TEMA 6

ARITMÉTICA BINARIA Y CIRCUITOS ARITMÉTICOS

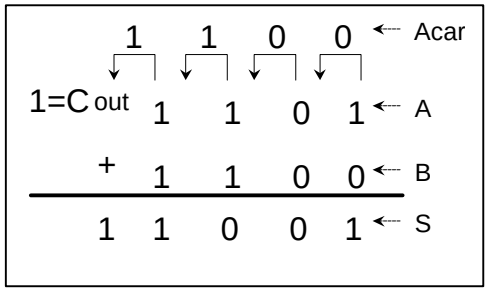
1. ARITMÉTICA BINARIA

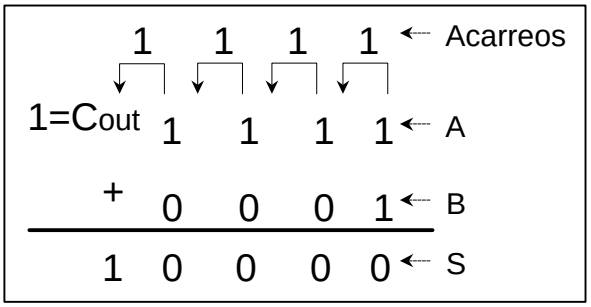
1.1 Aritmética binaria básica

a) Suma binaria

1.Sea C_i el acarreo (carry) generado al sumar los bits A_iB_i (A_i+B_i)
2. Sea i=0 y C_i =0
3. S_i = LSB(A_i + B_i + C_i),
4. C_{i+1} = MSB(A_i + B_i + C_i) y S_i = S_i -2
5. Incrementa i
6. Repite desde 3 mientras que i < n.

Ejemplos:

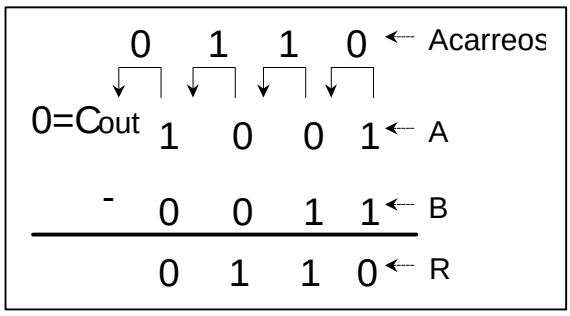


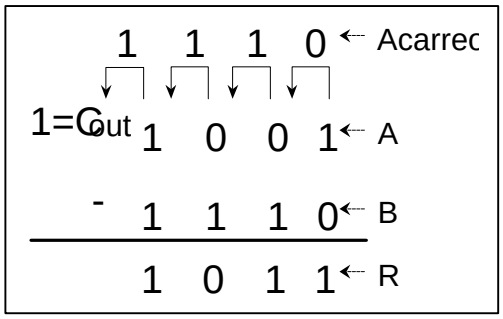


b) Resta binaria

1. Sea C_i el acarreo (borrow) generado al restar los bits A_i B_i
2. Sea i=0 y C_i =0
3. Si A_i >= (B_i + C_i) entonces R_i = A_i - (B_i + C_i) y C_{i+1} =0 en caso contrario R_i = LSB[(B_i + C_i) - A_i] y C_{i+1} =1. La función LSB[] devuelve el bit menos significativo del argumento
4. Incrementa i
5. Repite desde 3 mientras que i < n

Ejemplos:

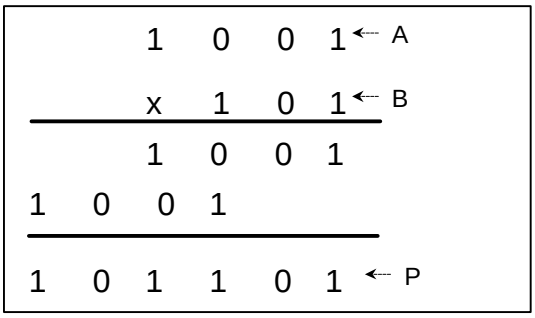


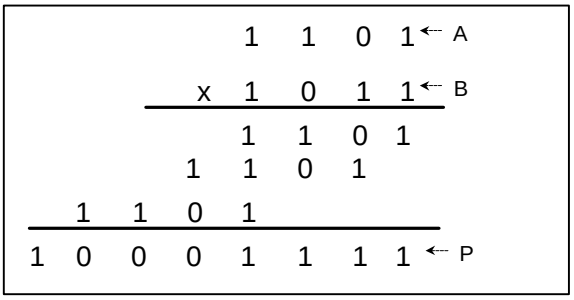


c) Multiplicación binaria

La multiplicación binaria sigue las mismas reglas que la correspondiente a la base diez, salvo que la suma final se realiza en binario.

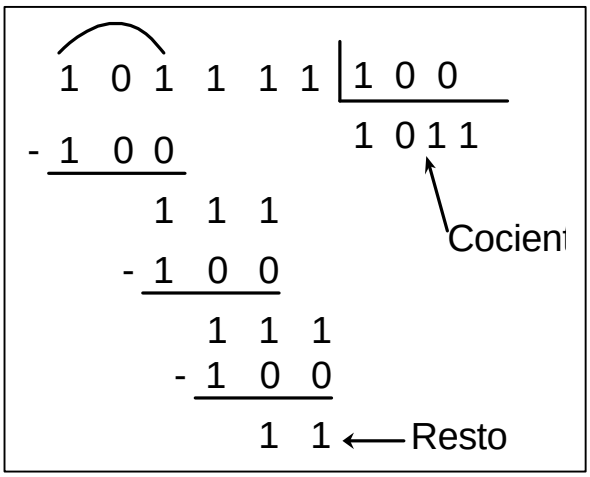
Ejemplos:

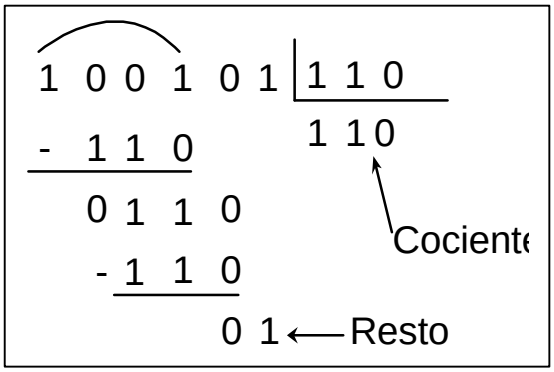




d) División binaria

Ejemplos:



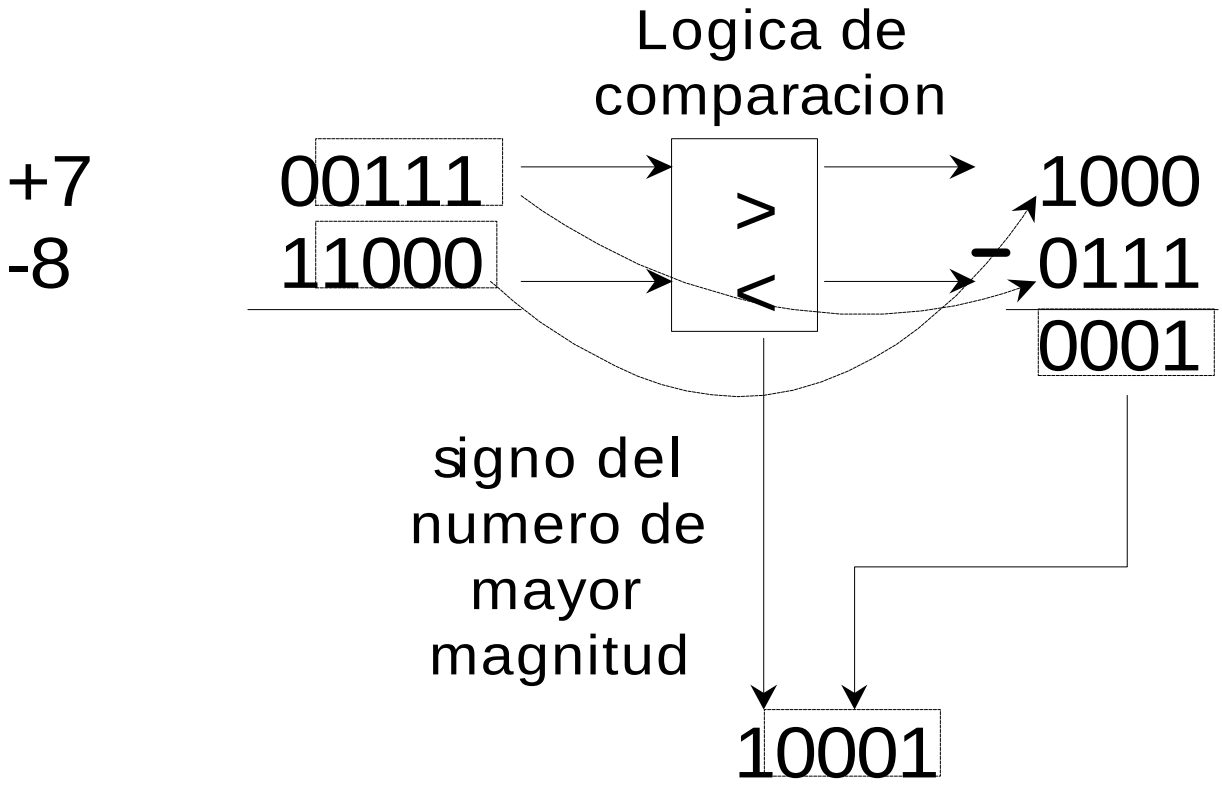


1.2 Aritmética en notación signo-magnitud

a) Si los dos números son del mismo signo, la magnitud del resultado se corresponde con la suma de las magnitudes de los números. Además, el bit de signo del resultado es el mismo que el de cualquiera de los sumandos

b) Si los dos números son de distinto signo, la magnitud del resultado se determina calculando la diferencia entre la magnitud mayor y menor de los dos números. El bit de signo se corresponde con el del número que tenga mayor magnitud.

