86.01 Técnica Digital

Unidades Aritméticas

Ing. Jorge H. Fuchs

Introducción

FACULTAD DE INGENIERIA
Universidad de Buenos Aires

Objetivos de la clase:

Analizar las celdas aritméticas básicas, sumadores, restadores, comparadores de magnitud.

Utilizar los bloques universales anteriores para la implementación de unidades aritméticas de más de un bit.

Estudiar los conceptos de sumador serie y sumador paralelo con acarreo serie y sus tiempos de respuesta.

Conocer las ventajas del sumador con acarreo anticipado (CLA).

Analizar el concepto de unidad aritmético lógica (ALU).

Analizar los circuitos comparadores de magnitud y su implementación.

Sumador de 2 Nros de 1 bit (Half Adder)

FACULTAD DE INGENIERIA
Universidad de Buenos Aires

Una de las operaciones más frecuentes en una computadora es **sumar** dos números.

Esta operación la realiza una unidad aritmético lógica (ALU), mediante un circuito **sumador.**

$$\begin{array}{r} 1\,0\,1\,1 \\ +\ \underline{1\,0\,0\,1} \\ 1\,0\,1\,0\,0 \end{array}$$

Asignaremos por convención en estos circuitos al 1 lógico el 1 aritmético binario, y al 0 lógico el 0 aritmético binario.

Sumador de 2 Nros de 1 bit (Half Adder)

FACULTAD DE INGENIERIA
Universidad de Buenos Aires

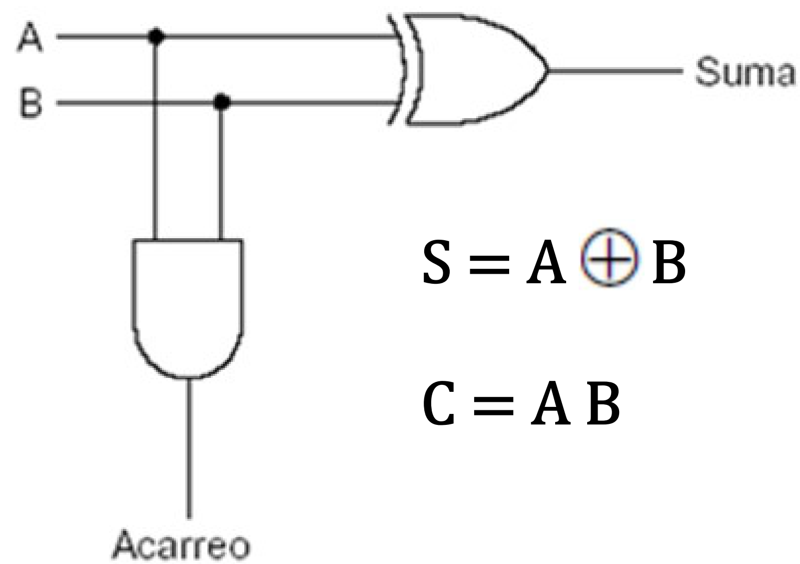
Para desarrollar un circuito binario que realice la suma aritmética de 2 bits (**Half Adder**), debemos basarnos en la tabla de la **suma binaria**:

$$\begin{array}{r} 1 \\ +\underline{\ 1} \\ 10 \end{array}$$

+	**0**	**1**
0	0	1
1	1	10

Vemos que el circuito tendrá **2 entradas** (A y B) y **2 salidas** (S y C).

A	B	Suma	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



$$S = A \oplus B$$

$$C = A\,B$$

Sumador de 3 Nros de 1 bit (Full Adder)

FACULTAD DE INGENIERIA
Universidad de Buenos Aires

En una suma de **2 números de n bits**, el circuito anterior queda restringido solo a la primer columna.

Para sumar la segunda columna y subsiguientes se deberá tener en cuenta el **acarreo** (me llevo) producido por la columna **anterior**.

$$
\begin{array}{r}
{\color{red}0\,1\,1\,} \\
1\,0\,1\,1 \\
+\;\underline{1\,0\,0\,1} \\
1\,0\,1\,0\,0
\end{array}
$$

Por lo tanto necesitaré un sumador de **3 números de 1 bit** cada uno (**Full Adder**).

Sumador de 3 Nros de 1 bit (Full Adder)

FACULTAD DE INGENIERIA
Universidad de Buenos Aires

En este caso tendremos un circuito de **3 entradas** (A_i, B_i y C_{i-1}) y **2 salidas** (S_i y C_i).

A_i	B_i	C_{i-1}	S_i	C_i
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$$S_i = A_i \oplus B_i \oplus C_{i-1}$$

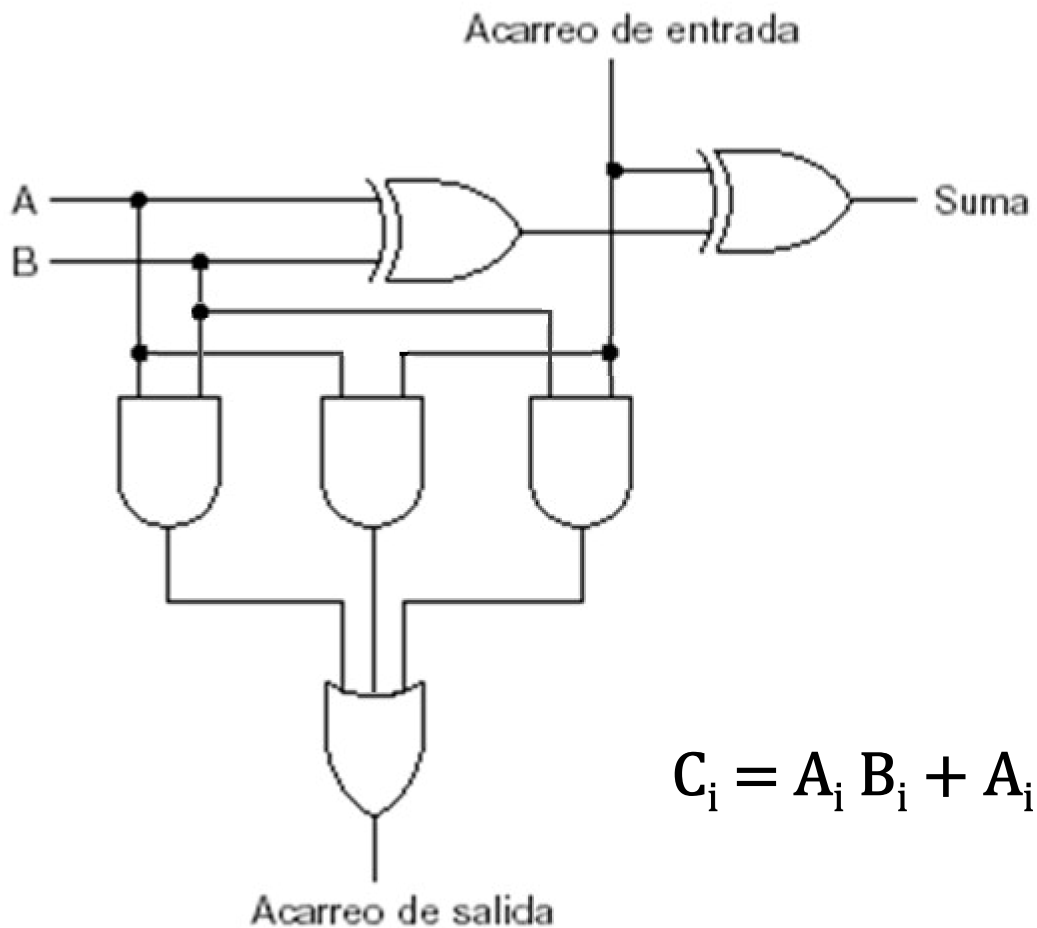
Ai \ BiCi-1	00	01	11	10
0	0	0	1	0
1	0	1	1	1

$$C_i = A_i\, B_i + A_i\, C_{i-1} + B_i\, C_{i-1}$$

Suponiendo implementaciones en 2 niveles, tanto $\mathbf{S_i}$ como $\mathbf{C_i}$ presentarán un retardo de 2 compuertas (**2** δ, donde δ es el retardo típico de una compuerta).

Sumador de 3 Nros de 1 bit (Full Adder)

El circuito del **FA** quedará:

