Tema 5: Circuitos Aritméticos

Escuela Técnica Superior de Ingeniería

Departamento de
Ingeniería Electrónica, Sistemas Informáticos y Automática

Operaciones aritméticas

En un computador se efectúan múltiples operaciones aritméticas. Veremos en este tema algunas de ellas, tales como la suma y la resta, así como los dispositivos que las realizan. Las unidades lógicas y aritméticas (ALU), como parte integrante de cualquier procesador, también se estudian aquí.

La descripción de operaciones aritméticas en VHDL también se presenta en este tema.

Suma binaria (I)

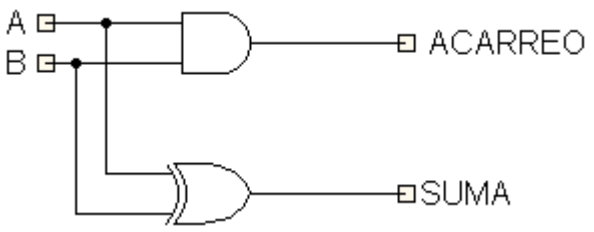
La suma binaria es muy parecida a la suma decimal que conocemos, salvo que opera sólo con dos dígitos, el 0 y el 1. La suma de dos números de n bits puede dar un resultado de n+1 bits, debido al acarreo que puede producirse en el bit de mayor peso (MSB).

$$\begin{array}{r} 1 \\ +2 \\ \hline 3 \end{array} \qquad \begin{array}{r} 01 \\ +10 \\ \hline 11 \end{array} \qquad\qquad \begin{array}{r} 2 \\ +2 \\ \hline 4 \end{array} \qquad \begin{array}{r} 10 \\ +10 \\ \hline 100 \end{array}$$

Como vemos la suma de números binarios de dos bits nos puede dar un número binario de tres bits. Abajo se muestra la tabla de la suma de dos bits.

A	B	ACARREO	SUMA
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Un dispositivo que realice esta operación se denomina **medio sumador o semisumador**:



Suma binaria (II)

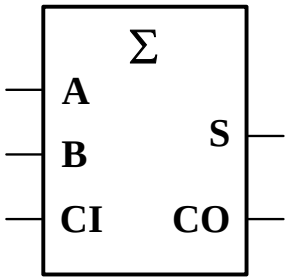
Ejemplo de suma binaria con más bits:

ACARREO	**1 1 1 1**	
SUMANDO	**1 0 1 1 1**	**→ 23**
SUMANDO	**+1 1 0 1 1**	**→ + 27**
	1 1 0 0 1 0	**→ 50**

Sumador completo

El semisumador no puede recibir el acarreo de una etapa anterior, en el caso de que queramos realizar una suma de más bits. Para ello, es necesario emplear el sumador completo, cuya tabla de funcionamiento, diagrama lógico y símbolo se muestran a continuación. Este sumador completo constituye el bloque básico para sumar palabras de mayor número de bits.

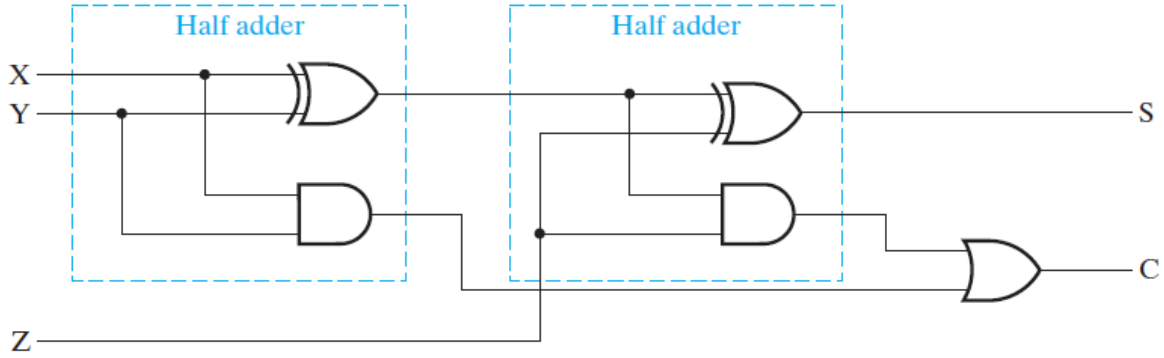
Y X	Cin (Z)	Cout (C)	A+B (S)
0 0	0	0	0
0 0	1	0	1
0 1	0	0	1
0 1	1	1	0
1 0	0	0	1
1 0	1	1	0
1 1	0	1	0
1 1	1	1	1