Ejemplo:



1.3 Aritmética en Complemento a 2

→ La representación de números binarios con signo en notación complemento a 2(o complemento a 1) eliminan la necesidad de utilizar circuitos restadores para la realización de las operaciones aritméticas básicas.

→ En complemento a 2, un número negativo se obtiene aplicando el operador Ca2 al módulo de dicho número (más un cero en la posición más significativa).

$$Ca2(M) = 2^n - M$$

→ La expresión de un número negativo en Ca2, lleva implícito la operación de resta.

En la aritmética en Ca2 deben considerarse algunas situaciones:

I) Se disponen de dos números binarios A y B positivos

La suma binaria de dos números A y B positivos expresados en Ca2 genera el resultado correcto también expresado en Ca2.

Ejemplo:

```
+5   00101
+6   00110
     -----
     01011  = +11
```

Puede darse la situación paradójica en que al sumar dos números positivos, el resultado sea un número negativo (bit más significativo a 1). En este caso se dice que se ha generado un **overflow** (desbordamiento). El desbordamiento se produce cuando la magnitud del resultado no puede ser expresada con el número de bits de los operandos.

Ejemplo:

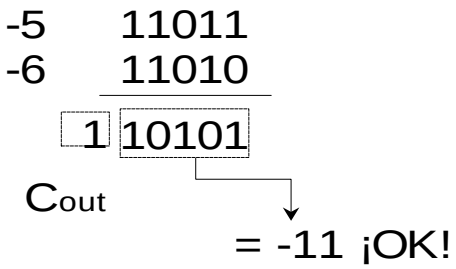
```
+12  01100
+13  01101
     -----
     11001  = ¡-7!
```

Se soluciona de forma sencilla sin más que añadir más bits a la representación de los números.

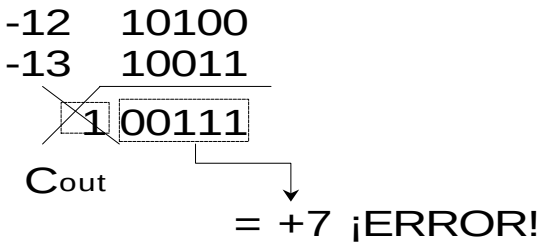
```
+12  001100
+13  001101
     ------
     011001  = +25
```

II) Se disponen de dos números binarios A y B negativos

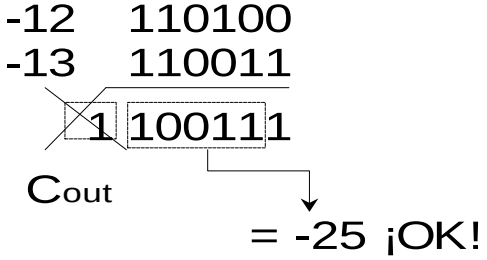
Ejemplo:



Ejemplo:



La solución es idéntica, se añaden más bits.



Si Ay B son números negativos, entonces en Ca2 expresan la cantidad

$$2^n - |A|$$
$$2^n - |B|$$

donde n es el número de bits de A y B y |A|, |B| son las magnitudes de ambos números. La suma de estas cantidades debe generar un resultado en Ca2 igual a

$$2^n - (A+B)$$

pero el resultado real de la suma es

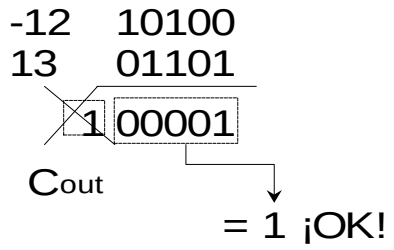
$$2^n - A + 2^n - B = 2^n - (A+B) + 2^n$$

Nos sobra, de la suma real, el término 2^n para obtener el resultado correcto. Obsérvese que si el número A y B tiene n bits, el término 2^n tiene n+1 bits, siendo el bit más significativo un 1, y los restantes n bits 0. La suma de este término, al resultado esperado, $2^n - (A+B)$, genera el acarreo de salida.

III) Se disponen de dos números binarios A y B de distinto signo.

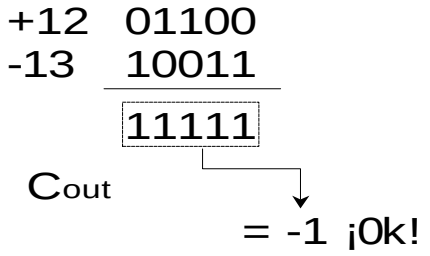
a) magnitud(A) < = magnitud(B) (|A|<=|B|)

Ejemplo:



b) magnitud(A) > magnitud(B) (|A|>|B|)

Ejemplo:



Sí A es un número negativo, entonces en Ca2 expresa la cantidad

$$2^n - |A|$$

Si |A|<=|B|, entonces el resultado debe ser un número positivo que en Ca2 se expresaría como:

$$B-A$$

Pero la suma de las cantidades $2^n - A$ con B genera el resultado

$$2^n + (B-A)$$

por tanto si B>=A, al resultado correcto se le suma el término 2^n, o sea, el acarreo de salida, que debe despreciarse.

En cambio si |A|>|B| el resultado es un número negativo que debe expresarse como:

$$2^n - (A-B)$$

De hecho, esta es la cantidad que se obtiene por la suma de ambos números, pero ahora no se genera acarreo puesto que A-B es una cantidad positiva que se resta a 2^n.

CONCLUSIONES:

La aritmética en Ca2 permite la realización de sumas y restas utilizando, exclusivamente, un sumador binario.

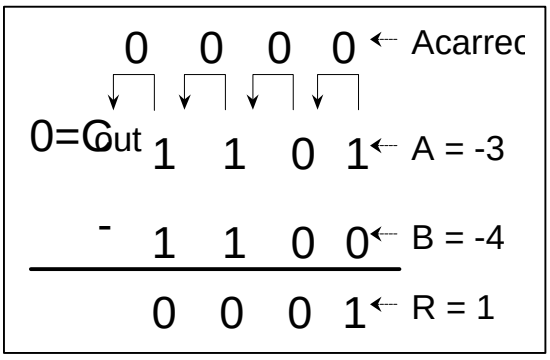
El resultado correcto de la operación aritmética se obtiene despreciando el acarreo de salida si este se genera.

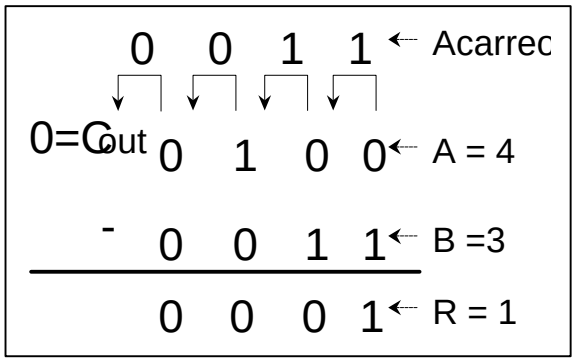
Muchos fabricantes de microprocesadores que usan la notación Ca2 para la representación de números con signo en sus máquinas incluyen circuitos sumadores y restadores en las unidades de cálculo de los mismos, para la realización de las operaciones aritméticas. Se pueden distinguir los siguientes casos:

a) signo (A)=signo(B)

a.1) Si A $\geq$ B, entonces R=A-B es positivo y no se genera acarreo de salida C_{out}=0

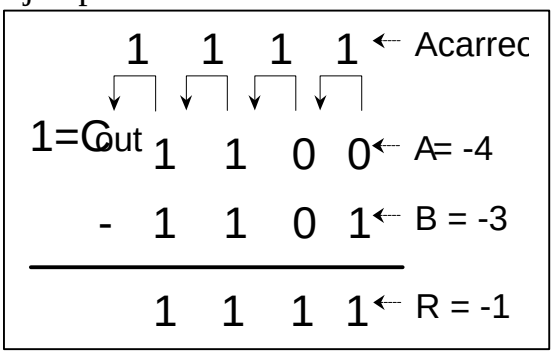
Ejemplos:

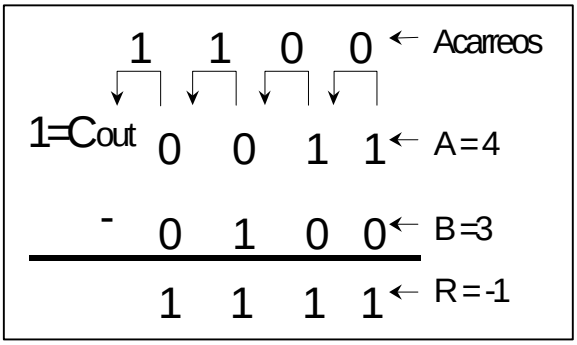




a.2) Si A <B, entonces el resultado de la resta es el complemento a 2 de la diferencia B-A y se genera acarreo de salida, C_{out}=1.

