4. CIRCUITOS ARITMÉTICOS

upna
Universidad
Pública de Navarra
Nafarroako
Unibertsitate Publikoa

4. CIRCUITOS ARITMÉTICOS

- **Introducción**

Suma y producto lógico:

Hemos visto que el conjunto B {0,1} y las operaciones (+,·) se definen como

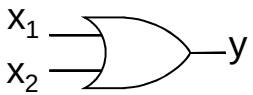
Suma lógica

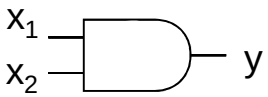
+	0	1
0	0	1
1	1	1

Producto lógico

·	0	1
0	0	0
1	0	1

Para representarlas en forma de circuito digital basta con emplear una puerta AND y una OR





Suma y producto aritmético:

La suma y el producto aritmético obedecen a las siguientes tablas:

Suma aritmética

+	0	1
0	0	1
1	1	2

Producto aritmético

·	0	1
0	0	0
1	0	1

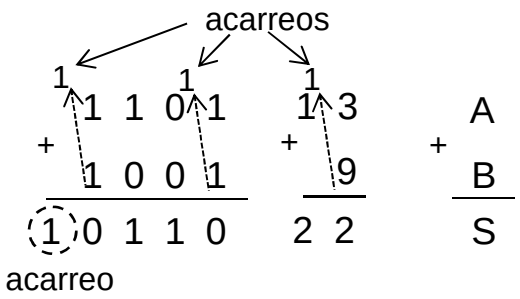
Para el producto aritmético la tabla es la misma que para el producto lógico, pero ya veremos que cuando se multiplican números de varios bit la cosa es diferente y no basta con emplear puertas AND

En cuanto a la suma aritmética, se aprecia la diferencia con respecto a la suma lógica. Al sumar 1+1 el resultado es 2 y se requiere un bit adicional (llamado bit de acarreo)

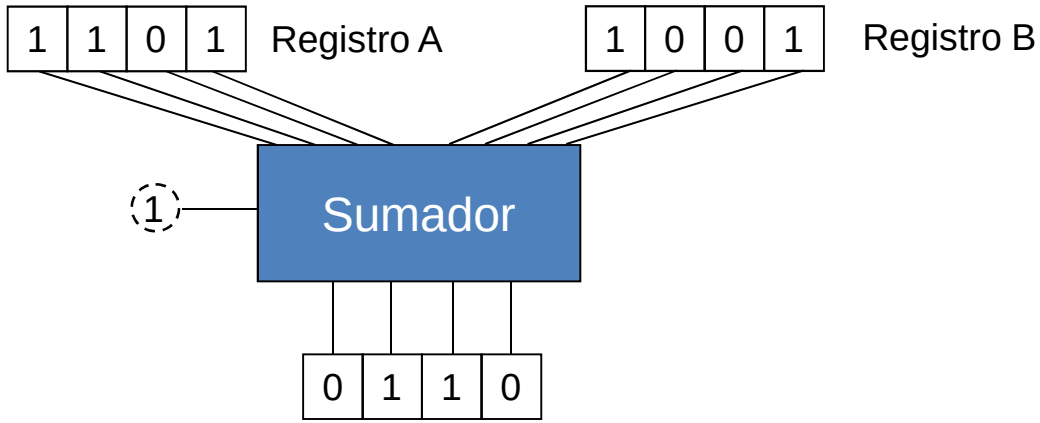
En general los circuitos aritméticos tienen una cierta complejidad que se abordará en este capítulo

- **Suma en binario**

Se va sumando cada pareja de bits con el acarreo de la pareja anterior



Los números A y B y el resultado S se almacenan en registros



Si el registro es de n bits ⇨ el número más grande que puede ser almacenado es 2^n-1

- Si $S = A+B <= 2^n-1$ ⇨ no hay problema (con n bits se puede representar el número resultante)
- Si $S = A+B > 2^n-1$ ⇨ desbordamiento (en inglés overflow; con n bits no se puede representar el número resultante)

$S = A+B-2^n$

- **Ejemplos de suma en binario**

  Para un registro de 4 bits, el número más grande es 2^4-1=15, que en binario es 1111

  Suma sin desbordamiento:

```
  1 0 1 0     1 0     A
+           +       +
  0 1 0 0       4     B
-------     ---     ---
  1 1 1 0     1 4     S
```

  Suma con desbordamiento:

```
  1   1   1     1 3     A
  1 1 0 1     1 9     B
+ 1 0 0 1   +       +
-------     ---     ---
1 0 1 1 0     2 2     S
```

- **Resta en binario**

  Se va restando cada pareja de bits con la llevada de la pareja anterior

```
    1 1
  1 1 0 1 1     2 7     A
-             -       -
  0 1 1 0 1     1 3     B
-----------     ---     ---
  0 1 1 1 0     1 4     S
```

  También se puede convertir la resta en suma de un número negativo. De esta manera se evitar realizar la operación de la resta

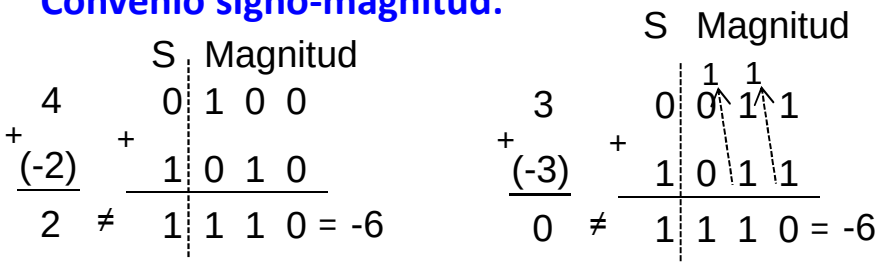
  En el capítulo 2 se vio que los números se representan de forma negativa de tres maneras:

  - Convenio signo magnitud
  - Complemento A 1
  - Complemento A 2

  Como se verá a continuación, no se puede realizar la resta como suma con todos ellos

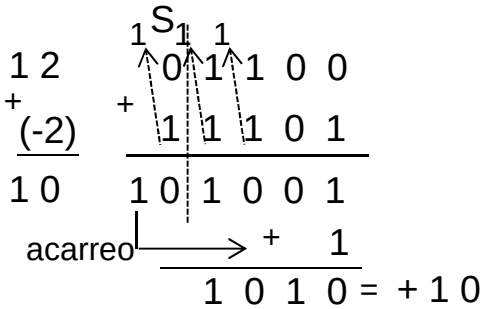
- **Resta como suma de número negativo**

