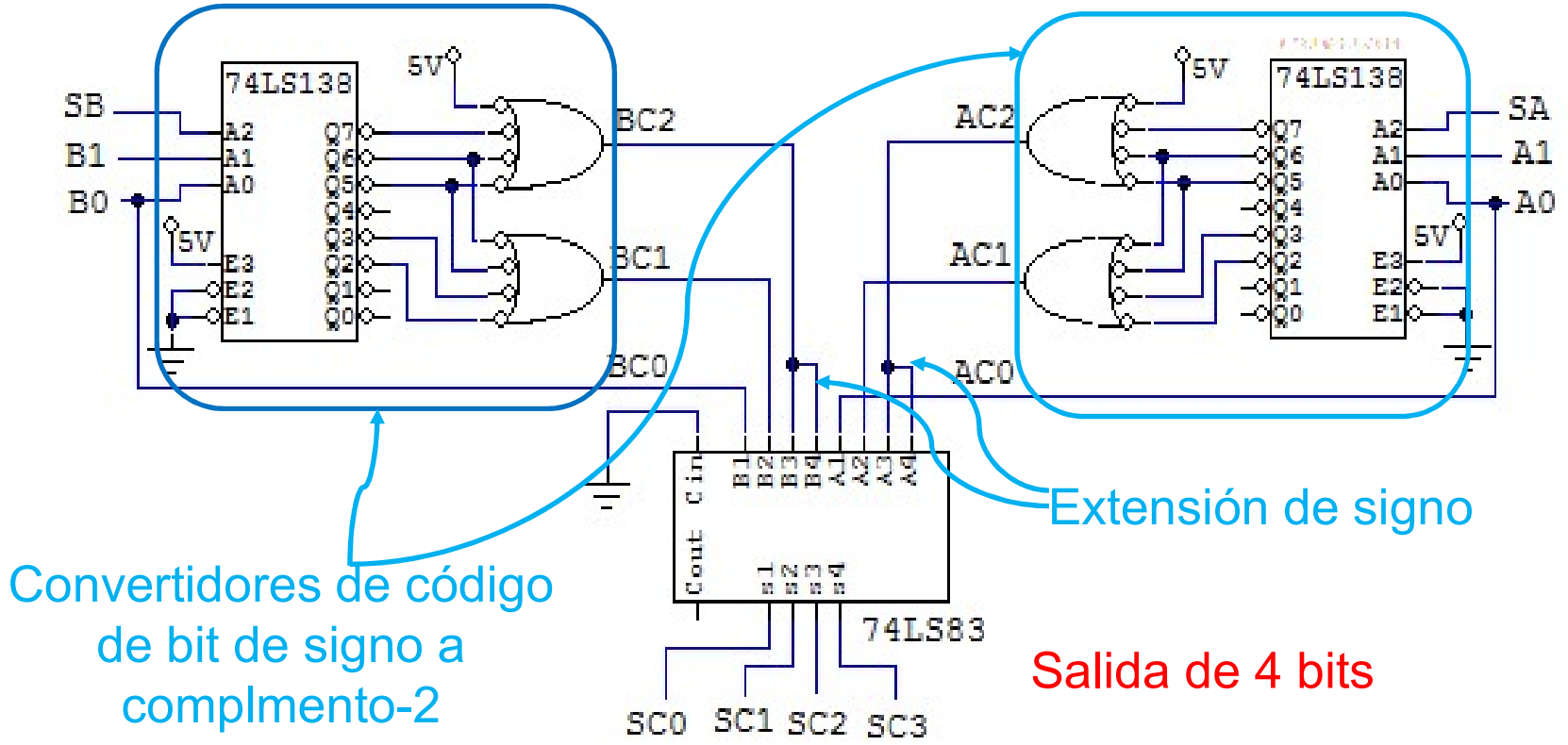
Se suman A y B en complemento-2 de 3 bits con un 74LS83 extendiendo el bit de signo

Conversión de bit de signo a complemento-2

$Y2 = \sum (5, 6, 7)$
$Y1 = \sum (2, 3, 5, 6)$
$Y0 = \sum (1, 3, 5, 7) = X0$

Se implementa con 1 DEC 74LS138 y 2 puertas NAND por entrada A y B

13.2. Se disponen de dos palabras de 3 bits A (Sa a1a0) y B (Sb b1b0) que representan números con signo en notación de bit de signo. Se quiere obtener la suma de A y B pero mostrando el resultado en complemento-2. La suma se hará convirtiendo los números A y B a complemento-2 usando decodificadores 74LS138 y puertas NAND de 4 entradas, y luego sumando los números convertidos usando un sumador completo de 4 bits como el 74LS83. Mostrar la implementación del circuito.



