ARITMÉTICA BINARIA

ARITMÉTICA BINARIA.

1.- Suma binaria.

2.- Semisumador (Half Adders).

3.- Sumador completo (Full Adders).

4.- Sumador paralelo con acarreo serie de 4 y 8 bit.

5.- Aritmética binaria con números positivos y negativos.

Sumas y restas en complemento a 1.

Sumas y restas en complemento a 2.

6.- Circuito detector de desbordamiento.

7.- Circuito sumador paralelo de 4 bit con entradas y salidas en complemento a uno.

8.- Circuito sumador paralelo de 4 bit con entradas y salidas en complemento a dos.

9.- Circuitos que evitan el desbordamiento.

10.- Suma o resta mediante señal de control.

Suma binaria.

Suma de dos bit.

Reglas básicas para sumar números en binario:

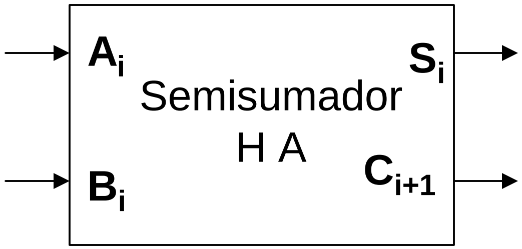
0	+	0	=	0	Suma 0 con acarreo 0
0	+	1	=	1	Suma 1 con acarreo 0
1	+	0	=	1	Suma 1 con acarreo 0
1	+	1	=	10	Suma 0 con acarreo 1 (Resultado de dos bit)

Acarreos	1	1	1	1			
		1	0	1	1	0	22
	+	0	1	1	1	0	+ 14
	1	0	0	1	0	0	36

Bits de acarreo

1	+	0	+	0	=	01	Suma 1 con acarreo 0
1	+	0	+	1	=	10	Suma 0 con acarreo 1
1	+	1	+	0	=	10	Suma 0 con acarreo 1
1	+	1	+	1	=	11	Suma 1 con acarreo 1

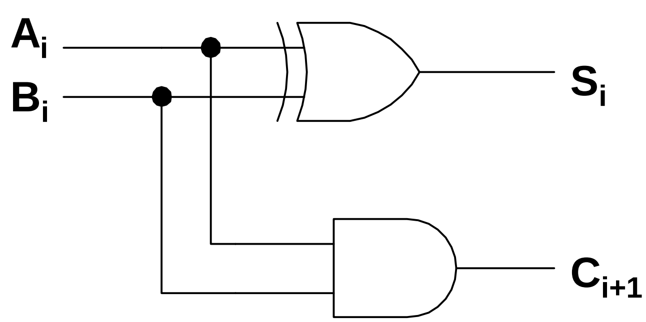
Semisumador (Half Adders).



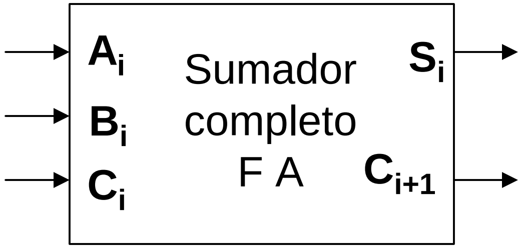
A_i	B_i	S_i	C_{i+1}
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$$\mathbf{S_i = \overline{A_i}\,B_i + A_i\,\overline{B_i} = A_i \oplus B_i}$$

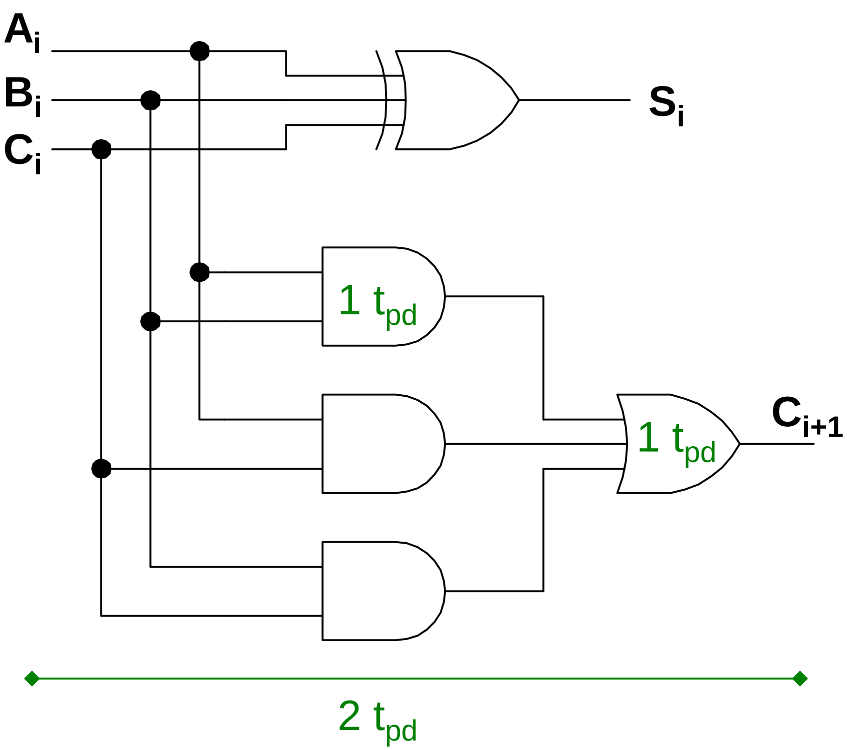
$$\mathbf{C_{i+1} = A_i \; B_i}$$



Sumador completo (Full Adders).