Convenio signo-magnitud:

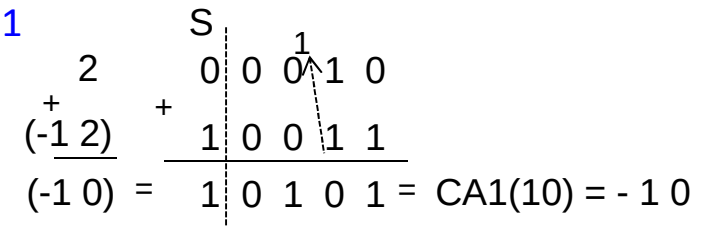


Convenio complemento A 1: (cuando la suma da acarreo se añade un 1 al resultado)

CA1(2)=11101

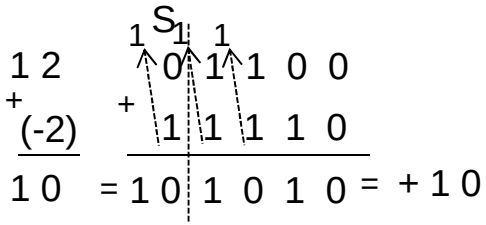


CA1(12)=10011

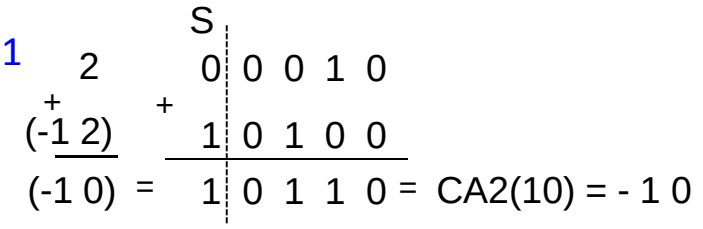


Convenio complemento A 2:

CA2(2)=11110

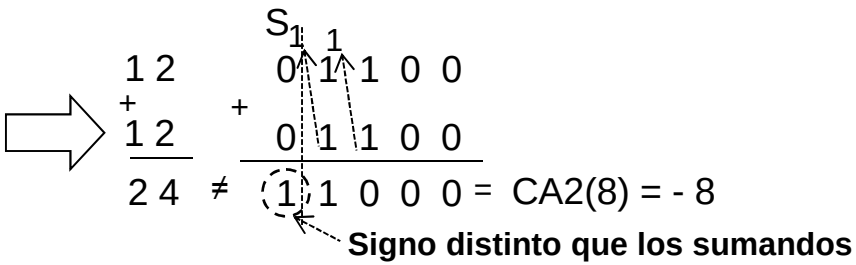


CA2(12)=10011



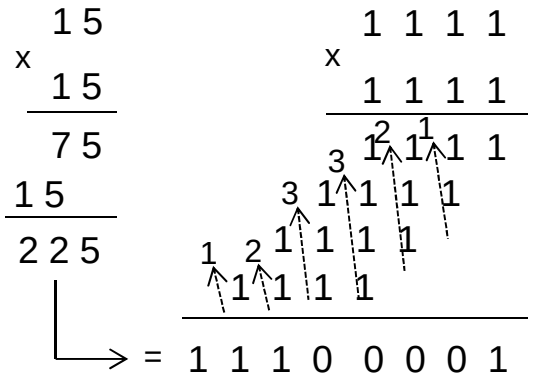
Desbordamiento en complemento A 1 y A 2:

Cuando los dos sumandos son positivos o negativos y el signo del resultado es contrario al de los sumandos



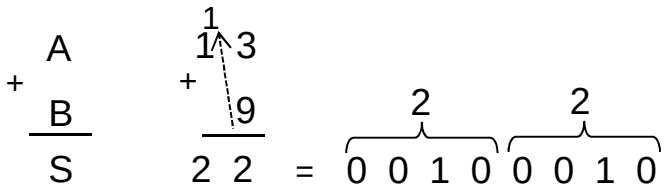
- **Multiplicación binaria**

  Se realiza como la decimal pero con dígitos binarios

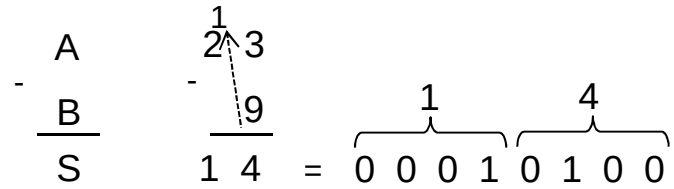


- **Operaciones aritméticas en BCD**

  Suma: se suma en decimal y el resultado se convierte a BCD

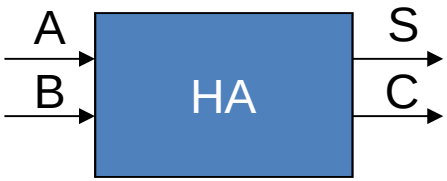


  Resta: se resta en decimal y el resultado se convierte a BCD



- **Semisumador básico (en inglés half adder - HA)**

Suma 1 bit + 1 bit



S (suma – en inglés sum)

C (acarreo – en inglés carry)

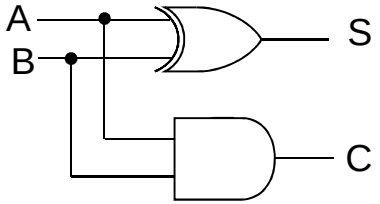
Tabla de verdad

A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Circuito:

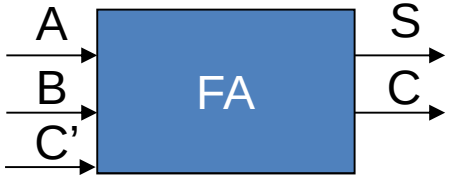
$S = \overline{A}\cdot B + A\cdot\overline{B} = A \oplus B$

$C = A\cdot B$



- **Sumador completo (en inglés full adder - FA)**

Suma 1 bit + 1 bit + acarreo (C')



S (suma – en inglés sum)

C (acarreo – en inglés carry)

