Escuela de Ingenierías Industriales

Titulación: **Grado en Ingeniería Electrónica, Robótica y Mecatrónica**

Asignatura: **Electrónica Digital**

Tema 6: **Bloques Funcionales Combinacionales**

ANDALUCÍA TECH
Campus de Excelencia Internacional
International Campus of Excellence

TEMA 6: BLOQUES FUNCIONALES COMBINACIONALES

6.1. Bloques para el procesamiento y enrutado de datos.

6.1.1 Decodificadores. Codificadores. Conversores de código.

6.1.2 Multiplexores y demultiplexores.

6.1.3 Comparadores

6.2. Bloques aritméticos.

6.2.1 Sumadores binarios: semisumador y sumador completo. Sumador de n bits con acarreo en serie. Sumador de n bits con acarreo anticipado. Sumador/restador.

6.2.2 Multiplicadores binarios.

6.2.3 ALU: Unidad aritmético lógica

6.3. Bloques Reconfigurables. Lógica programable.

6.3.1 Generadores Universales de Funciones Booleanas: ROM, PLA PAL

6.4. Bloques funcionales en HDL

Bloques Aritméticos

Sumadores

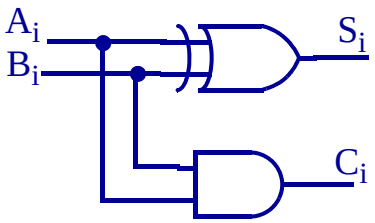
Sumador binario de un bit. Semisumador o “half adder”

A_i B_i	S_i C_i
0 0	0 0
0 1	1 0
1 0	1 0
1 1	0 1

Se tiene por tanto que:

$$S_i = A_i \oplus B_i$$

$$C_i = A_i \cdot B_i$$



No contempla el acarreo de la etapa anterior

Bloques Aritméticos. Sumadores

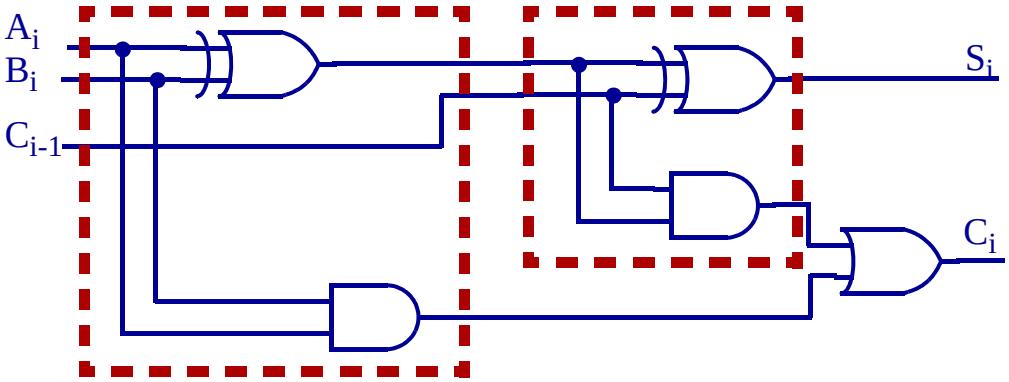
- **Sumador completo.** "Full adder". Incorpora como entrada el acarreo

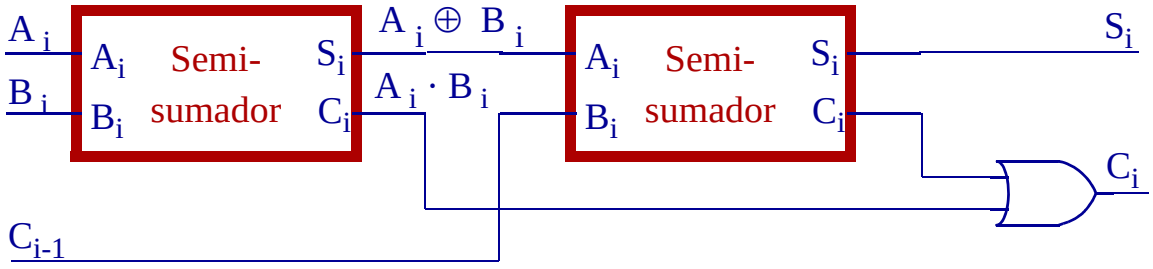
C_{i-1} A_i B_i	S_i C_i
0 0 0	0 0
0 0 1	1 0
0 1 0	1 0
0 1 1	0 1
1 0 0	1 0
1 0 1	0 1
1 1 0	0 1
1 1 1	1 1

Se tiene por tanto que:

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

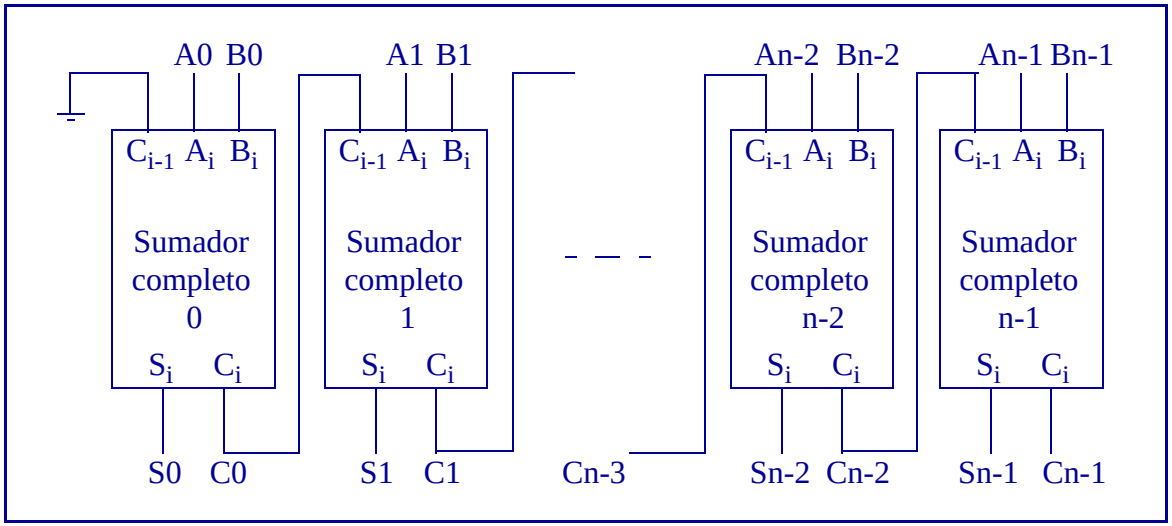
$$C_i = (A_i \oplus B_i) \cdot C_{i-1} + A_i \cdot B_i$$