A_i	B_i	C_i	S_i	C_{i+1}
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



$$
\begin{aligned}
S_i &= \overline{A_i}\,\overline{B_i}\,C_i + \overline{A_i}\,B_i\,\overline{C_i} + A_i\,\overline{B_i}\,\overline{C_i} + A_i\,B_i\,C_i \\
&= \overline{A_i}\left(\overline{B_i}\,C_i + B_i\,\overline{C_i}\right) + A_i\left(\overline{B_i}\,\overline{C_i} + B_i\,C_i\right) \\
&= \overline{A_i}\left(B_i \oplus C_i\right) + A_i\left(\overline{B_i \oplus C_i}\right) = A_i \oplus B_i \oplus C_i
\end{aligned}
$$

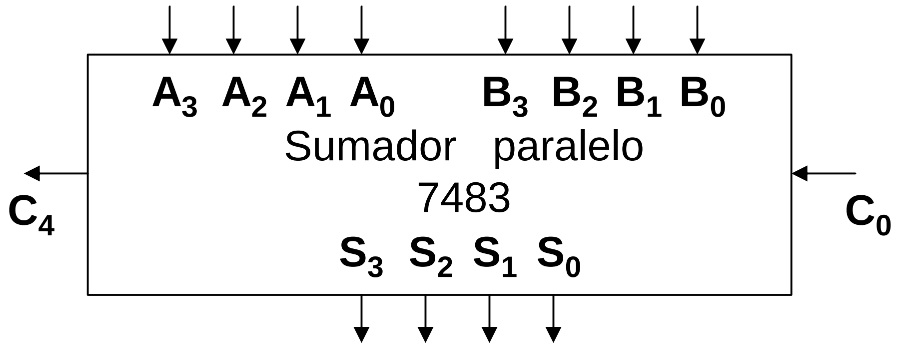
C_i \ $A_i B_i$	00	01	11	10
0	0	0	1	0
1	0	1	1	1

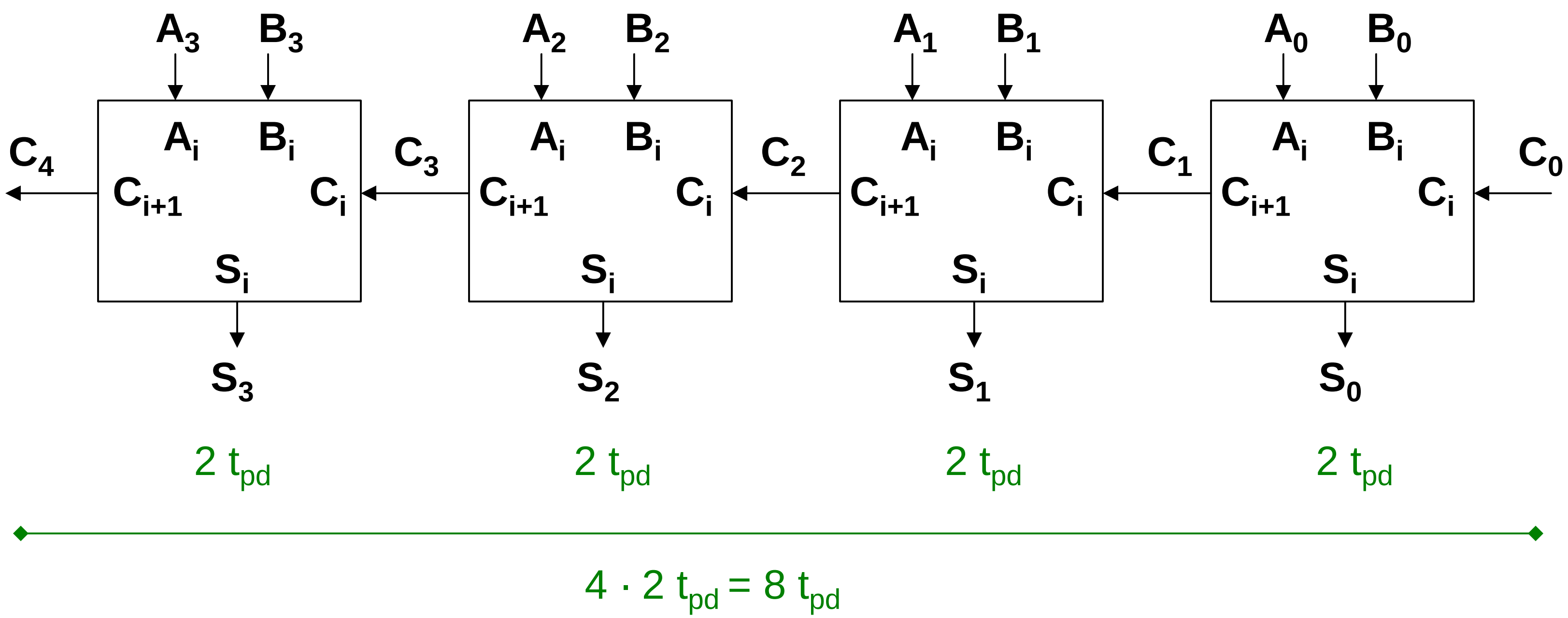
C_{i+1}

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

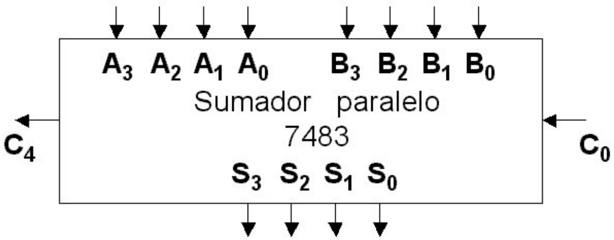
t_{pd} = tiempo de retardo de una puerta.

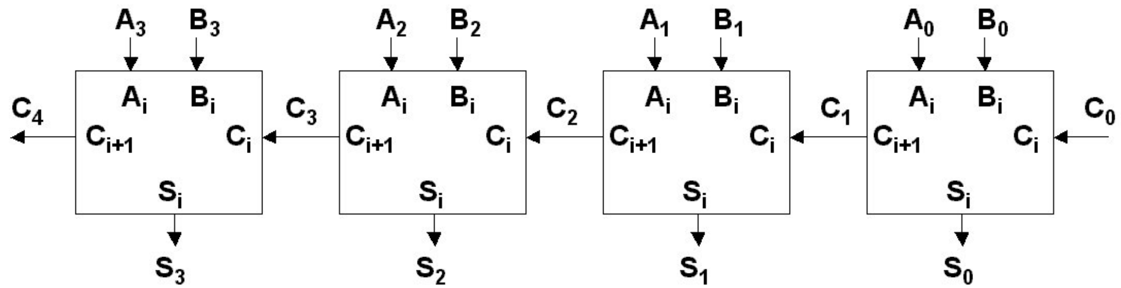
Sumador paralelo con acarreo serie de 4 bit.





Sumador paralelo con acarreo serie de 4 bit.