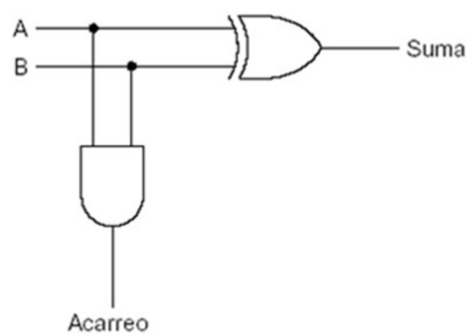


$$C_i = A_i\, B_i + A_i\, C_{i\text{-}1} + B_i\, C_{i\text{-}1}$$

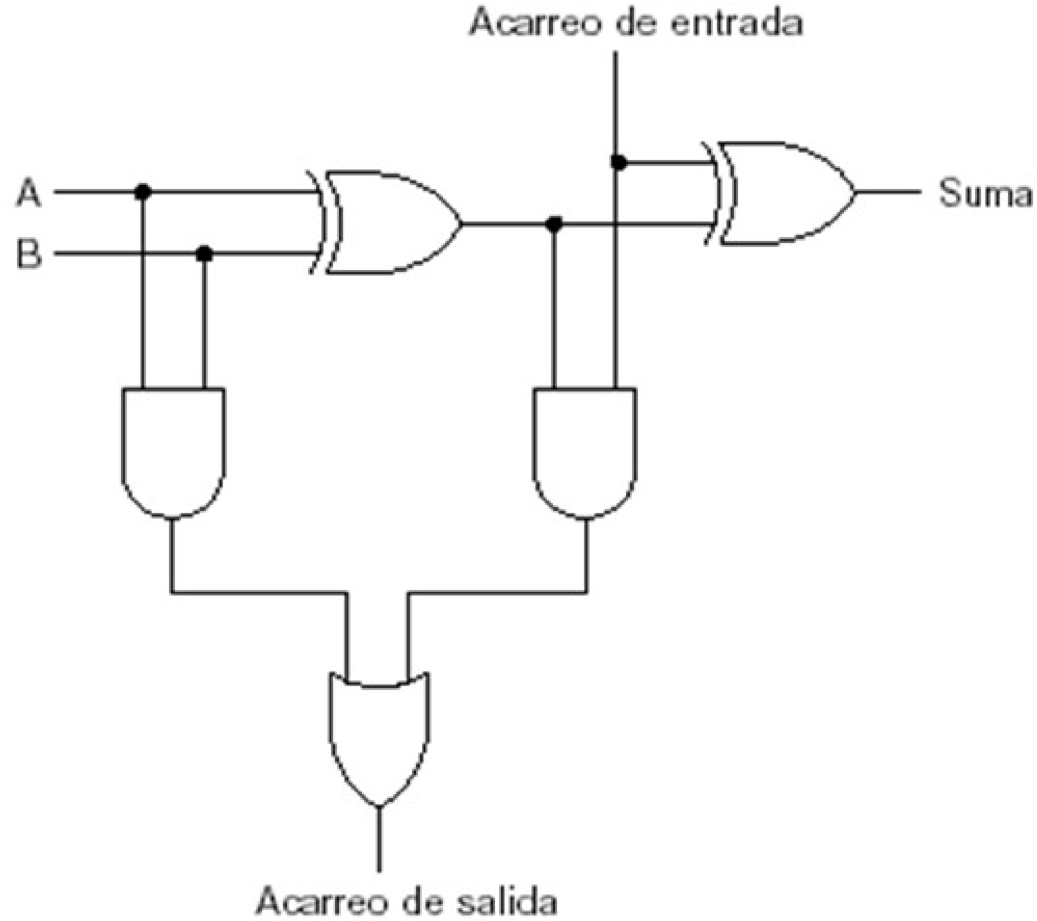
Sumador de 3 Nros de 1 bit (Full Adder)

También podremos implementar un **FA** como la combinación de 2 **HA** y una compuerta **OR**. Atención con los retardos, 3 niveles.

HA

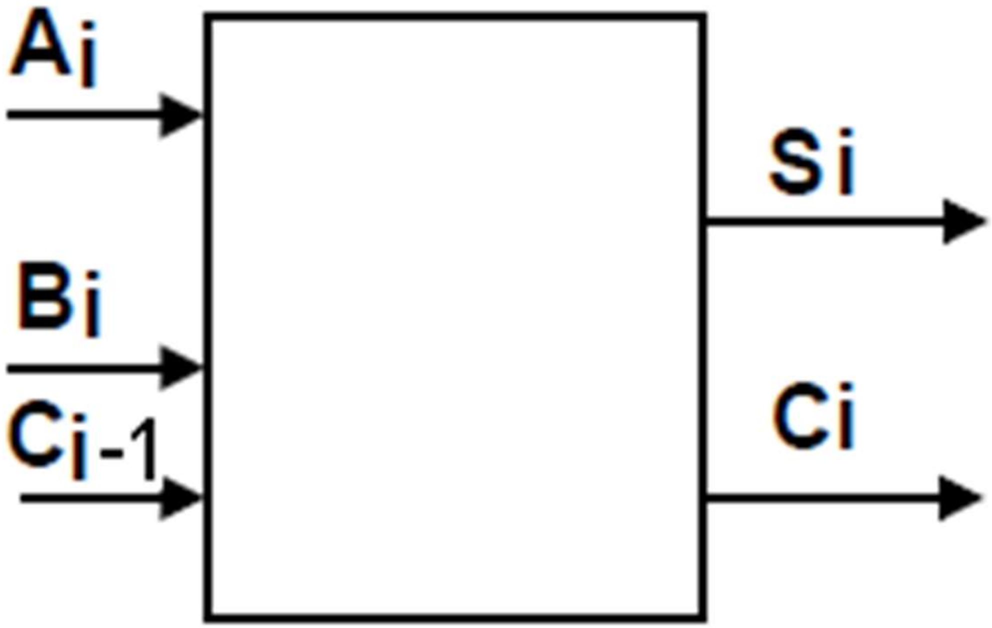


FA



Sumador de 3 Nros de 1 bit (Full Adder)

Tenemos entonces un **bloque funcional FA**.



Asumimos que tanto $\mathbf{S_i}$ como $\mathbf{C_i}$ presentarán un retardo de 2 compuertas (**2** δ).

Restador de 2 números de 1 bit

Ahora desarrollaremos un circuito binario que realice la **resta** de **2 bits**, debemos basarnos en la siguiente TV:

A	B	Resta	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

$$R = A \oplus B$$

$$Bw = \overline{A}\ B$$

Al igual que con los sumadores, este restador solo sirve para la primer columna.

Para restar la segunda columna y subsiguientes se deberá tener en cuenta el **borrow** (pedir prestado) producido por la columna **anterior**.

Restador de 3 números de 1 bit

FACULTAD DE INGENIERIA Universidad de Buenos Aires

En este caso tendremos un circuito de **3 entradas** (A_i, B_i y Bw_{i-1}) y **2 salidas** (R_i y Bw_i).

A_i	B_i	Bw_{i-1}	R_i	Bw_i
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

$$R_i = A_i \oplus B_i \oplus Bw_{i-1}$$

Ai \ BiBwi-1	00	01	11	10
0	0	1	1	1
1	0	0	1	0

$$Bw_i = \bar{A}_i\, B_i + \bar{A}_i\, Bw_{i-1} + B_i\, Bw_{i-1}$$

Equivalencias

Vemos que los sumadores y restadores son **circuitalmente similares**, por lo que podemos implementar unos con otros y viceversa agregando alguna compuerta o inversor:

Sumador de 3 bits a partir de 2 sumadores de 2 bits

Sumador de 3 bits a partir de 1 sumador y 1 restador de 2 bits

Sumador de 3 bits a partir de 1 restador de 3 bits

Sumador de 2 bits a partir de 1 restador de 2 bits

Restador de 2 bits a partir de 1 sumador de 2 bits

Restador de 3 bits a partir de 1 sumador de 3 bits

Restador de 3 bits a partir de 2 sumadores de 2 bits

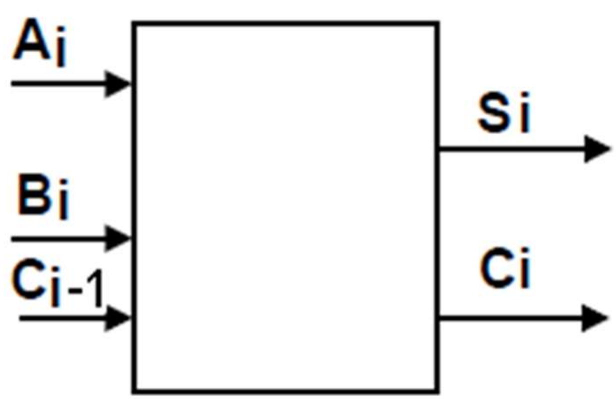
Tipos de sumadores

Basándonos en el **bloque funcional FA** ya visto, ahora analizaremos varios conceptos distintos de **sumadores de más de 1 bit** y sus tiempos de cálculo:

Sumador Serie

Sumador Paralelo con Acarreo Serie (Ripple Adder)

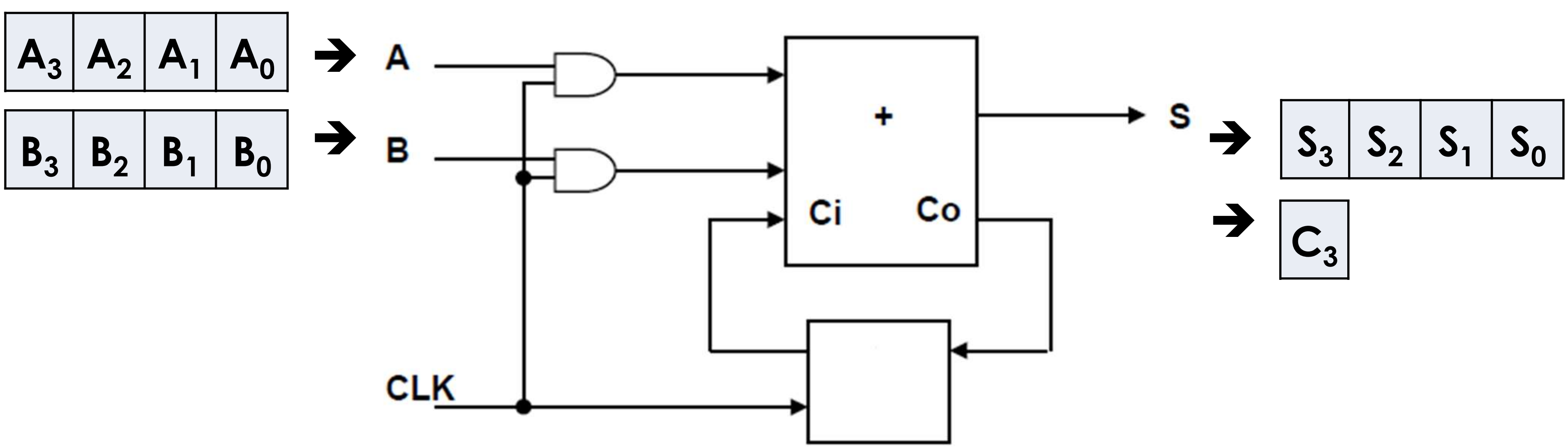
Sumador Paralelo con Acarreo Anticipado (Carry Look Ahead)