Ejemplos:



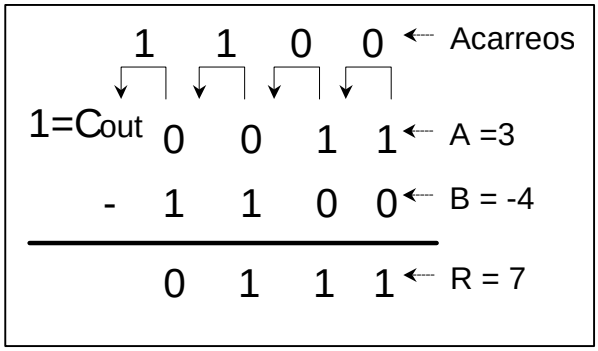


Nótese que al restar dos números con el mismo signo, es imposible que se genere desbordamiento (overflow).

b) A es un número positivo y B es negativo (A>0 y B<0)

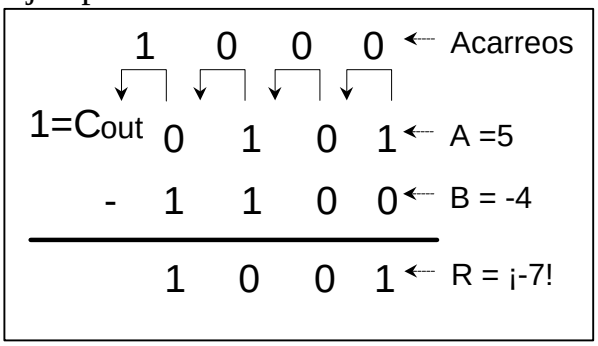
El resultado obtenido debe ser positivo y se genera acarreo (borrow) de salida C_{out}=1.

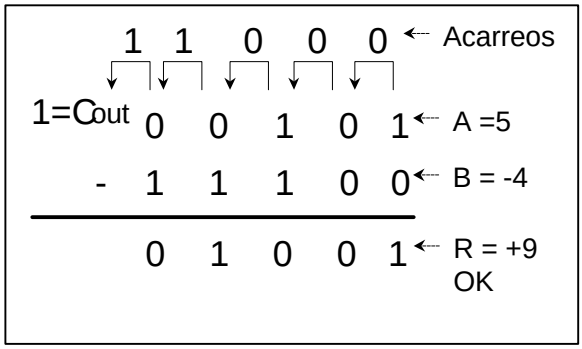
Ejemplo:



La generación de desbordamientos es posible en este caso

Ejemplos:

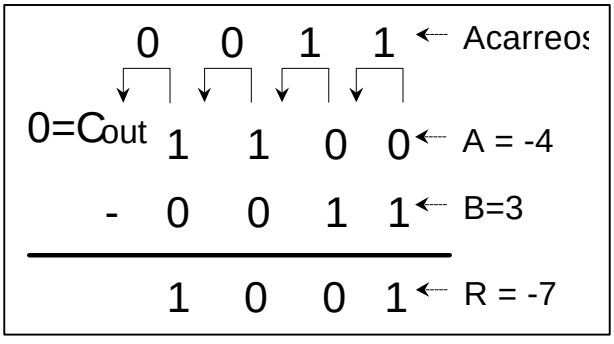




c) A es un número negativo y B es positivo (A<0 y B>0)

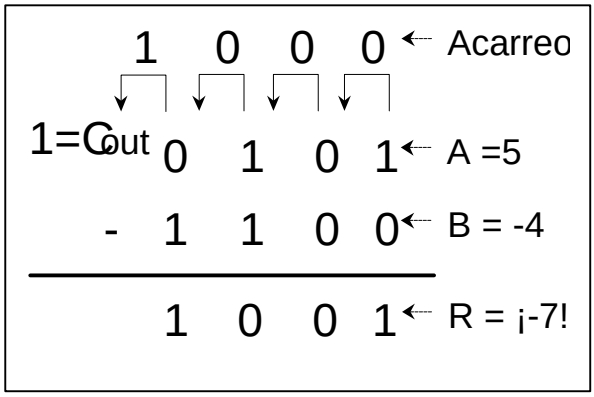
El resultado obtenido debe ser negativo y no se genera acarreo (borrow) final C_{out} =0.

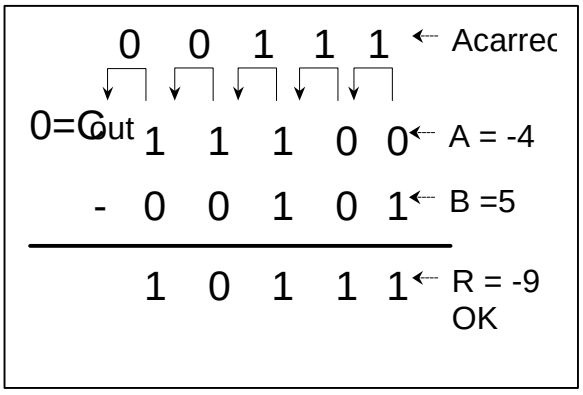
Ejemplo:



También es posible en este caso la aparición de desbordamientos.

Ejemplos:





1.4 Aritmética en Complemento a 1

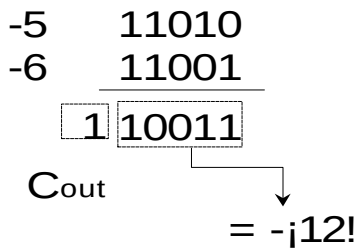
Al igual que sucedió con la aritmética en Ca2, en Ca1 deben considerarse algunas situaciones especiales:

I) Se disponen de dos números A y B positivos

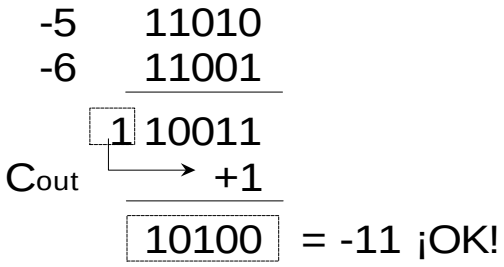
Este caso es idéntico al del Ca2 puesto que los números positivos se expersan igual

II) Se disponen de dos números A y B negativos

Ejemplo:



En Ca1, el acarreo generado se desprecia y además se debe añadir una unidad al resultado obtenido por el sumador para conseguir el valor correcto.



También se puede dar situación de overflow

Si Ay B son números negativos sin parte fraccionaria, entonces en Ca1 expresan la cantidad

$$2^n - |A| - 1$$
$$2^n - |B| - 1$$

donde n es el número de bits de A y B y |A|, |B| son las magnitudes de ambos números.

El resultado correcto, expresado en Ca1, de la suma de A y B debe ser

$$2^n - (A+B) - 1$$

Pero la suma de estas cantidades genera el siguiente resultado

$$2^n - A - 1 + 2^n - B - 1 = 2^n - (A+B) - 1 + 2^n - 1$$

Nos sobra, de la suma real, el término 2^n, y además el resultado correcto aparece con una unidad menos.

2. Se disponen de dos números binarios A y B de distinto signo.

a) magnitud(A) < = magnitud(B) (|A|<=|B|)

Ejemplo:

```
-12   10011
+13   01101
     ------
    1 00001
Cout      +1
     ------
      00001   = +1 ¡OK!
```

b) magnitud(A) > magnitud(B) (|A|>|B|)

Ejemplo:

```
+12 01100
-13  10010
      11110 = -1
```

Si A es un número negativo, entonces en Ca1 expresa la cantidad

$$2^n - |A| - 1$$

Si |A|<=|B|, entonces el resultado debe ser un número positivo que en Ca1 se expresaría como:

$$(B-A)$$

Pero la suma de las cantidades $2^n - A - 1$ con B genera el resultado

$$2^n + (B-A) - 1$$

por tanto si B>=A, al resultado correcto se le suma el término 2^n, o sea, el acarreo de salida, que debe despreciarse y se le resta una unidad.

En cambio si |A|>|B| el resultado es un número negativo que debe expresarse como:

$$2^n - (A-B) - 1$$

De hecho, esta es la cantidad que se obtiene por la suma de ambos números, pero ahora no se genera acarreo puesto que A-B es una cantidad positiva que se resta a $2^n - 1$.

CONCLUSIONES:

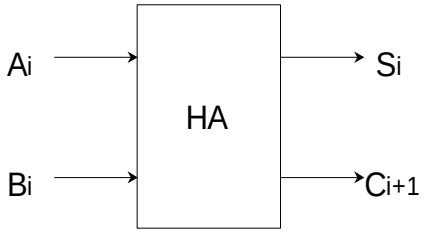
La aritmética en Ca1 permite la realización de sumas y restas utilizando, exclusivamente, un sumador binario.

El resultado correcto de la operación aritmética se obtiene despreciando el acarreo de salida que se utiliza para sumar una unidad al resultado.

2.**CIRCUITOS ARITMÉTICOS BÁSICOS**

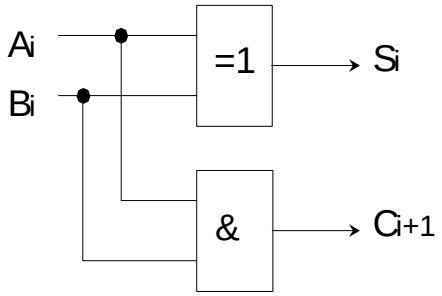
2.1 Circuitos semisumador y sumador completo

Un semisumador o sumador medio (HA), es un circuito combinacional con dos entradas A_i y B_i y dos salidas S_i y C_{i+1}. La salida S_i representa el resultado de la suma aritmética de las entradas A_i y B_i y la salida C_{i+1}, el acarreo.

