Tema 2

LÓGICA COMBINACIONAL (I): FUNCIONES ARITMÉTICO-LÓGICAS (Tema 5 del libro)

http://Prof.mfbarcell.es

TEMA 2: LÓGICA COMBINACIONAL (I): FUNCIONES ARITMÉTICO-LÓGICAS

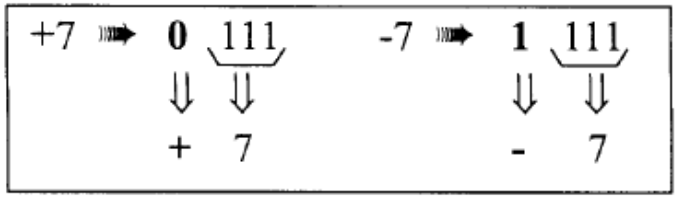
- *Contexto*
- *Conocimiento Previo necesario*
- *Objetivos del Tema*
- *Guía de Estudio*
- *Contenido del Tema*
  - 2.1. Representación Conjunta de Números Positivos y Negativos
  - 2.2. Sumadores y Restadores
    - 2.2.1. Semisumadores
    - 2.2.2. Sumadores
    - 2.2.3. Semirrestadores
    - 2.2.4. Restadores Completos
    - 2.2.5. Sumador Serie
    - 2.2.6. Sumador Paralelo con Acarreo Adelantado
  - 2.3. Sumadores en Complemento a 1: Gestión del Problema del Rebose
  - 2.4. Comparadores
  - 2.5. Unidades Aritmético-Lógicas (ALUs)
  - 2.6. Problemas
- *Preparación de la Evaluación*
- *Referencias Bibliográficas*

Objetivo 1: *Conocer los fundamentos de la aritmética binaria. Distintas formas de representación de los números positivos y negativos.*

Objetivo 2: *Conocer los distintos tipos de circuitos sumadores y restadores (soluciones básicas y rápidas y problema del rebose).*

Objetivo 3: *Comprender el funcionamiento de los comparadores para palabras de* n *bits.*

Objetivo 4: *Comprender el funcionamiento y la estructura interna de las ALU's tipo* SN74181.



En general, para palabras de n bits, tendremos 2^{n-1} números positivos y 2^{n-1} negativos.

Hay tres formas básicas que permiten la representación conjunta de números positivos y negativos: (1) *Signo y Magnitud* (**S-M**), (2) *Complemento a* 1 (***C-1***) y (3) *Complemento a* 2 (***C-2***).

5.1 Representación conjunta de números positivos y negativos

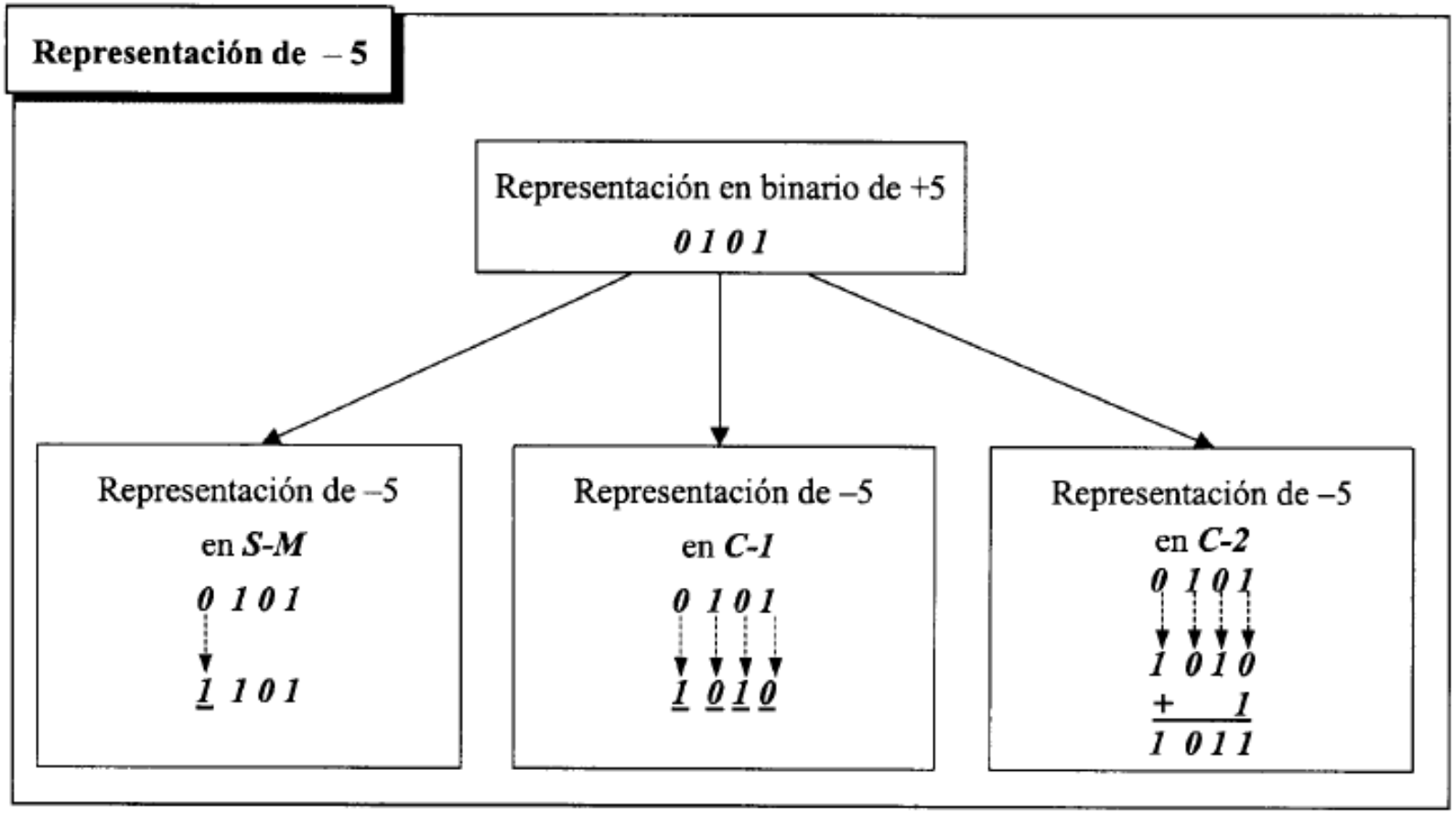


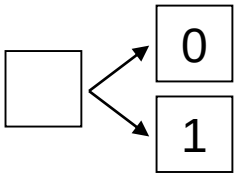
Figura 5.3. Ilustración del proceso de obtención de las tres representaciones de –5 (*S-M*, *C-1* y *C-2*).