Tabla de verdad

A	B	C'	S	C
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

A B \ C'	00	01	11	10
0	0 (0)	1 (2)	0 (6)	1 (4)
1	1 (1)	0 (3)	1 (7)	0 (5)

C

A B \ C'	00	01	11	10
0	0 (0)	0 (2)	1 (6)	0 (4)
1	0 (1)	1 (3)	1 (7)	1 (5)

$S = \overline{A}\cdot\overline{B}\cdot C' + \overline{A}\cdot B\cdot\overline{C'} + A\cdot B\cdot C' + A\cdot\overline{B}\cdot\overline{C'} = A \oplus B \oplus C$

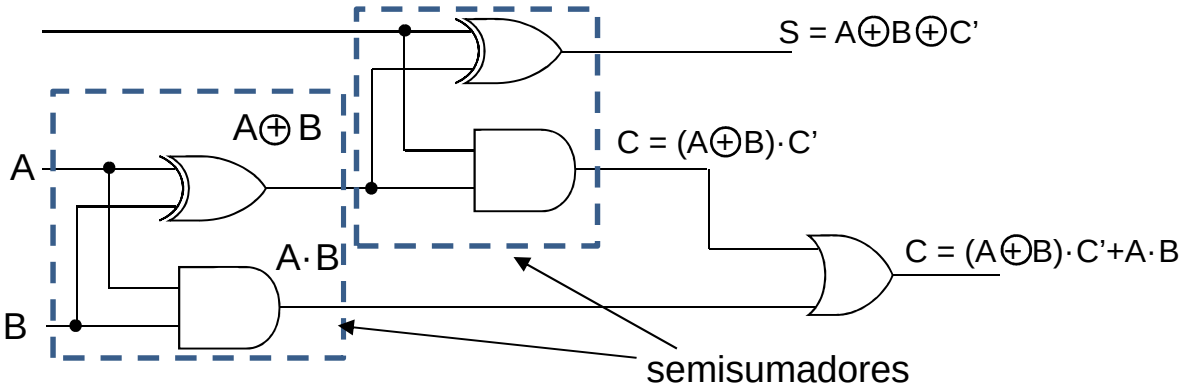
$C = A\cdot B + A\cdot C' + B\cdot C'$

- **Sumador completo (FA) usando semisumadores (HA)**

Si se obtiene la ecuación del acarreo de otra manera:

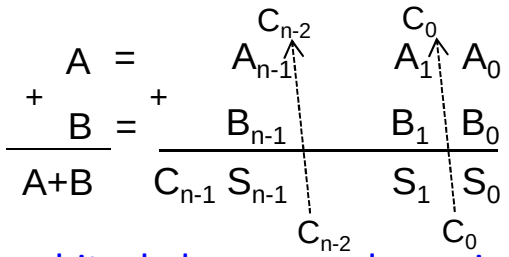
$$C = \overline{A}\cdot B\cdot C' + A\cdot\overline{B}\cdot C' + A\cdot B\cdot\overline{C}' + A\cdot B\cdot C' = (\overline{A}\cdot B + A\cdot\overline{B})\cdot C' + AB\cdot\underbrace{(C'+\overline{C}')}_{1} = (A \oplus B)\cdot C' + A\cdot B$$

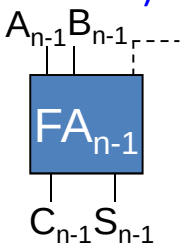
Entonces se puede representar el sumador completo con dos semisumadores HA y una puerta OR

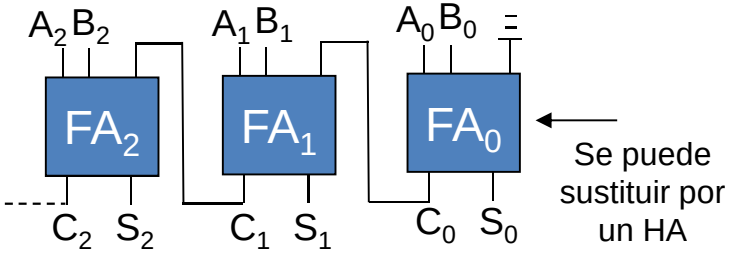


- **Sumador en paralelo con acarreo serie**

Suma dos números de n bits cada uno mediante n sumadores completos (también se puede sustituir uno de los sumadores completos por un semisumador):







Los bits de los sumandos se introducen simultáneamente.

Ventaja: simplicidad Desventaja: lentitud (retardo n veces el tiempo de retardo de un FA)

- **Sumador en paralelo con acarreo paralelo**

  Se emplea para reducir el tiempo del sumador con acarreo serie, pero es más complejo

  Utiliza funciones de generación y propagación de acarreo:

		A_i	B_i	C_{i-1}	S_i	C_i		G_i	P_i
Sumando 0+0	0	0	0	0	0	0	No se genera acarreo C_i=0	0	0
	1	0	0	1	1	0		0	0
Sumando 0+1	2	0	1	0	1	0	Se propaga el acarreo C_i C_i=C_{i-1}	0	1
	3	0	1	1	0	1		0	1
Sumando 1+0	4	1	0	0	1	0		0	1
	5	1	0	1	0	1		0	1
Sumando 1+1	6	1	1	0	0	1	Se genera acarreo C_i=1	1	0
	7	1	1	1	1	1		1	0

G_i

C_{i-1} \ A_iB_i	00	01	11	10
0	0 (0)	0 (2)	1 (6)	0 (4)
1	0 (1)	0 (3)	1 (7)	0 (5)

$$G_i = A_i \cdot B_i$$

P_i

C_{i-1} \ A_iB_i	00	01	11	10
0	0 (0)	1 (2)	0 (6)	1 (4)
1	0 (1)	1 (3)	0 (7)	1 (5)

$$P_i = \overline{A_i} \cdot B_i + A_i \cdot \overline{B_i} = A_i \oplus B_i$$

Ci en función de Gi y Pi

$$C_i = \overline{A_i} \cdot B_i \cdot C_{i-1} + A_i \cdot \overline{B_i} \cdot C_{i-1} + A_i \cdot B_i \cdot \overline{C_{i-1}} + A_i \cdot B_i \cdot C_{i-1} = \underbrace{(\overline{A_i} \cdot B_i + A_i \cdot \overline{B_i})}_{A_i \oplus B_i} \cdot C_{i-1} + A_i \cdot B_i \underbrace{(\overline{C_{i-1}} + C_{i-1})}_{1} = (A_i \oplus B_i) \cdot C_{i-1} + A_i \cdot B_i = G_i + P_i \cdot C_{i-1}$$

$$C_1 = G_1 + P_1 \cdot C_0$$

$$C_2 = G_2 + P_2 \cdot C_1 = G_2 + P_2 \cdot (G_1 + P_1 \cdot C_0) = G_2 + P_2 \cdot G_1 + P_2 \cdot P_1 \cdot C_0$$