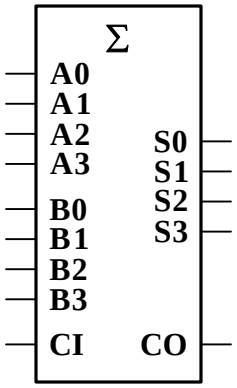
Suma binaria (III)

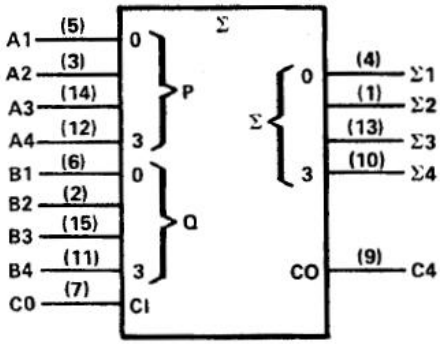
Sumador completo (II)

La siguiente figura muestra la implementación de un sumador completo de un bit. Como se ve, está basado en dos semisumadores.



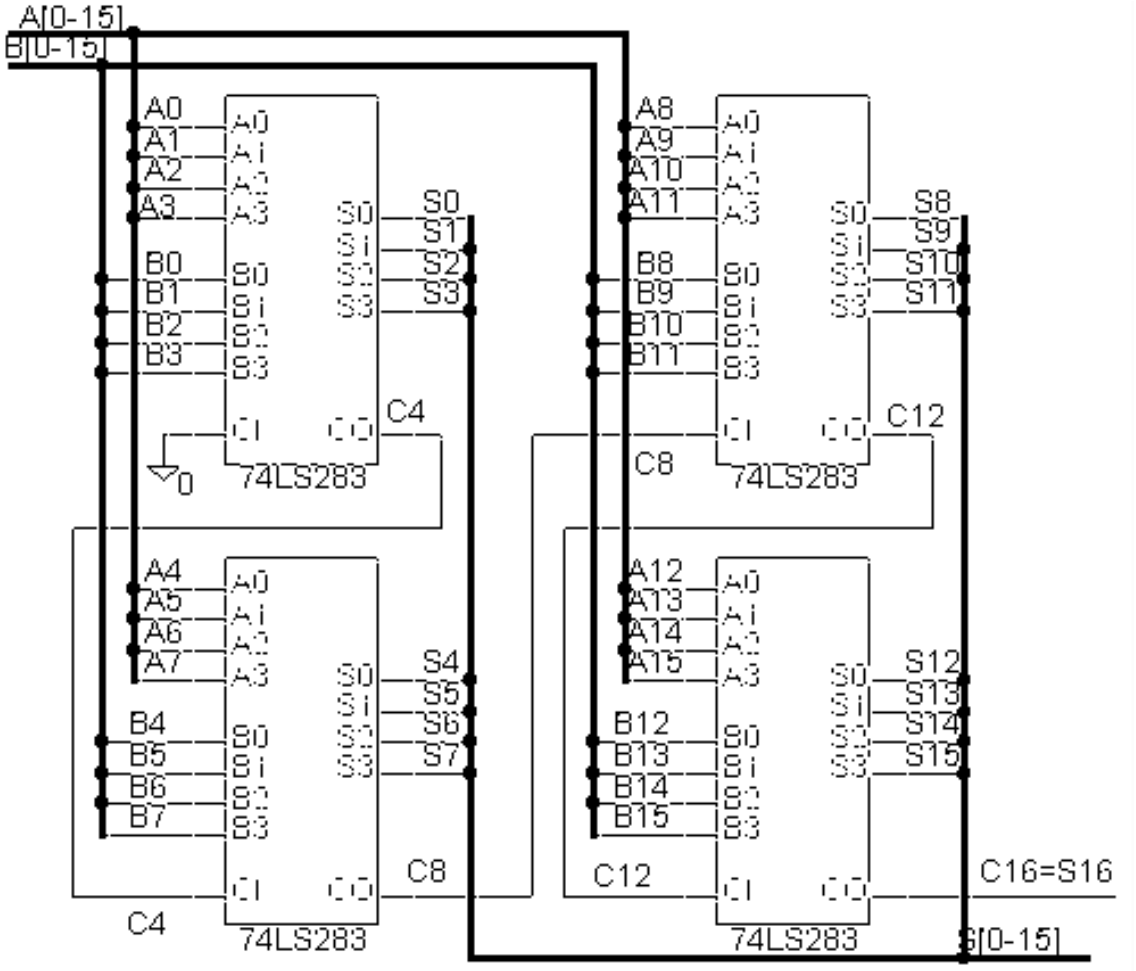
El sumador completo que opera con números de 4 bits es un bloque funcional que existe en circuito integrado. A continuación se presenta un símbolo no normalizado, el símbolo IEEE y el circuito 74283, que es un sumador completo de 4 bits.