Sumador serie

Este concepto requiere poco hardware (**1 único FA**) pero necesita almacenar los sumandos y resultados (memoria) y también secuenciar las sumas (software).

$$
\begin{array}{rccccc}
 & & \color{red}{C_2} & \color{red}{C_1} & \color{red}{C_0} & \\
 & & A_3 & A_2 & A_1 & A_0 \\
+ & & \underline{B_3} & \underline{B_2} & \underline{B_1} & \underline{B_0} \\
 & C_3 & S_3 & S_2 & S_1 & S_0
\end{array}
$$



Sumador serie

Debemos considerar que por cada bit (columna) tendremos los retardos propios del **FA**, a esto deberemos adicionar los tiempos de transporte de los sumandos, almacenamiento de los resultados en la memoria, etc.

$$
\begin{array}{rcccc}
 & C_2 & C_1 & C_0 & \\
 & A_3 & A_2 & A_1 & A_0 \\
+ & \underline{B_3} & \underline{B_2} & \underline{B_1} & \underline{B_0} \\
C_3 & S_3 & S_2 & S_1 & S_0
\end{array}
$$

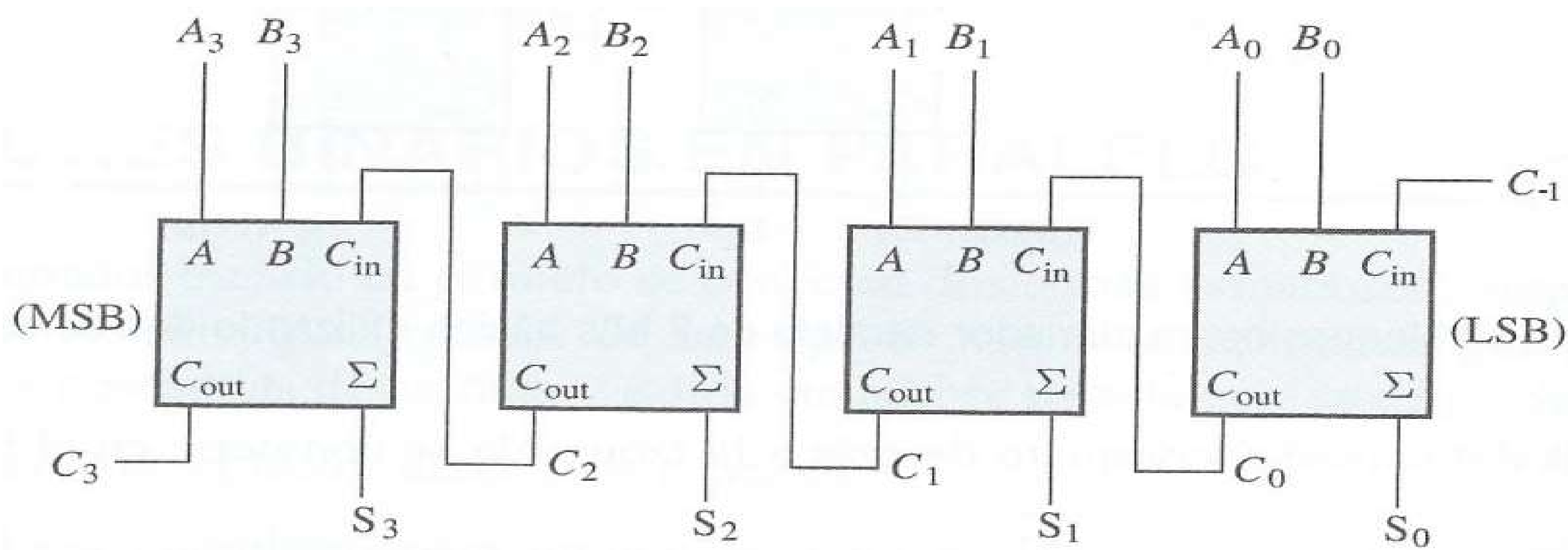
El tiempo total depende **linealmente** de la cantidad de bits del sumador más los tiempos de almacenamiento. Considerando el retardo del FA como 2δ tenemos:

Para **n bits**: $\mathbf{Tt = 2\,n\,\delta + ts}$

Sumador paralelo con acarreo serie

Este concepto es puro hardware (**1 FA por cada bit**).

$$
\begin{array}{r}
\color{red}{C_2\ C_1\ C_0\ \ } \\
A_3\ A_2\ A_1\ A_0 \\
+\ \underline{B_3\ B_2\ B_1\ B_0} \\
C_3\ \ S_3\ \ S_2\ \ S_1\ \ S_0
\end{array}
$$



8δ 8δ 6δ 6δ 4δ 4δ 2δ 2δ

Sumador paralelo con acarreo serie

Debemos considerar que cada FA debe esperar los resultados válidos del anterior, y a partir de ese momento debe transcurrir su propio tiempo de cálculo.

$$
\begin{array}{rcccc}
 & {\color{red}C_2} & {\color{red}C_1} & {\color{red}C_0} & \\
 & A_3 & A_2 & A_1 & A_0 \\
+ & \underline{B_3} & \underline{B_2} & \underline{B_1} & \underline{B_0} \\
C_3 & S_3 & S_2 & S_1 & S_0
\end{array}
$$

El tiempo total depende **linealmente** de la cantidad de bits del sumador. Nuevamente considerando el retardo del FA como 2δ tenemos:

Para **n bits**: $\mathbf{Tt = 2\ n\ \delta}$

Sumador paralelo con acarreo anticipado

FACULTAD DE INGENIERIA Universidad de Buenos Aires

Este sumador (**CLA: Carry Look Ahead**) precalcula todos los acarreos para así tener una velocidad independiente de la cantidad de bits a sumar:

$$
\begin{array}{rcccc}
 & \color{red}{C_2} & \color{red}{C_1} & \color{red}{C_0} & \\
 & A_3 & A_2 & A_1 & A_0 \\
+ & \underline{B_3} & \underline{B_2} & \underline{B_1} & \underline{B_0} \\
\color{red}{C_3} & S_3 & S_2 & S_1 & S_0
\end{array}
$$

Debemos precalcular los C_i para cada columna, de la solución del **FA** tenemos:

$$C_i = A_i B_i + A_i C_{i-1} + B_i C_{i-1} \quad \rightarrow \quad C_i = A_i B_i + C_{i-1} (A_i + B_i)$$

si: $g_i = A_i B_i$ y $p_i = A_i + B_i$

$$C_i = g_i + C_{i-1} p_i$$

Sumador paralelo con acarreo anticipado

FACULTAD DE INGENIERIA
Universidad de Buenos Aires

Debemos precalcular los C_i para las 4 columnas.

$$\mathbf{C_i = g_i + C_{i-1}\, p_i}$$

$$\mathbf{C_0 = g_0 + C_{-1}\, p_0}$$

$$\mathbf{C_1 = g_1 + g_0\, p_1 + C_{-1}\, p_0\, p_1}$$

$$\mathbf{C_2 = g_2 + g_1\, p_2 + g_0\, p_1\, p_2 + C_{-1}\, p_0\, p_1\, p_2}$$

$$\mathbf{C_3 = g_3 + g_2\, p_3 + g_1\, p_2\, p_3 + g_0\, p_1\, p_2\, p_3 + C_{-1}\, p_0\, p_1\, p_2\, p_3}$$

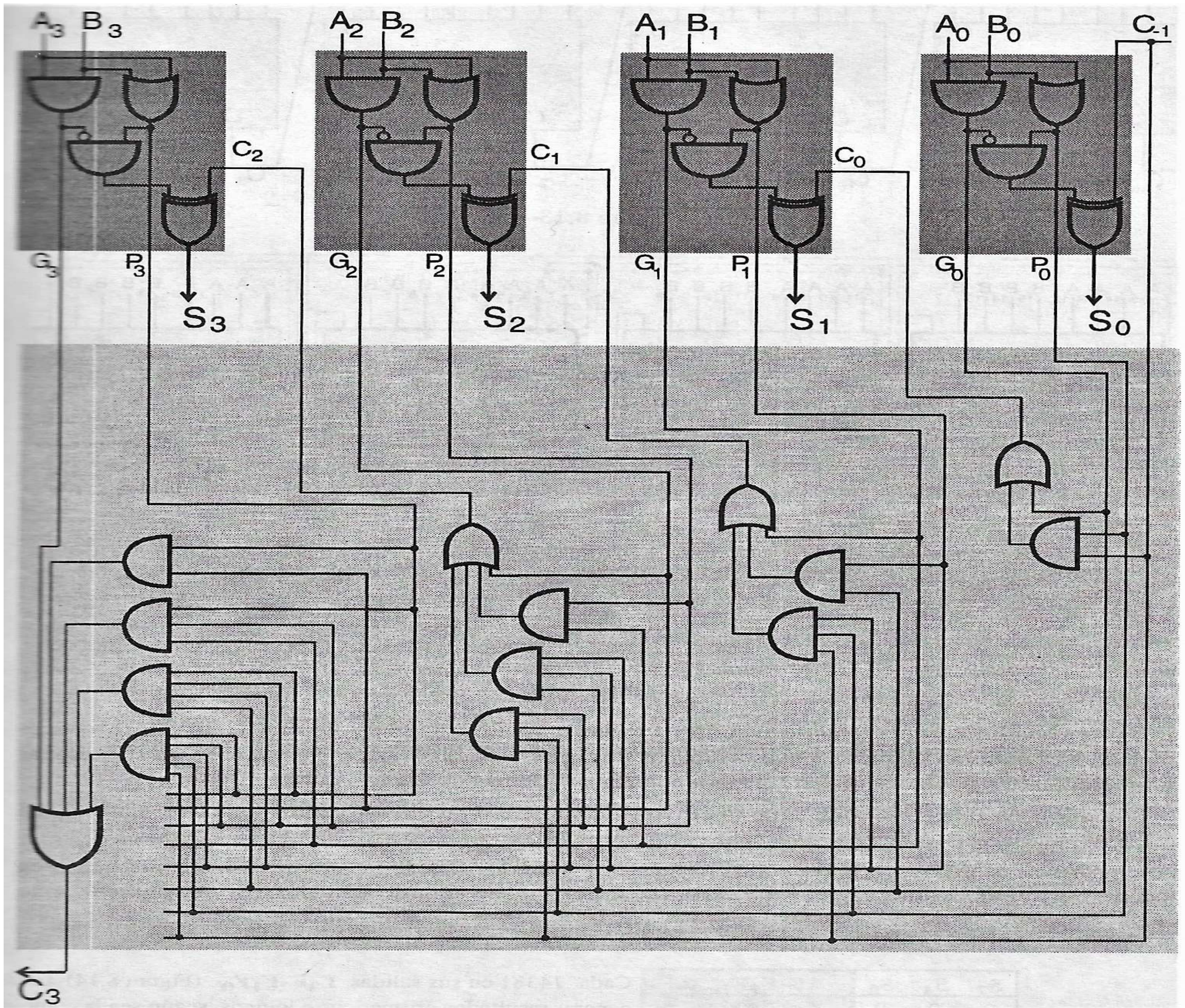
Son todas funciones de 2 niveles, que se hacen 3 considerando g_i y p_i.