14.1. Diseñar un circuito sumador para números binarios A y B de cinco bits descritos en código binario con signo, de la forma (Sa a3a2a1a0) y (Sb b3b2b1b0), donde S es el bit de signo (0 positivo, 1 negativo), y (a3-a0), (b3-b0) la codificación binaria del módulo de los números. El circuito tiene que generar un número de seis bits como resultado (Sf f4f3f2f1f0, f4-f0 bits de módulo y Sf de signo).
Utilizar como base del diseño un único circuito sumador/restador (S/R) de 4 bits, y otros circuitos lógicos MSI (comparadores, multiplexores, etc) y puertas lógicas. El circuito S/R utiliza dos operandos X e Y de cuatro bits, una señal de control C del tipo de operación (X PLUS Y, X MINUS Y), una salida Z de 4 bits y un bit de salida de acarreo Co. Los operandos X e Y son números binarios positivos, con la restricción de que en el modo resta X debe ser mayor o igual que Y.
Explicar en qué se basa el diseño realizado y representar el diseño en modo esquemático con notación de tipo bus para simplificar el dibujo final.
Generar un circuito sumador/restador, añadiendo una señal de control K y sólo una puerta lógica al circuito anterior.

Operaciones:

Sa = Sb => Suma: MF = MA PLUS MB; Sf = Sa (o Sb)

Sa ≠ Sb => Resta: MA ≥ MB; MF = MA MINUS MB; Sf = Sa

MA < MB; MF = MB MINUS MA; Sf = Sb

Señales de control:

SC = (Sa ≠ Sb) = Sa ⊕ Sb

MM = (MA < MB) -- Salida < de un comparador de 4 bits

S/R

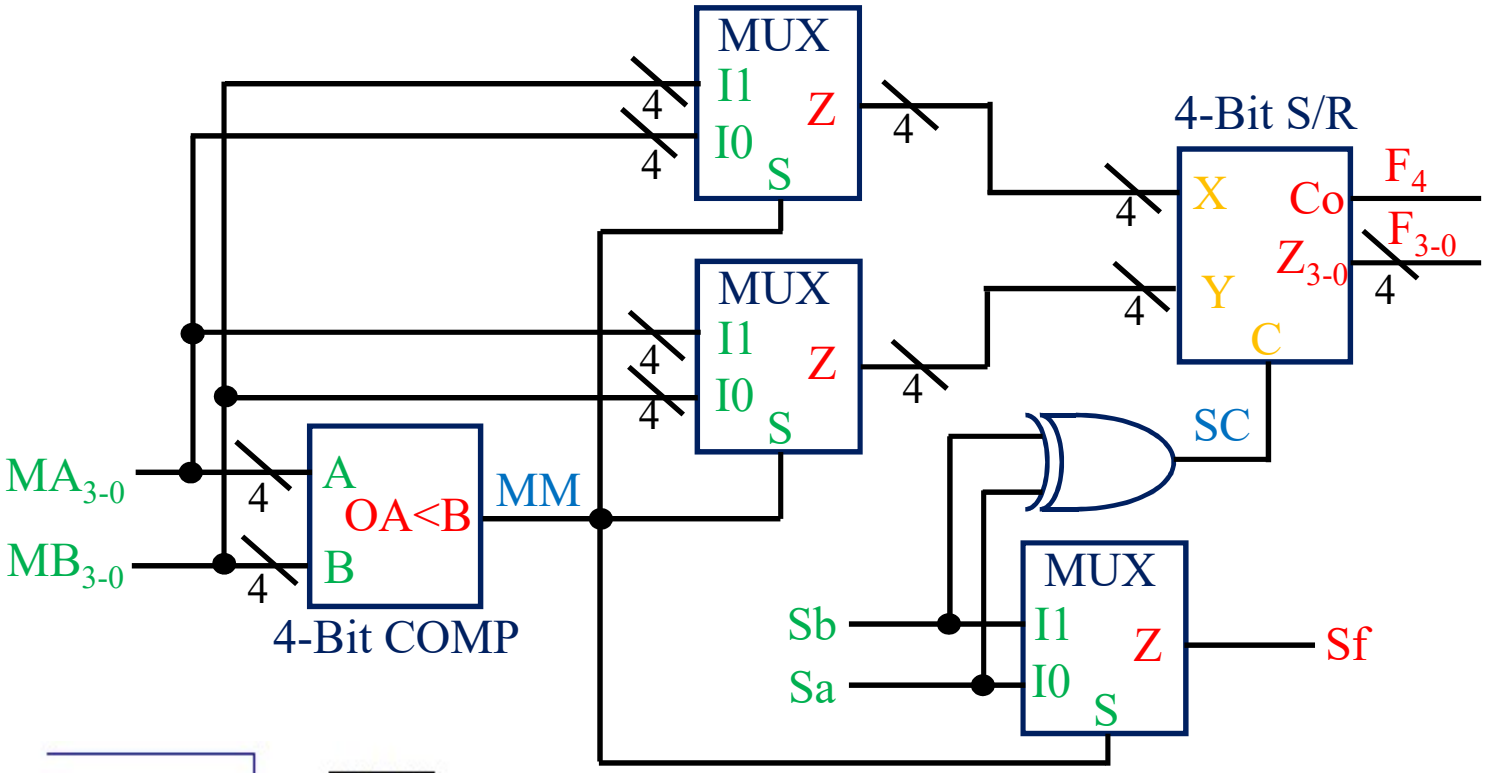
SC	MM	Sf	C	X	Y	Op.	
0	0	Sa	0 (+)	MA	MB	MA PLUS MB	A PLUS B
0	1	Sb	0 (+)	MB	MA	MB PLUS MA	
1	0	Sa	1 (–)	MA	MB	MA MINUS MB	A MINUS B
1	1	Sb	1 (–)	MB	MA	MB MINUS MA	B MINUS A