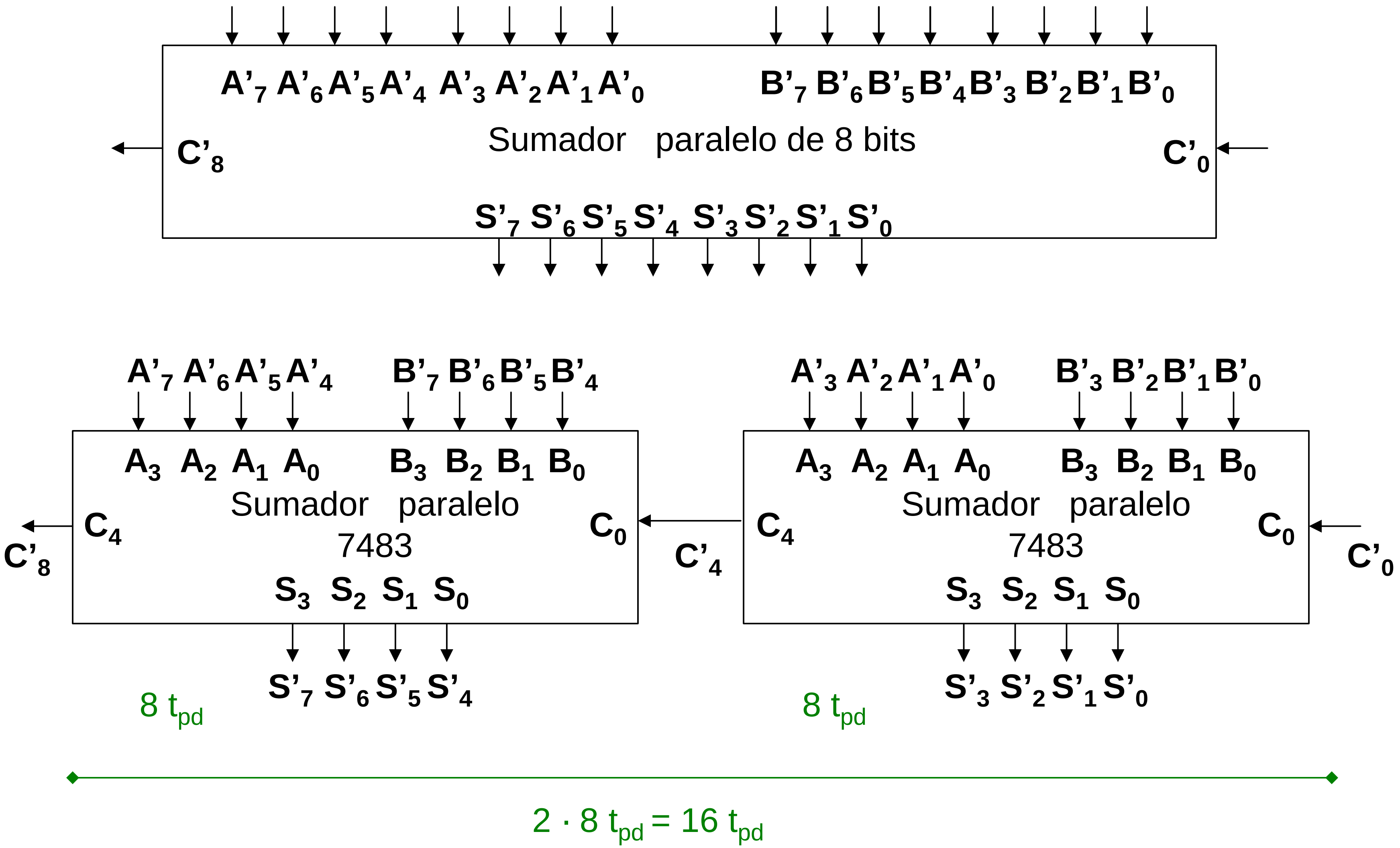


Ejemplo: Suma de A + B siendo A = 1111 y B = 0001

	C_4 C_3 C_2 C_1 C_0			C_4	
C_{i+1}	0 0 0 1 0	0 0 1 1 0	0 1 1 1 0	1 1 1 1 0	
C_i	0 0 0 0 0	0 0 0 1 0	0 0 1 1 0	0 1 1 1 0	
A	1 1 1 1	1 1 1 1	1 1 1 1	1 1 1 1	(A_3 A_2 A_1 A_0)
B	0 0 0 1	0 0 0 1	0 0 0 1	0 0 0 1	(B_3 B_2 B_1 B_0)
S	1 1 1 0	1 1 0 0	1 0 0 0	0 0 0 0	(S_3 S_2 S_1 S_0)
	$2\ t_{pd}$	$2\ t_{pd}$	$2\ t_{pd}$	$2\ t_{pd}$	

$$4 \cdot 2\ t_{pd} = 8\ t_{pd}$$

Sumador paralelo con acarreo serie de 8 bit.



Aritmética binaria con números positivos y negativos.

Comprende la suma y la resta, ya que una resta **A - B** no es más que **A + (-B)**, complementando **B** a 1 o a 2.

Algunos sistemas de representación de números binarios:

BINARIO PURO:

Se toman todos como **positivos**.

El **rango** de representación, para **n** bits, es **{0 , + 2^n-1 }** .   Ej., para **n** = 4 bits, {0 , +15}.

No permite trabajar con números negativos, y tampoco hacer "restas".

Ejemplos, con **n** = 4 bits: 15 = 1111; 8 = 1000; 7 = 0111; 3 = 0011; 0 = 0000.

Aritmética binaria con números positivos y negativos.

VALOR ABSOLUTO Y SIGNO:

Se dedica uno de los bits a indicar si el número es **positivo** (**bit de signo a 0**), o **negativo** (**bit de signo a 1**).

El resto de bits indican el **valor absoluto**, ("leído como binario puro").

El formato es: $b_S , b_{n-2} , b_{n-3} , \ldots , b_1 , b_0$; para un número con **n** bits en total (incluido el bit de signo).

El **rango** de representación, para un total de **n** bits, es de $\{-(2^{n-1}-1) , +(2^{n-1}-1)\}$.

Ej., para **n** = 4 bits, {-7 , +7}.

Permite representar números negativos y positivos, pero **no es posible operar con ellos en sumas y restas directamente**, (salvo que sólo se tomen los positivos y sólo sumas).