Suma binaria (IV)

A continuación se muestra un sumador completo de 16 bits realizado con sumadores completos de 4 bits.



Suma binaria (V)

Propagación en serie del acarreo del sumador de 16 bits (ver diapositiva anterior)

Si el tiempo que tarda un sumador de 4 bits en obtener, a partir de los datos de entrada, los valores de la salida lo denominamos τ, el tiempo que se tarda en obtener la palabra **s[0-16]** como la suma binaria de **a[0-15]** y **b[0-15]**, con este circuito es **4** τ. Esto se debe a que cada etapa debe esperar a que la anterior termine la suma parcial que está realizando. Esta forma de propagar el acarreo entre etapas se denomina propagación de acarreo en serie.

Propagación en paralelo del acarreo:

Sería conveniente que para producir un acarreo no tuviéramos que esperar a la etapa anterior. Esto puede conseguirse mediante un circuito especial denominado generador de acarreo en paralelo. Este circuito permite que los acarreos intermedios lleguen al mismo tiempo a las distintas etapas.

Otros ejemplos de uso de los sumadores

Ejemplo1:

Para obtener una palabra del código **BCD exceso 3**, basta sumar la palabra **0011** (3) al número **BCD natural**.

Ejemplo 2:

Realización de un convertidor de código **BCD natural** a **BCD Aiken**, usando un circuito sumador de 4 bits y puertas lógicas:

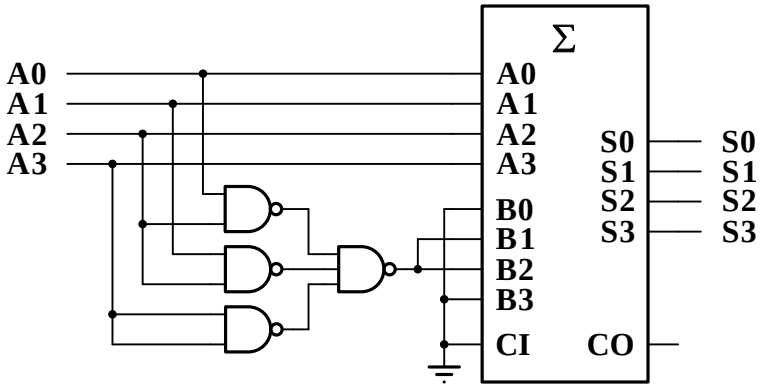
Suma binaria (VI)

Ejemplo 2 (cont.):

	BCD-natural				Palabra a sumar				BCD-Aiken			
n	**A3**	**A2**	**A1**	**A0**	**B3**	**B2**	**B1**	**B0**	**S3**	**S2**	**S1**	**S0**
0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	0	0	0	0	1
2	0	0	1	0	0	0	0	0	0	0	1	0
3	0	0	1	1	0	0	0	0	0	0	1	1
4	0	1	0	0	0	0	0	0	0	1	0	0
5	0	1	0	1	0	1	1	0	1	0	1	1
6	0	1	1	0	0	1	1	0	1	1	0	0
7	0	1	1	1	0	1	1	0	1	1	0	1
8	1	0	0	0	0	1	1	0	1	1	1	0
9	1	0	0	1	0	1	1	0	1	1	1	1
X	1	0	1	0	X	X	X	X	X	X	X	X