Representación conjunta de números positivos y negativos.

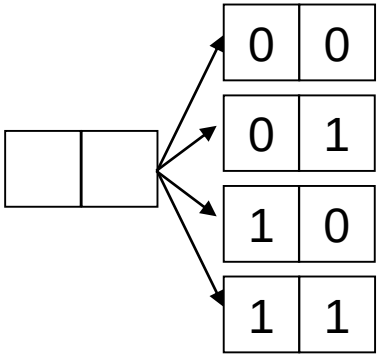
Representación de números enteros.

Ejemplo:

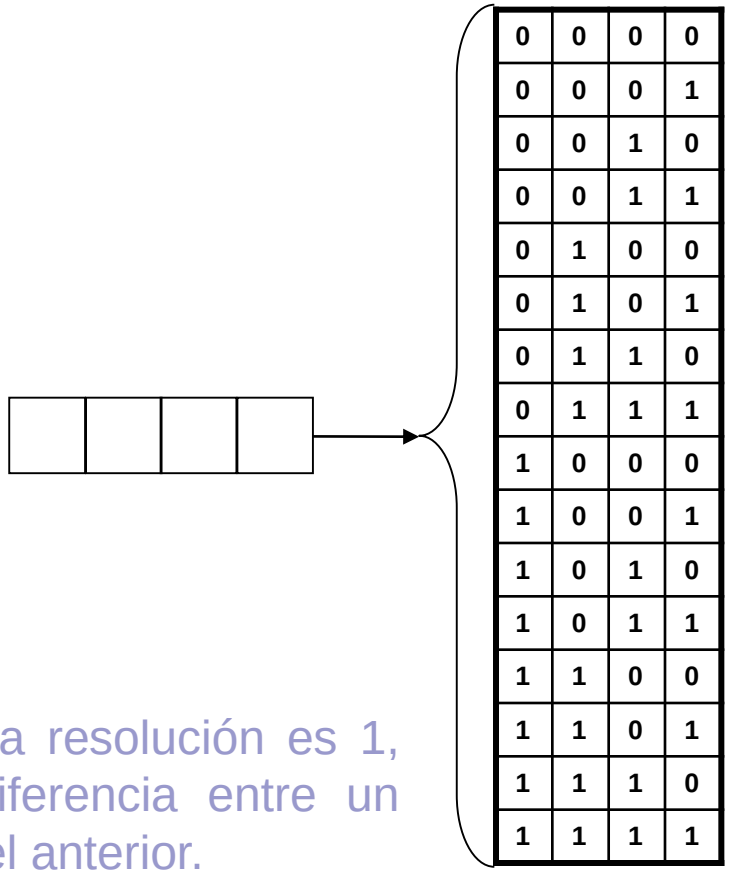
Un número binario con 1 bit, su rango de representación es del 0 al $2^1-1=1$.



Un número binario con 2 bits, su rango de representación es del 0 al $2^2-1=3$.



Un número binario con 4 bits, su rango de representación es del 0 al $2^4-1=15$.



0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1

En todos estos casos, la resolución es 1, ya que es la mayor diferencia entre un número y el siguiente o el anterior.

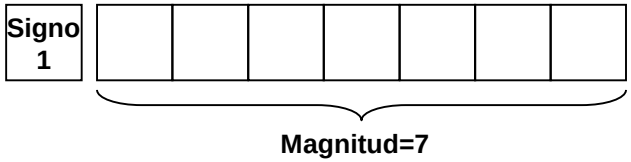
Representación de números positivos y negativos.

Representación de números enteros.

Ejemplo:

Calcular el rango de un número de 8 bits expresado en signo y magnitud, teniendo en cuenta que el bit de signo está incluido.

De los 8 bits, utilizaremos 1 para el bit de signo, y el resto (7 bits) para la magnitud.



Como de los ***n=8*** bits, hemos utilizado 1 para el signo, nos quedarán ***n-1=7*** bits para representar la magnitud del número, por lo que el rango de representación será:

$$[-2^{7}-1, +2^{7}-1] = \mathbf{[-127, +127]}$$

Representación de números positivos y negativos.

Representación de números enteros.

Complementos.

- *Convenio de complemento a 2 en números binarios.*

Existen 2 métodos para representar un número negativo en complemento a 2.

- ***Método 1***: Cambiar todos los unos por ceros y los ceros por unos y luego sumarle 1.
- ***Método 2***: Comenzando por la derecha, dejar intactos todos los dígitos hasta el primer 1 inclusive y cambiar del resto los unos por ceros y los ceros por unos.

Ejemplo: Representar los números decimales 42 y –42, en complemento a dos.

Como el 42 es positivo, se representa en signo y magnitud: **0 101010**

Para representar el -42 probamos los 2 métodos:

Método 1.

	0	1	0	1	0	1	0	Cambiamos los 0 por 1
	1	0	1	0	1	0	1	Cambiamos los 1 por 0
+							1	Sumamos 1
	1	0	1	0	1	1	0	

Método 2.

0	1	0	1	0	1	0	Dejamos intactos los bits de la derecha hasta el primer 1
1	0	1	0	1	1	0	Complementamos el resto de bits cambiando unos por ceros y ceros por unos.

Representación de números enteros.

Complementos.

- *Convenio de complemento a 2 en números binarios.*

En la siguiente tabla se puede ver la representación de los números binarios de 4 bits incluido el signo en complemento a 2.

Decimal	C 2	
7	0	111
6	0	110
5	0	101
4	0	100
3	0	011
2	0	010
1	0	001
0	0	000

Decimal	C 2	
-1	1	111
-2	1	110
-3	1	101
-4	1	100
-5	1	011
-6	1	010
-7	1	001
-8	1	000

La representación en C2, no tiene estructura posicional de pesos.
Para conocer la magnitud del número complementado a 2, hay que complementarlo para obtener su correspondiente magnitud en positivo.

La representación de los números positivos es la misma que en binario con signo.

Tan sólo sufren transformación los número negativos.

El rango de representación en C2 es **$[-2^{n-1}, 2^{n-1}-1]$**.

Representación de números enteros.

Complementos.

- *Convenio de complemento a 1 en números binarios.*

En la siguiente tabla se puede ver la representación de los números binarios de 4 bits incluido el signo en complemento a 1.

Decimal	C 1	
7	0	111
6	0	110
5	0	101
4	0	100
3	0	011
2	0	010
1	0	001
0	0	000

La representación de los números positivos es la misma que en binario con signo.

Decimal	C 1	
-0	1	111
-1	1	110
-2	1	101
-3	1	100
-4	1	011
-5	1	010
-6	1	001
-7	1	000

Tan sólo sufren transformación los número negativos.