Ejemplos, con **n** = 4 bits: + 7 = 0,111; - 7 = 1,111; + 3 = 0,011; - 3 = 1,011;

0 = 0,000 = 1,000; (el **cero** tiene dos posibles representaciones).

Aritmética binaria con números positivos y negativos.

COMPLEMENTO A UNO (C1):

En **C1** se pueden representar números positivos y negativos. Los **positivos** tienen el **bit de mayor peso a 0** y los **negativos a 1**.

Los **números positivos** se representan **igual que en binario** puro. Ej. $\mathbf{0110_{(C1)}} = +\ (0110)_{(2} = \mathbf{+\ 6_{(10}}$

Los **números negativos** se representan mediante el **complemento a uno ($A_{(C1)}$)** del número correspondiente positivo **A**.

El **complemento a uno** de un número entero $A_{(10}$, se define como $A_{(C1)} = (2^n - A_{(10)} - 1)$ expresado en binario, siendo **n** el número de bits con el que se esté trabajando.

Ej.: Número $\mathbf{-2_{(10}}$ en **C1** con **n**=4 bit. Complemento a uno de $A_{(10}=2_{(10}=(0010)_{(2}$; $A_{(C1)} = 16\text{-}2\text{-}1=13_{(10} = (1101)_{(2}$
$\mathbf{(1101)_{(C1)} = -2_{(10}}$

Rango de representación (para un total de **n** bits) $\{-(2^{n-1}-1)\ ,\ +(2^{n-1}-1)\}$. Ej., n = 4 bits, {-7 , +7}.

En complemento a uno se pueden realizar sumas y restas, obteniendose el resultado en **C1 si es negativo (bit de mas peso a 1)**, o en **binario puro si es positivo (bit de más peso a cero)**.

Forma práctica de obtener el C1 de un número:

La forma de hallar el **C1** de un número es **invertir cada uno de sus bits**, o sea, aplicar la función NOT o negación a cada bit.

Ej., con **n** = 4 bits:

$\mathbf{-\ 7_{(10}} = \mathbf{C1}\ (\ 7_{(10}\ ) = \mathbf{C1}\ (\ 0111_{(2}\ ) = \mathbf{1000_{(C1)}}$; $\mathbf{1010_{(C1)}} = -\ \mathbf{C1}\ (1010_{(C1)}) = -\ (0101)_{(2} = \mathbf{-\ 5_{(10}}$

$\mathbf{-\ 3_{(10}} = \mathbf{C1}\ (3_{(10}) = \mathbf{C1}\ (\ 0011_{(2}\ ) = \mathbf{1100_{(C1)}}$; $\mathbf{1110_{(C1)}} = -\ \mathbf{C1}\ (1110_{(C1)}) = -\ (0001)_{(2} = \mathbf{-1_{(10}}$

$\mathbf{0_{(10}} = \mathbf{0000_{(C1)}} = \mathbf{1111_{(C1)}}$, (el **cero** admite dos representaciones).

Aritmética binaria con números positivos y negativos.

COMPLEMENTO A DOS (C2):

En **C2** también se pueden representar números positivos y negativos. Los **positivos** tienen el **bit de mayor peso a 0** y los **negativos a 1**.

Los **números positivos** se representan **igual que en binario** puro. Ej. $\mathbf{0101_{(C2)}} = +\ (0101)_{(2} = \mathbf{+\ 5_{(10}}$

Los **números negativos** se representan mediante el **complemento a dos ($A_{(C2)}$)** del número correspondiente positivo **A**.

El **complemento a dos** de un número entero $A_{(10}$, se define como $A_{(C2)} = (2^n - A_{(10})$ expresado en binario, siendo **n** el número de bits con el que se esté trabajando.

Ej.: Número $\mathbf{-2_{(10}}$ en **C2** con **n**=4 bit. Complemento a dos de $A_{(10}=2_{(10}=(0010)_{(2}$; $A_{(C2)}=16-2=14_{(10} = (1110)_{(2}$

$\mathbf{(1110)_{(C2)} = -2_{(10}}$

Rango de representación (para un total de **n** bits) $\{-(2^{n-1}) , +(2^{n-1}-1)\}$. Ej., n = 4 bits, {-8 , +7}.

En complemento a dos se pueden realizar sumas y restas, obteniendose el resultado en **C2 si es negativo (bit de mas peso a 1)**, o en **binario puro si es positivo (bit de más peso a cero)**.

Forma práctica de obtener el C2 de un número:

1er método: Obtener el **C1 y sumar 1**, ya que $A_{(C2)} = (2^n - A_{(10}) = (2^n - A_{(10} - 1) + 1 = A_{(C1)} + 1$.

Ej. $\mathbf{-7_{(10}}$ = **C2** $(0111_{(2})$ = **C1**$(0111_{(2})$ **+ 1** = $1000_{(C1)}$ **+ 1** = $\mathbf{1001_{(C2)}}$. (Para **n** = 4 bits).

$\mathbf{1100_{(C2)}}$ = **- C2** $(1100_{(2})$ = - [**C1**$(1100_{(2})$ **+ 1**] = - $[0011_{(C1)}$ **+ 1**] = - $(\mathbf{0100_{(2}})$ = - $\mathbf{4_{(10}}$.

2º método: Dejar todos los bits como están hasta encontrar, empezando **por la derecha, el primer "1"; a partir del bit siguiente hasta el de más peso se invierten** (ó compl. a 1).

Ej. $\mathbf{-7_{(10}}$ = **C2** $(011\mathbf{1}_{(2})$ = $100\mathbf{1}_{(C2)}$; $\mathbf{1010_{(C2)}}$ = -**C2** $(10\mathbf{10}_{(C2)})$ = - $(01\mathbf{10}_{(2})$ = $\mathbf{-6_{(10}}$; $\mathbf{-\ 0_{(10}}$= **C2** $(0000_{(2})$ = $0000_{(C2)}$.

Aritmética binaria con números positivos y negativos.