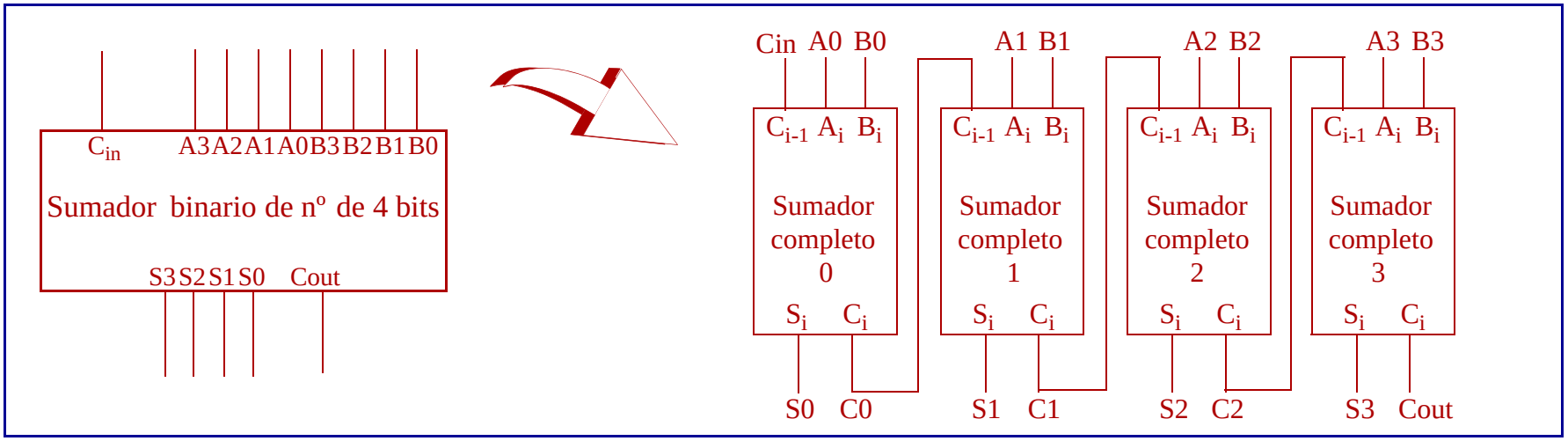


Bloques Aritméticos. Sumadores

- **Sumador paralelo de números binarios de N bits, con acarreo en serie**. "Paralell-adder-ripple-carry".



Ejemplo: Sumador binario paralelo con acarreo en serie de palabras de 4 bits



Para este tipo de sumadores, el retardo de propagación depende de nº etapas. Puede resultar excesivo.

Bloques Aritméticos. Sumadores

- **Sumador paralelo de números binarios de N bits, con acarreo anticipado**. "Paralell-adder-look-ahead-carry".

Para cada etapa sumador completo se tiene que:

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

$$C_i = (A_i \oplus B_i) \cdot C_{i-1} + A_i \cdot B_i$$

"El acarreo de la etapa i-ésima, se obtiene de la suma lógica del acarreo en dicha etapa $A_i \cdot B_i$ y el acarreo de la etapa anterior C_{i-1}, si esta permitido por la salida suma en la etapa i-ésima."

$C_i = G_i + P_i \cdot C_{i-1}$

G_i se denomina término generador de acarreo. $G_i = A_i \cdot B_i$

P_i se denomina término propagador de acarreo. $P_i = A_i \oplus B_i$

$S_i = P_i \oplus C_{i-1}$

G_i y P_i solamente dependen de las entradas de la etapa i-ésima.

Desarrollamos C_i para todo i, deducimos la expresión para un sumador de n etapas.

$$C_0 = G_0 + P_0 \cdot C_{-1}$$
$$C_1 = G_1 + P_1 \cdot C_0 = G_1 + P_1 \cdot (G_0 + P_0 \cdot C_{-1}) = G_1 + P_1 \cdot G_0 + P_1 \cdot P_0 \cdot C_{-1}$$
$$C_2 = G_2 + P_2 \cdot C_1 = G_2 + P_2 \cdot (G_1 + P_1 \cdot C_0) = G_2 + P_2 \cdot G_1 + P_2 \cdot P_1 \cdot G_0 + P_2 \cdot P_1 \cdot P_0 \cdot C_{-1}$$
$$C_3 = G_3 + P_3 \cdot C_2 = G_3 + P_3 \cdot G_2 + P_3 \cdot P_2 \cdot G_1 + P_3 \cdot P_2 \cdot P_1 \cdot G_0 + P_3 \cdot P_2 \cdot P_1 \cdot P_0 \cdot C_{-1}$$

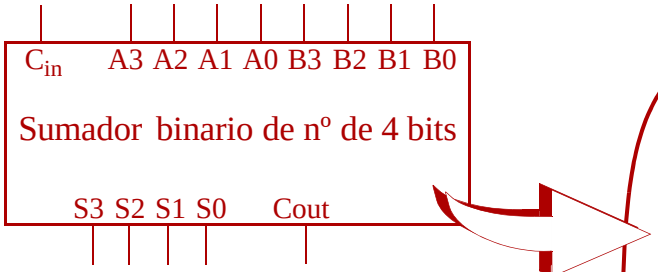
$$C_n = G_n + P_n \cdot G_{n-1} + P_n \cdot P_{n-1} \cdot G_{n-2} + \cdots + P_n \cdot P_{n-1} \cdot \cdots P_1 \cdot G_0 + P_n \cdot P_{n-1} \cdot \cdots P_1 \cdot P_0 \cdot C_{-1}$$
$$S_n = P_n \oplus C_{n-1}$$

Independiza el retardo de propagación del nº de etapas.

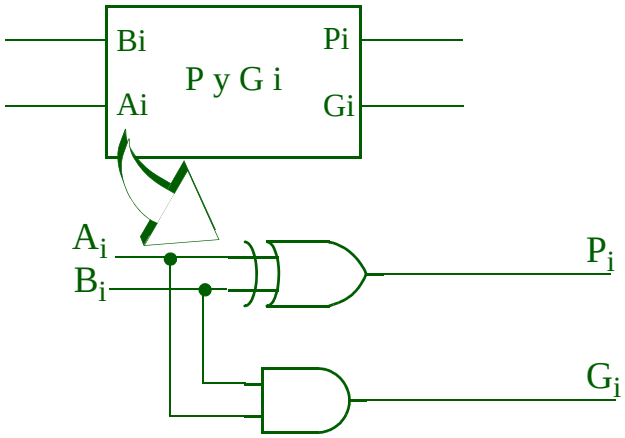
Bloques Aritméticos. Sumadores

- **Sumador paralelo de números binarios de N bits, con acarreo anticipado**. "Paralell-adder-look-ahead-carry".