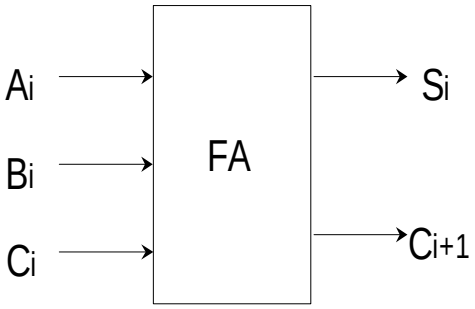


Entradas	Salidas
A_i B_i	C_{i+1} S_i
0 0	0 0
0 1	0 1
1 0	0 1
1 1	1 0

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



El sumador completo o sumador total es un circuito combinacional con tres entradas, A_i B_i C_i y dos salidas S_i y C_{i+1}. La salida S_i representa la suma binaria de las tres entradas y C_{i+1} el accarreo.



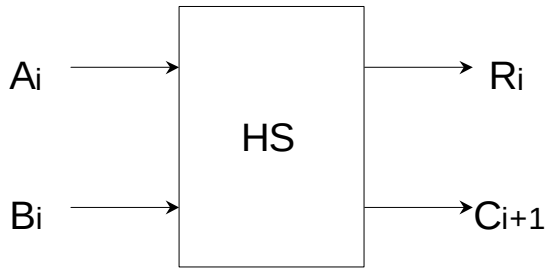
Entradas	Salidas
A_i B_i C_i	C_{i+1} S_i
0 0 0	0 0
0 0 1	0 1
0 1 0	0 1
0 1 1	1 0
1 0 0	0 1
1 0 1	1 0
1 1 0	1 0
1 1 1	1 1

$S_i = A_i \oplus B_i \oplus C_i$

$$C_{i+1} = A_i B_i + A_i C_i + B_i C_i$$

2.2 Circuitos semirestador y restador completo

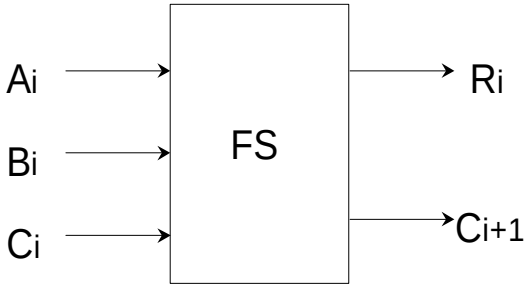
Un semirestador (HS) o restador medio es un circuito combinacional con dos entradas, A_i y B_i y dos salidas R_i y C_{i+1}. La salida R_i representa el resultado de la resta aritmética de las entradas A_i y B_i ($A_i - B_i$)y la salida C_{i+1}, el arrastre de la resta ("me llevo uno").



Entradas	Salidas
A_i B_i	C_{i+1} R_i
0 0	0 0
0 1	1 1
1 0	0 1
1 1	0 0

$$R_i = A_i \oplus B_i$$
$$C_{i+1} = A_i' \, B_i$$

Un restador completo (FS) es un circuito combinacional con tres entradas A_i, B_i y C_i y dos salidas R_i y C_{i+1}. La salida R_i representa el resultado de la resta aritmética de las entradas A_i, B_i y C_i ($A_i - B_i - C_i$) y la salida C_{i+1}, el arrastre.



Entradas	Salidas
A_i B_i C_i	C_{i+1} R_i
0 0 0	0 0
0 0 1	1 1
0 1 0	1 1
0 1 1	1 0
1 0 0	0 1
1 0 1	0 0
1 1 0	0 0
1 1 1	1 1

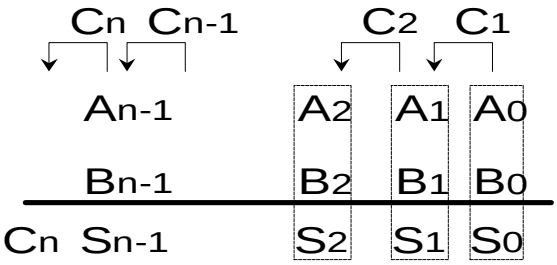
$$R_i = A_i \oplus B_i \oplus C_i$$

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

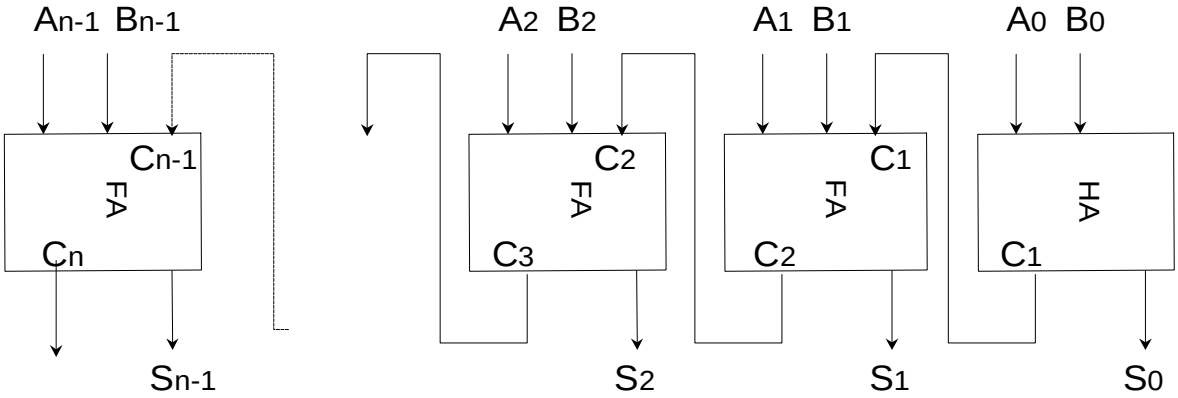
2.3 Sumadores y restadores de n bits

Los sumadores y restadores anteriores pueden combinarse para formar circuitos sumadores y restadores de números de n bits.

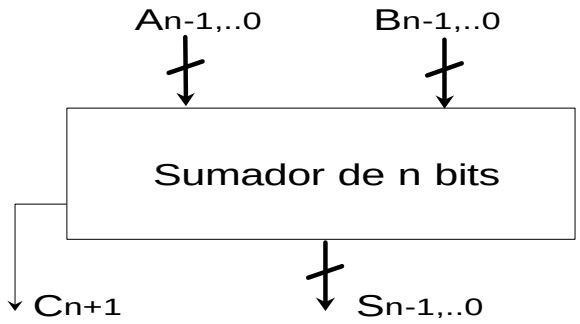
La suma de dos números binarios de n bits A ($A_{n-1...}A_0$) y B ($B_{n-1...}B_0$) genera un resultado, también de n bits, S ($S_{n-1...}S_0$) y un acarreo C_n.



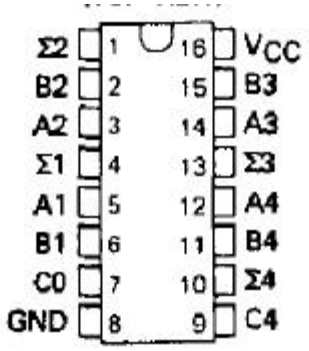
Para la etapa i, se suman los bits A_i B_i C_i para dar el resultado S_i y se genera el acarreo para la siguiente etapa (C_{i+1}).

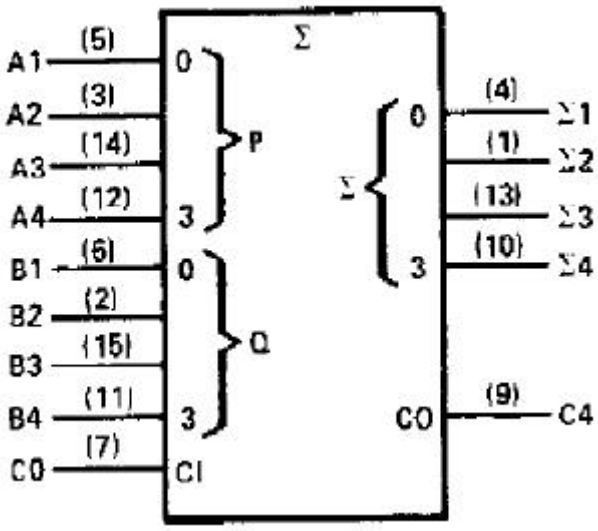


Los sumadores de n bits pueden representarse como bloques funcionales que disponen de 2*n entradas (números A y B), generan n salidas (resultado S) y un acarreo, C_n.