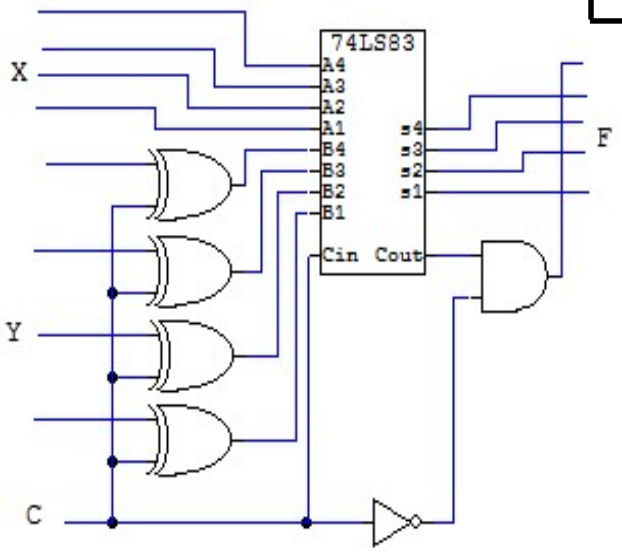
Sf = $\overline{MM}$ Sa + MM Sb => 2-Inp MUX (S => MM, I0 => Sa, I1 => Sb)

C = SC

X = $\overline{MM}$ MA + MM MB => cuatro 2-Inp MUX (S => MM, I0 => MA, I1 => MB)

Y = $\overline{MM}$ MB + MM MA => cuatro 2-Inp MUX (S => MM, I0 => MB, I1 => MA)