Ejemplo: Sumador binario paralelo con acarreo anticipado de palabras de 4 bits



Bloque Generador y Propagador



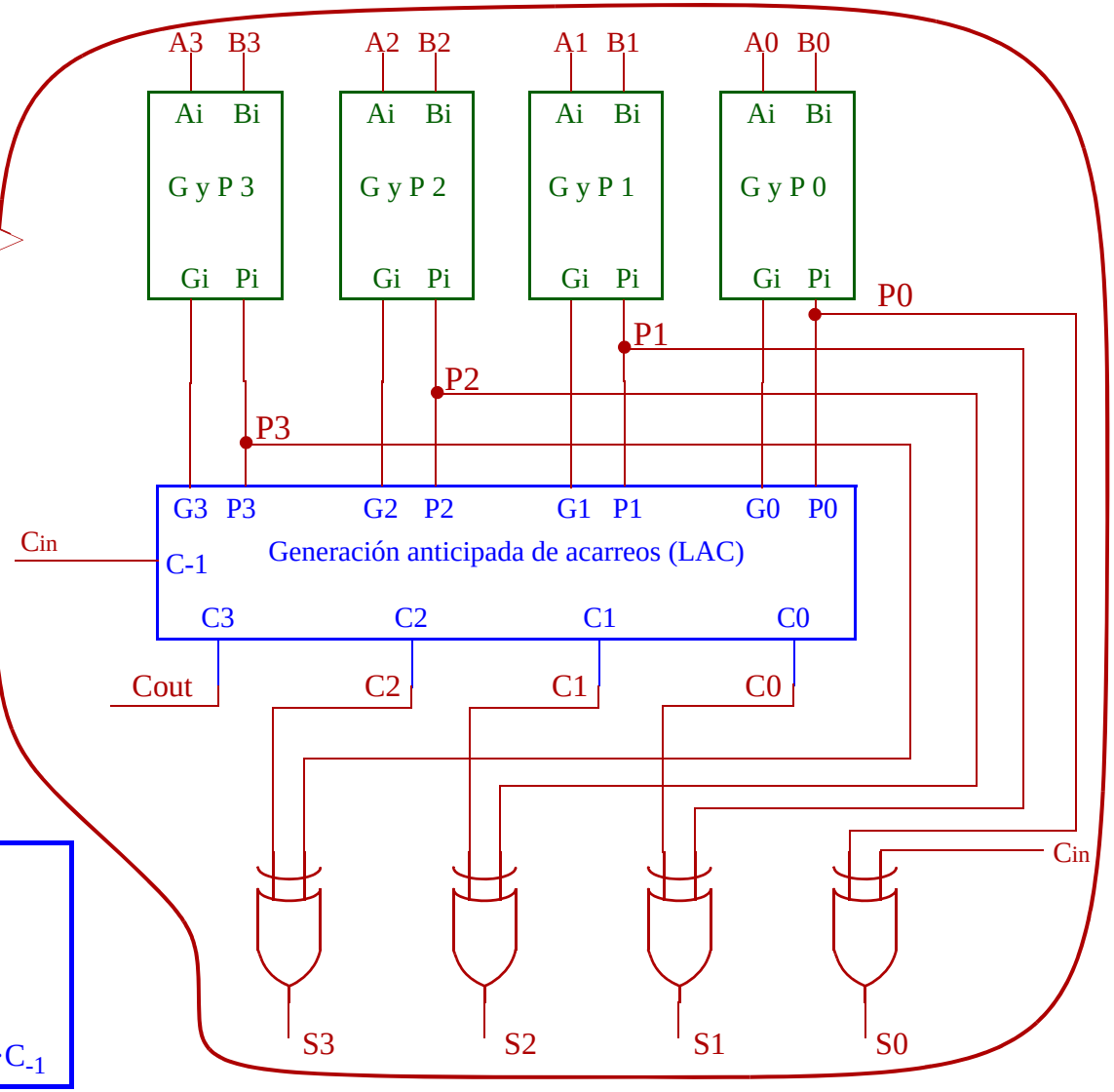
Bloque Generación anticipada de Acarreo (LAC)

$C_0 = G_0 + P_0 \cdot C_{-1}$

$C_1 = G_1 + P_1 \cdot G_0 + P_1 \cdot P_0 \cdot C_{-1}$

$C_2 = G_2 + P_2 \cdot G_1 + P_2 \cdot P_1 \cdot G_0 + P_2 \cdot P_1 \cdot P_0 \cdot C_{-1}$

$C_3 = G_3 + P_3 \cdot G_2 + P_3 \cdot P_2 \cdot G_1 + P_3 \cdot P_2 \cdot P_1 \cdot G_0 + P_3 \cdot P_2 \cdot P_1 \cdot P_0 \cdot C_{-1}$



Bloques Aritméticos. Restadores

Restador.

La operación A-B entre números binarios de n bits se realiza como A+(-B).

Donde -B es el opuesto de B en su representación en complemento a dos.

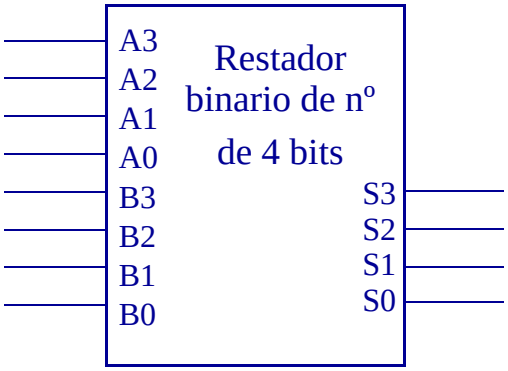
En general se tiene R= A+(-B) = 2^n + (A-B)

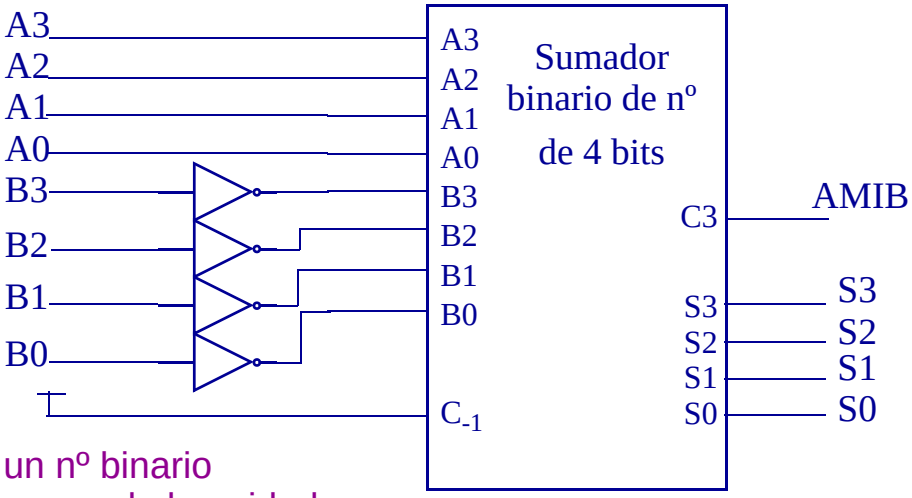
El resultado de esta operación se interpreta según la relación entre |A| y |B|:

- Si |A| ≥ |B|. A-B es cero o un nº positivo. La suma de A con (-B) siempre se produce acarreo, por lo que R = (A-B), despreciando el acarreo (término 2^n).