En el C1, se tiene en cuenta el bit de acarreo a la hora de realizar la resta.

La representación en C1, al igual que el C2, permite la representación de números negativos.

Si a un número binario se le suma su complemento a 1, da como resultado 0, ***siempre y cuando se sume el bit de acarreo si se produce***,

Para conocer la magnitud del número complementado a 1, hay que complementarlo para obtener su correspondiente magnitud en positivo.

El rango en C1 es ***simétrico*** **$[-(2^{n-1}-1), +(2^{n-1}-1)]$**.

El 0 se representa tanto con signo positivo como negativo.

Comparación entre las representaciones diferentes de números binarios con signo.

- ✓ Los complementos permiten realizar las sumas y restas con el mismo circuito digital, mientras que la representación en signo-magnitud no.
- ✓ En complemento a 2 no es necesario tener en cuenta el bit de acarreo a la hora de realizar las sumas o restas, mientras que en complemento a 1 sí.
- ✓ El rango de representación es simétrico.
- ✓ En complemento a 1 la transformación es más sencilla, pero tiene el inconveniente de la doble representación del 0 (±0).

Decimal	Signo-Magnitud		C 1		C2	
7	0	111	0	111	0	111
6	0	110	0	110	0	110
5	0	101	0	101	0	101
4	0	100	0	100	0	100
3	0	011	0	011	0	011
2	0	010	0	010	0	010
1	0	001	0	001	0	001
0	0	000	0	000	0	000
-0	1	000	1	111	- - -	- - -
-1	1	001	1	110	1	111
-2	1	010	1	101	1	110
-3	1	011	1	100	1	101
-4	1	100	1	011	1	100
-5	1	101	1	010	1	011
-6	1	110	1	001	1	010
-7	1	111	1	000	1	001
-8	- - -	- - -	- - -	- - -	1	000

Configuraciones Binarias (4 bits)	*Nº equivalente en decimal* según las distintas representaciones en binario			
	Binario Puro	***S-M***	***C-1***	***C-2***
0 000	0	+0	+0	+0
0 001	1	+1	+1	+1
0 010	2	+2	+2	+2
0 011	3	+3	+3	+3
0 100	4	+4	+4	+4
0 101	5	+5	+5	+5
0 110	6	+6	+6	+6
0 111	7	+7	+7	+7
1 000	8	-0	-7	-8
1 001	9	-1	-6	-7
1 010	10	-2	-5	-6
1 011	11	-3	-4	-5
1 100	12	-4	-3	-4
1 101	13	-5	-2	-3
1 110	14	-6	-1	-2
1 111	15	-7	-0	-1

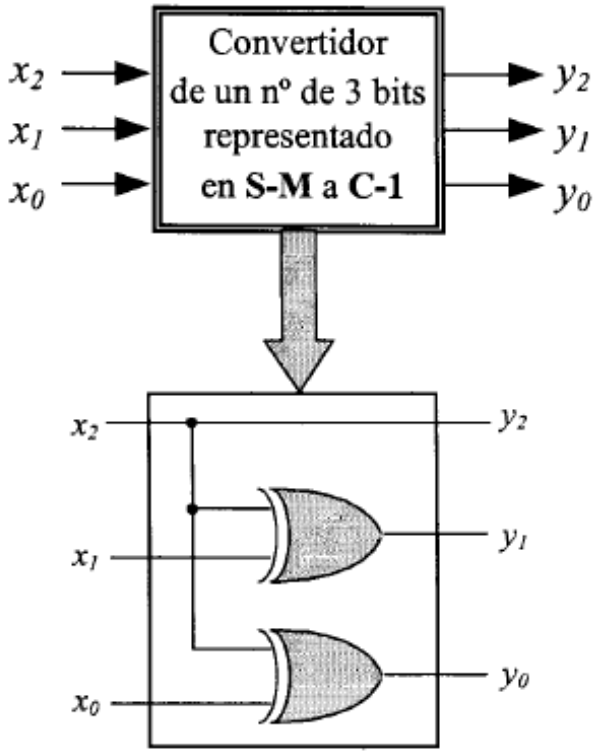
Figura 5.1. Tabla resumen de las distintas formas de representación conjunta de números positivos y negativos. Pasamos de binario al número equivalente en decimal.

$$y_2 = x_2$$

$$y_1 = \bar{x}_2 x_1 x_0 + \bar{x}_2 x_1 \bar{x}_0 + x_2 \bar{x}_1 \bar{x}_0 + x_2 \bar{x}_1 x_0 = x_2 \oplus x_1 \qquad [5.1]$$

$$y_0 = \bar{x}_2 x_1 x_0 + \bar{x}_2 \bar{x}_1 x_0 + x_2 \bar{x}_1 \bar{x}_0 + x_2 x_1 \bar{x}_0 = x_2 \oplus x_0$$

Signo y Magnitud (S-M)
x_2 x_1 x_0
0 1 1
0 1 0
0 0 1
0 0 0
1 0 0
1 0 1
1 1 0
1 1 1

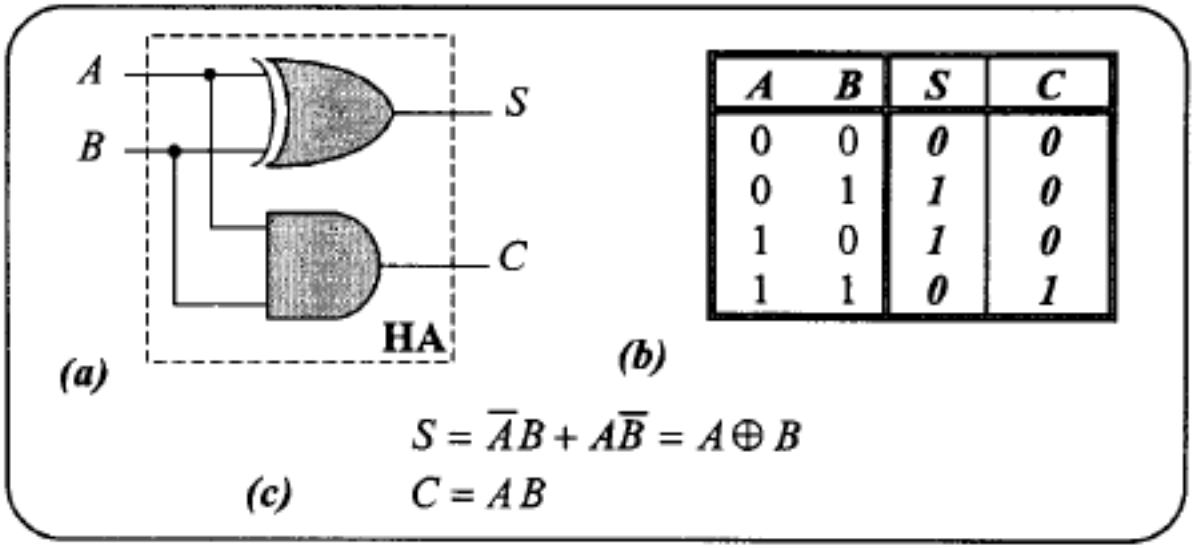


Complemento a 1 (C-1)
y_2 y_1 y_0
0 1 1
0 1 0
0 0 1
0 0 0
1 1 1
1 1 0
1 0 1
1 0 0

Figura 5.4. Circuito convertidor de S-M a C-1.