Por lo tanto, en **3** δ tendremos precalculados todos los $\mathbf{C_i}$.

Adicionando **2** δ de los FA tendremos: **Tt = 5** δ (**independiente de n**)

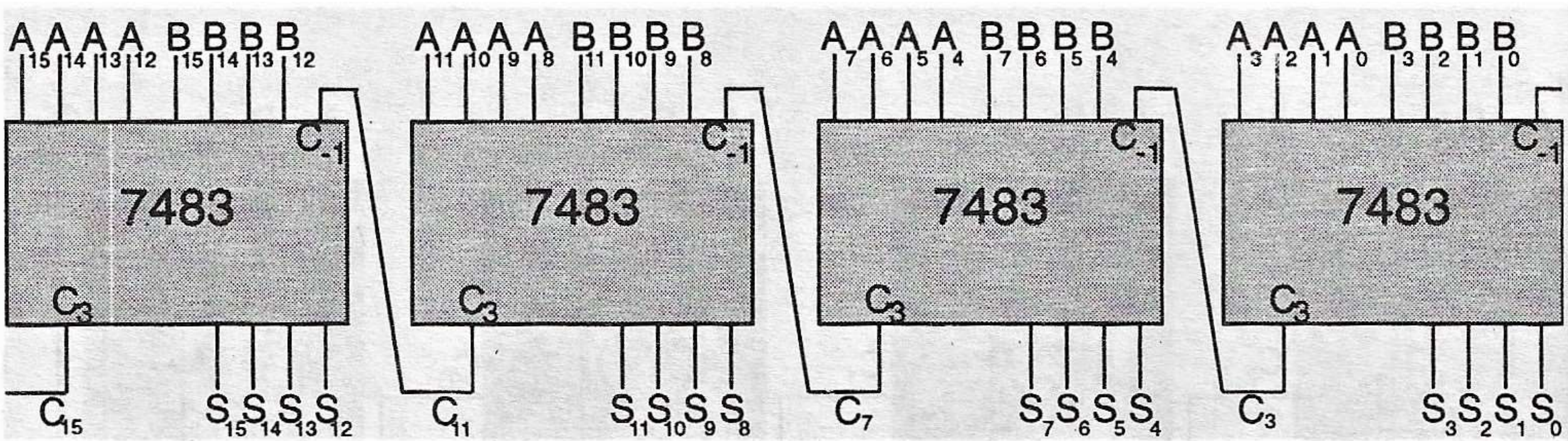
Sumador paralelo con acarreo anticipado

El circuito completo para 4 bits queda (CI 7483 Sumador de 4 bits CLA):



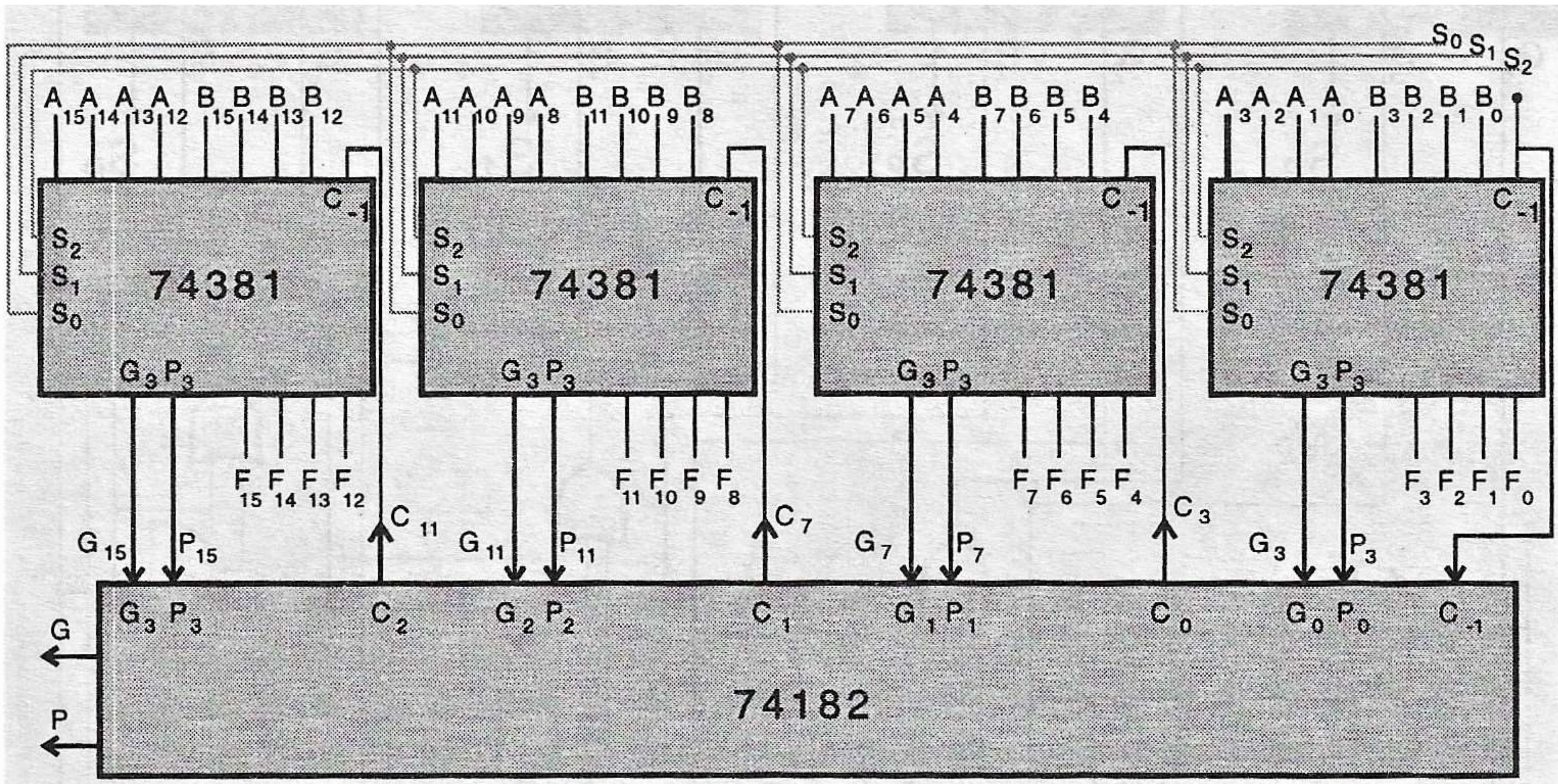
Sumador paralelo con acarreo anticipado

Sumador paralelo de 16 bits con acarreo anticipado (4 CI 7483 Sumador de 4 bits CLA) y acarreo serie entre ellos:



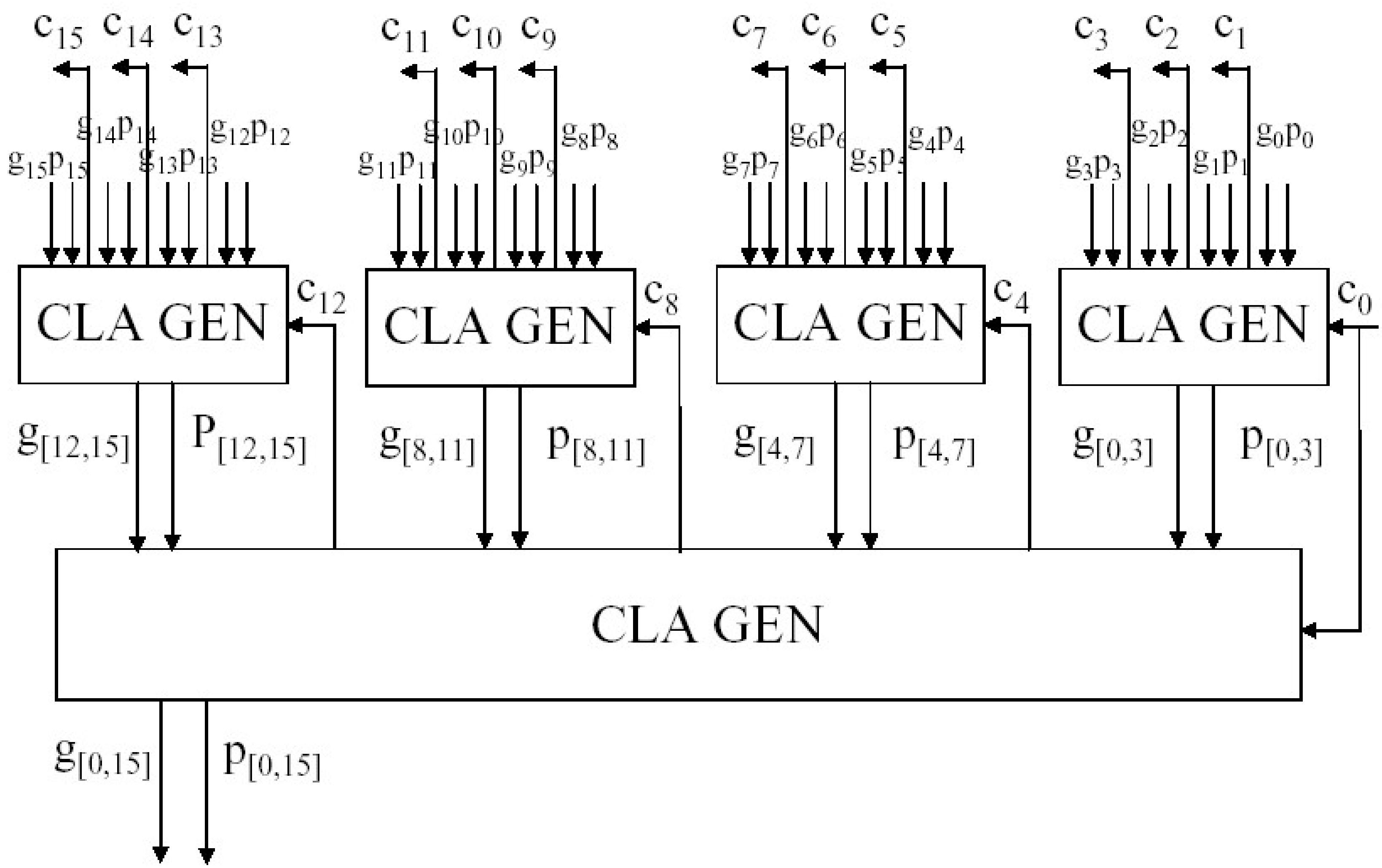
Sumador paralelo con acarreo anticipado

Sumador paralelo de 16 bits con acarreo anticipado (74381/74182):



Sumador paralelo con acarreo anticipado

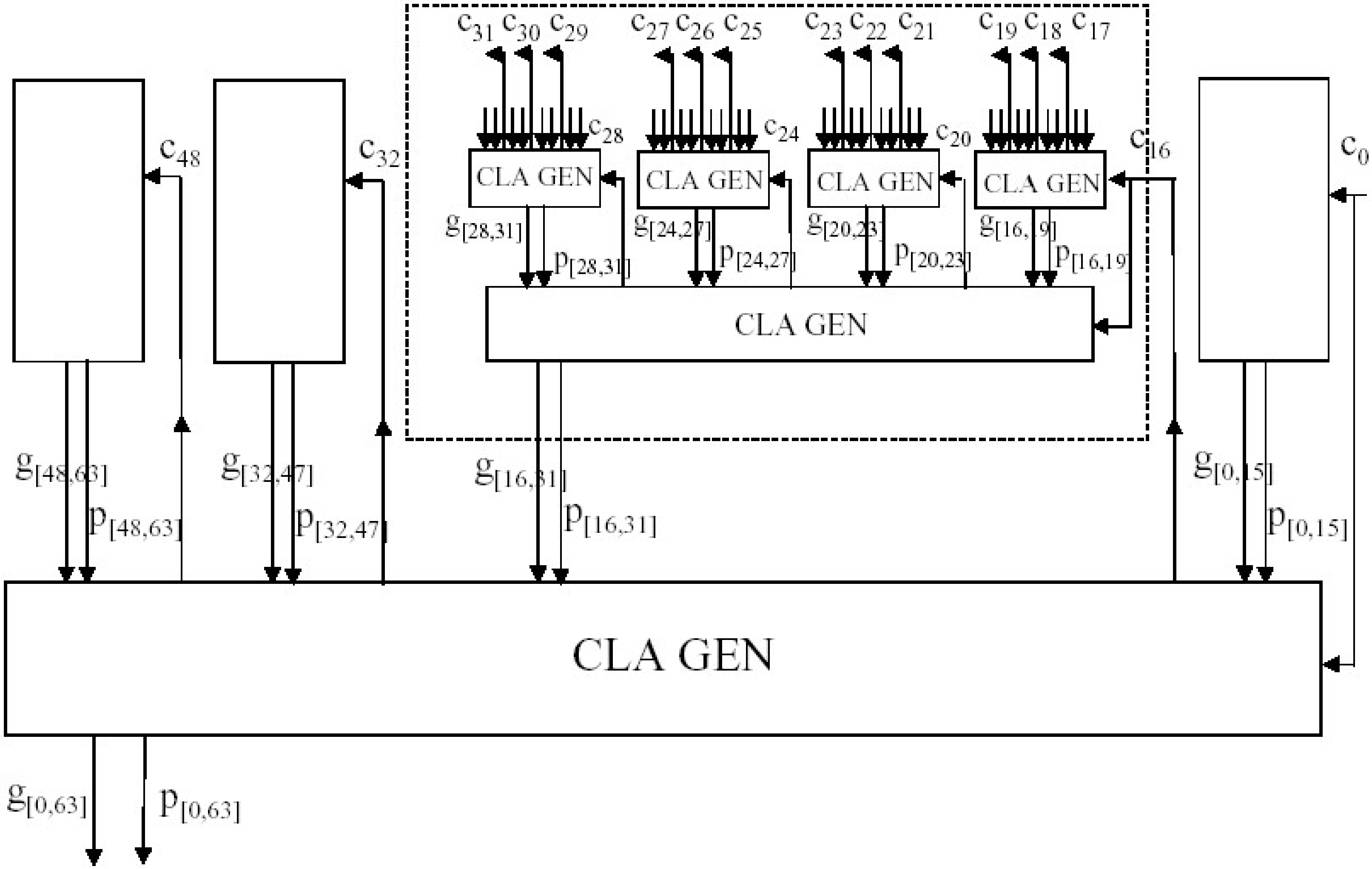
Generador de acarreo de 16 bits implementado en 2 niveles (sin los FA):



Sumador paralelo con acarreo anticipado

FACULTAD DE INGENIERIA Universidad de Buenos Aires

Generador de acarreo de 64 bits implementado en 3 niveles (sin los FA):



Tipos de sumadores (resumen)

FACULTAD DE INGENIERIA
Universidad de Buenos Aires

Sumador Serie

Ventaja: 1 solo FA

Desventaja: Necesita Software y memoria

Tiempo de cálculo: $Tt = 2\ n\ \delta + ts$

P. ej.: para 4 bits: $> 8\ \delta$ para 64 bits: $> 128\ \delta$

Sumador Paralelo con Acarreo Serie (Ripple Adder)

Ventaja: No necesita Software ni memoria

Desventaja: 1 FA por cada bit

Tiempo de cálculo: $Tt = 2\ n\ \delta$

P. ej.: para 4 bits: $8\ \delta$ para 64 bits: $128\ \delta$

Sumador Paralelo con Acarreo Anticipado (Carry Look Ahead)

Ventaja: Velocidad

Desventaja: 1 FA por cada bit

Tiempo de cálculo: $Tt = 5\ \delta$

Independiente de la cantidad de bits

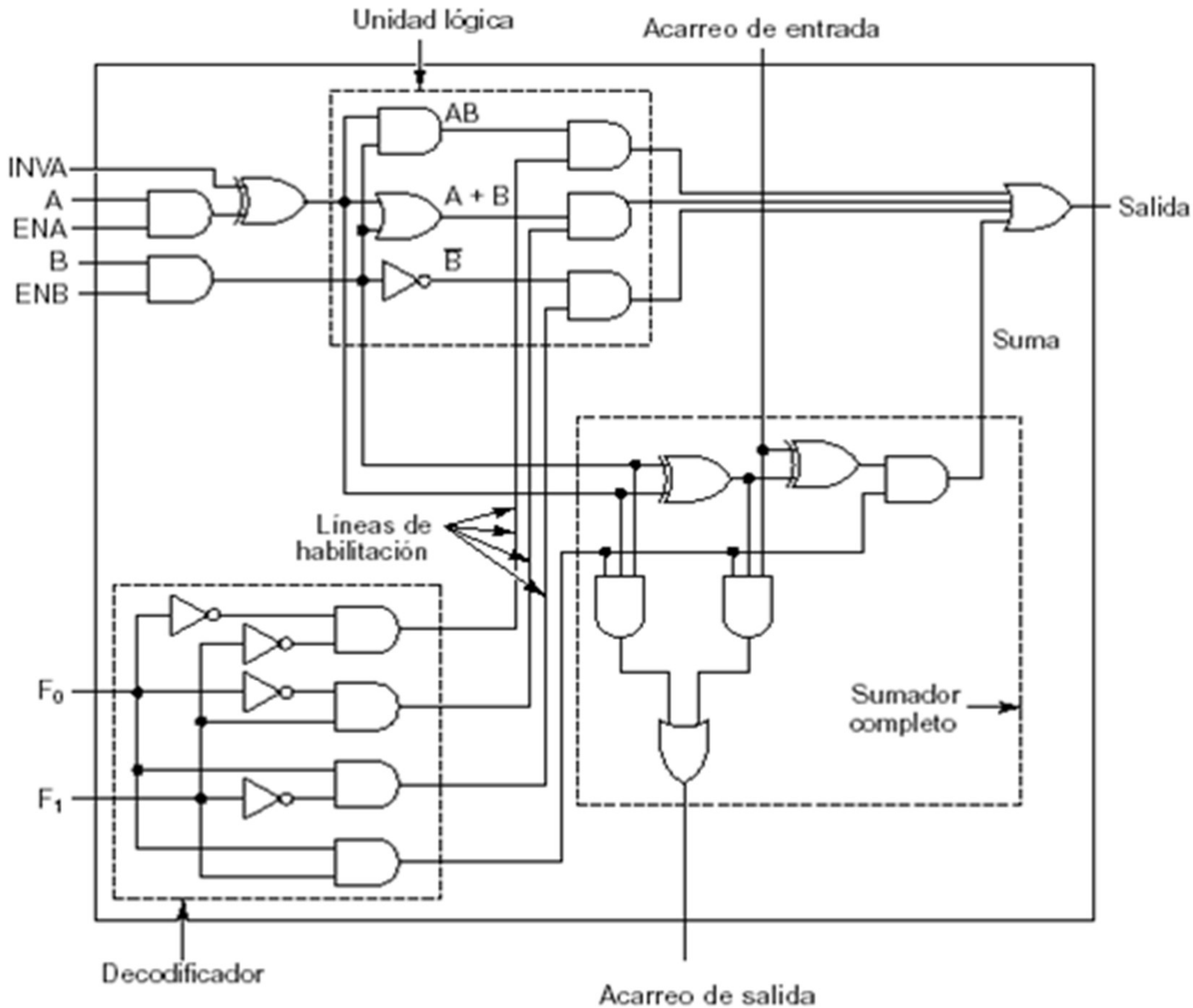
Sumador (implementación alternativa)