- Si |A| < |B|. A-B es un nº negativo. La suma de A con (-B) nunca produce acarreo y el resultado R debe interpretarse como un nº negativo expresado en complemento a dos. Esto es, R representa a -(B-A) expresado en complento a dos.

Ejemplo: Diseño de un restador de nº de 4 bits a partir de un bloque sumador.



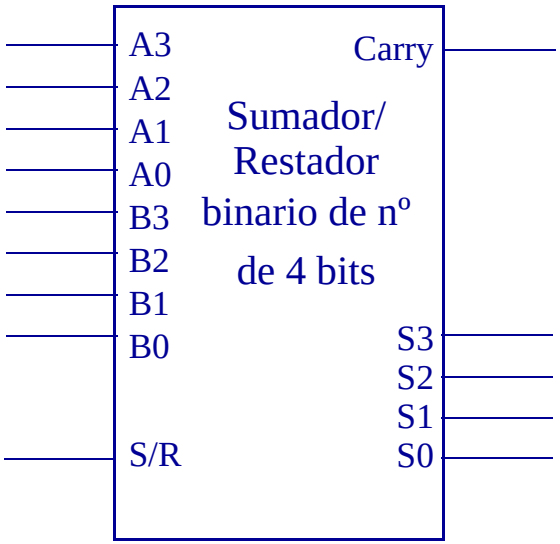


Recuérdese que el complemento a dos de un nº binario puede obtenerse invirtiendo éste bit a bit y sumando la unidad

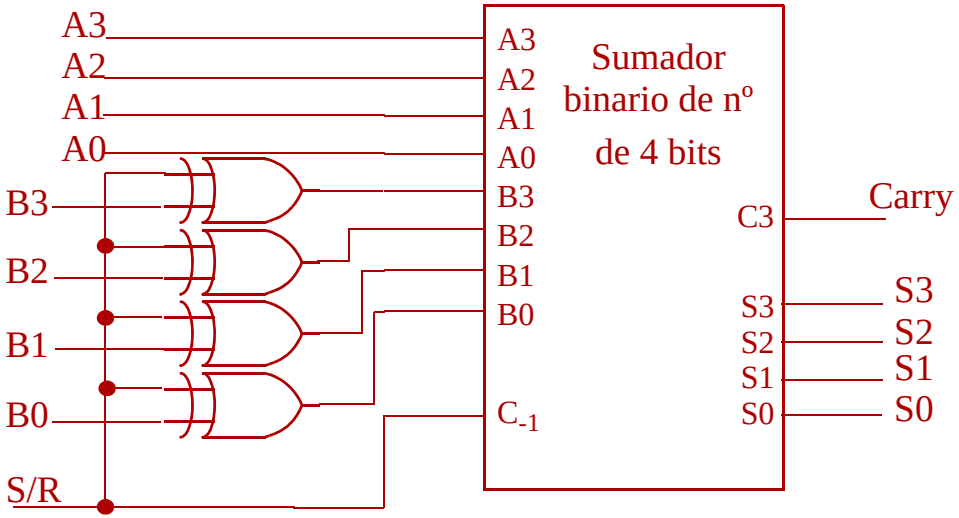
Bloques Aritméticos. Restadores

Bloque Sumador/Restador.

Ejemplo: Sumador/restador de palabras de 4 bits.



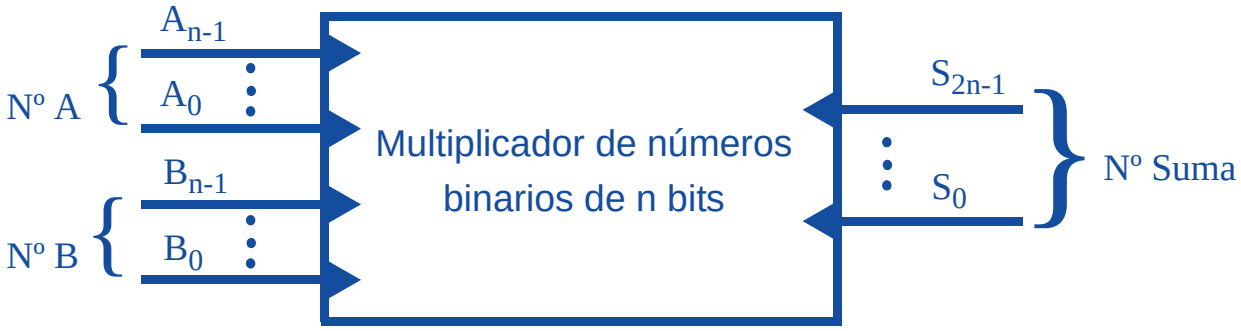
S/R	Operación	Salida
0	A+B	Carry S=A+B
1	A - B	Si $\|A\| \geq \|B\|$ Carry = 1 S= A-B ≥ 0
		Si $\|A\| < \|B\|$ Carry = 0 S= $(A-B)_{C\,D}$ < 0



Bloques Aritméticos. Multiplicadores binarios

Tabla de multiplicar

A B	Producto
0 0	0
0 1	0
1 0	0
1 1	1



Producto = AxB = AND(A,B)

Ej: Nº 4 Bits P = BxA

$B = b_3 b_2 b_1 b_0$

$A = a_3 a_2 a_1 a_0$

$$\begin{array}{cccccccc} & & b_3 & b_2 & b_1 & b_0 & & \\ \times & & a_3 & a_2 & a_1 & a_0 & & \\ \hline & & a_0xb_3 & a_0xb_2 & a_0xb_1 & a_0xb_0 & & \\ & a_1xb_3 & a_1xb_2 & a_1xb_1 & a_1xb_0 & & & \\ & a_2xb_3 & a_2xb_2 & a_2xb_1 & a_2xb_0 & & & \\ + & a_3xb_3 & a_3xb_2 & a_3xb_1 & a_3xb_0 & & & \\ \hline P_7 & P_6 & P_5 & P_4 & P_3 & P_2 & P_1 & P_0 \end{array}$$