Números de 4 bit (**n** = 4)
$A = A_3\ A_2\ A_1\ A_0$

Números de 3 bit (**n** = 3) **$A = A_2\ A_1\ A_0$**

A_2	A_1	A_0	Binario puro	V.A. y S.	C.1	C.2
0	0	0	0	0	0	0
0	0	1	1	1	1	1
0	1	0	2	2	2	2
0	1	1	3	3	3	3
1	0	0	4	-0	-3	-4
1	0	1	5	-1	-2	-3
1	1	0	6	-2	-1	-2
1	1	1	7	-3	-0	-1

A_3	A_2	A_1	A_0	Binario puro	V.A. y S.	C.1	C.2
0	0	0	0	0	0	0	0
0	0	0	1	1	1	1	1
0	0	1	0	2	2	2	2
0	0	1	1	3	3	3	3
0	1	0	0	4	4	4	4
0	1	0	1	5	5	5	5
0	1	1	0	6	6	6	6
0	1	1	1	7	7	7	7
1	0	0	0	8	-0	-7	-8
1	0	0	1	9	-1	-6	-7
1	0	1	0	10	-2	-5	-6
1	0	1	1	11	-3	-4	-5
1	1	0	0	12	-4	-3	-4
1	1	0	1	13	-5	-2	-3
1	1	1	0	14	-6	-1	-2
1	1	1	1	15	-7	-0	-1

Sumas y restas en complemento a 1.

Ejemplo: n = 5 bits. (Datos de entrada y salida en C.1).

El mayor nº positivo es **01111** (+15) y el menor nº negativo el **10000** (-15).

Cuando el resultado sobrepasa estas cantidades (no cabe en 5 bits) ⇒ **desbordamiento**.

A + B ; A y B > 0:

01000 = + 8	**0**1100 = + 12	**0**1110 = + 14	**0**1111 = + 15
+ **0**0111 = +7	+ **0**1111 = + 15	+ **0**0010 = + 2	+ **0**0000 = + 0
01111 = +15	**1**1011 = - 4 (+27)	**1**0000 = - 15 (+16)	**0**1111 = +15

A + B ; A y B < 0: ***El acarreo de salida debe sumarse al de entrada***

10111 = - 8	**1**0011 = - 12	**1**0001 = - 14	**1**0000 = -15
+ **1**1000 = - 7	+ **1**0000 = - 15	+ **1**1101 = - 2	+ **1**1111 = -0
CARR = 1 01111	1 00011	1 01110	1 01111
→ +1	→ +1	→ +1	→ +1
10000 = -15	**0**0100 = + 4 (-27)	**0**1111 = + 15 (-16)	**1**0000 = -15

Sumas y restas en complemento a 1.

Ejemplo: n = 5 bits. (Datos de entrada y salida en C.1).

(A + B) > 0 ; A y B de DISTINTO SIGNO:

	10111 = - 8	01110 = + 14	00011 = + 3	01111 = + 15
	+ 01100 = + 12	+ 10010 = - 13	+ 11101 = - 2	+ 11111 = - 0
CARR =	1 00011	1 00000	1 00000	1 01110
	+1	+1	+1	+1
	00100 = + 4	00001 = + 1	00001 = + 1	01111 = + 15

<u>En complemento a 1, el acarreo de salida debe sumarse al de entrada</u>

(A + B) £ 0 ; A y B de DISTINTO SIGNO:

01000 = + 8	10001 = - 14	00011 = + 3	10000 = - 15
+ 10011 = - 12	+ 01101 = + 13	+ 11100 = - 3	+ 00000 = + 0
11011 = -4	11110 = - 1	11111 = - 0	10000 = - 15

Al ser los operandos de distinto signo es imposible que se produzca desbordamiento.

Sumas y restas en complemento a 2.

Ejemplo: n = 5 bits. (Datos de entrada y salida en C.2).

El mayor nº positivo es **01111** (+15) y el menor nº negativo el **10000** (-16).

Cuando el resultado sobrepasa estas cantidades (no cabe en 5 bits) ⇒ **desbordamiento**.

A + B ; A y B > 0:

01000 = + 8	**0**1100 = + 12	**0**1110 = + 14	**0**1111 = + 15
+ **0**0111 = + 7	+ **0**1111 = + 15	+ **0**0010 = + 2	+ **0**0000 = + 0
01111 = + 15	**1**1011 = - 5 (+27)	**1**0000 = - 16 (+16)	**0**1111 = + 15

A + B ; A y B < 0:

	11000 = - 8	**1**0100 = - 12	**1**0010 = - 14	**1**0001 = - 15
	+ **1**1110 = - 2	+ **1**0001 = - 15	+ **1**1101 = - 3	+ **1**1111 = - 1
CARR =	1 **1**0110 = - 10	1 **0**0101 = + 5 (-27)	1 **0**1111 = + 15 (-17)	1 **1**0000 = -16

En complemento a 2, el acarreo de salida se desprecia

Sumas y restas en complemento a 2.

Ejemplo: n = 5 bits. (Datos de entrada y salida en C.2).

(A + B) ≥ 0 ; A y B de DISTINTO SIGNO:

	11000 = - 8	01110 = + 14	00011 = + 3	01111 = + 15
	+ 01100 = + 12	+ 10011 = - 13	+ 11101 = - 3	+ 11111 = - 1
CARR = 1	00100 = + 4	1 00001 = + 1	1 00000 = 0	1 01110 = + 14

En complemento a 2, el acarreo de salida se desprecia

(A + B) < 0 ; A y B de DISTINTO SIGNO:

01000 = + 8	10010 = - 14	00011 = + 3	10000 = - 16
+ 10100 = - 12	+ 01101 = + 13	+ 11100 = - 4	+ 00000 = 0
11100 = - 4	11111 = - 1	11111 = -1	10000 = - 16

Al ser los operandos de distinto signo es imposible que se produzca desbordamiento.

Sumas (restas) en C.1 y C. 2. Resumen.

• En **c.1 se suma el acarreo de salida al de entrada**.

• En **c.2 se ignora el acarreo**.

• Existirá **desbordamiento** siempre que los dos bits de más peso de los operandos sean iguales, pero el del resultado sea distinto:

$$\textbf{DESBORDAMIENTO} = D_E = A_{n-1} B_{n-1} \overline{S_{n-1}} + \overline{A_{n-1}}\,\overline{B_{n-1}}\, S_{n-1}$$

donde **A** $_{n-1}$, **B** $_{n-1}$ y **S** $_{n-1}$ son los bits de más peso (indican signo) de los dos operandos y del resultado, respectivamente.