5.2 Sumadores y Restadores

- 5.2.1 Semisumador
  - No tiene en cuenta el acarreo de la fase anterior
  - Suma dos bits

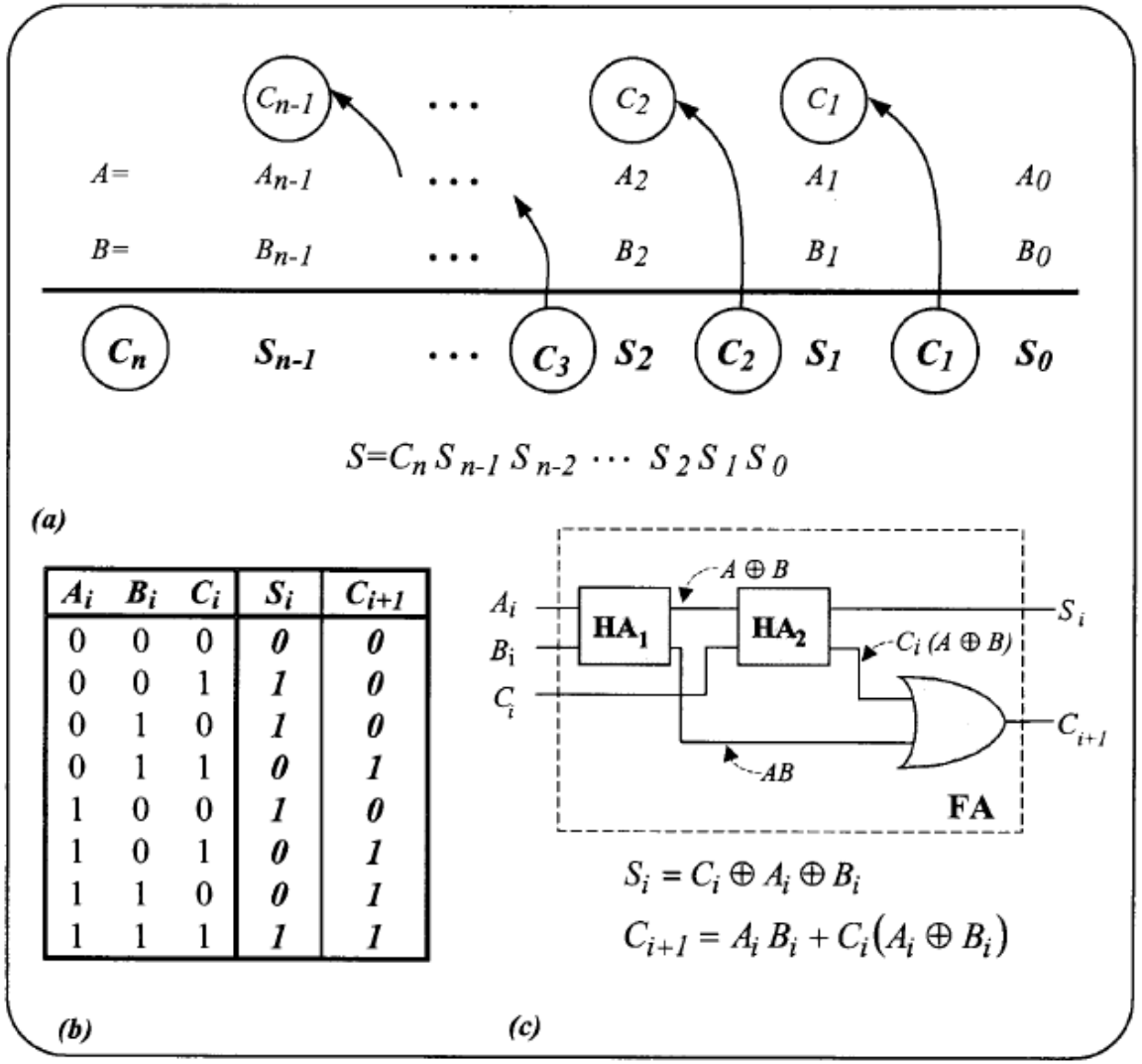


A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Figura 5.5. Semisumador. ***(a)*** Circuito. ***(b)*** Tablas de verdad. ***(c)*** Expresiones de la suma y el arrastre.

5.2.2 Sumadores

- Tiene en cuenta el acarreo de la fase anterior
- Para su síntesis necesitamos tener de un módulo que sumará tres bits A_i, B_j , C_i y que produzca la suma local y el arrastre S_i, C_i



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

Figura 5.6. Sumador completo. ***(a)*** Algoritmo de suma de dos números de *n* bits. ***(b)*** Síntesis a partir de semisumadores. ***(c)*** Tabla de verdad del sumador completo.