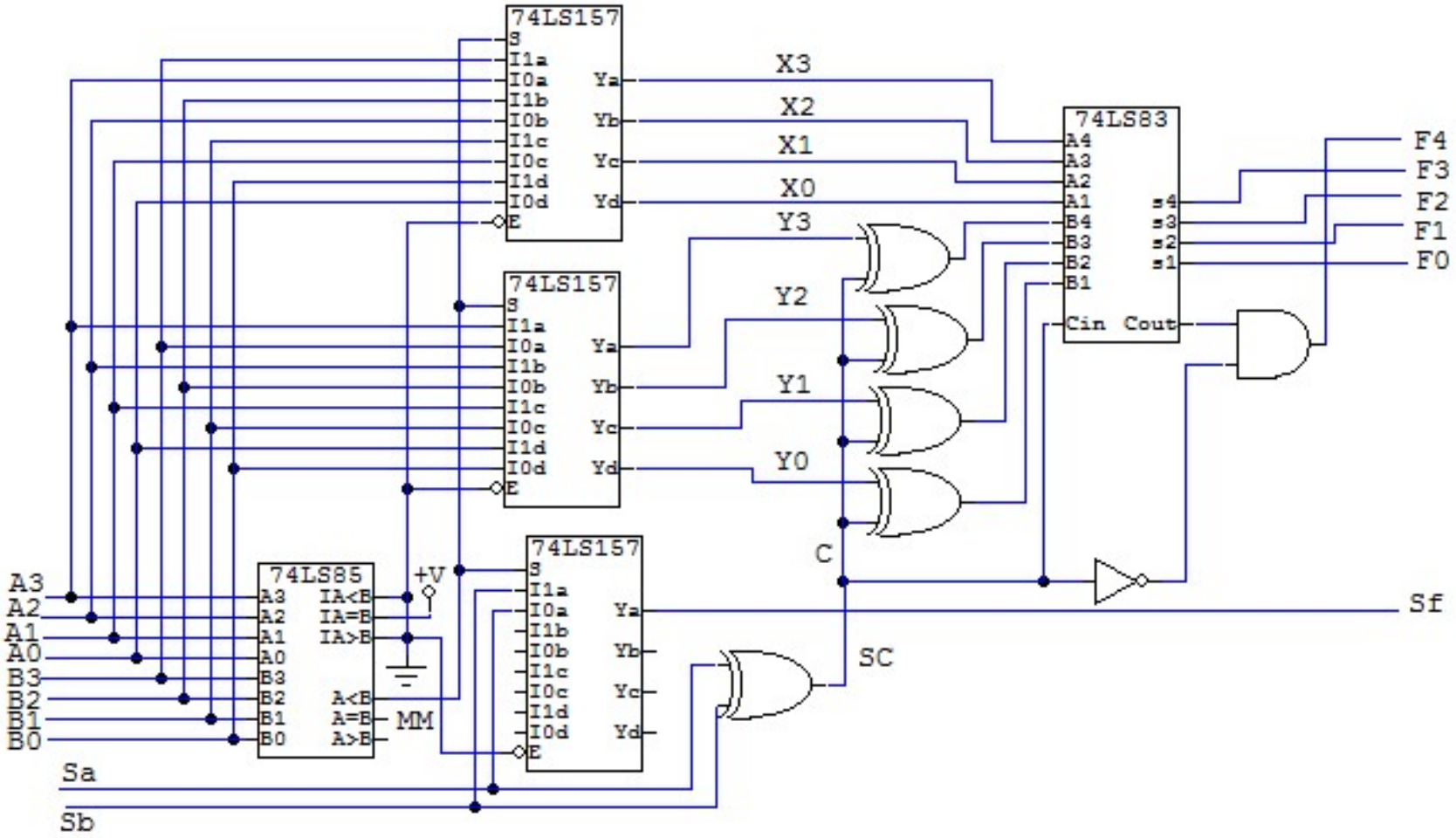


4-bit Sumador/Restador

Opera bien en 4 bits para X, Y en c-a-2, o números sin signo en suma y, si X ≥ Y, en resta.

F4 es 0 en la resta y es Cout en la suma

(C = 0) => F4 = $\overline{C} \cdot Cout$.



Generar un circuito sumador/restador, añadiendo una señal de control K y sólo una puerta lógica al circuito anterior.

A – B = A + (-B) => complementar signo de B.

Si K = 0 (+), SBI = SB; si K = 1 (-) SBI = $\overline{SB}$ => SBI = $K \oplus Sb$

Hay que añadir una XOR y sustituir SB por SBI en la figura.

15.1. La multiplicación de dos números binarios sin signo A y B puede hacerse utilizando puertas AND para generar los términos producto parciales (Xij = ai • bj), para sumar posteriormente estos términos mediante diferentes algoritmos (por filas, por columnas, etc). Cuando se trabaja con números con signo en complemento-2, el proceso se complica ya que los Xij pueden tener peso positivo o negativo en el resultado final por lo que deben sumarse o restarse. En este problema se debe realizar la multiplicación de dos números de tres bits A (a2 a1 a0) y B (b2 b1 b0) en notación en complemento-2, cuyos pesos de más significativo a menos significativo son (-4, 2, 1).