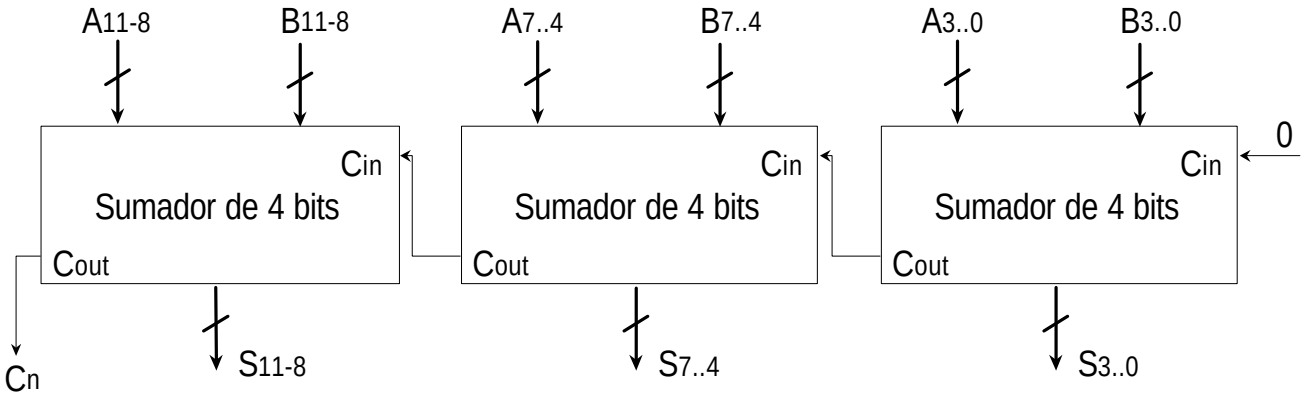


En la siguiente figura se representa un circuito aritmético de 4 bits y su bloque funcional. Se aprecia que además del acarreo de salida CO(C4) existe un acarreo de entrada CI(C0).

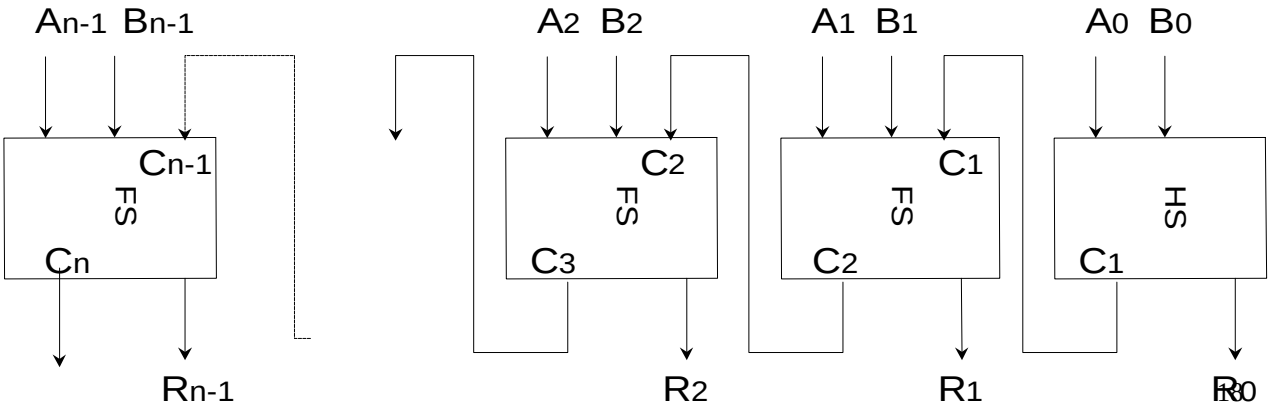




En la siguiente figura se representa una asociación de sumadores de cuatro bits para sumar dos números de 12 bits.



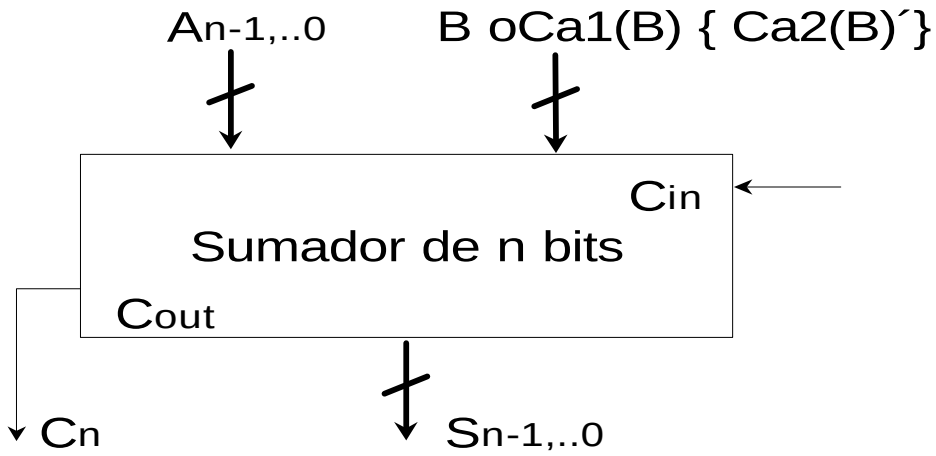
Un **restador** de dos números de n bits, puede construirse como muestra la siguiente figura.



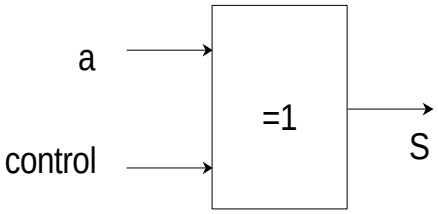
2.4 Circuito sumador-restador

La operación aritmética de la resta puede implementarse con una suma si los números se introducen en complemento a 1 o en complemento a 2.

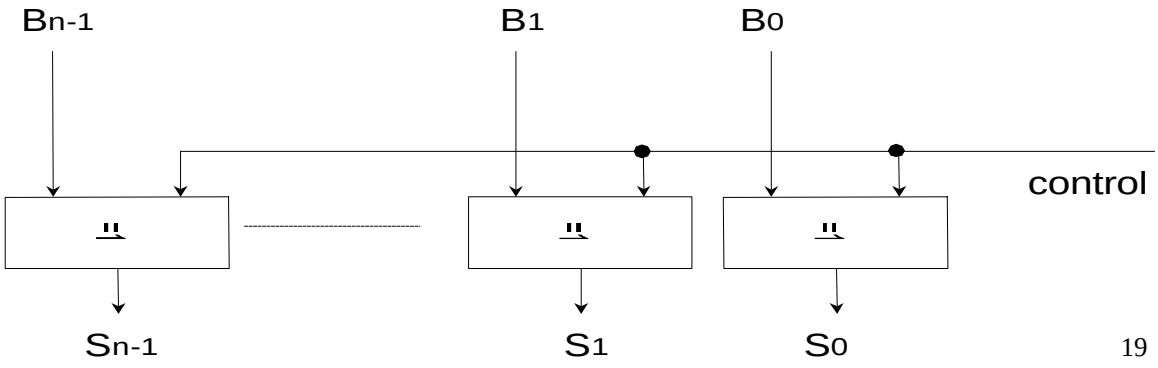
En este apartado, se diseñará un circuito sumador/restador, utilizando las propiedades de las notaciones Ca1 y Ca2. Este circuito debe disponer de una señal de control RESTAR/#SUMAR.



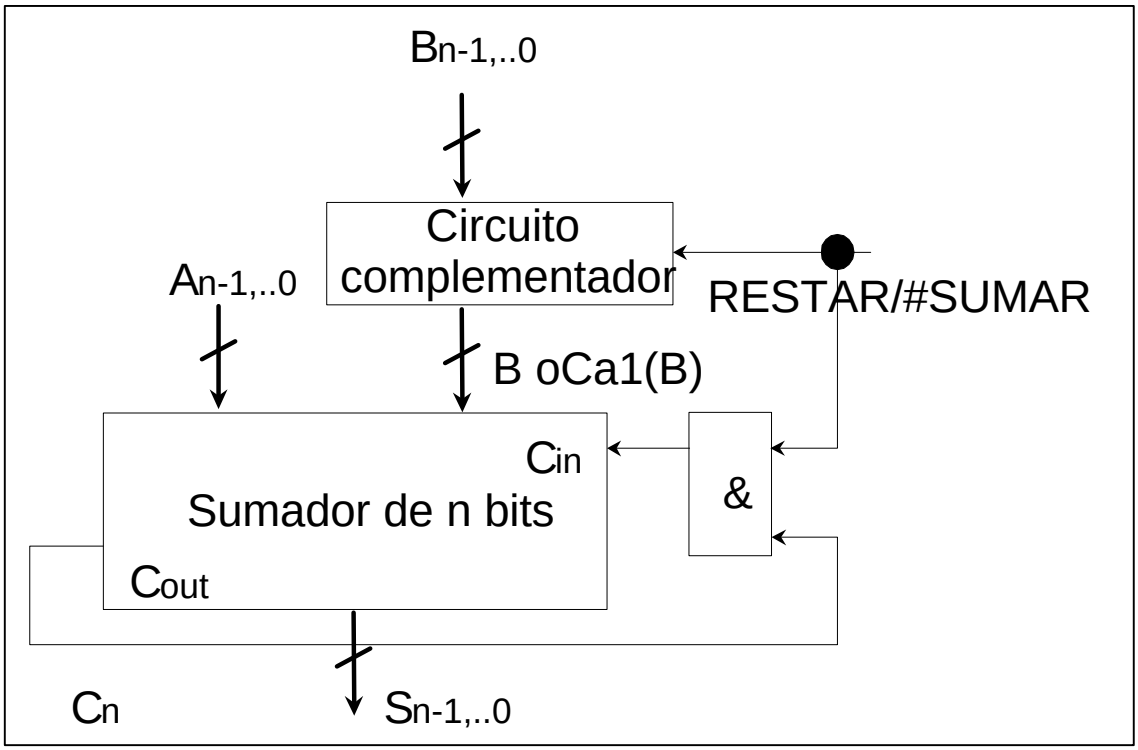
En la siguiente figura se ha representado una puerta exor de dos entradas denominadas, “a” y “control”; y una salida S.