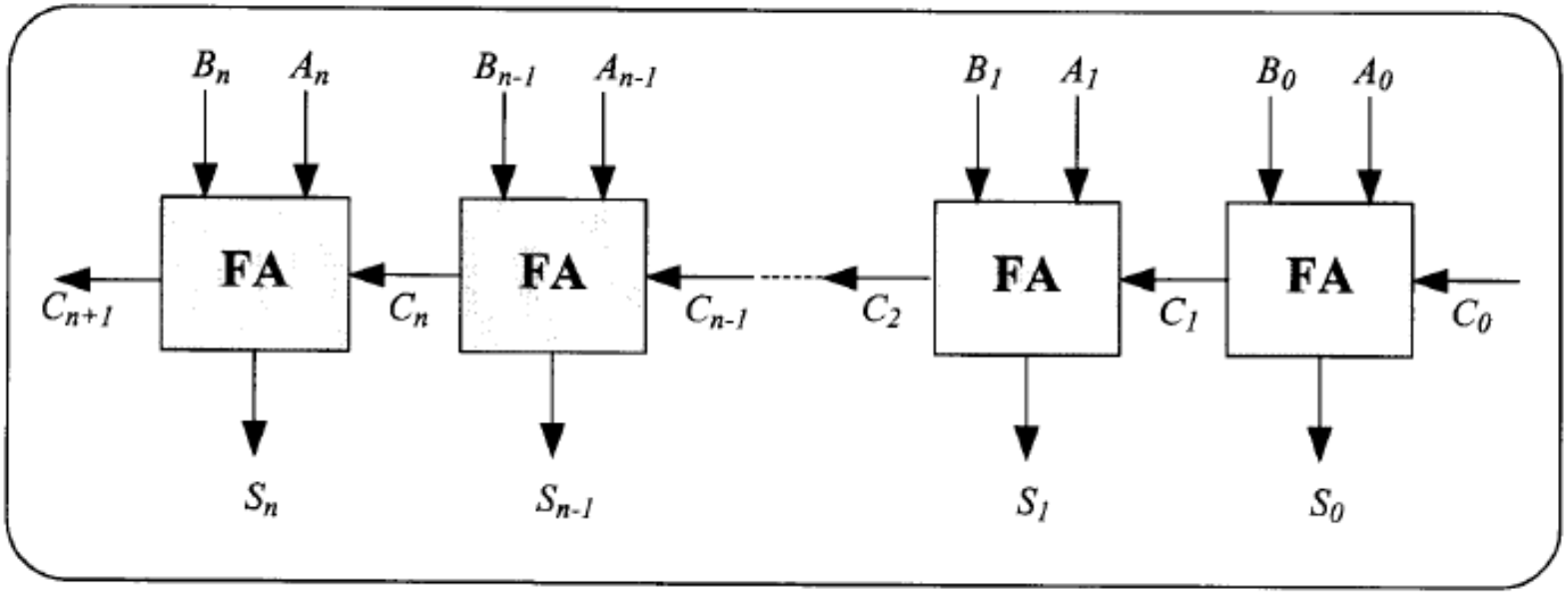
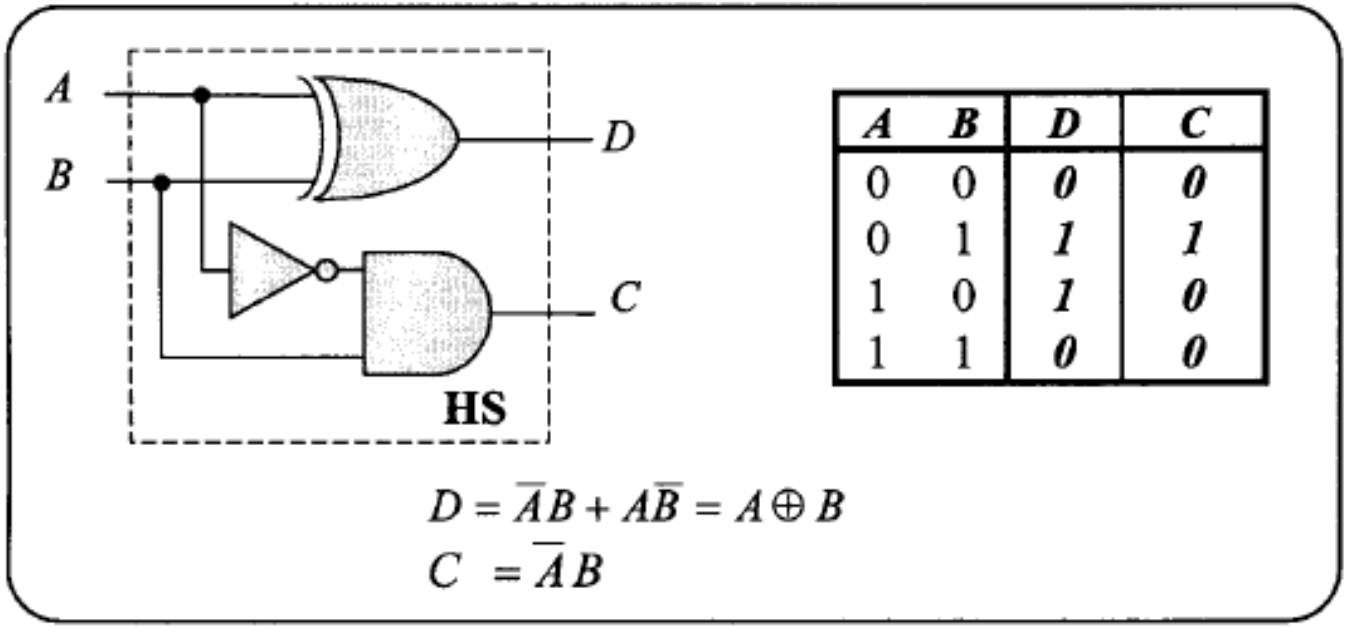


Figura 5.7. Extensión del sumador completo de la *figura 5.6.c* a un sumador paralelo para palabras de 4 bits.

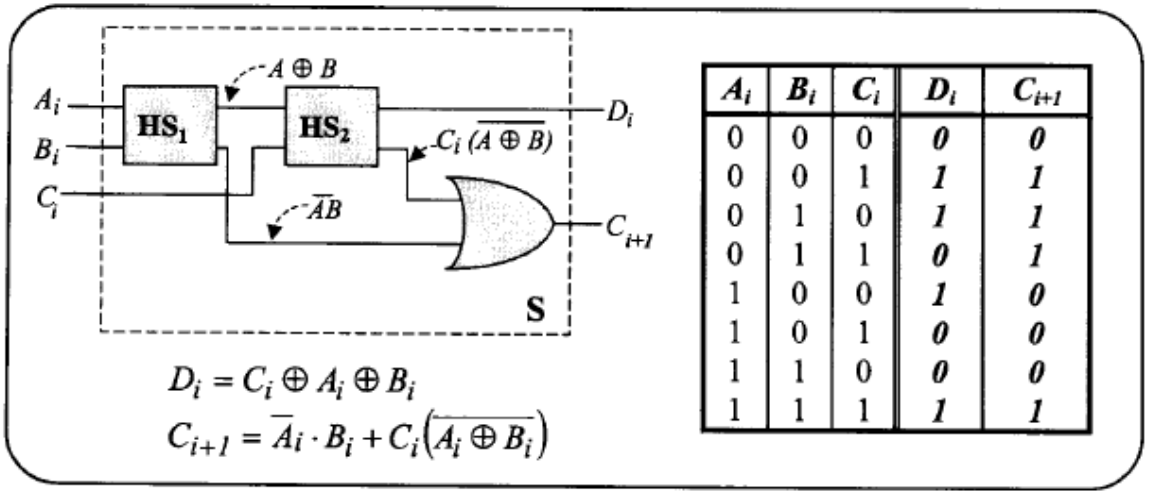
5.2.3 Semirrestadores



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

Figura 5.8. Semirrestador

5.2.4 Restadores Completo



A_i	B_i	C_i	D_i	C_{i+1}
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*

Figura 5.9. Restador.