De las tablas obtenemos:

B3 = B0 = 0
B1 = B2 = A3 + A2A1 + A2A0



Resta binaria (I)

La resta binaria de dos números puede realizarse también con sumadores, teniendo en cuenta que:

A - B = A + (-B)

Es decir, en este caso, el número B debe venir expresado en negativo y así obtendremos la resta. Los números negativos se expresan en binario usando una de estas dos convenciones:

- ***Complemento a 1***: Se obtiene cambiando ceros por unos y unos por ceros en el número original:

  Ejemplo: Suponiendo que disponemos de 4 bits para representar los números:

  7 -> 0111 ; -7 -> 1000

- **Complemento a 2**: Se obtiene sumándole 1 al complemento a 1

  Ejemplo:

  +5 -> 0101; -5 -> 1011

Normalmente, cuando trabajamos con números que pueden ser positivos o negativos, necesitamos indicar **el signo del número**. La forma más usual es usar una posición concreta, la de mayor peso, para indicar el signo. **Un '0' indica signo positivo y un '1' signo negativo**. En los ejemplos anteriores, el MSB indica el signo en cada caso.

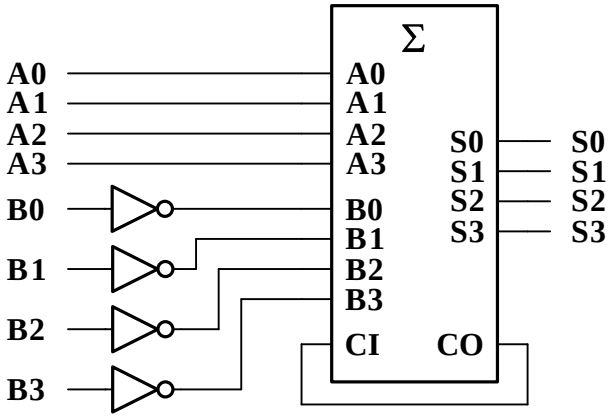
El resto del número (desde el bit menos significativo hasta el anterior al bit de signo, representa la MAGNITUD. Esta magnitud tiene un número fijo de bits y por tanto tenemos un número máximo y un número mínimo que podemos representar. Si nos pasamos de estos límites, se producirá un ***overflow*** (desbordamiento).

Este desbordamiento puede ser fácilmente detectado si se producen estas situaciones:

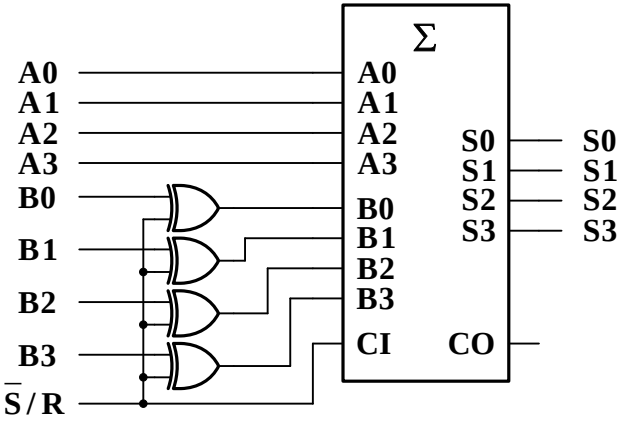
- **Si los dos operandos aplicados a los sumadores son positivos y el resultado es negativo.**
- **Si los dos operandos aplicados a los sumadores son negativos y el resultado es positivo.**

Resta binaria (II)