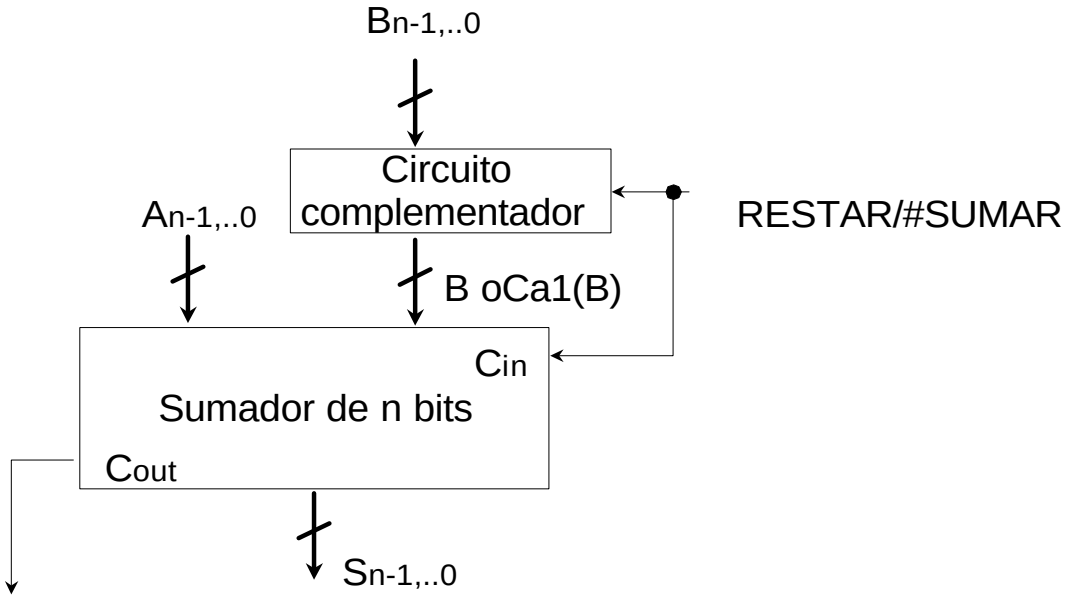
El circuito complementador



La aritmética en Ca1 exige que al resultado de la suma o resta, se debe añadir una unidad, en caso de generarse acarreo, para obtener el resultado correcto. El circuito sumador/restador en complemento a 1 se muestra en la siguiente figura.



Circuito sumador/restador en Ca2



2.5 Sumador BCD: aritmética decimal

Algunos aspectos deben ser considerados en la aritmética BCD. En la suma de dos digitos BCD con sumadores bianrios de 4 bits, pueden darse los siguientes casos:

a) El resultado es un número BCD y no existe arrastre:

```
  0 1 0 1    = 5
+ 0 1 0 0    = 4
  1 0 0 1    = 9
```

b)El resultado no es un número BCD

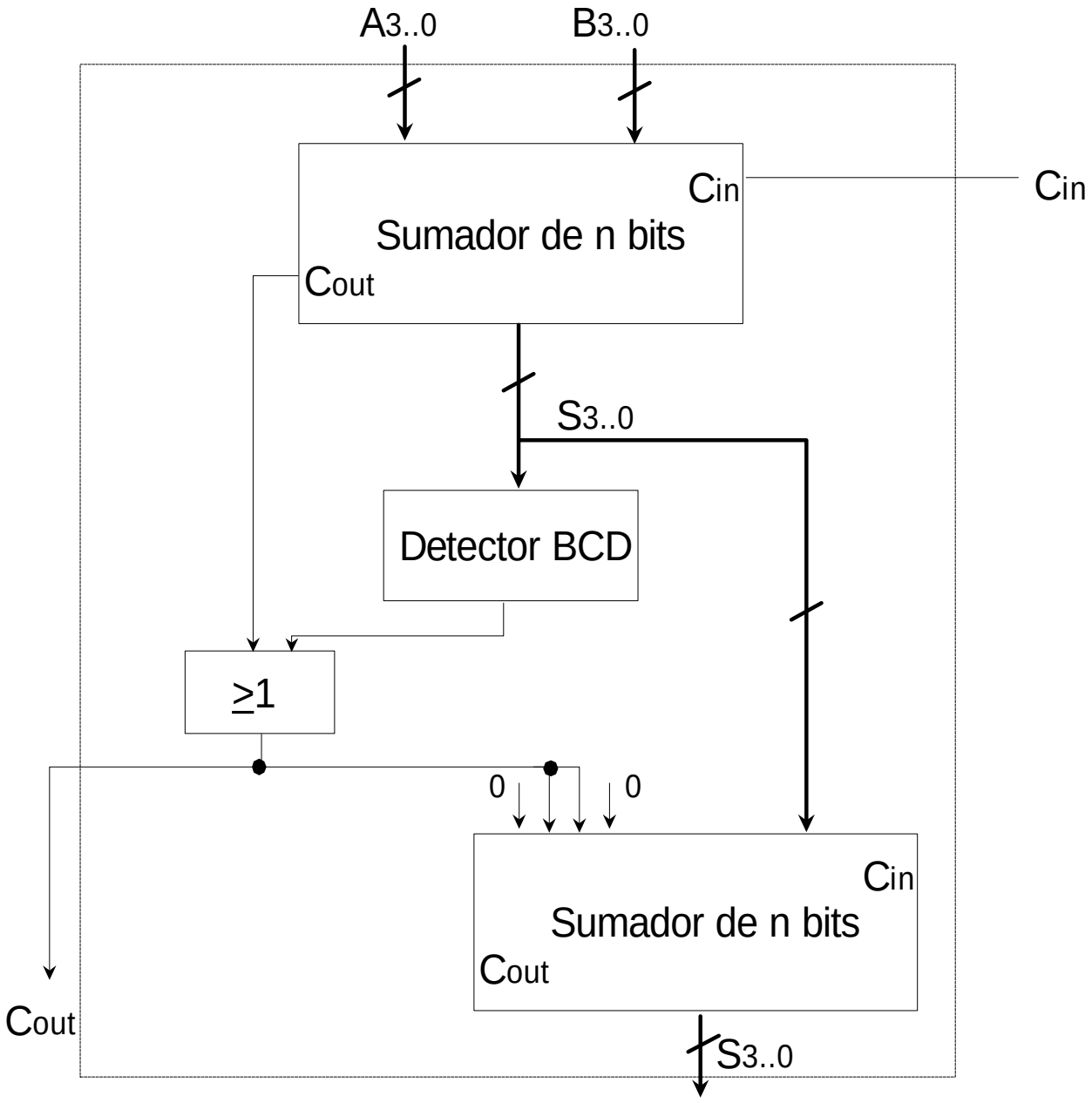
```
  1 0 0 1    = 9
+ 0 1 1 0    = 6
  1 1 1 1    = no BCD
```

c)El resultado es un número BCD y se genera arrastre

```
  1 0 0 1    = 9
+ 1 0 0 0    = 8
1 0 0 0 1  = 11 en BCD
```

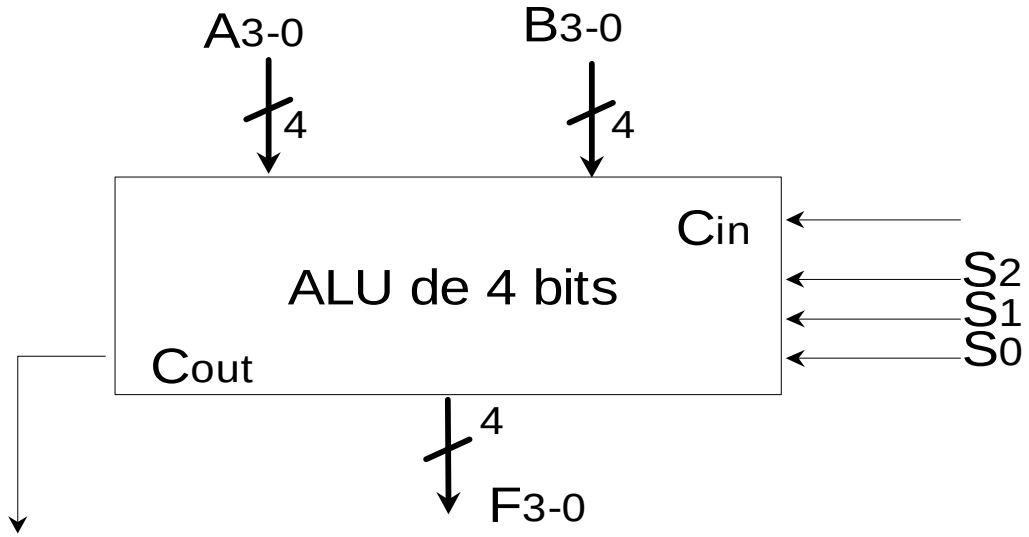
Los casos b) y c) pueden resolverse si se le suma la maginitud 0110 (6) al resultado obtenido por el sumador binario.

En la siguiente figura se representa el sumador BCD



3.- UNIDAD ARITMÉTICO LÓGICA (ALU)

Una ALU de n bits es un dispositivo combinacional que acepta dos palabras de entrada A y B, de n bits cada una y genera un resultado de n bits (además de cierta información como acarreo, overflow, etc.) procedente de la realización de alguna operación aritmética o lógica identificada por unas señales se selección.