$P_0 = a_0xb_0$

$P_1 = a_0xb_1 + a_1xb_0$

$P_2 = a_0xb_2 + a_1xb_1 + a_2xb_0 + C_1$

$P_3 = a_0xb_3 + a_1xb_2 + a_2xb_1 + a_3xb_0 + C_2$

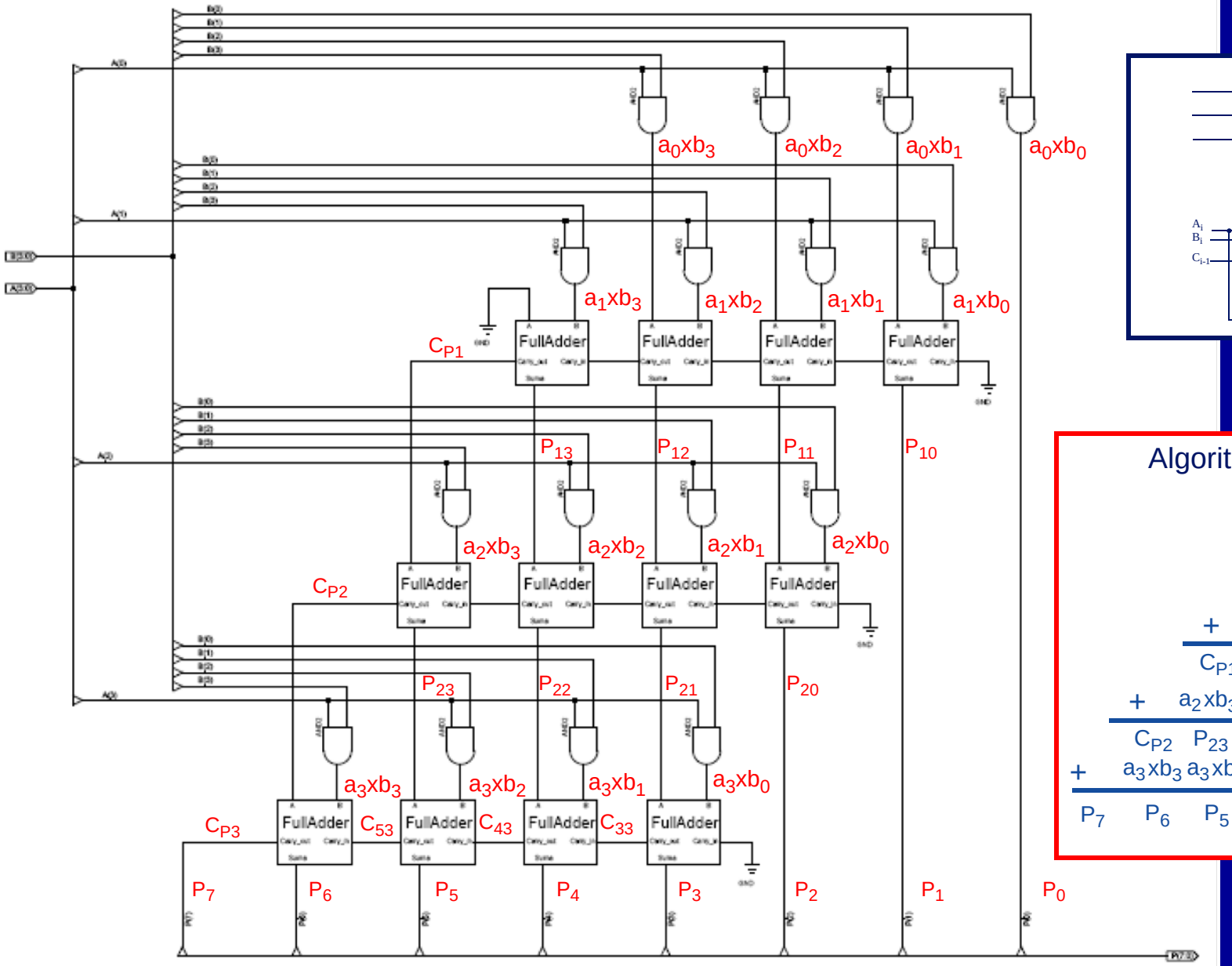
$P_4 = a_1xb_3 + a_2xb_2 + a_3xb_1 + C_3$

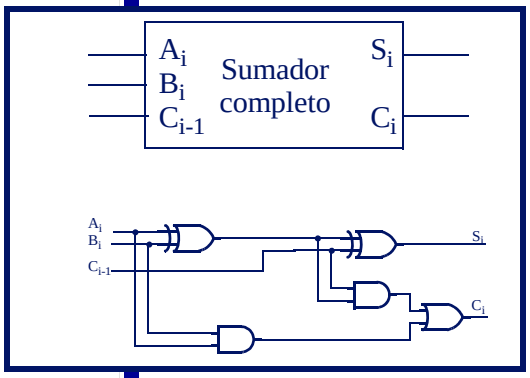
$P_5 = a_2xb_3 + a_3xb_2 + C_4$

$P_6 = a_3xb_3 + C_5$

$P_7 = C_6$

Multiplicadores binarios: Multiplicador con acarreo en serie





Algoritmo de Multiplicación

$$
\begin{array}{cccccccc}
 & & b_3 & b_2 & b_1 & b_0 & & \\
\times & & a_3 & a_2 & a_1 & a_0 & & \\
\hline
 & & a_0xb_3 & a_0xb_2 & a_0xb_1 & a_0xb_0 & & \\
+ & a_1xb_3 & a_1xb_2 & a_1xb_1 & a_1xb_0 & & & \\
\hline
 & C_{P1} & P_{13} & P_{12} & P_{11} & P_{10} & & \\
+ & a_2xb_3 & a_2xb_2 & a_2xb_1 & a_2xb_0 & & & \\
\hline
 & C_{P2} & P_{23} & P_{22} & P_{21} & P_{20} & & \\
+ & a_3xb_3 & a_3xb_2 & a_3xb_1 & a_3xb_0 & & & \\
\hline
P_7 & P_6 & P_5 & P_4 & P_3 & P_2 & P_1 & P_0
\end{array}
$$