Circuito restador en complemento a 1



Circuito restador en complemento a 2

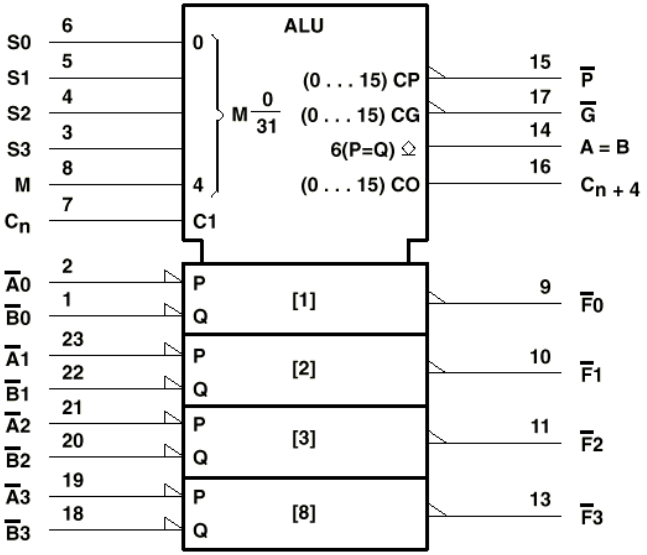


Decimal	Binario	C-1	C-2
+8	1000		
+7	0111	0111	0111
+6	0110	0110	0110
+5	0101	0101	0101
+4	0100	0100	0100
+3	0011	0011	0011
+2	0010	0010	0010
+1	0001	0001	0001
+0	0000	0000	0000
–0		1111	
–1		1110	1111
–2		1101	1110
–3		1100	1101
–4		1011	1100
–5		1010	1011
–6		1001	1010
–7		1000	1001
–8			1000

Unidades Lógicas y Aritméticas (ALU)

Se trata de circuitos que pueden realizar diferentes operaciones aritméticas y lógicas con dos palabras de n bits. Son un elemento imprescindible en los procesadores de propósito general.

Por ejemplo, esta ALU de 4 bits puede realizar hasta 32 funciones diferentes (16 lógicas y 16 aritméticas):



SELECTION				ACTIVE-LOW DATA		
				M = H	M = L; ARITHMETIC OPERATIONS	
S3	S2	S1	S0	LOGIC FUNCTIONS	C_n = L (no carry)	C_n = 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 PLUS 1	F = A PLUS 1

† Each bit is shifted to the next more significant position.

Operaciones aritméticas en VHDL (I)

Suma aritmética

Para realizar la suma de dos números:

S <= A + B;

- **Los números A y B pueden ser vectores y enteros. Los operandos A y B pueden ser de distinta longitud. Si S se declara de la misma longitud que el operando de mayor longitud, el acarreo no se tiene en cuenta. Si se quiere conservar el acarreo, S debe tener un bit más que el operando de mayor longitud, y en ese caso la suma se escribirá:**

  S <= '0' & A + B;

- **Si operamos con números positivos y negativos, deben declararse S, A y B de la misma longitud, y la operación R <= A + B usará el complemento a 2 para representar números negativos. Se puede añadir un circuito que detecte el *overflow* comprobando los bits de signo.**
- **Cuando A o B son de menor longitud que S, el paquete *std_logic_unsigned* añade ceros hasta completar la longitud de la suma. Si se emplea el paquete *std_logic_signed*, la longitud se completa repitiendo el bit de signo.**

Resta aritmética

R <= A – B;

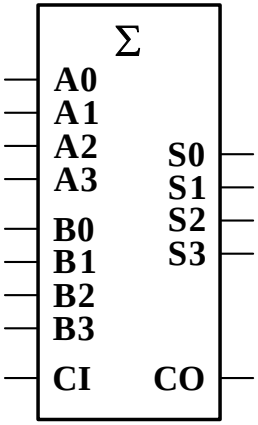
Los operandos pueden ser de distinta longitud, y el resultado R ha de ser de longitud igual o mayor que el operando de mayor longitud.

Ambas operaciones requieren el uso de uno de estos dos paquetes, dependiendo de si trabajamos con números positivos y negativos, o sólo positivos:

```
use ieee.std_logic_unsigned.all;        use ieee.std_logic_signed.all;
```

Operaciones aritméticas en VHDL (II)

Ejemplo: Realizar una descripción VHDL de un sumador completo de 4 bits.