• **Si se añaden "unos" a la izquierda de un número negativo, o bien "ceros" a la izquierda de un positivo, el número no cambia**. Esto permite trabajar con un número de bits incluso mayor del necesario, sin más que **extender el bit de más peso** (que indica el signo) **hacia la izquierda**. Ejemplos:

En c.1:

n = 4 bits	n = 8 bits	Valor decimal
0111	**00000**111	+ 7
0000	**00000**000	+ 0
1111	**11111**111	- 0
1000	**11111**000	- 7

En c.2:

n = 4 bits	n = 8 bits	Valor decimal
0111	**00000**111	+ 7
0000	**00000**000	+ 0
1111	**11111**111	- 1
1000	**11111**000	- 8

Circuito detector de desbordamiento.

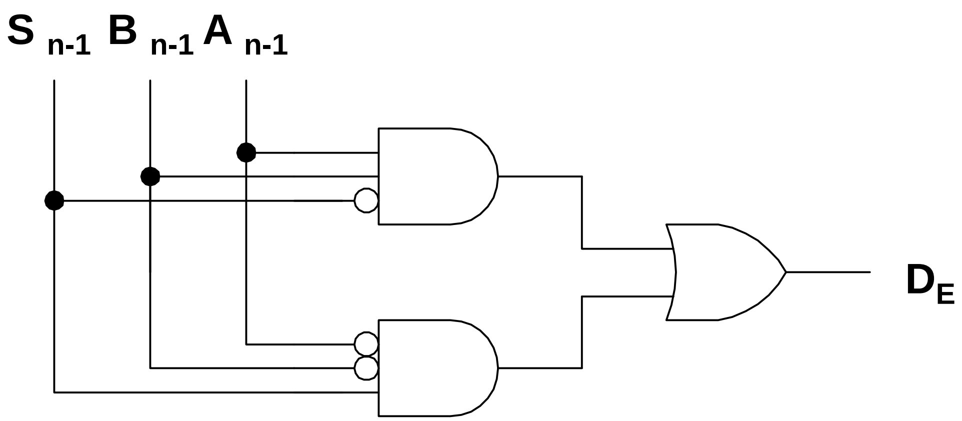
Al efectuar la suma de dos números de n bits

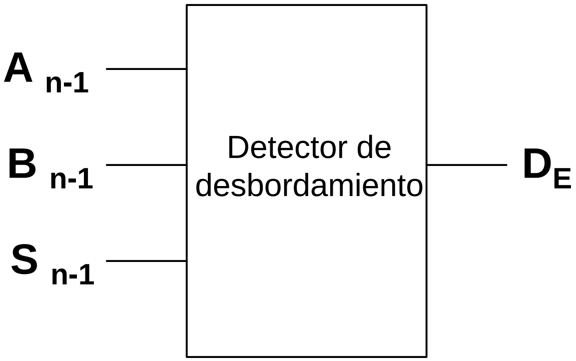
$$\mathbf{A} = \mathbf{A}_{n-1}\ \mathbf{A}_{n-2} \ldots \mathbf{A}_{1}\ \mathbf{A}_{0} \qquad \text{y} \qquad \mathbf{B} = \mathbf{B}_{n-1}\ \mathbf{B}_{n-2} \ldots \mathbf{B}_{1}\ \mathbf{B}_{0}$$

se obtiene un resultado de n bit $\mathbf{S} = \mathbf{S}_{n-1}\ \mathbf{S}_{n-2} \ldots \mathbf{S}_{1}\ \mathbf{S}_{0}$.

Existirá **desbordamiento** siempre que los dos bits de más peso de los operandos ($\mathbf{A}_{n-1}$ y $\mathbf{B}_{n-1}$) sean iguales, pero el del resultado ($\mathbf{S}_{n-1}$) distinto. Por tanto:

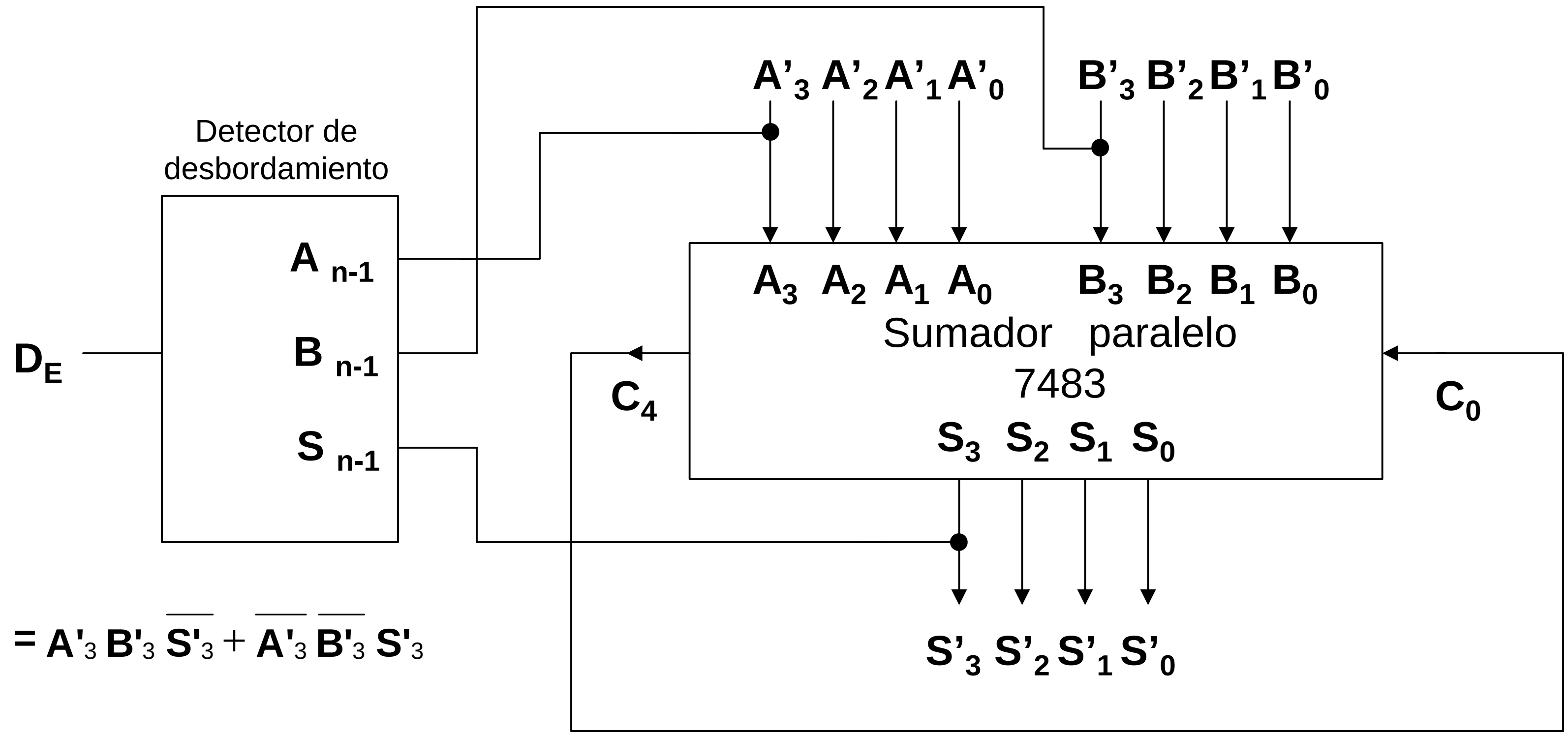
$$\textbf{DESBORDAMIENTO} = \quad D_E = A_{n-1} B_{n-1} \overline{S_{n-1}} + \overline{A_{n-1}}\,\overline{B_{n-1}}\, S_{n-1}$$