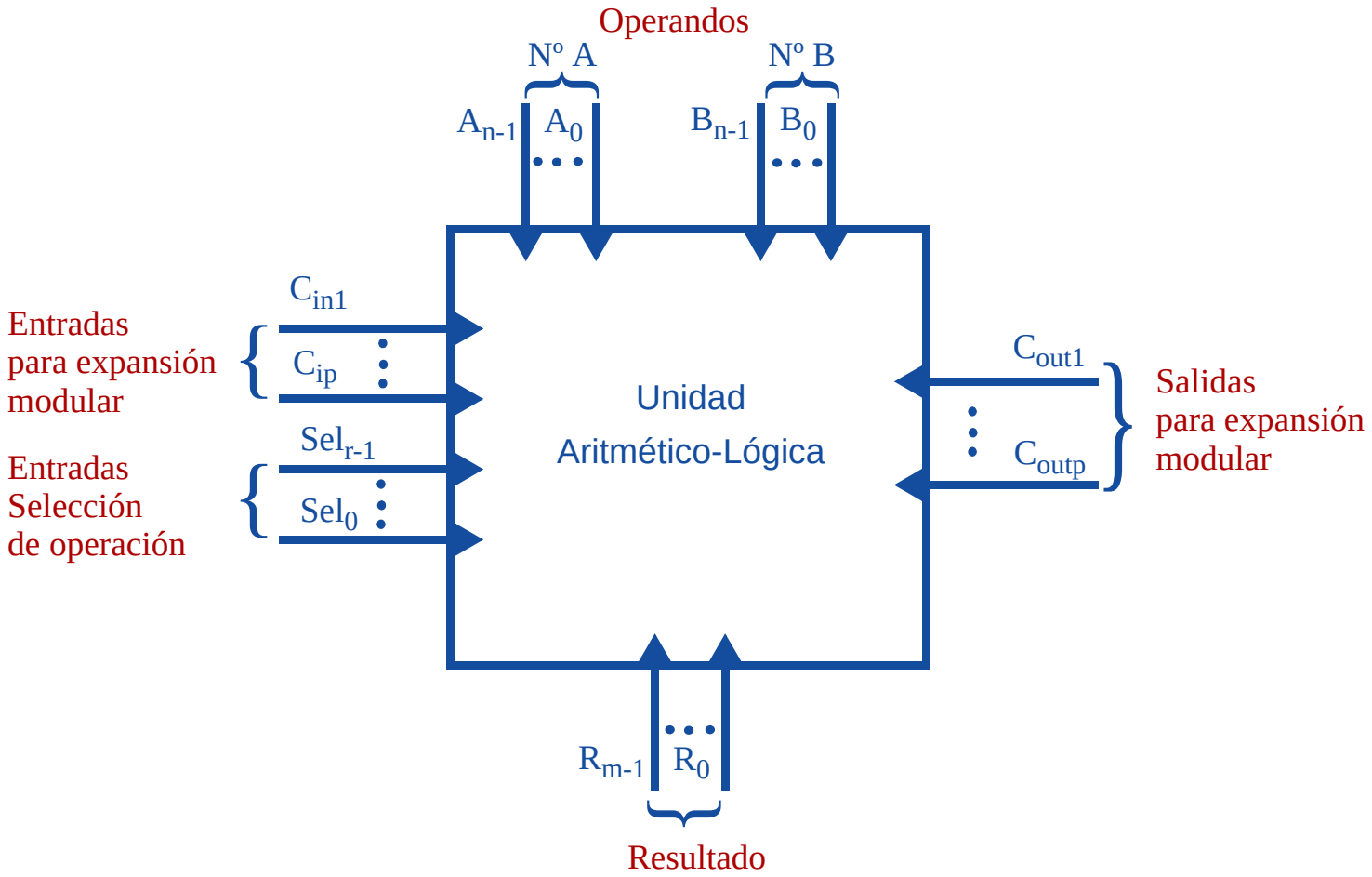
Unidad Aritmético-lógica: ALU (Arithmetic-Logic Unit)

Bloque **MSI** combinacional, que ofrece la posibilidad de realizar diversas operaciones aritméticas y lógicas, que se seleccionan mediante señales de control.

Actua sobre dos operandos de N bits, y proporciona como resultado una palabra de M bits.

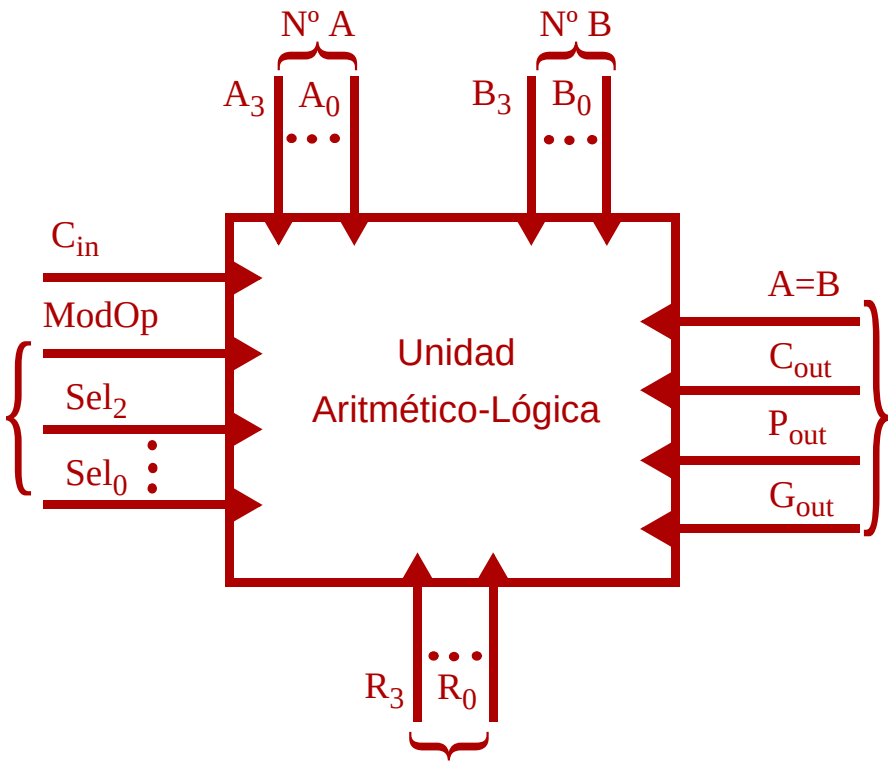
Incluye además entradas y salidas que facilitan su expansión modular.

Es un bloque fundamental en todos los procesadores digitales.



Unidad Aritmético-lógica

Ej: ALU 4 bits.



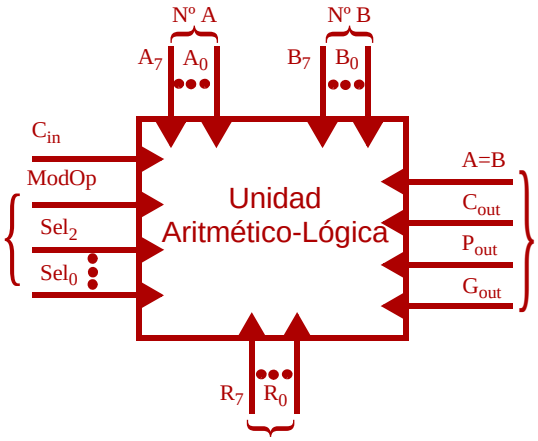
SIgnificado de las señales I/O

A[3.0]: Operando A de 4 bits
B[3.0]: Operando B de 4 bits
R[3.0]: Resultado de la operación (4 bits)
C_{in}: entada de acarreo precedente
ModOp: entrada de modo de operación Aritm/Lógic
Sel[2:0]: entrada de selec operación según ModOp
A=B: Salida que indica igualdad de operandos
C_{out}: salida de accarreo (expansión acarreo en serie)
P_{out}: Salida término propagador de accarreo
G_{out}: Salida termino generador accarreo
(ambas para expansión con look-ahead-carry)