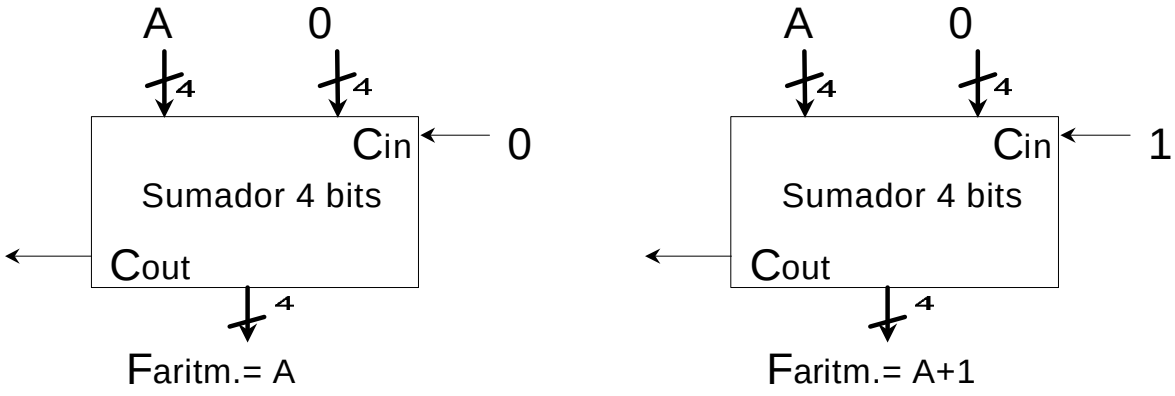


El diseño de una ALU se realiza en tres etapas: diseño del circuito aritmético, diseño del circuito lógico y unión de las partes anteriores.

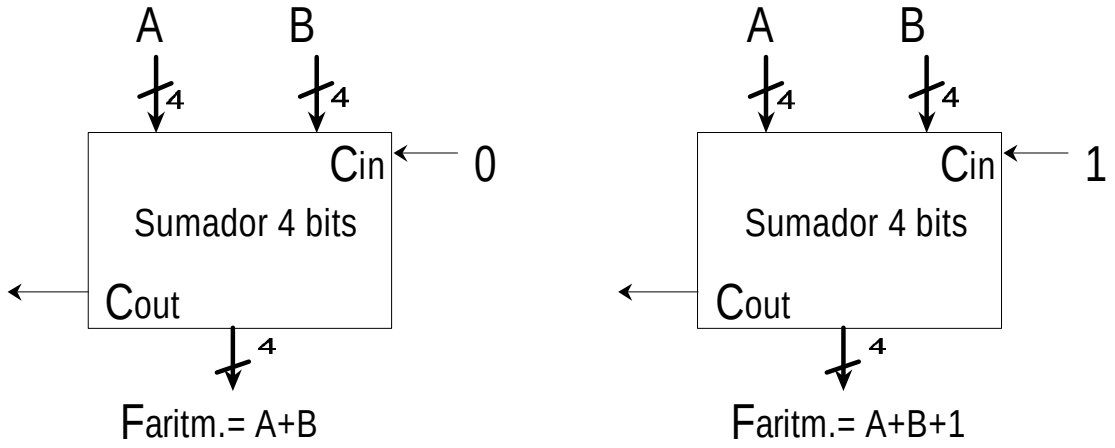
3.1 Diseño del circuito aritmético

El componente básico del circuito aritmético de una ALU de n bits es un sumador paralelo de n bits. Si se controlan las entradas de dicho sumador, su salida puede generar distintas funciones aritméticas:

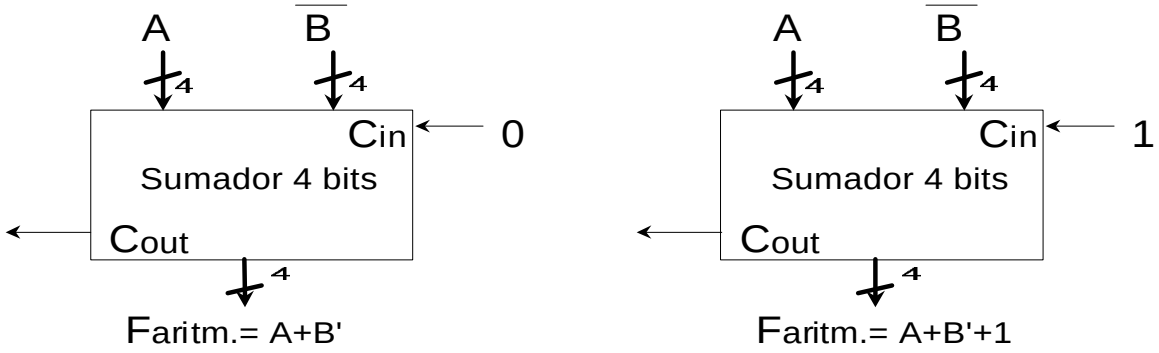
a) Si las entradas del sumador son A y (Todo 0's) (En este caso, tampoco se utiliza el número B de entrada de la ALU), se pueden obtener funciones de transferencia (F=A si C_{in} =0) o de incremento (F=A+1 si C_{in} =1).



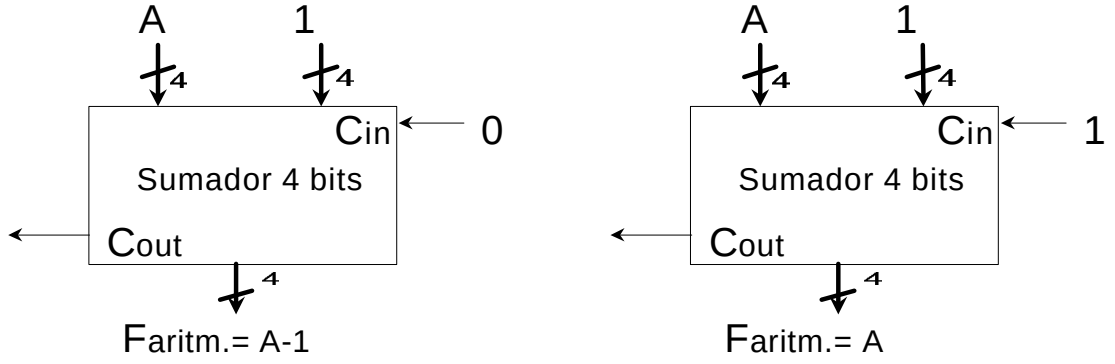
b) Si las entradas del sumador son A y B (los números de entrada de la ALU), se pueden obtener funciones de suma F= A+B (si C_{in} =0) o F=A+B+1 (si C_{in} =1).



c) Si las entradas del sumador son A y B' (se complementan los bits del número B de entrada de la ALU), se pueden obtener operaciones de resta en Ca1 – F=A+B' si C_{in} =0- o de resta en Ca2 – F= A+b'+1 si C_{in} =1.



d) Si las entradas del sumador son A y (Todo 1's) (En este caso no se utiliza el número B de entrada de la ALU), se pueden obtener funciones de transferencia (F=A si C_{in} =1) o de decremento (F=A-1 si C_{in} =0).



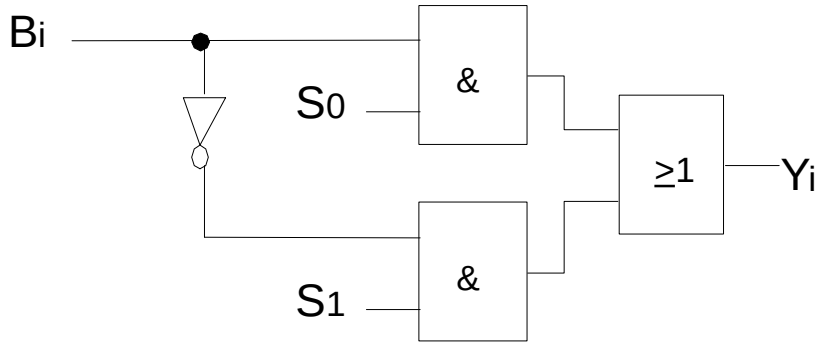
Se debe diseñar un circuito combinacional que, en función de S_1 y S_0, permita seleccionar la entrada B del sumador completo y, por tanto, la operación aritmética.