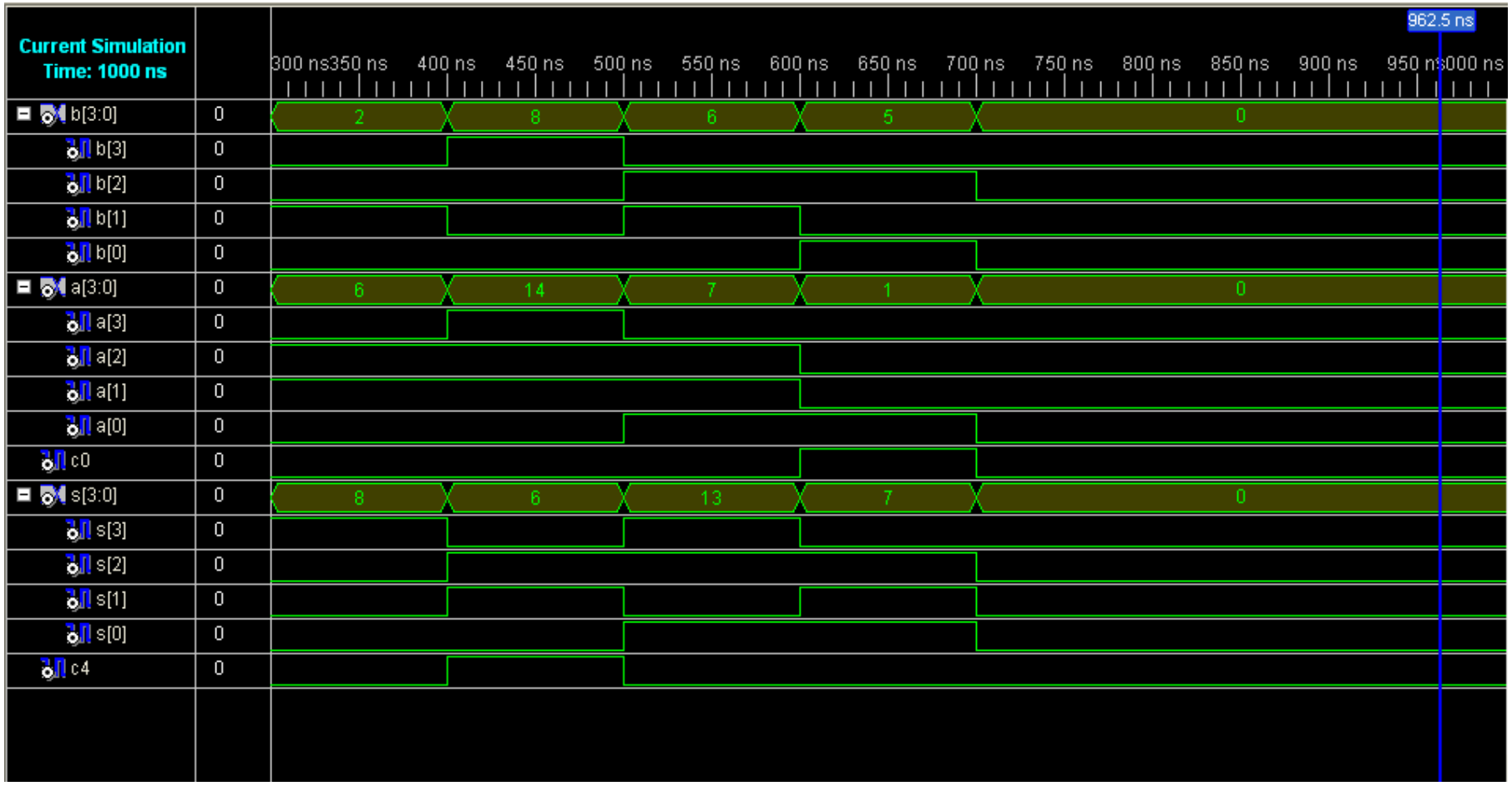

```
-- Sumador de 4 bits
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity sumador_4 is
      port (B, A : in std_logic_vector (3 downto 0);
            C0 : in std_logic;
            S : out std_logic_vector (3 downto 0);
            C4: out std_logic);
end sumador_4;

architecture suma of sumador_4 is
      signal sum : std_logic_vector (4 downto 0);
      begin
            sum <= ('0' & A) + ('0' & B) + ("0000" & C0);
            C4 <= sum(4);
            S <= sum(3 downto 0);
end suma;
```