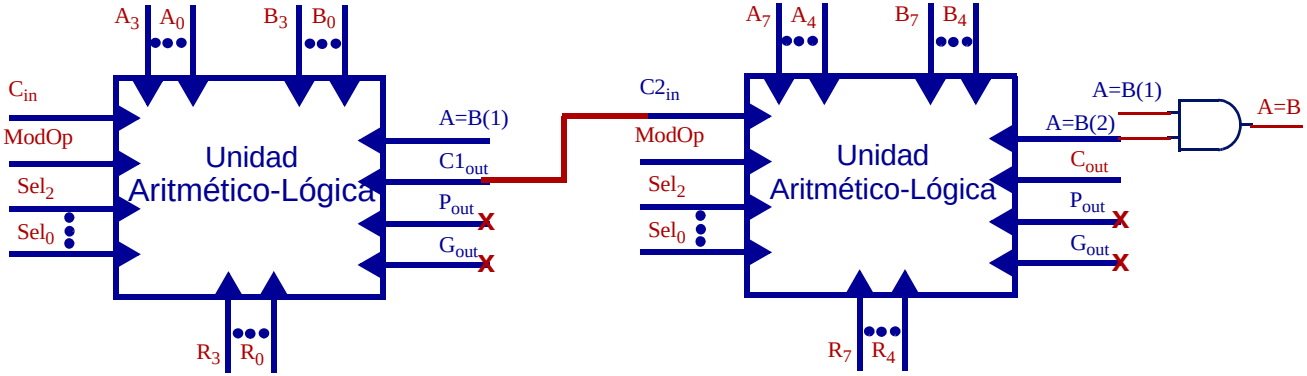
Tabla de Operaciones	
Operaciones Aritméticas (Sobre operandos de 4 bits)	**Operaciones Lógicas (Operación bit a bit)**
A+B **A+B+1** **A+1** **A-B** **A-B-1** **CD A** **A-1**	**OR(A,B)** **AND(A,B)** **NOR(A,B)** **NAND(A,B)** **XOR(A,B)** **XNOR(A,B)** **OR(NOT(A),B)**

Unidad Aritmético-lógica: Expansión de bloques

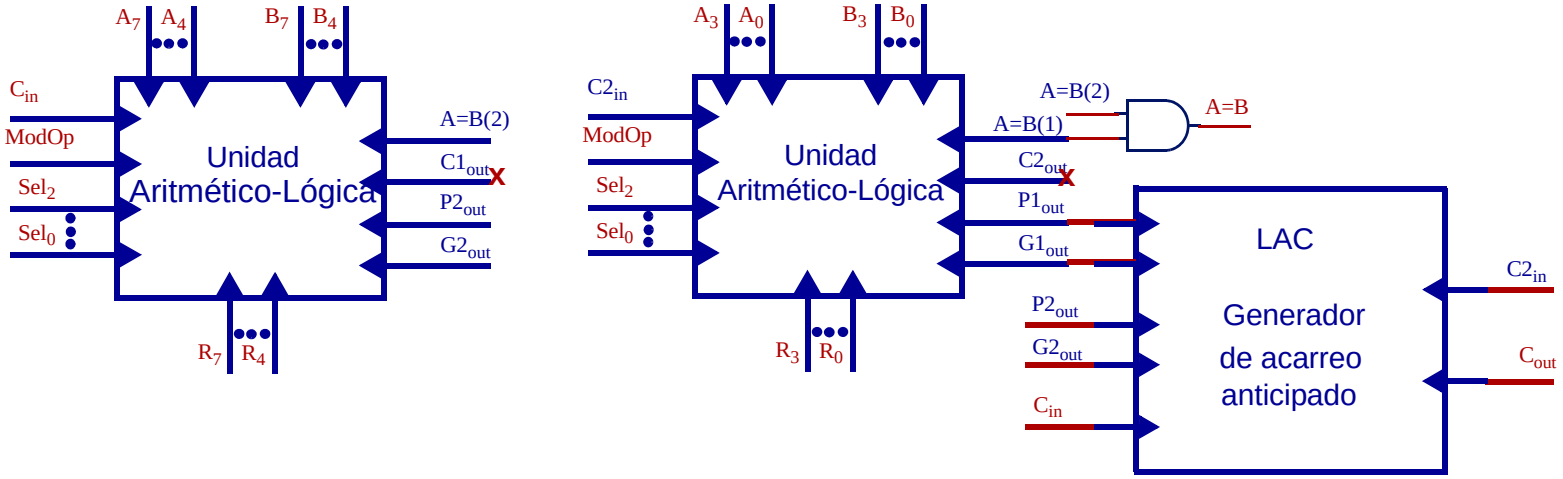
Ej: ALU 8 bits



Asociación de ALU´s de 4bits con acarreo en serie



Asociación de ALU´s de 4bits con acarreo anticipado



Unidad Aritmético-lógica. Ejemplo de implementación

Ej: ALU 4 bits

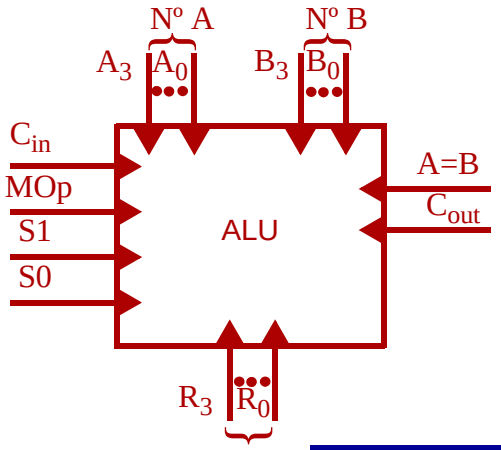
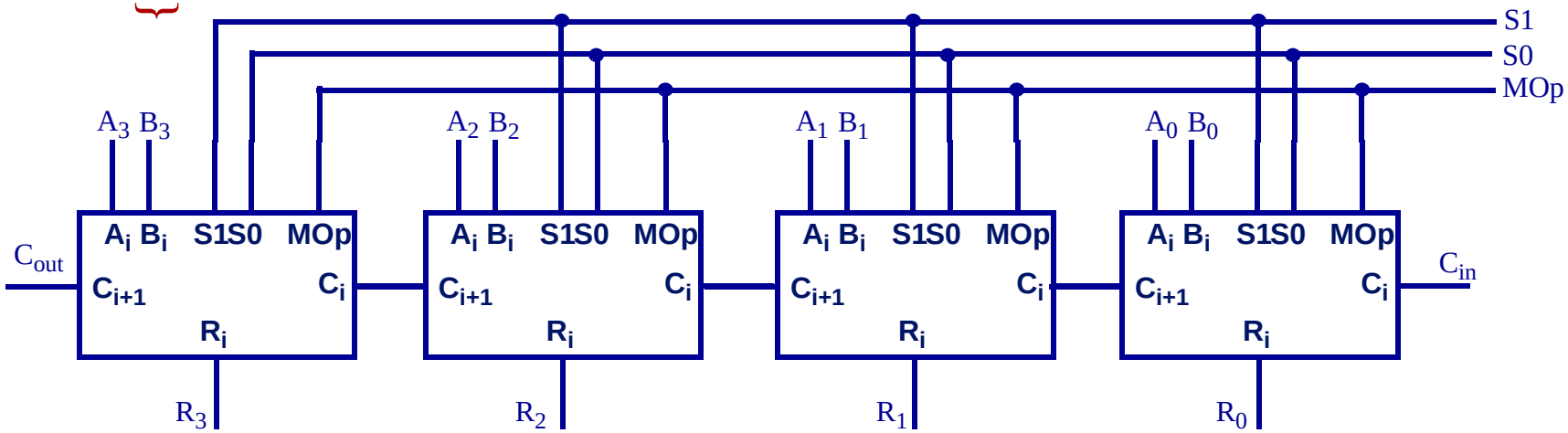
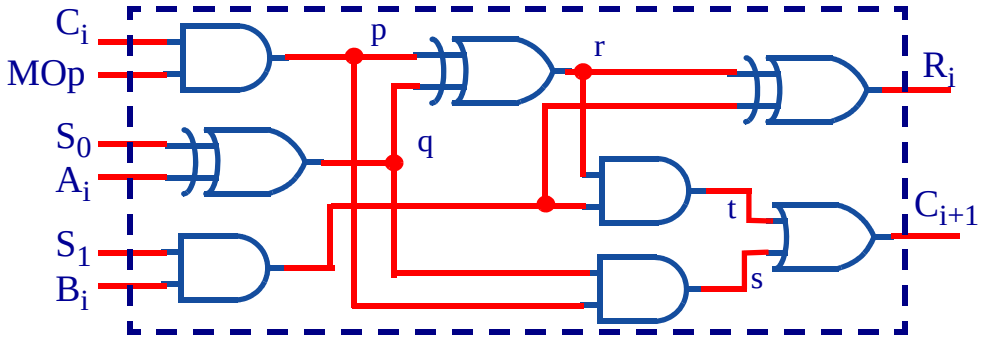