Se considera que el sumador completo dispone de dos entradas (X e Y). La entrada X se conecta al número A, o lo que es lo mismo, $X_i = A_i$, mientras que la entrada Y se conecta al circuito cuyas salidas, para cada bit, se representa en la siguiente tabla de verdad

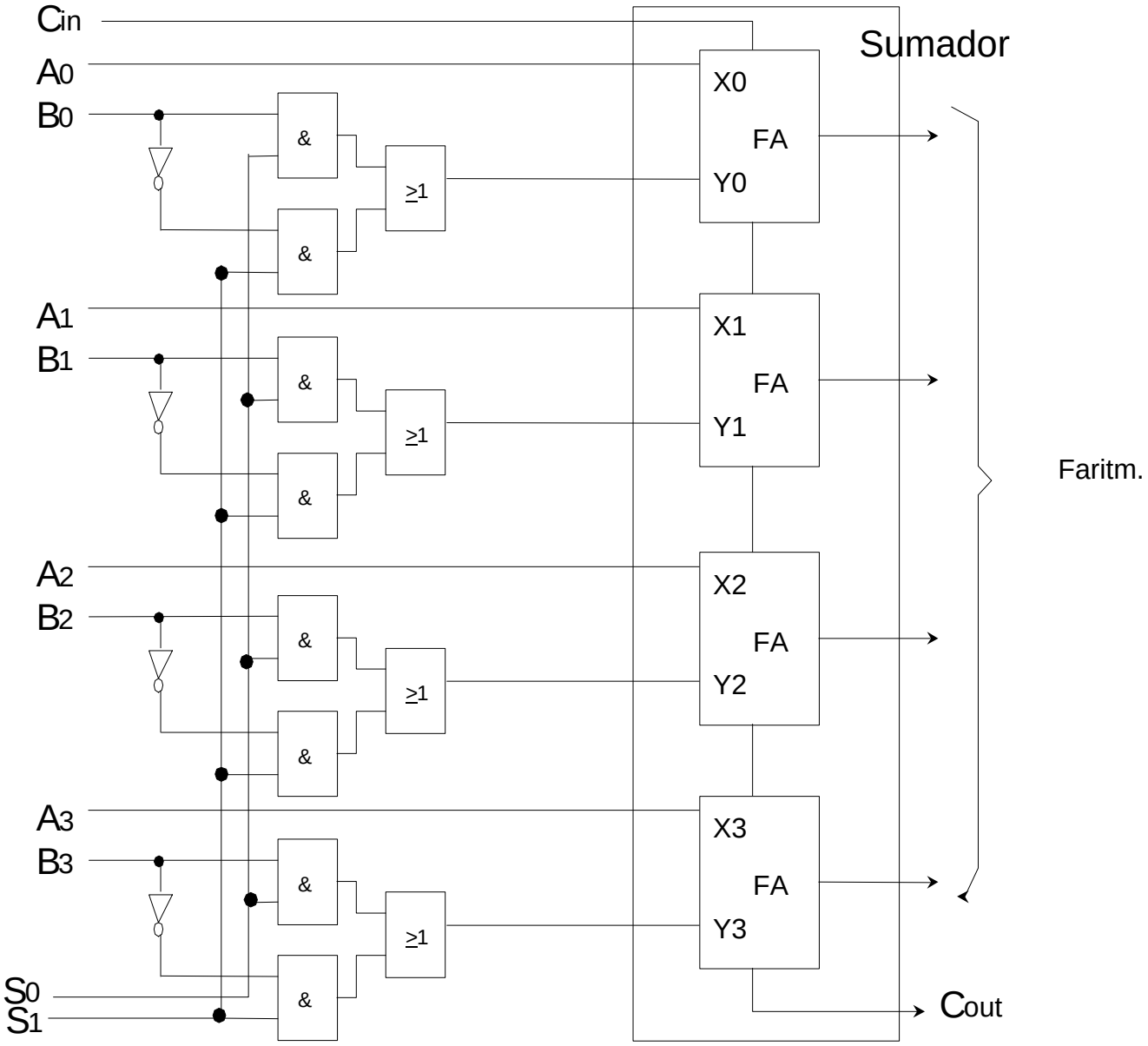
S_1 S_0	Y_i
0 0	0
0 1	B_i
1 0	B'_i
1 1	1

Por tanto, la salida será

$$Y_i = B_i S_1' S_0 + B_i' S_1 S_0' + S_1 S_0 = B_i S_0 + B_i' S_1$$



El circuito aritmético es, por tanto, el siguiente:



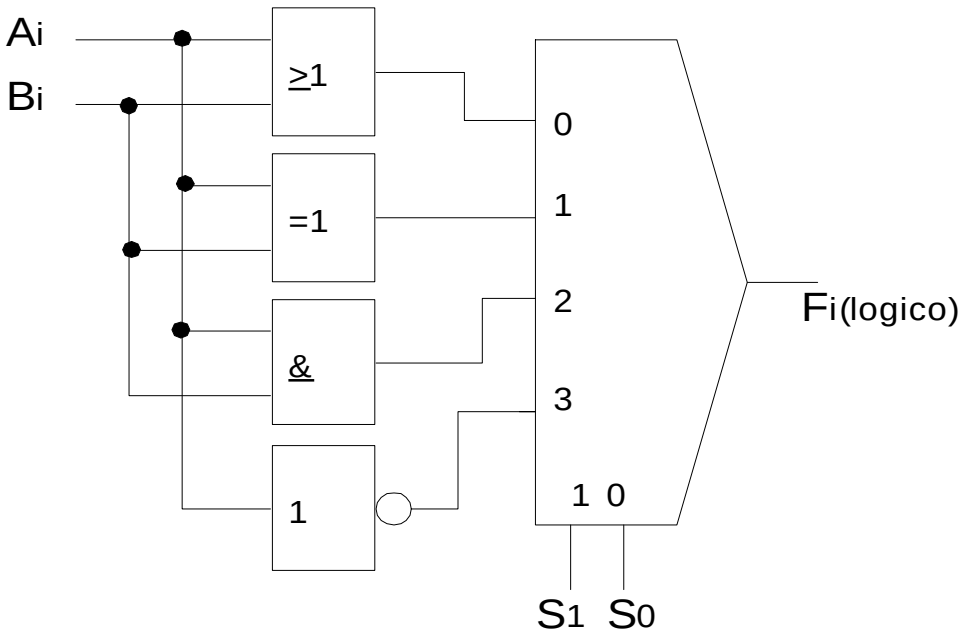
El acarreo de salida nos puede dar una información muy importante:

S_1	S_0	C_{in}		Operación	C_{out} =1 si	Comentario
0	0	0	F=A	Transferir A	--------	C_{out} =0 siempre
0	0	1	F=A+1	Incrementar A	$A=2^n-1$	Si C_{out} =1 , F=0
0	1	0	F=A+B	Sumar A+B	$A+B\geq 2^n$	Overflow si C_{out} =1
0	1	1	F=A+B+1	Incrementar A+B	$A+B\geq 2^n-1$	Overflow si C_{out} =1
1	0	0	F=A+B'	Restar A-B en Ca1	A>B	Si C_{out} =0 → A≤B y F=Ca1(B-A)
1	0	1	F=A+B'+1	Restar A-B en Ca2	A≥B	Si C_{out} =0 → A<B y F=Ca1(B-A)
1	1	0	F=A-1	Decrementar A	A≠0	Si C_{out} =0 → A=0
1	1	1	F=A	Transferir A	-------	C_{out} =1 siempre

3.2 Diseño del circuito lógico

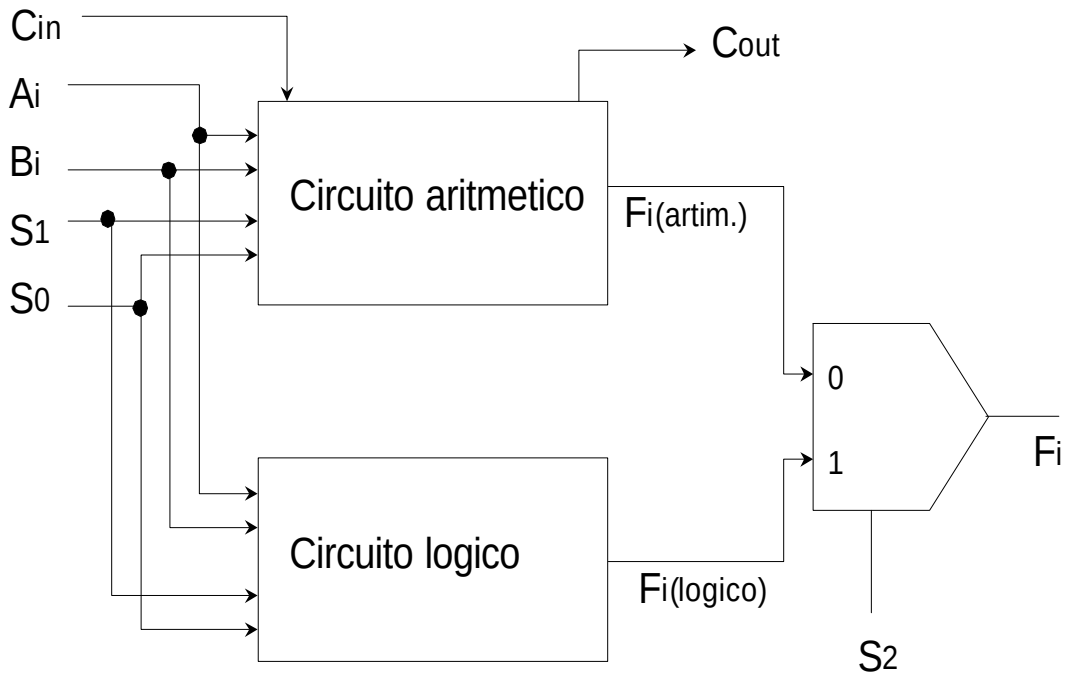
Supongamos que las operaciones lógicas a realizar son: A AND B, A OR B,A EXOR B y NOT A, donde A y B son las palabras de entrada en la ALU. (Se entiende que A AND B – o cualquier otra operación lógica- genera un resultado F, donde cada bit, $F_i = A_i$ AND Bi)

El circuito que permite calcular la operación lógica aparece en la siguiente figura.



3.3 Unión de la secciónа aritmética y lógica

Para formar la ALU se han de unir las dos secciones que la forman. Las variables S_1 S_0 serán comunes para las dos secciones siempre que S_2 distinga entre una etapa y otra. La siguiente figura representa una posible solución para la etapa i de una ALU.



Las siguientes figuras muestran una ALU comercial