Circuito sumador paralelo de 4 bit con entradas y salidas en complemento a uno.

En complemento a 1, el acarreo de salida debe sumarse al de entrada



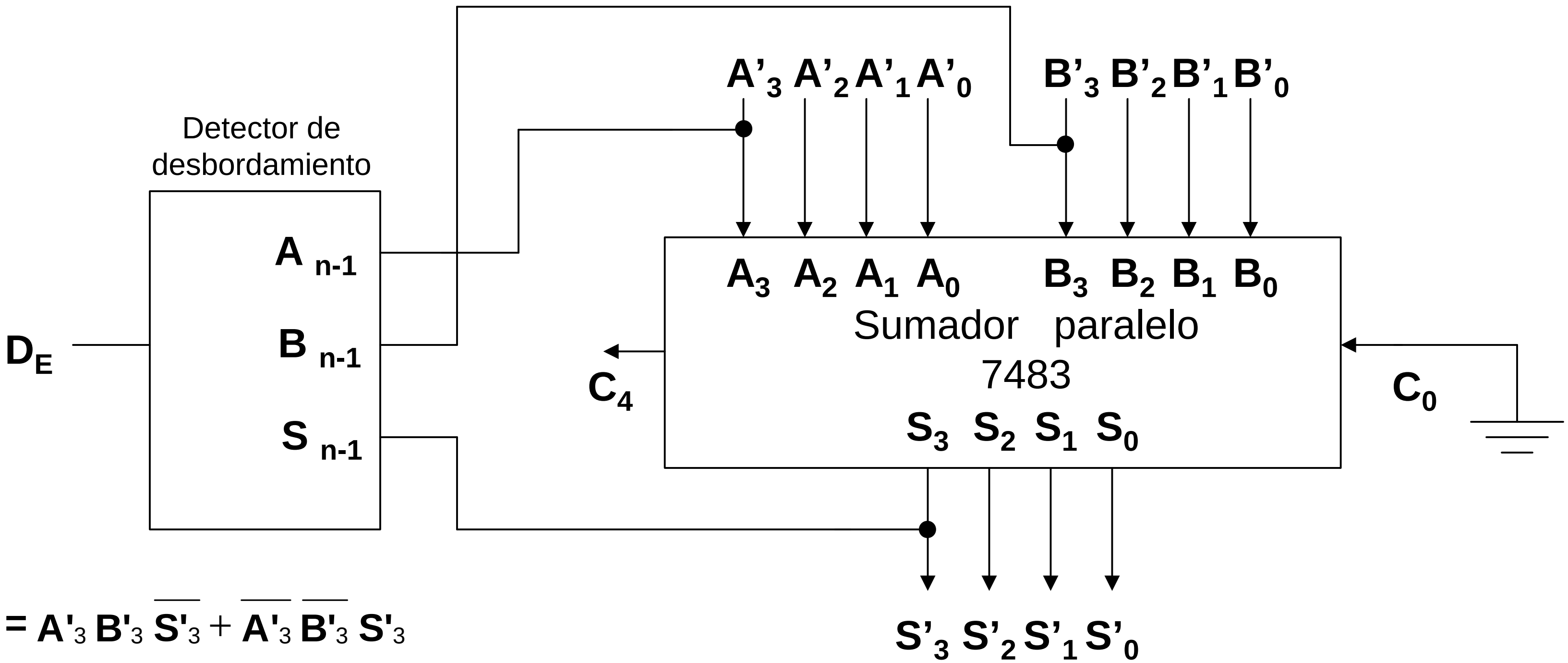
$$D_E = A'_3\, B'_3\, \overline{S'_3} + \overline{A'_3}\, \overline{B'_3}\, S'_3$$

ESCUELA DE INGENIERÍAS INDUSTRIALES

UVa

Circuito sumador paralelo de 4 bit con entradas y salidas en complemento a dos.

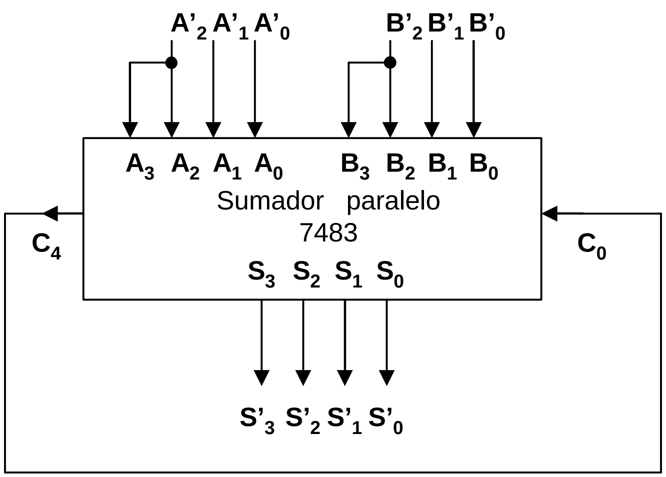
En complemento a 2, el acarreo de salida se desprecia