Tabla de Operaciones

	Op. Lógicas	Op. Aritméticas	
S1 S0	(MOp = 0)	(MOp = 1, C_{in} = 0)	(MOp = 1, C_{in} = 1)
0 0	$F_i = A_i$	F = A	F = A + 1
0 1	$F_i = NOT(A_i)$	$F = \overline{A}$	$F = \overline{A} + 1$
1 0	$F_i = XOR(A_i,B_i)$	F = A + B	F = A + B + 1
1 1	$F_i = XOR(NOT(A_i),B_i)$	$F = \overline{A}+B$	$F = \overline{A}+B + 1$





Si MOp = 0 Op. Lógica. No se propaga el acarreo

$$R_i = (S_0 \oplus A_i) \oplus S_1 B_1$$

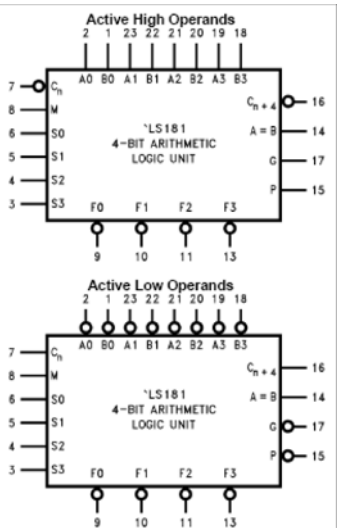
Si MOp = 1 Op. Aritmética. Se propaga el acarreo

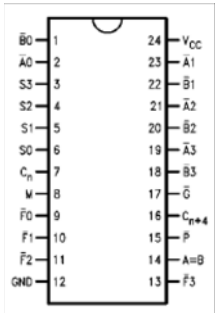
$$R_i = C_i \oplus (S_0 \oplus A_i) \oplus S_1 B_1$$

$$C_{i+1} = C_i(S_0 \oplus A_i) + (S_1 B_1)(C_i \oplus S_0 \oplus A_i)$$

Unidad Aritmético-lógica

Ej: ALU 4 bits comercial: 74LS182





Pin Names	Description
$\overline{A}0$–$\overline{A}3$	Operand Inputs (Active LOW)
$\overline{B}0$–$\overline{B}3$	Operand Inputs (Active LOW)
S0–S3	Function Select Inputs
M	Mode Control Input
C_n	Carry Input
$\overline{F}0$–$\overline{F}3$	Function Outputs (Active LOW)
A = B	Comparator Output
$\overline{G}$	Carry Generate Output (Active LOW)
$\overline{P}$	Carry Propagate Output (Active LOW)
C_{n+4}	Carry Output

Function Table

Mode Select Inputs				Active LOW Operands & F_n Outputs		Active HIGH Operands & F_n Outputs	
				Logic	Arithmetic (Note 2)	Logic	Arithmetic (Note 2)
S3	S2	S1	S0	(M = H)	(M = L) (C_n = L)	(M = H)	(M = L) (C_n = H)
L	L	L	L	$\overline{A}$	A minus 1	$\overline{A}$	A
L	L	L	H	$\overline{AB}$	AB minus 1	$\overline{A}+\overline{B}$	A + B
L	L	H	L	$\overline{A}+\overline{B}$	$A\overline{B}$ minus 1	$\overline{A}B$	$A+\overline{B}$
L	L	H	H	Logic 1	minus 1	Logic 0	minus 1
L	H	L	L	$\overline{A}+\overline{B}$	A plus (A + $\overline{B}$)	$\overline{AB}$	A plus $A\overline{B}$
L	H	L	H	$\overline{B}$	AB plus (A + $\overline{B}$)	$\overline{B}$	(A + B) plus $A\overline{B}$
L	H	H	L	$\overline{A}\oplus\overline{B}$	A minus B minus 1	$A\oplus B$	A minus B minus 1
L	H	H	H	$A+\overline{B}$	$A+\overline{B}$	$A\overline{B}$	AB minus 1
H	L	L	L	$\overline{A}B$	A plus (A + B)	$\overline{A}+B$	A plus AB
H	L	L	H	$A\oplus B$	A plus B	$\overline{A}\oplus\overline{B}$	A plus B
H	L	H	L	B	$A\overline{B}$ plus (A + B)	B	(A + $\overline{B}$) plus AB
H	L	H	H	A + B	A + B	AB	AB minus 1
H	H	L	L	Logic 0	A plus A (Note 1)	Logic 1	A plus A (Note 1)
H	H	L	H	$A\overline{B}$	AB plus A	$A+\overline{B}$	(A + B) plus A
H	H	H	L	$A\overline{B}$	$A\overline{B}$ minus A	$A+\overline{B}$	(A + $\overline{B}$) plus A
H	H	H	H	A	A	A	A minus 1

Note 1: Each bit is shifted to the next most significant position.
Note 2: Arithmetic operations expressed in 2s complement notation.