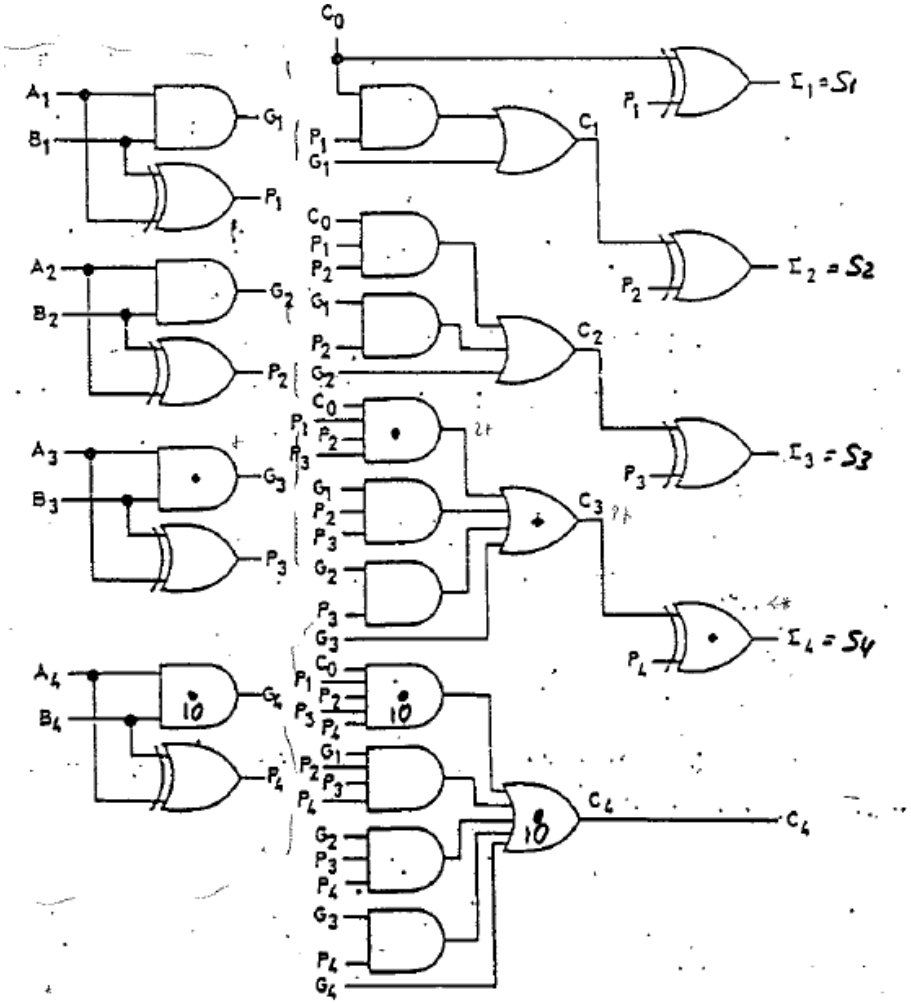
$$C_3 = G_3 + P_3 \cdot C_2 = G_3 + P_3 \cdot G_2 + P_3 \cdot P_2 \cdot G_1 + P_3 \cdot P_2 \cdot P_1 \cdot C_0$$

$$C_4 = G_4 + P_4 \cdot C_3 = G_4 + P_4 \cdot G_3 + P_4 \cdot P_3 \cdot G_2 + P_4 \cdot P_3 \cdot P_2 \cdot G_1 + P_4 \cdot P_3 \cdot P_2 \cdot P_1 \cdot C_0$$

S_i en función de G_i y P_i

$S_i = \underbrace{A_i \oplus B_i}_{P_i} \oplus C_{i-1} = P_i \oplus C_{i-1}$ $\quad S_1 = P_1 \oplus C_0$ $\quad S_2 = P_2 \oplus C_1$ $\quad S_3 = P_3 \oplus C_2$ $\quad S_4 = P_4 \oplus C_3$

Sumador paralelo de 4 bits con acarreo paralelo (el comercial es el 7483A: http://bit.ly/1lddPUg):



El acarreo y la suma pasan respectivamente por 3 y 4 puertas sin importar el número de bits que se suman

Ventaja: es más rápido que el sumador con acarreo serie

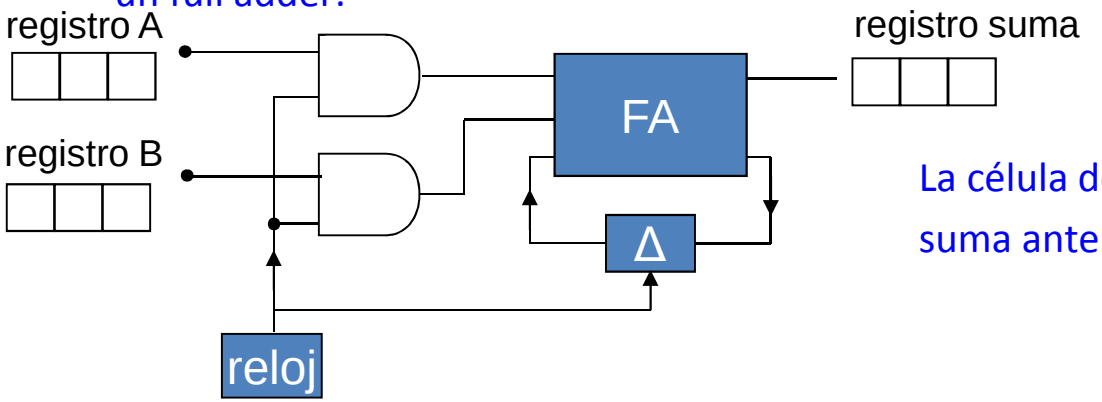
Inconveniente: es muy complejo (muchas puertas lógicas)