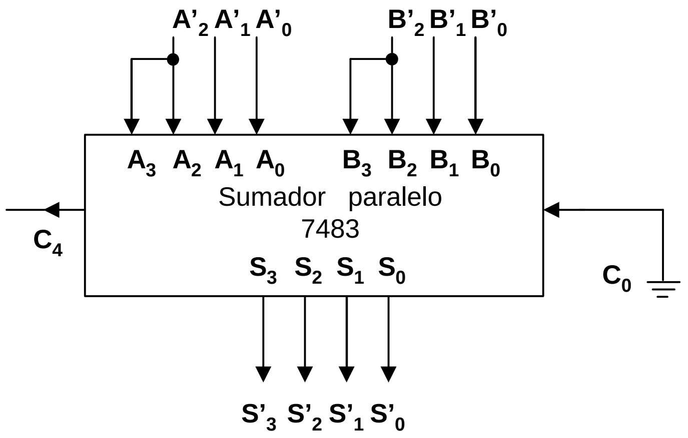
$$D_E = A'_3\,B'_3\,\overline{S'_3} + \overline{A'_3}\,\overline{B'_3}\,S'_3$$

Circuitos sumadores que evitan el desbordamiento.

Se puede evitar que aparezca desbordamiento calculando cual es el rango del resultado, y añadiendo los bits que sean necesarios extendiendo el bit de mayor peso (bit de signo) hacia la izquierda.
Si se añaden “unos” a la izquierda de un número negativo, o “ceros” a la izquierda de un número positivo, el número no cambia.



Sumador con entradas de 3 bits, salida en C.1, evitando el desbordamiento.

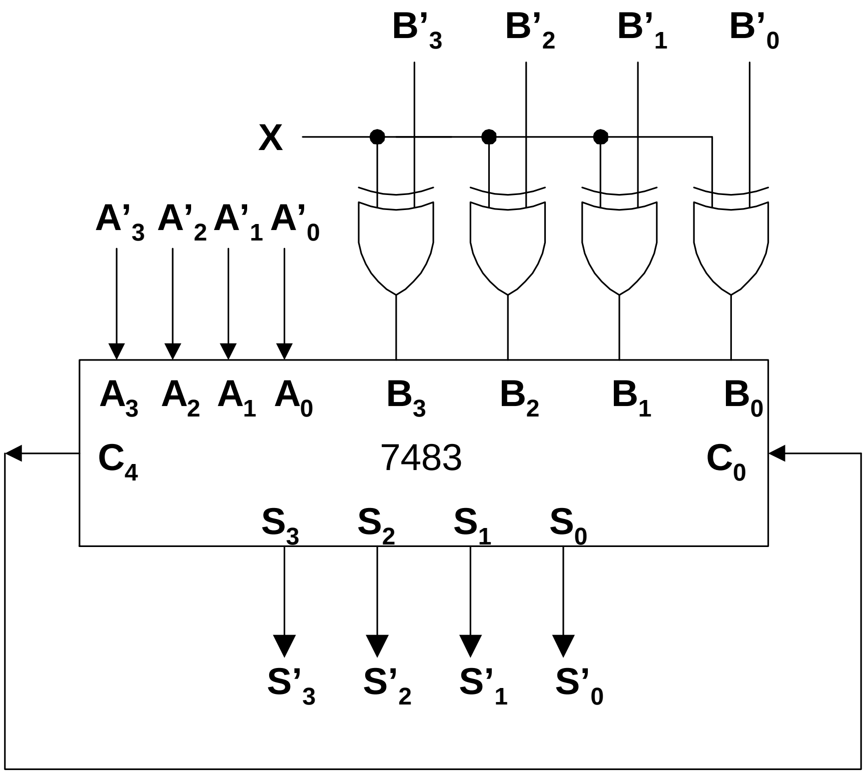


Sumador con entradas de 3 bits, salida en C.2, evitando el desbordamiento.

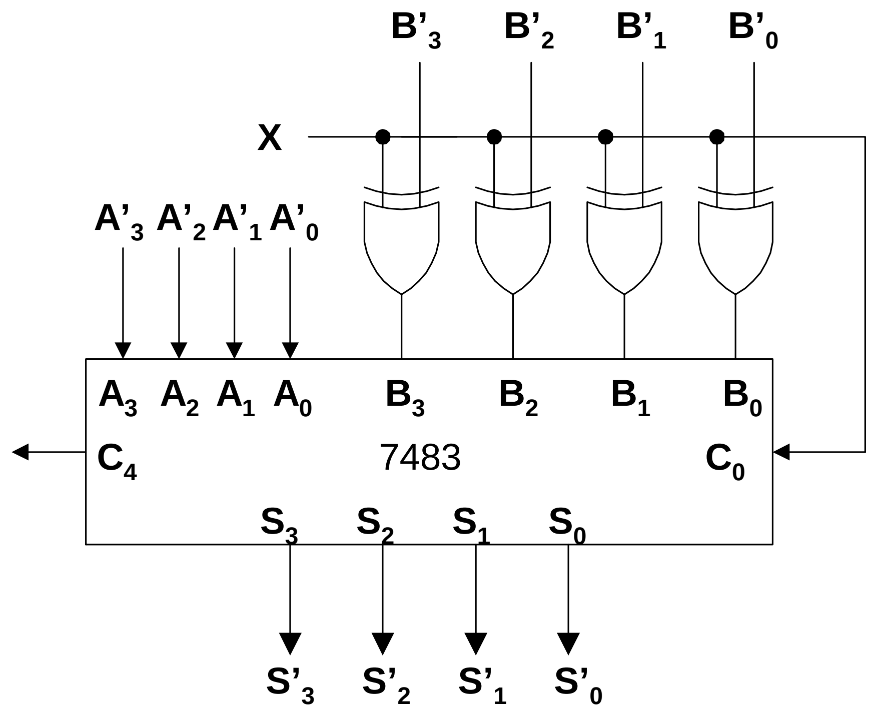
Suma o resta mediante señal de control.

Señal de control **X ($\overline{S}$ / R)**

X = 0 þ Suma de A + B
X = 1 þ Suma de A + (- B) : resta de A - B



Suma en complemento a uno.



C2 = C1 + 1

Suma en complemento a dos.