Considerando que al sumar 2 números de n bits el resultado podrá tener hasta n + 1 bits, otro abordaje podría plantear un circuito de 2 niveles con 2 n variables de entrada y n + 1 salidas. Ventajas y desventajas. Retardos.

En nuestro ejemplo tendríamos 8 entradas y 5 salidas.

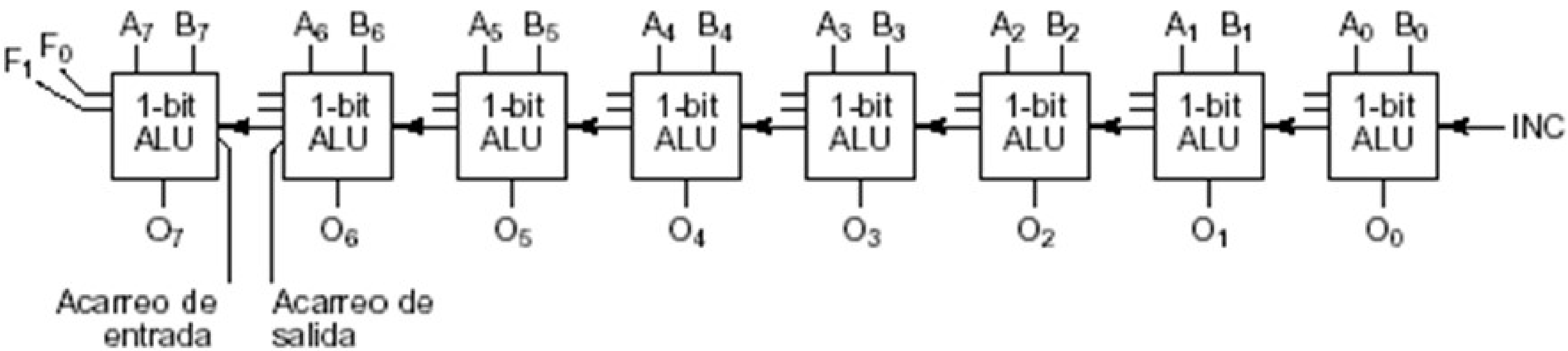
$A_3\ A_2\ A_1\ A_0$	4 entradas
$+\ \underline{B_3\ B_2\ B_1\ B_0}$	4 entradas
$C_3\ S_3\ S_2\ S_1\ S_0$	5 salidas

Unidad aritmético lógica (ALU)



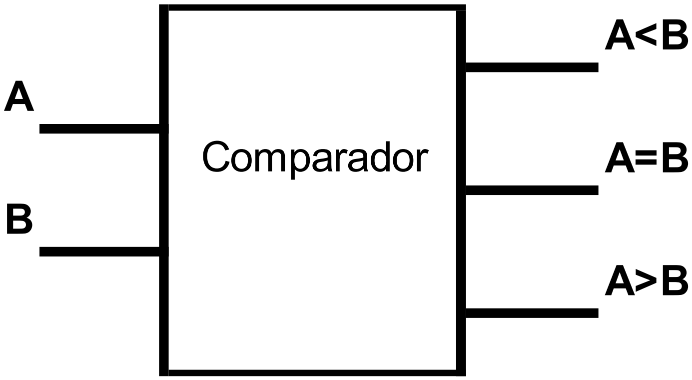
Unidad aritmético lógica (ALU)

Conjunto de 8 ALUs de 1 bit formando una ALU de 8 bits:



Comparador de magnitud de 1 bit

Consideremos un circuito que compare 2 números de **1 bit** cada uno:



A	B	$A<B$	$A\leq B$	$A=B$	$A\geq B$	$A>B$	$A\neq B$
0	0	0	1	1	1	0	0
0	1	1	1	0	0	0	1
1	0	0	0	0	1	1	1
1	1	0	1	1	1	0	0

$$S_{A\leq B} = S_{A<B} + S_{A=B}$$

$$S_{A\geq B} = S_{A>B} + S_{A=B}$$

$$S_{A\leq B} = \overline{S_{A>B}}$$

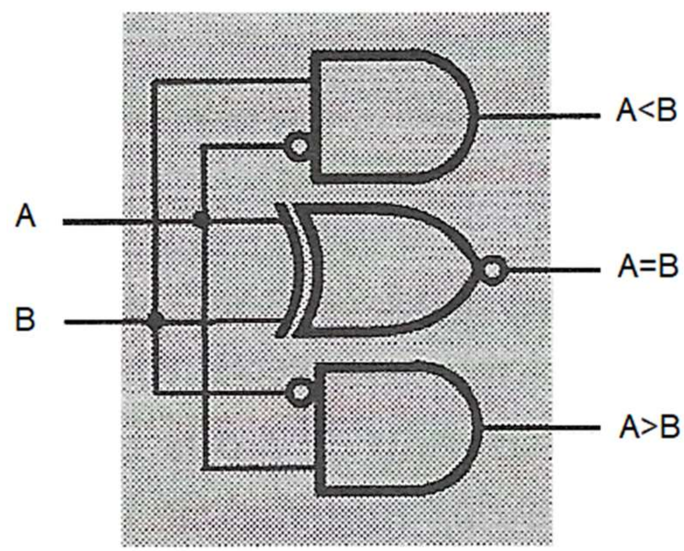
$$S_{A\geq B} = \overline{S_{A<B}}$$

$$S_{A\neq B} = \overline{S_{A=B}}$$

$$S_{A<B} = \bar{A}\,B$$

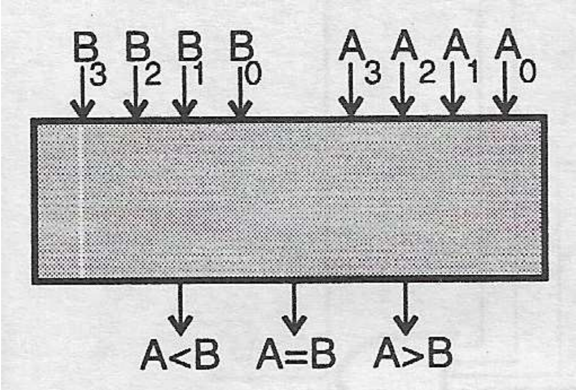
$$S_{A=B} = \overline{A \oplus B}$$

$$S_{A>B} = A\,\bar{B}$$



Comparador de magnitud de 4 bits

Veamos ahora un circuito que compare 2 números de **4 bits** cada uno:



Tendríamos 3 funciones de 8 variables cada una. Ventajas y desventajas.

A3	A2	A1	A0	B3	B2	B1	B0	A<B	A=B	A>B
0	0	0	0	0	0	0	0	0	1	0
		256 combinaciones								
1	1	1	1	1	1	1	1	0	1	0

Comparador de magnitud de 4 bits

FACULTAD DE INGENIERIA Universidad de Buenos Aires

Utilizando comparadores de **1 bit**, implementando solo **A>B** y **A=B** tendríamos:

A=B cuando $A_3=B_3$ **y** $A_2=B_2$ **y** $A_1=B_1$ **y** $A_0=B_0$

A>B cuando $A_3>B_3$ **o** $A_3=B_3$ **y** $A_2>B_2$ **o** $A_3=B_3$ **y** $A_2=B_2$ **y** $A_1>B_1$ **o**