- **Sumador en serie**

Existen dos formas de sumar:

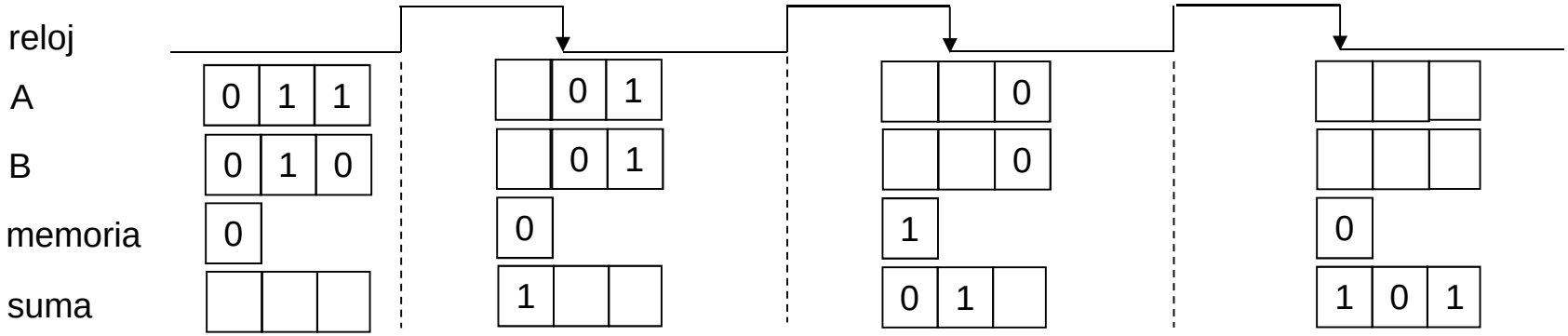
En paralelo: es la que se ha visto hasta ahora. Las parejas de bits (1 bit de cada sumando) se introducen simultáneamente en el circuito. En el caso del sumador con acarreo serie se emplean tantos Full Adder como bits tienen los sumandos

En serie: las parejas de bits se introducen sucesivamente (primero el bit menos significativo de cada sumando, luego el segundo menos significativo, hasta el más significativo). Esto provoca una mayor lentitud de respuesta. La gran ventaja frente a los dos sumadores paralelo vistos es que sólo emplea un full adder:



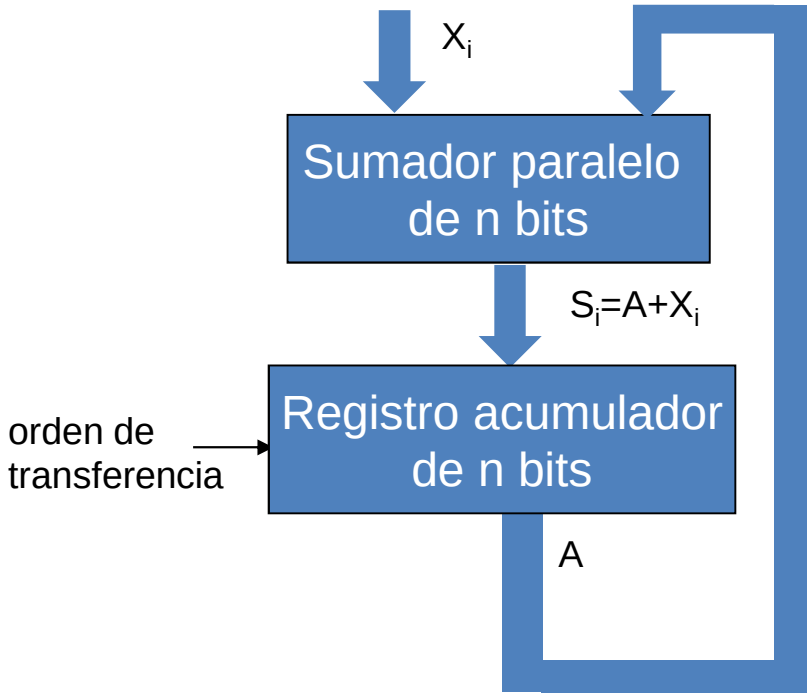
La célula de memoria Δ almacena el acarreo de la suma anterior

Los impulsos de reloj desplazan los registros de A, B y suma, y cargan la memoria:



- **Aplicación de sumadores: el acumulador**

  Se emplea para sumar varios números X_i de n bits cada uno



  El acumulador almacena las sumas parciales

  Este método se emplea en las calculadoras y en los microprocesadores

- **Semirestador básico (en inglés half substractor – HS)**

resta 1 bit - 1 bit

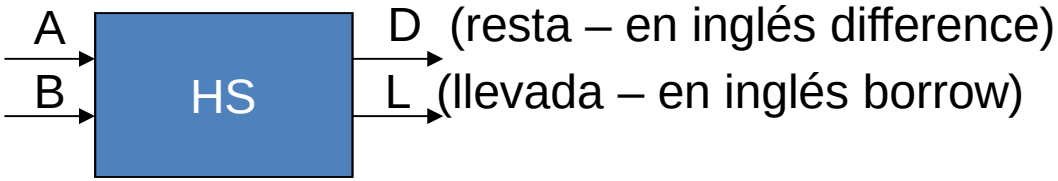


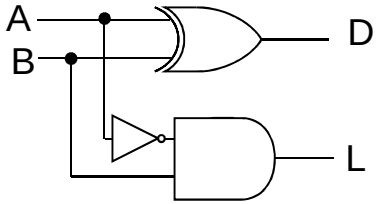
Tabla de verdad

A	B	D	B
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Circuito:

$D = \overline{A}\cdot B + A\cdot\overline{B} = A \oplus B$

$L = \overline{A}\cdot B$



- **Restador completo (en inglés full substractor - FS)**

Suma 1 bit + 1 bit + acarreo (C')