¿Cuántos bits debe tener el resultado P (también en complemento-2) de la multiplicación?

Teniendo en cuenta que

$$P = A \cdot B = 16\,X22 - 8(X21 + X12) - 4(X20 + X02) + 4X11 + 2(X01 + X10) + X00.$$

y ordenando de forma adecuada las operaciones de suma y resta a realizar, diseñar un circuito que realice la multiplicación propuesta utilizando el menor número de sumadores (completos o semisumadores, indicando el número de bits de cada sumador) y puertas lógicas básicas.

A, B de 3 bits en c-a-2 => A, B ∈ [-4, +3]

Pmax = (-4) * (-4) = +16; Pmin = (-4) * 3 = -12 => P es de 6 bits

+16 en c-a-2 => (010000) de pesos (-32 16 8 4 2 1)

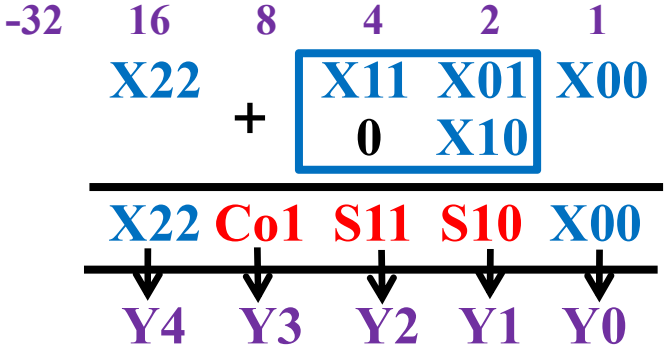
Xij = Ai * Bj (AND)

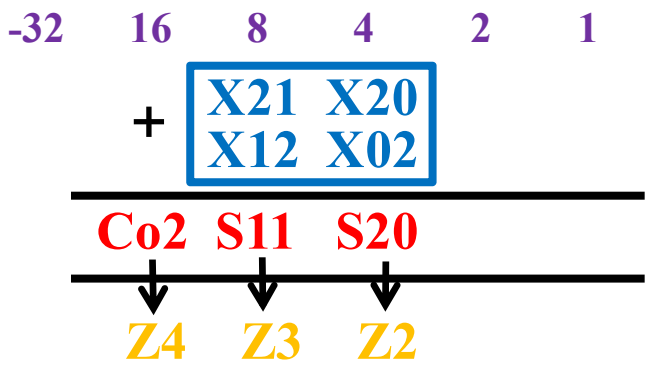
P = A • B = (-4 A2 + 2 A1 + A0) * (-4 B2 + 2 B1 + B0) =

= 16 X22 – 8 X21 – 4 X20 – 8 X12 + 4 X11 + 2 X10 – 4 X02 + 2 X01 + X00

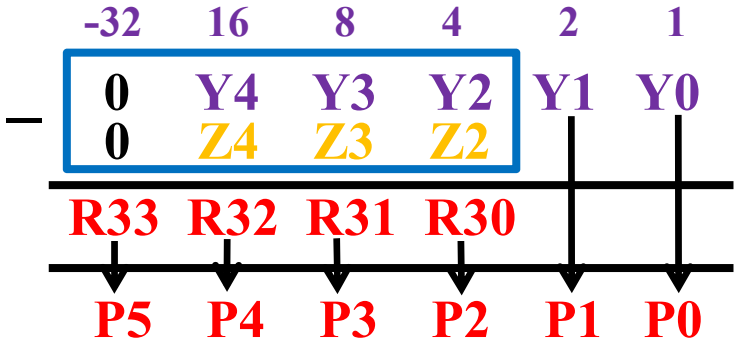
= 16 X22 + 4 X11 + 2 (X10 + X01) + X00 – [8 (X21 + X12) + 4 (X20 + X02)]

Sumo por separado la parte positiva Y, y la parte negativa Z.





Genero P = Y – Z con un restador (sumador y puertas NOT).