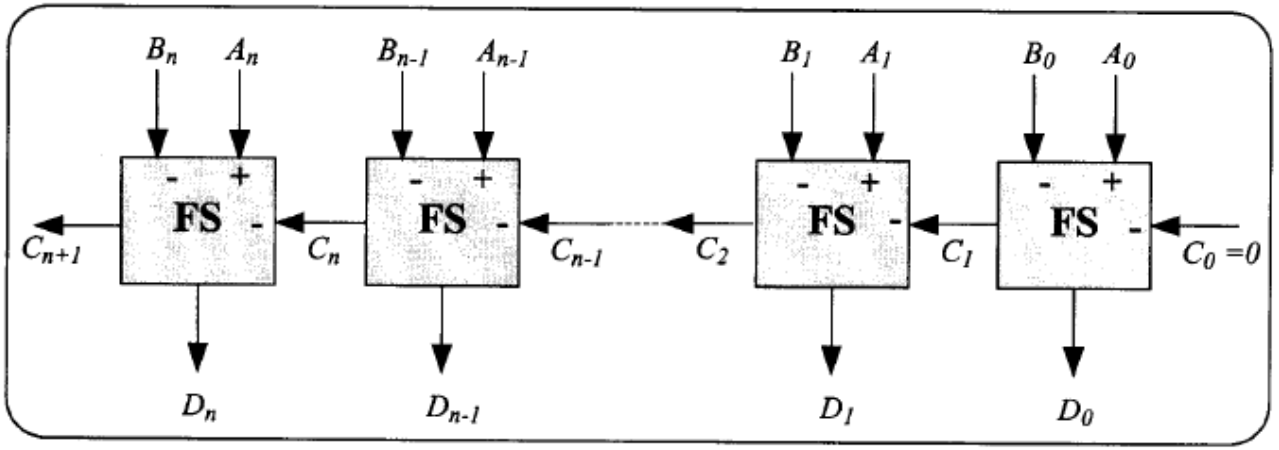


Figura 5.10. Restador paralelo

5.2.5 Sumador Serie

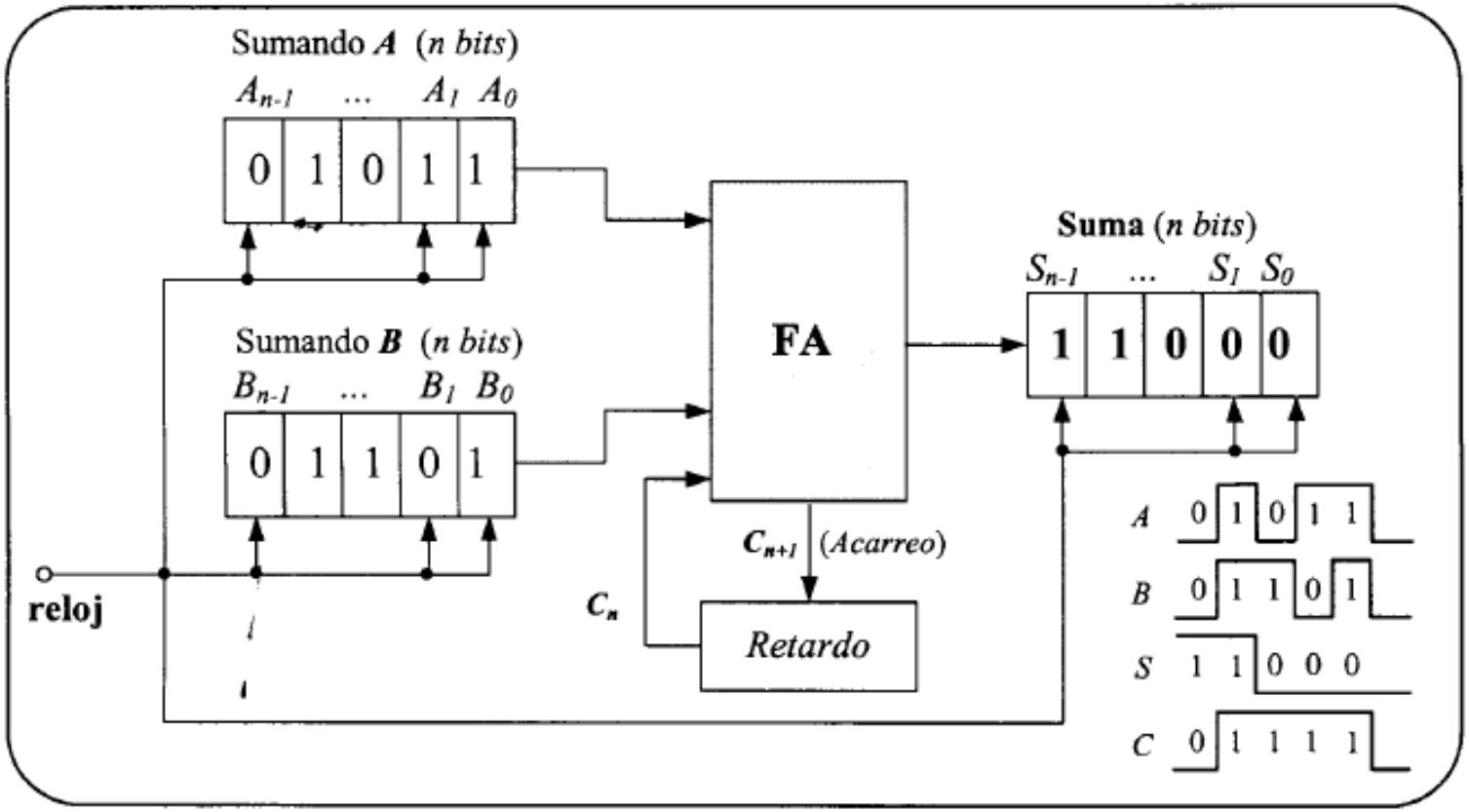
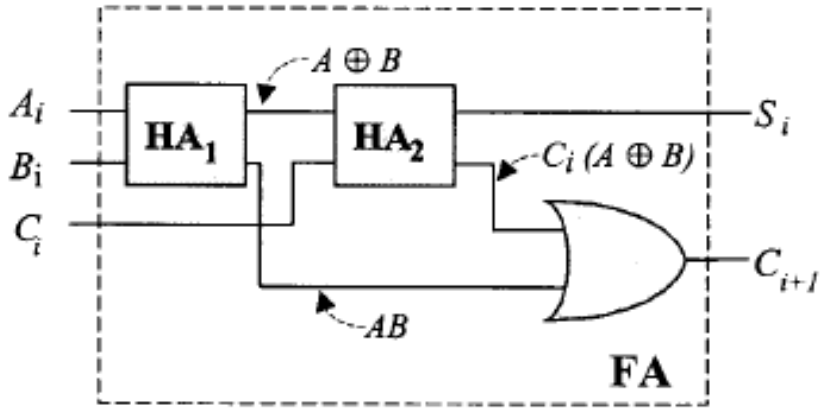


Figura. 5.11. Esquema cualitativo de un sumador serie para palabras de *n* bits.