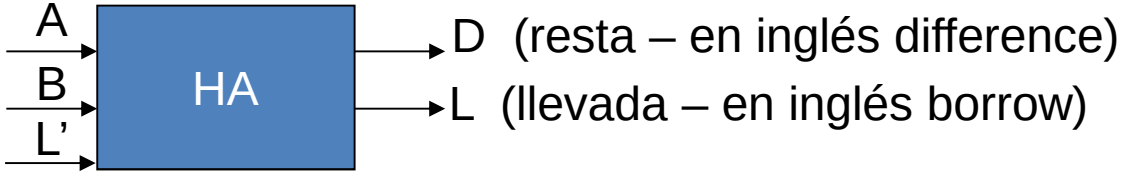


Tabla de verdad

A	B	L'	S	L
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

D

L' \ AB	00	01	11	10
0	0	1	0	1
1	1	0	1	0

L

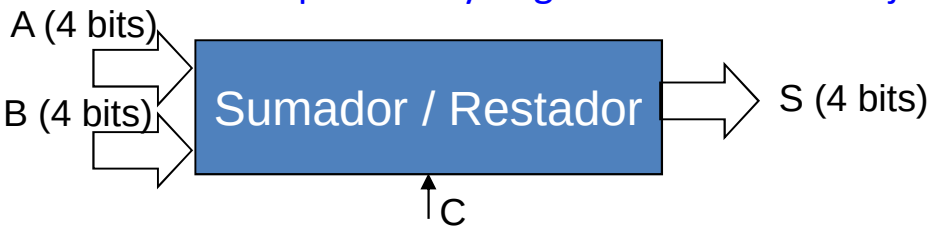
L' \ AB	00	01	11	10
0	0	1	0	0
1	1	1	1	0

$D = \overline{A}\cdot\overline{B}\cdot L' + \overline{A}\cdot B\cdot\overline{L'} + A\cdot\overline{B}\cdot\overline{L'} + A\cdot B\cdot L' = A \oplus B \oplus L'$

$C = \overline{A}\cdot B + \overline{A}\cdot L' + B\cdot L'$

- **Sumador-restador**

Ya se vio al principio de este capítulo que se puede restar haciendo la suma de un número negativo. Para implementar un circuito que realice sumas y restas la mejor opción es por tanto que permita sumar números positivos y negativos. Se hará un ejemplo para números de 4 bits:

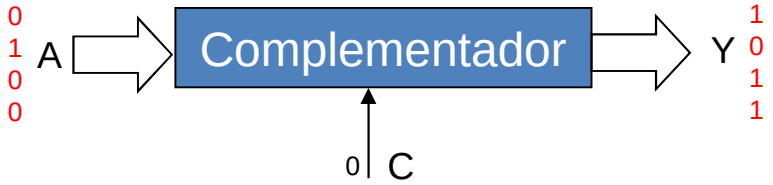


C	S	
0	B-A	Resta
1	B+A	Suma

Suma: B+A ⇨ Sumador

Resta: B-A=B+(-A)= | B+CA1(A) / B+CA2(A) ⇨ Complementador y sumador

Paso 1: Implementar un circuito complementador

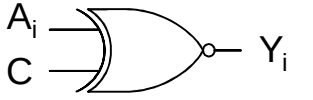


C	Y	
0	$\overline{A}$	Complementa
1	A	No complementa

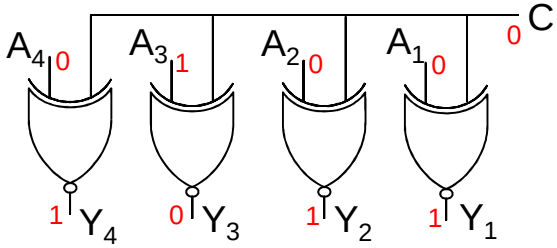
Tabla para complementar uno de los bit de A: A_i

C	A_i	Y_i
0	0	1
0	1	0
1	0	0
1	1	1

$$D = \overline{A_i}\cdot\overline{C}+A_i\cdot C = \overline{A_i \oplus C}$$



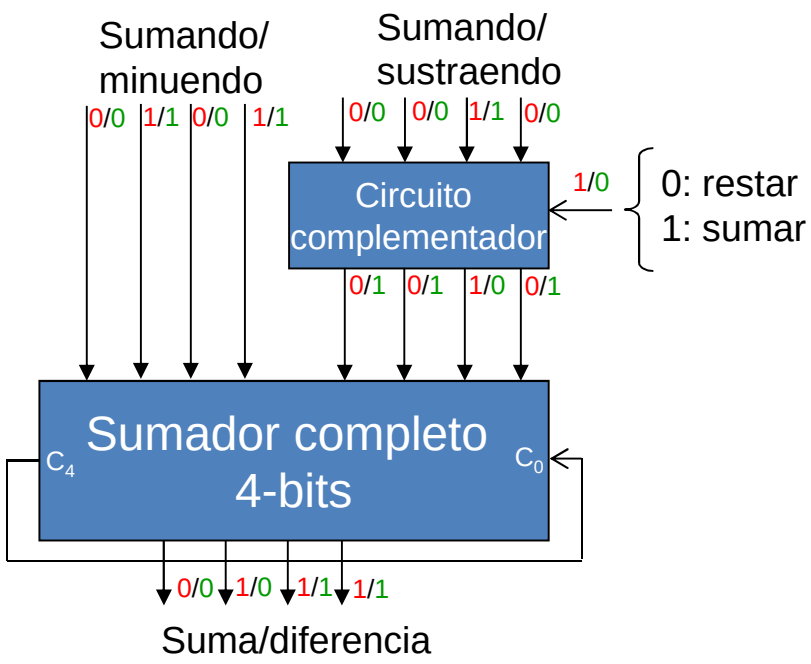
Complementador de 4 bits:



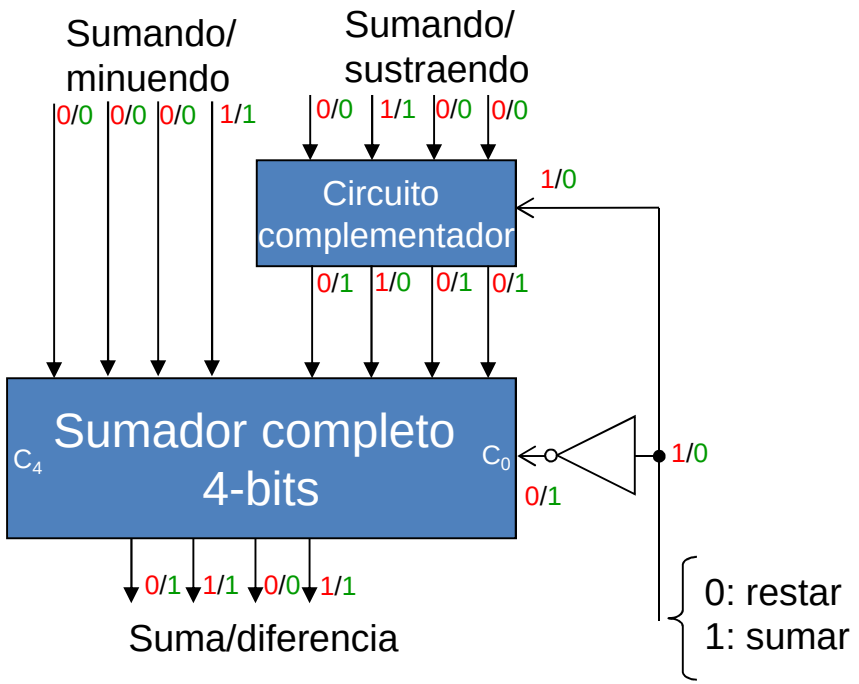
- **Sumador-restador**

Paso 2: Implementación del circuito completo:

a) En Complemento A 1



b) En Complemento A 2



$$\begin{array}{rcl} 5 & & 0\,1\,0\,1 \\ +\ 2 & + & 0\,0\,1\,0 \\ \hline 7 & = & 0\,1\,1\,1 \end{array} \qquad \begin{array}{rcl} 5 & & 0\,1\,0\,1 \\ +\ (-2) & + & 1\,1\,0\,1 \\ \hline 3 & & 1\,0\,0\,1\,0 \end{array}$$

acarreo → +1

$$0\,0\,1\,1 = 3$$

$$\begin{array}{rcl} 1 & & 0\,0\,0\,1 \\ +\ 4 & + & 0\,1\,0\,0 \\ \hline 5 & = & 0\,1\,0\,1 \end{array} \qquad \begin{array}{rcl} 1 & & 0\,0\,0\,1 \\ +\ (-4) & + & 1\,1\,0\,0 \\ \hline (-3) & = & 1\,1\,0\,1 = \text{CA2(3)} \end{array}$$

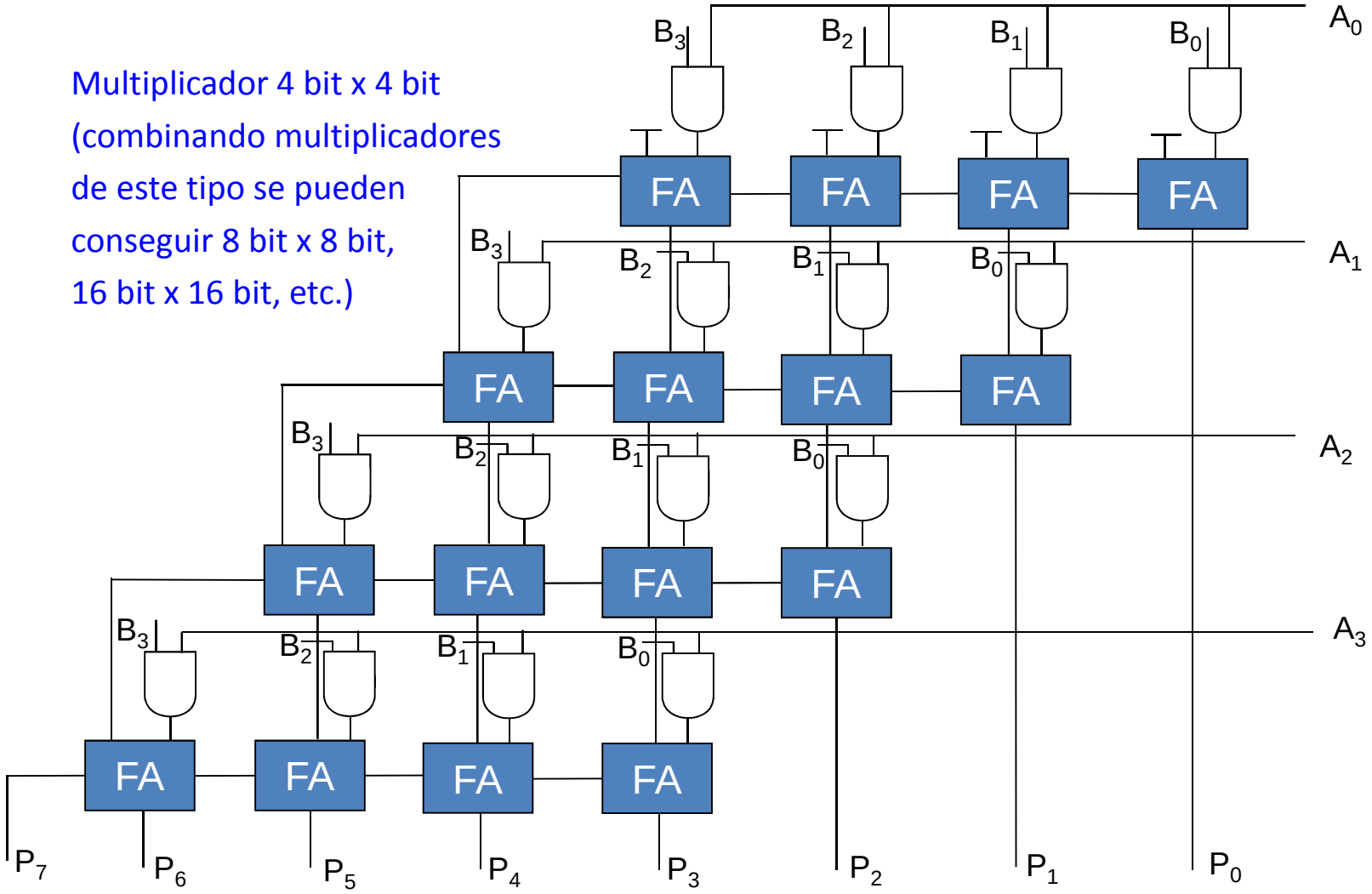
- **Multiplicadores**

Una forma simple de implementarlo es con:

- Puertas AND para los productos bit a bit
- Full Adders para las sumas

Ejemplo comercial de un multiplicador de 4bit x 4bit: http://bit.ly/1kmPVed (Texas 74284)