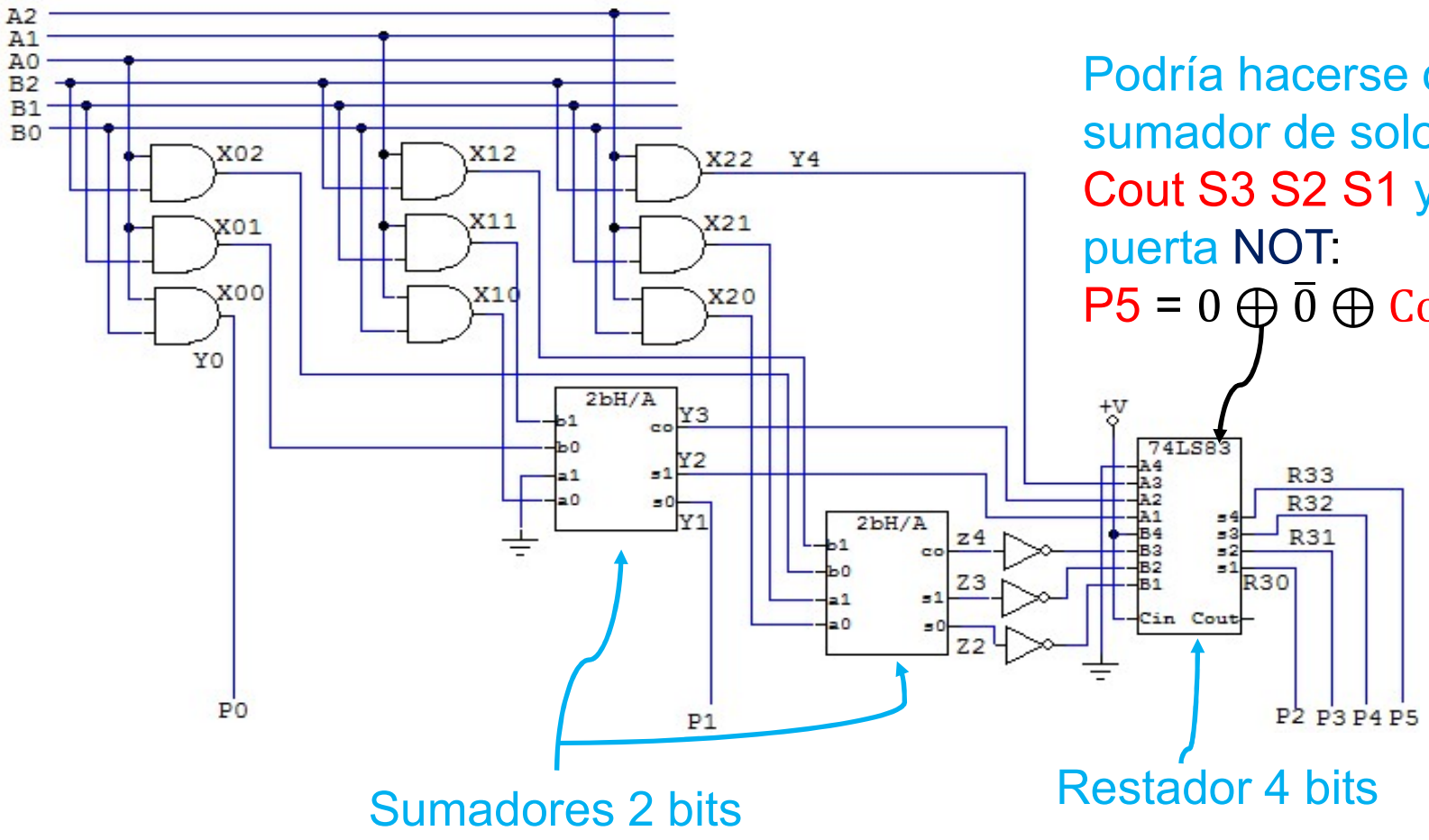


P = Y – Z = Y PLUS $\bar{Z}$ PLUS 1

Se extiende el sumador a 4 bits para evitar desbordamiento.

Y y Z positivos: se extiende un 0.

Xij: 9 puertas AND



Podría hacerse con un sumador de solo 3 bits: Cout S3 S2 S1 y una puerta NOT:

P5 = $0 \oplus \bar{0} \oplus Cout = \overline{Cout}$

Sumadores 2 bits

Restador 4 bits

16.1. La división A/B de dos números A (a3a2a1a0) y B (b3b2b1b0) de 4 bits, calculando el cociente Q (q3q2q1q0) y el resto R (r3r2r1r0), puede realizarse según el siguiente método:

- Tomar el dividendo como (000a3a2a1a0) y el divisor como (b3b2b1b0).