5.2.6 Sumador paralelo con acarreo adelantado



$$S_i = C_i \oplus A_i \oplus B_i$$

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

$$P_i = A_i \oplus B_i \quad y \quad G_i = A_i \cdot B_i \qquad [5.9]$$

Las salidas del sumador se pueden expresar en términos de P y G como

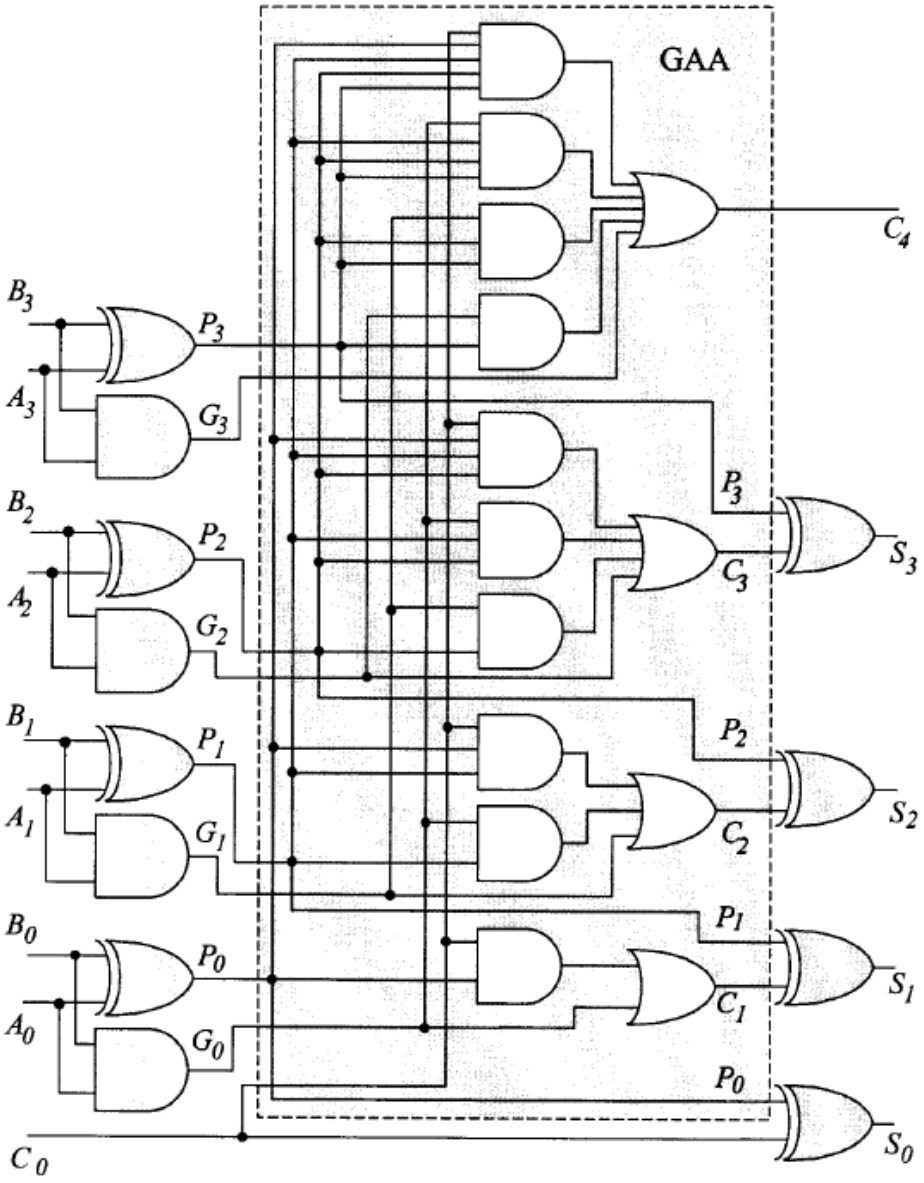
$$S_i = P_i \oplus C_i \quad y \quad C_i = G_i + P_i C_i \qquad [5.10]$$

La señal interna, G_i, se denomina "*generación de acarreo*" y produce un estado de alta cuando $A_i = B_i$ ="1". La señal P_i se denomina de "*propagación del acarreo*" y es la que se compone con el acarreo de entrada (C_i) para producir el de salida (C_{i+1}).

$$S_i = P_i \oplus C_i \quad y \quad C_i = G_i + P_i C_i \qquad [5.10]$$

Escribamos ahora las funciones para la salida de arrastre de cada etapa en función de las señales P y G.

$$\begin{aligned}
C_1 &= G_0 + P_0 C_0 \\
C_2 &= G_1 + P_1 C_1 = G_1 + P_1(G_0 + P_0 C_0) = G_1 + P_1 G_0 + P_1 P_0 C_0 \qquad [5.11] \\
C_3 &= G_2 + P_2 C_2 = G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_0 \\
C_4 &= G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0 + P_3 P_2 P_1 P_0 C_0
\end{aligned}$$



$$P_i = a_i \oplus b_i$$
$$G_i = a_i \bullet b_i$$

$$S_i = P_i \oplus C_i$$
$$C_i = G_{i-1} + P_{i-1}C_{i-1}$$

$$C_1 = G_0 + P_0C_0$$
$$C_2 = G_1 + P_1C_1 = G_1 + P_1(G_0 + P_0C_0) = G_1 + P_1G_0 + P_1P_0C_0$$

Figura 5.12. Sumador paralelo de 4 bits con lógica de acarreo adelantado.