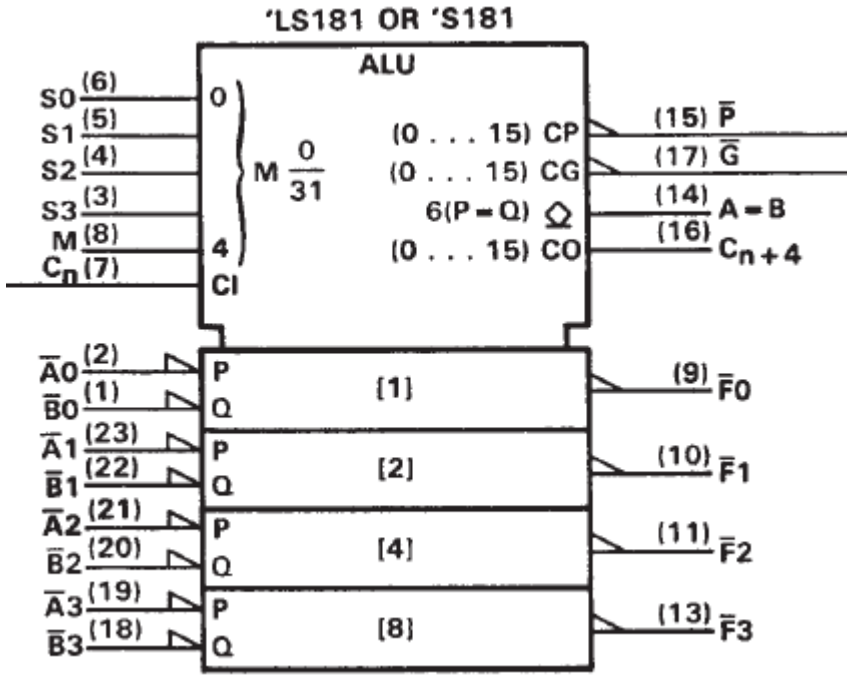
Multiplicador 4 bit x 4 bit (combinando multiplicadores de este tipo se pueden conseguir 8 bit x 8 bit, 16 bit x 16 bit, etc.)



- **Unidad aritmético lógica**

Circuito combinacional que realiza operaciones aritméticas y lógicas (como una pequeña calculadora)

Un circuito clásico es el 74181 (http://bit.ly/1le2vHx)



En él podemos distinguir:

Dos dígitos de entrada

A: $\overline{A}_3\overline{A}_2\overline{A}_1\overline{A}_0$ } 0 es el bit menos significativo y los bits están complementados

B: $\overline{B}_3\overline{B}_2\overline{B}_1\overline{B}_0$ }

El resultado de la operación a realizar:

F: $\overline{F}_3\overline{F}_2\overline{F}_1\overline{F}_0$ } 0 es el bit menos significativo y los bits están complementados

El código de la operación a realizar:

S: $S_3S_2S_1S_0$

El grupo al que pertenece el código de operación:

Se controla mediante M (si vale 1 es operación lógica y si vale 1 es operación aritmética)

Las operaciones aritméticas se subdividen en dos grupos dependiendo del valor del bit Cn (en realidad es un bit que si está a 1 le suma una unidad al resultado de la operación aritmética)

El resto de pines del circuito CP, CG, C0, se usan para conexión con otros circuitos

- **Unidad aritmético lógica**

A continuación está la tabla de operaciones del 74181

SELECTION				ACTIVE-LOW DATA		
				M = H	M = L; ARITHMETIC OPERATIONS	
S3	S2	S1	S0	LOGIC FUNCTIONS	Cn = L (no carry)	Cn = H (with carry)
L	L	L	L	$F = \overline{A}$	F = A MINUS 1	F = A
L	L	L	H	$F = \overline{AB}$	F = AB MINUS 1	F = AB
L	L	H	L	$F = \overline{A} + B$	F = $A\overline{B}$ MINUS 1	F = $A\overline{B}$
L	L	H	H	F = 1	F = MINUS 1 (2's COMP)	F = ZERO
L	H	L	L	$F = \overline{A + B}$	F = A PLUS (A + $\overline{B}$)	F = A PLUS (A + $\overline{B}$) PLUS 1
L	H	L	H	$F = \overline{B}$	F = AB PLUS (A + $\overline{B}$)	F = AB PLUS (A + $\overline{B}$) PLUS 1
L	H	H	L	$F = \overline{A \oplus B}$	F = A MINUS B MINUS 1	F = A MINUS B
L	H	H	H	$F = A + \overline{B}$	F = A + $\overline{B}$	F = (A + $\overline{B}$) PLUS 1
H	L	L	L	$F = \overline{AB}$	F = A PLUS (A + B)	F = A PLUS (A + B) PLUS 1
H	L	L	H	$F = A \oplus B$	F = A PLUS B	F = A PLUS B PLUS 1
H	L	H	L	F = B	F = $A\overline{B}$ PLUS (A + B)	F = $A\overline{B}$ PLUS (A + B) PLUS 1
H	L	H	H	F = A + B	F = (A + B)	F = (A + B) PLUS 1
H	H	L	L	F = 0	F = A PLUS A‡	F = A PLUS A PLUS 1
H	H	L	H	$F = A\overline{B}$	F = AB PLUS A	F = AB PLUS A PLUS 1
H	H	H	L	F = AB	F = $A\overline{B}$ PLUS A	F = $A\overline{B}$ PLUS A PLUS 1
H	H	H	H	F = A	F = A	F = A PLUS 1

Ejemplo: Se introducen el número $\overline{A}_3\overline{A}_2\overline{A}_1\overline{A}_0 = 1000$, que en realidad es el 0111 (7 en decimal) ya que los bits están complementados.

Se inserta el código de operación $S_3S_2S_1S_0 = 0000$ (en la tabla L indica low=0 y H indica high=1). Por tanto estamos en la primera fila

El grupo de operación es M=0 (operación aritmética) y acarreo Cn=L (primera columna)

Por tanto la operación a realizar es F=A minus 1. Como A vale 7, 7-1=6. F=0110, pero al estar sus bits complementados $F= \overline{F}_3\overline{F}_2\overline{F}_1\overline{F}_0=1001$