- Sea D1 los 4 bits de la izquierda (000a3) del dividendo, comparar D1 con el divisor B, si D1 ≥ B el bit del cociente q3 es 1 y X (x3x2x1x0) = D1 – B; si no el cociente q3 es 0 y X (x3x2x1x0) = D1.

- Tomar D2 como (x2x1x0a2), si D2 ≥ B el bit del cociente q2 es 1 e Y (y3y2y1y0) = D2 – B; si no el cociente q2 es 0 e Y (y3y2y1y0) = D2.

- Tomar D3 como (y2y1y0a1), si D3 ≥ B el bit del cociente q1 es 1 y Z (z3z2z1z0) = D3 – B; si no el cociente q1 es 0 y Z (z3z2z1z0) = D3.

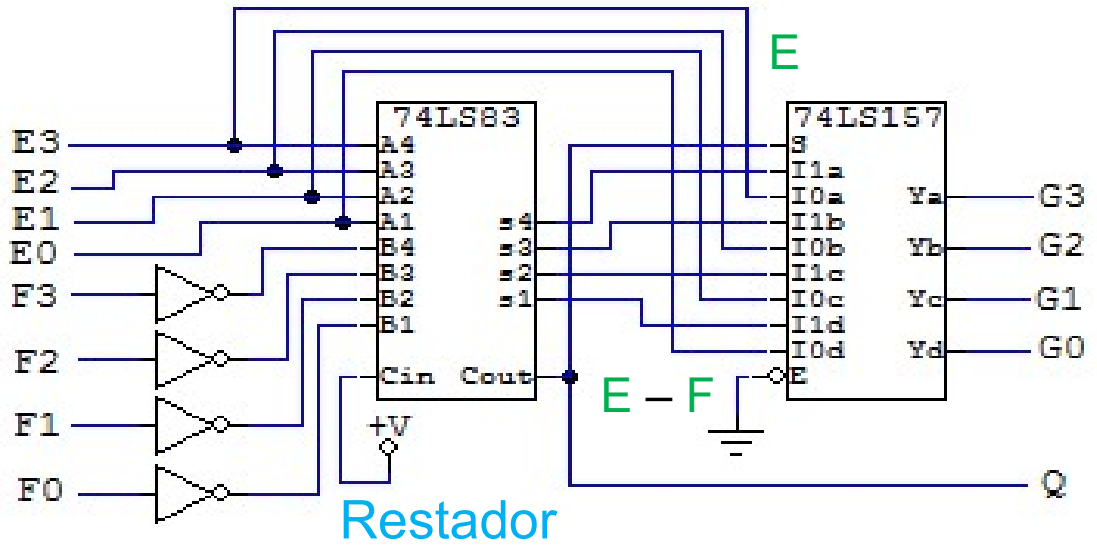
- Tomar D4 como (z2z1z0a0), si D4 ≥ B el bit del cociente q0 es 1 y se genera el resto R (r3r2r1r0) = D4 – B; si no el cociente q0 es 0 y el resto R (r3r2r1r0) = D4.

Diseñar un circuito digital que realice la división usando sumadores 74LS83 (4 chips), multiplexores 74LS157 (4 chips), y cuatro inversores 74LS04 (1 chip), utilizando el algoritmo anterior basado en operaciones de comparaciones y restas.

Calcular el tiempo de propagación máximo del circuito.