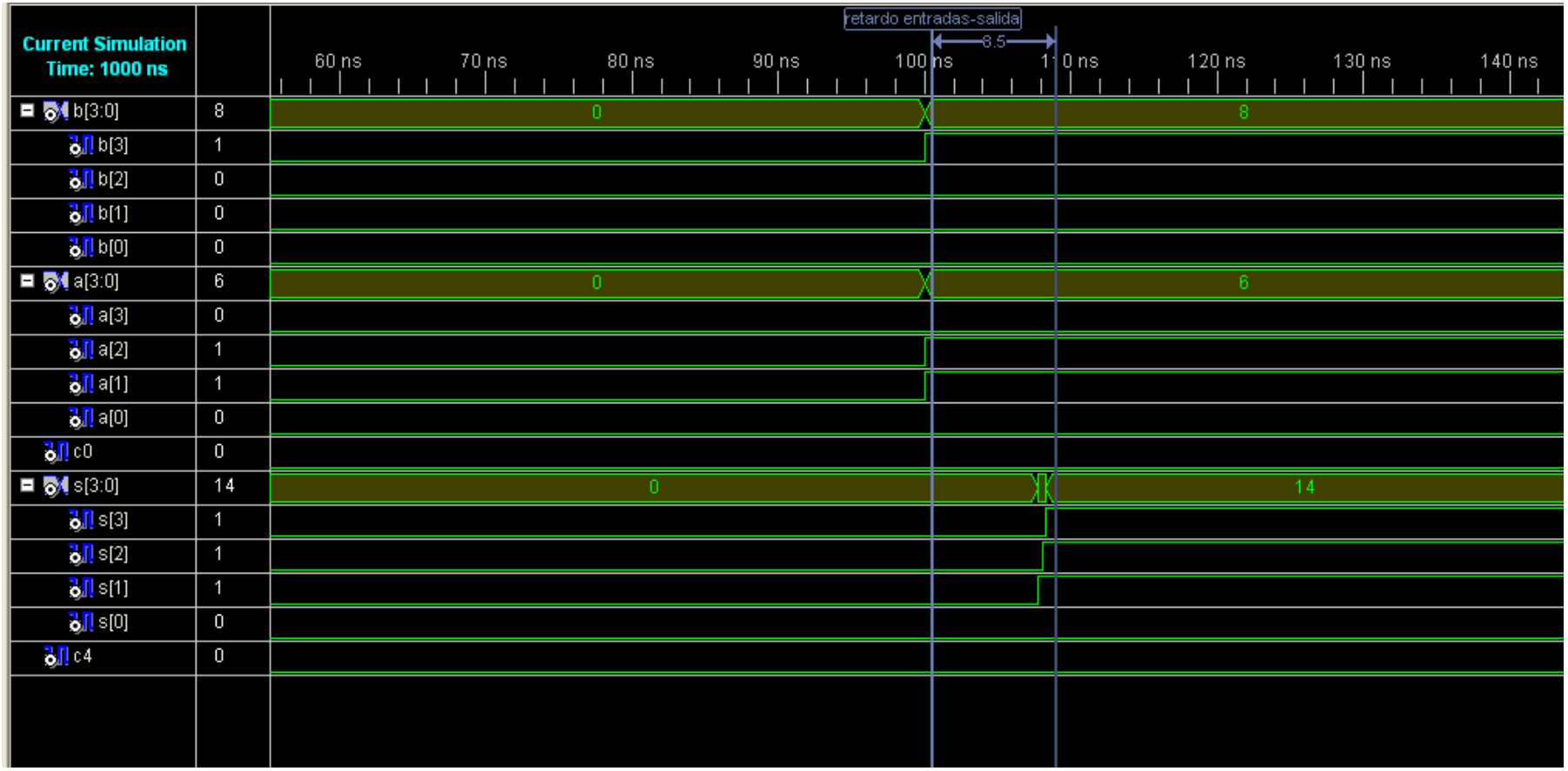
Operaciones aritméticas en VHDL (III)

Simulación funcional del sumador de 4 bits. Si trabajamos sólo con números positivos nunca se produce overflow (si tenemos en cuenta C4).



Operaciones aritméticas en VHDL (IV)

Simulación temporal del sumador de 4 bits una vez implementado en un dispositivo programable real.



Operaciones aritméticas en VHDL (V)

Simulación funcional del sumador de 4 bits. Significado de las salidas en el caso de trabajar con números positivos y negativos (en complemento a 2)