$A_3=B_3$ **y** $A_2=B_2$ **y** $A_1=B_1$ **y** $A_0>B_0$

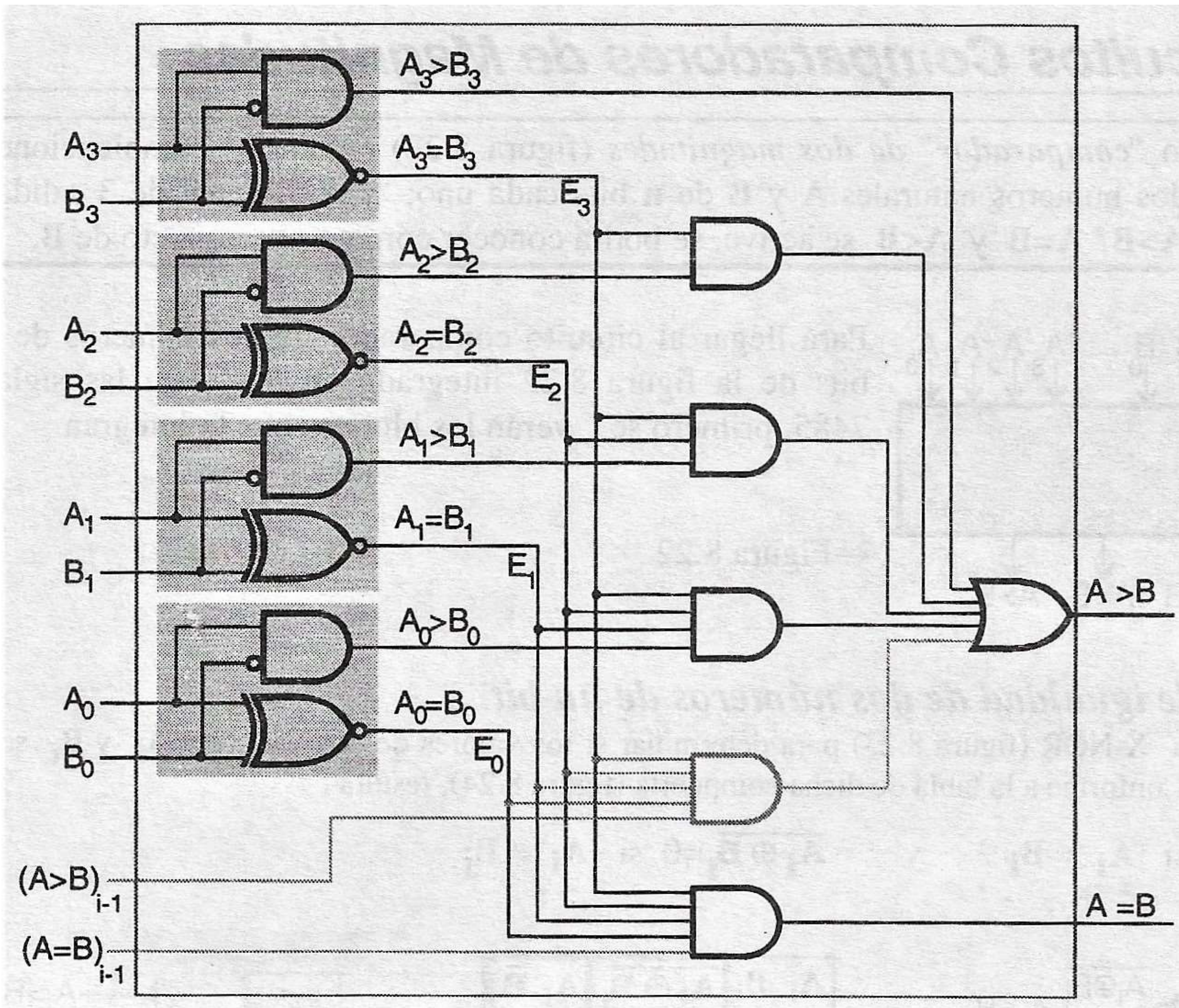
$$S_{A=B} = \overline{A_3 \oplus B_3}\ \overline{A_2 \oplus B_2}\ \overline{A_1 \oplus B_1}\ \overline{A_0 \oplus B_0}$$

$$S_{A>B} = A_3\,\overline{B}_3 + \overline{A_3 \oplus B_3}\ A_2\,\overline{B}_2 + \overline{A_3 \oplus B_3}\ \overline{A_2 \oplus B_2}\ A_1\,\overline{B}_1 +$$

$$+\ \overline{A_3 \oplus B_3}\ \overline{A_2 \oplus B_2}\ \overline{A_1 \oplus B_1}\ A_0\,\overline{B}_0$$

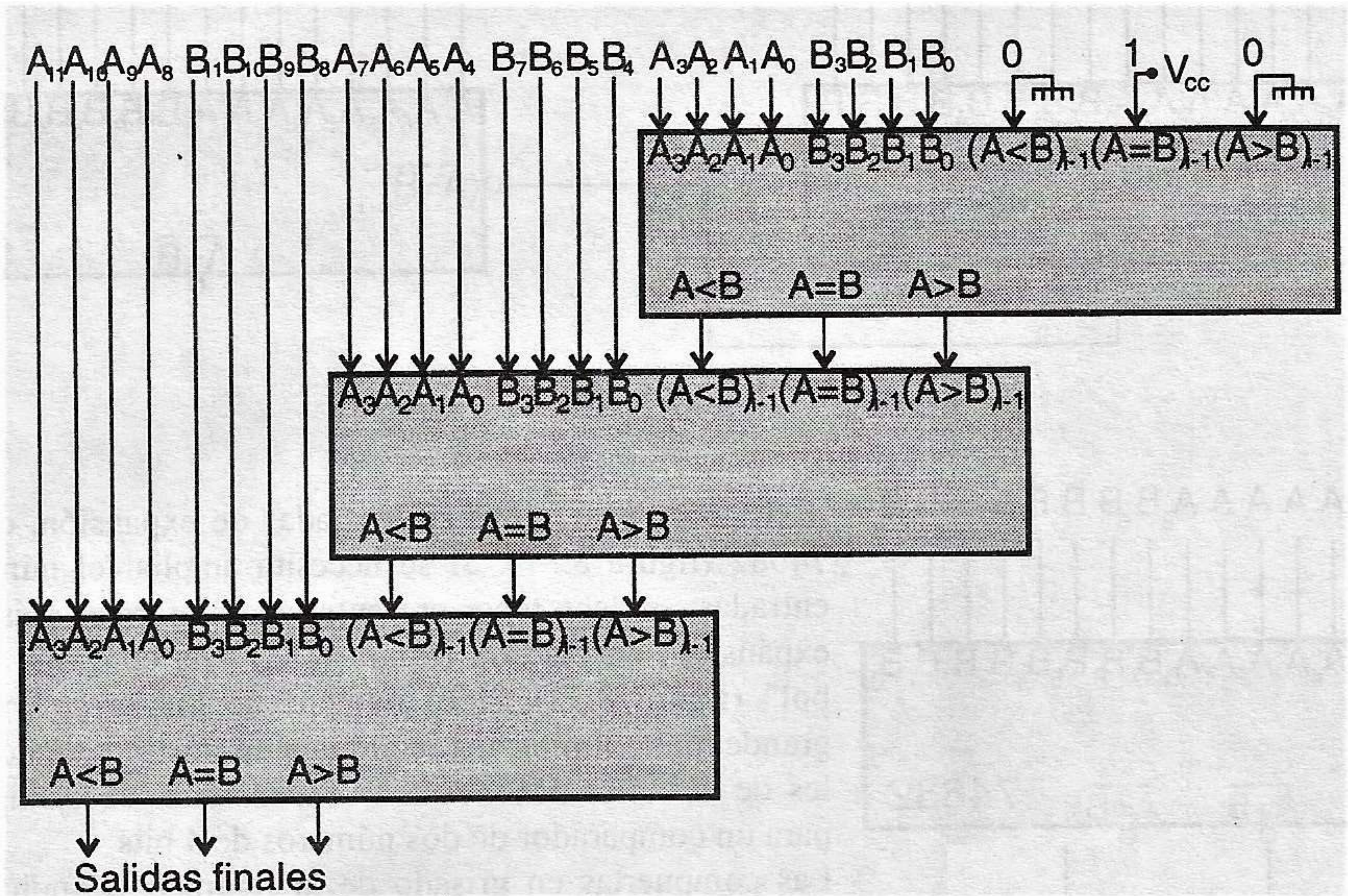
Comparador de magnitud de 4 bits

Agregando entradas adicionales para otro comparador anterior (en gris) queda:



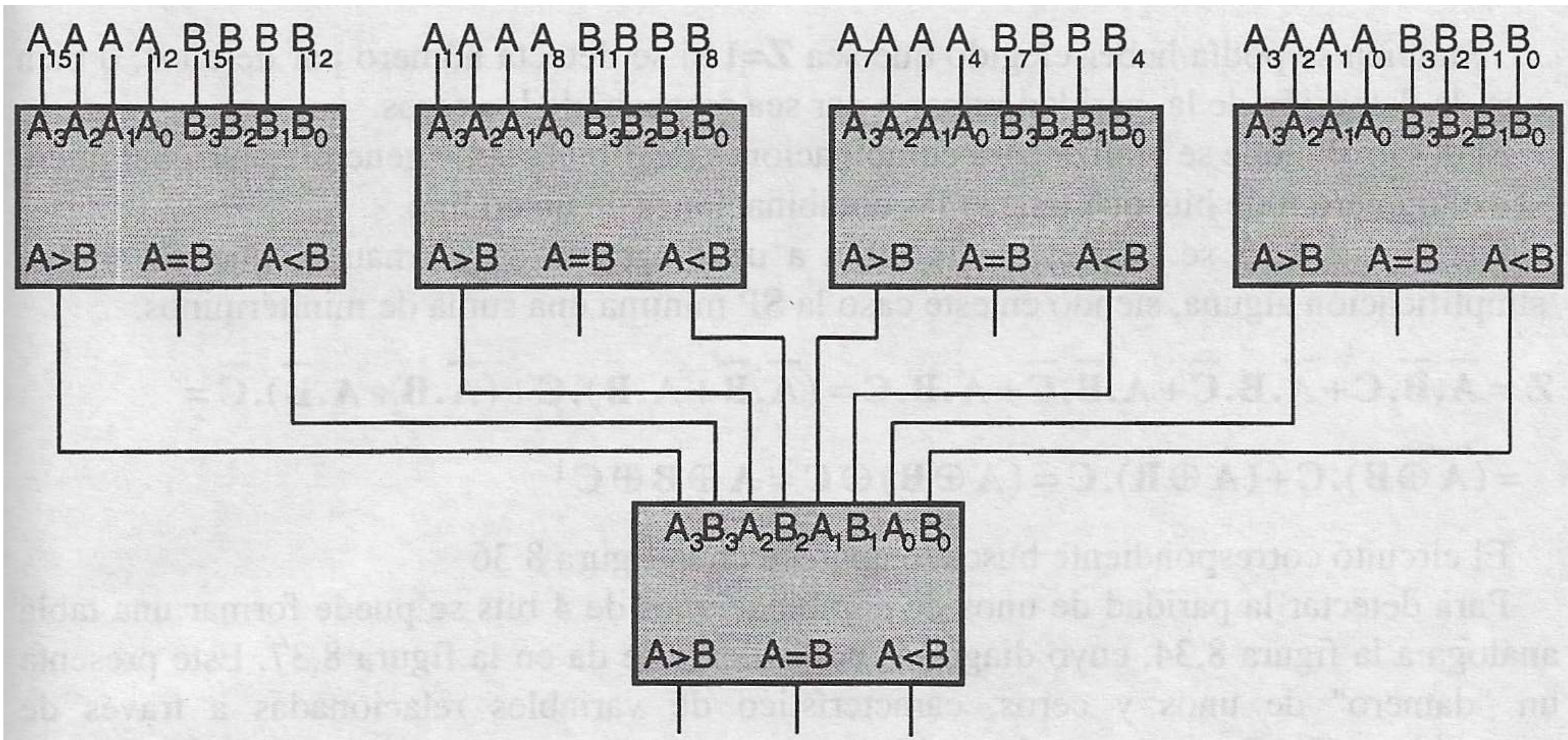
Comparador de magnitud de 12 bits

Conectando 3 comparadores de 4 bits en cascada (CI 7485):



Comparador de magnitud de 16 bits

Conectando 5 comparadores de 4 bits en árbol:

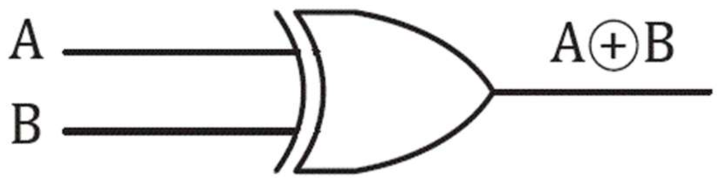


Generador / detector de paridad

FACULTAD DE INGENIERIA
Universidad de Buenos Aires

Las compuertas exclusivas permiten conocer la **paridad** de sus entradas:

A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0



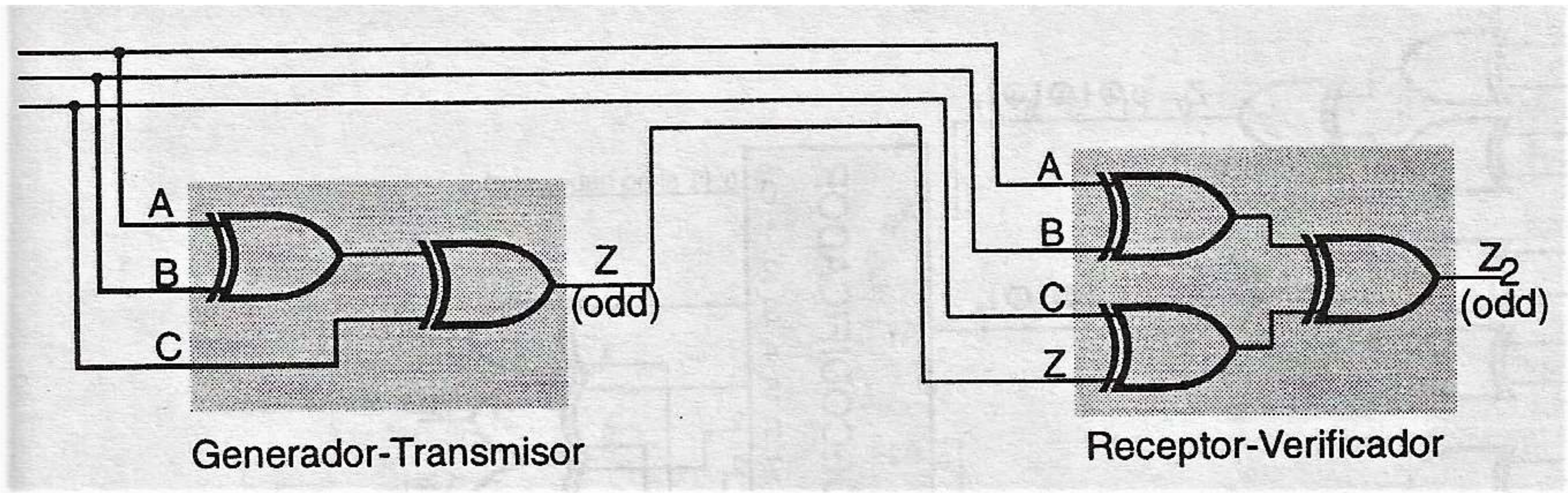
Generador de paridad **PAR**

A	B	$\overline{A \oplus B}$
0	0	1
0	1	0
1	0	0
1	1	1

Generador de paridad **IMPAR**

Generador / detector de paridad

Ejemplo: Generación y detección de las paridades en el método de Hamming. (atención con la cantidad de niveles en el receptor).

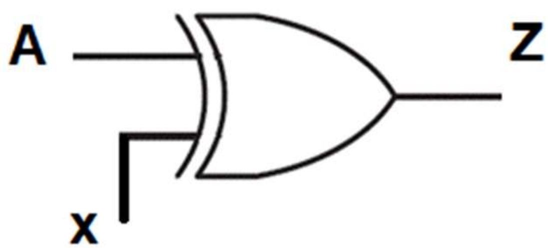


Inversor controlado

Utilizando una de las entradas de una XOR como entrada de control (x) puedo **controlar la inversión** o no del dato presente en la otra entrada (A):

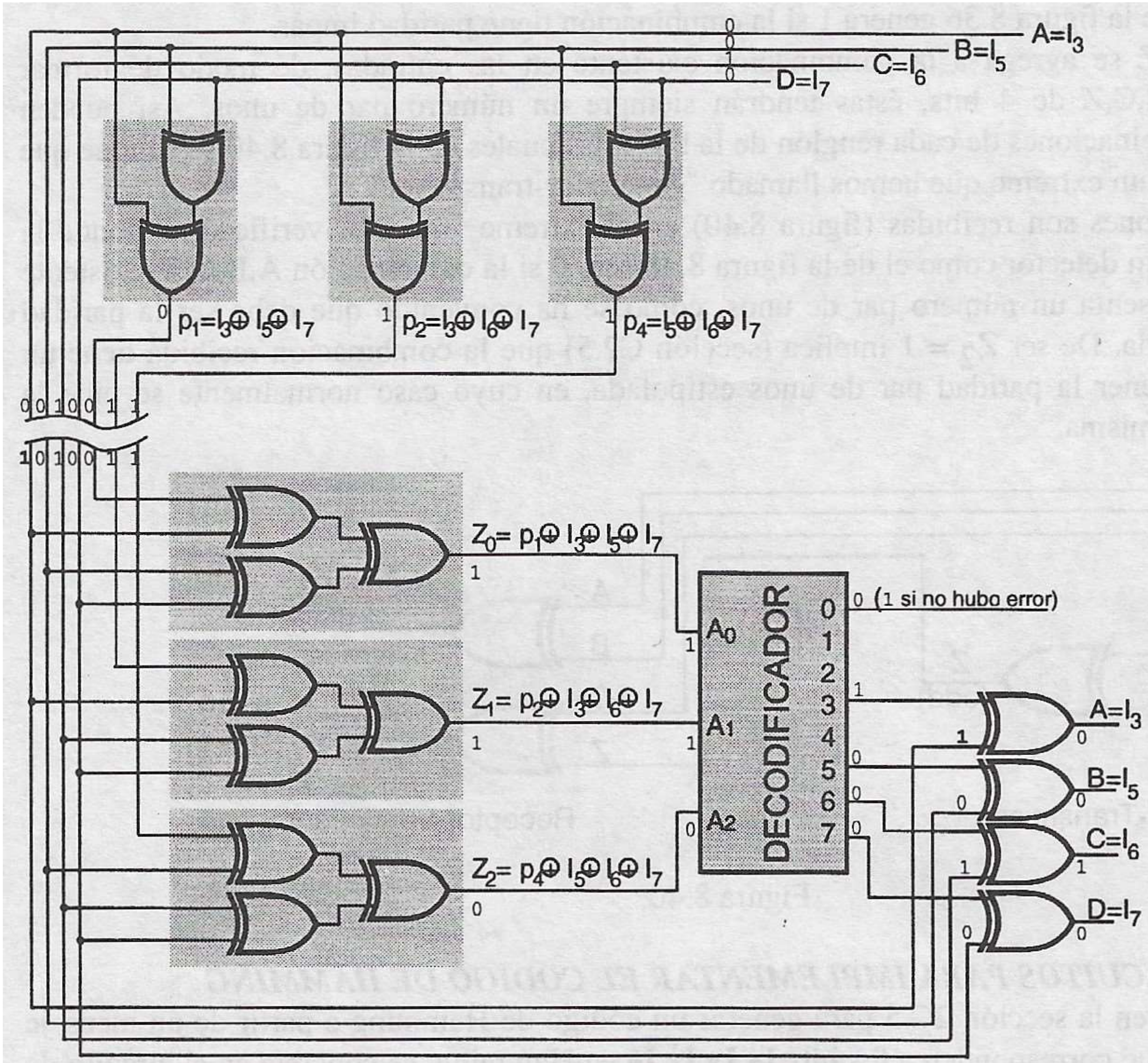
x	Z
0	A
1	Not A

x	A	Z
0	0	0
0	1	1
1	0	1
1	1	0



Circuito de Hamming

Aquí vemos varias de las unidades funcionales vistas:



FACULTAD DE INGENIERIA
Universidad de Buenos Aires