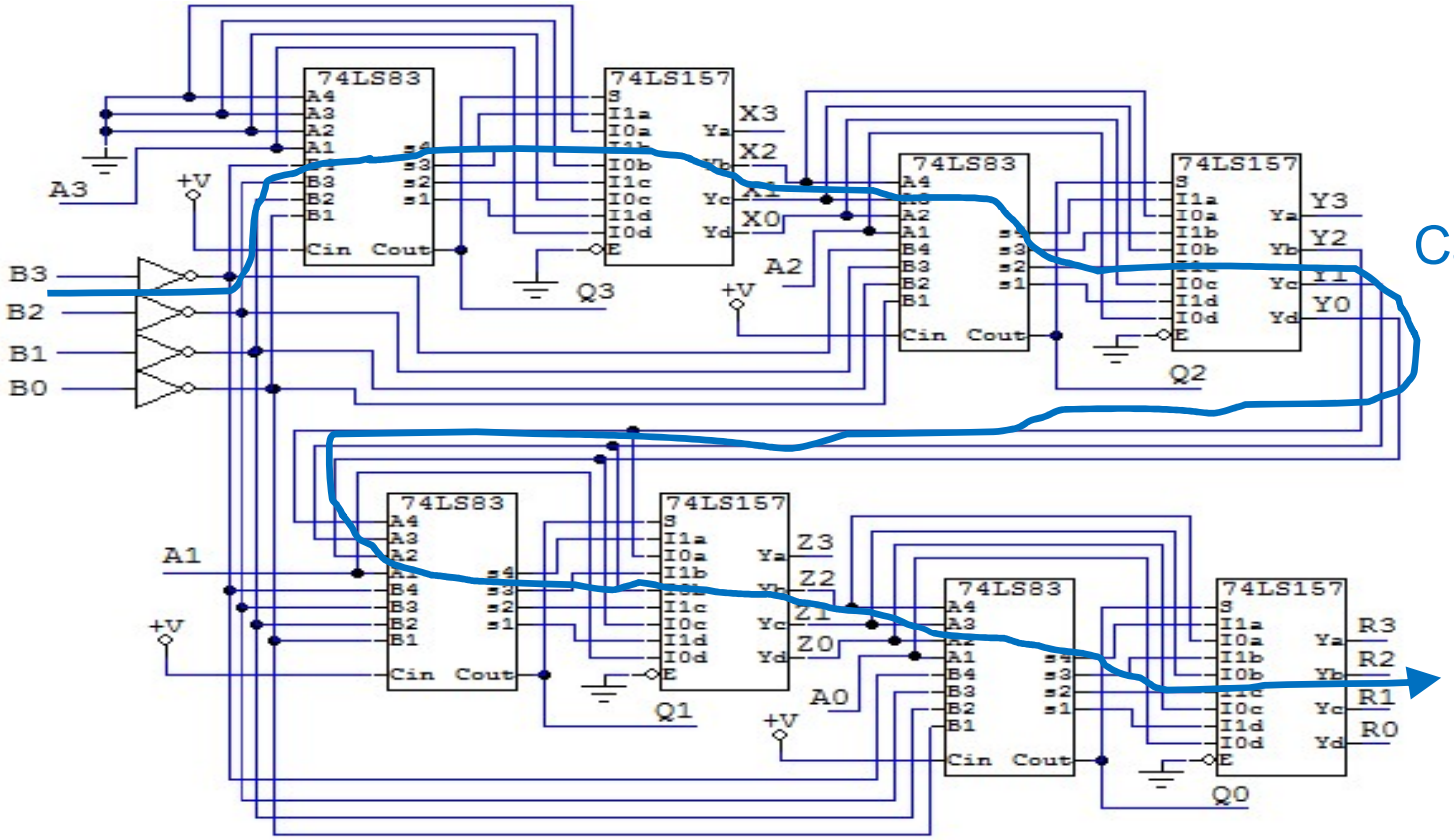
La celda básica (se repite 4 veces) es:
Si E ≥ F: Q es 1 y G = E – F; si no: Q es 0 y G = E (en 4 bits)

El comparador es un restador (1 sumador 74LS83 y 4 NOTs 74LS04):
E – F = E + $(F)_{2,c}$ = 2^N + (E – F) => si E ≥ F , E – F ≥ 2^N => Co = 1 => Q = Co
E – F = E PLUS $\bar{F}$ PLUS 1 (en c-a-2, o en números sin signo si E ≥ F)
G se selecciona con 2-Inp MUXs (74LS157): S es Q, I1 = E – F, I0 = E



4 etapas

Etapa	E	F	Q	G
1	0 0 0 A3	B3 B2 B1 B0	Q3	X3 X2 X1 X0
2	X2 X1 X0 A2	B3 B2 B1 B0	Q2	Y3 Y2 Y1 Y0
3	Y2 Y1 Y0 A1	B3 B2 B1 B0	Q1	Z3 Z2 Z1 Z0
4	Z2 Z1 Z0 A0	B3 B2 B1 B0	Q0	R3 R2 R1 R0



Tp(etapa) -- 2 caminos

Tp83(A-S) + Tp157(I-Y) = 24 ns + 14 ns = 38 ns

Tp83(A-Cout) + Tp157(S-Y) = 17 ns + 27 ns = 44 ns

Tp(etapa)max = 44 ns

Tpmax = Tp(NOT) + 4 Tp(etapa) = 15 ns + 4 * 44 ns = 191 ns