Obsérvese que los cuatro bits de salida, S_0 a S_3, sufren el mismo retardo de propagación, el correspondiente a los cuatro niveles de puertas lógicas usados:

a) Los semisumadores que generan las señales P y G.

b) Las puertas AND que generan los términos mínimos del acarreo adelantado.

c) Las puertas OR.

d) Las XOR de salida

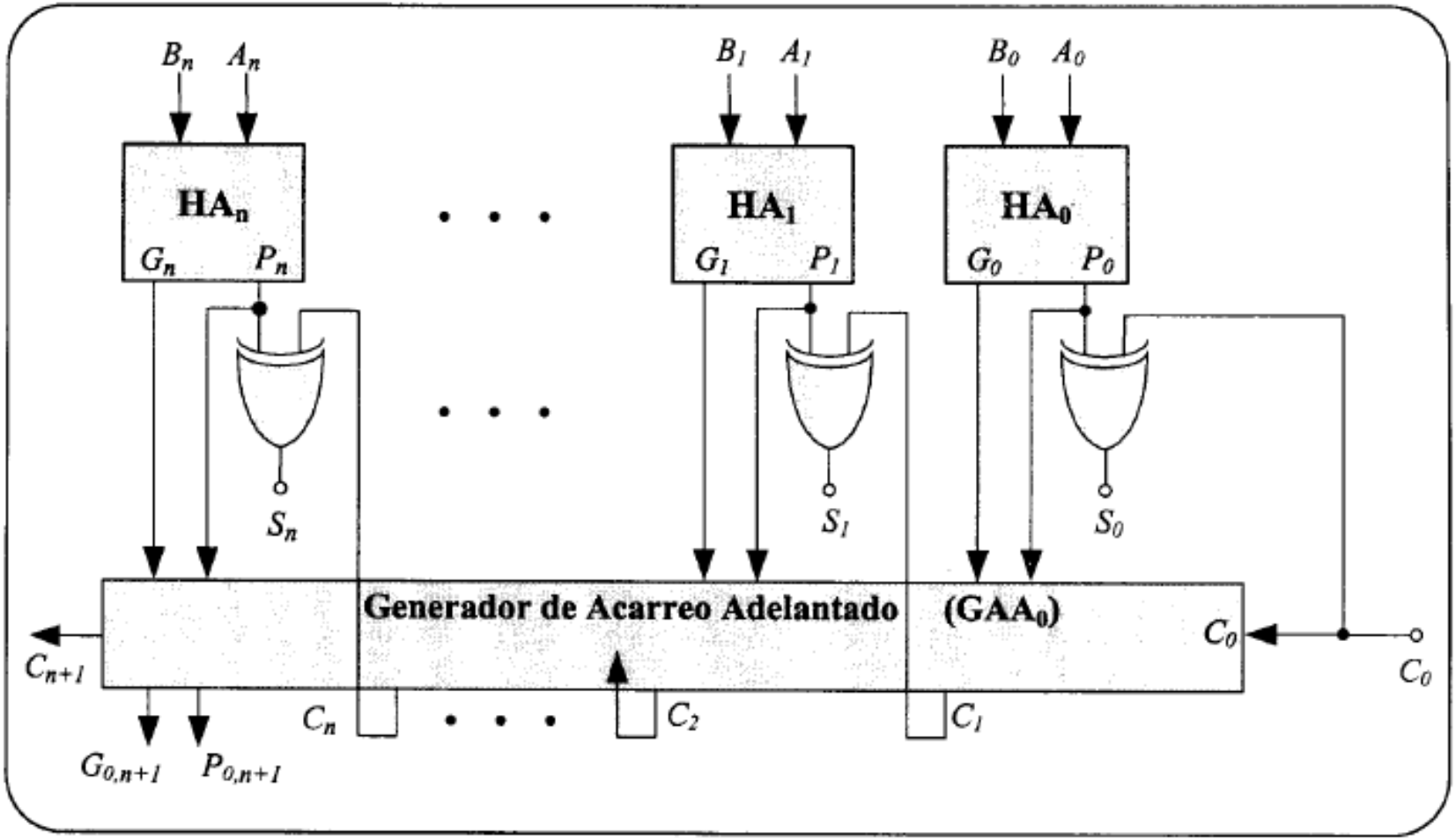


Figura 5.13. Versión compacta del circuito de sumador paralelo con acarreo adelantado de la *figura 5.12*.

5.3 Sumadores en complemento a 1: gestión del problema del rebose

I. Para obtener los números negativos en C-1 hay que cambiar los "0" por "1" y los "1" por "0".

II. Para sumar números positivos y negativos en C-1, se suma en binario y se desprecia el bit de acarreo cuando éste es cero. Cuando es uno se desprecia también, pero en este caso se le suma un uno al resultado previo.

En este caso, como queremos encontrar un circuito que sume palabras de sólo dos bits, (A_1, A_0) y (B_1, B_0), sólo tenemos 2 bits para representar los números positivos y negativos y el bit más significativo es el bit de signo. Por consiguiente, realmente sólo nos queda 1 bit para representar las magnitudes. Así los números que podemos representar en C-1 (con 2 bits) son:

C-1	00	01	10	11
Decimal	+0	+1	-1	-0

Obsérvese que el rebose sólo se puede producir cuando *los dos sumandos son del mismo signo* (ambos positivos o ambos negativos) pues cuando son de signo distinto el resultado siempre será menor que uno de los operandos y podrá representarse usando el mismo número de bits que hemos usado para los operandos. Para saber si se produce o no el rebose hay que comprobar el *signo de la suma*. Si es correcto no hay rebose. Si es incorrecto, si.

- *Sin problemas ni necesidad de sumar "1" al resultado:*

```
  00     (+0)          01     (+1)
+ 01     (+1)        + 10     (-1)
-----                -----
  01 →   (+1)          11 →   (-0)
```

- *Con necesidad de sumar "1" al resultado:*

```
   10     (-1)
+  11     (-0)
-------
 1 01 →   (+1)
```

Obsérvese que si nos quedamos con los dos últimos bits (01), el resultado sería erróneo ya que 01 corresponde a +1 en C-1. Por consiguiente hay que sumarle "1" al resultado Así,

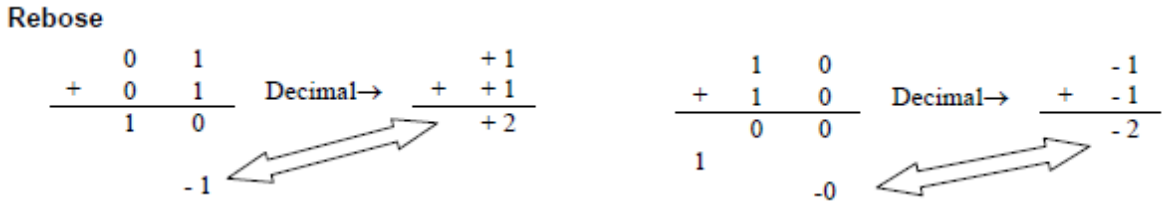
```
   1 0      (-1)
+  1 1      (-0)
-------
   1 0 1
   └+ 1
-------
   1 0  →   (-1)
```

- *Con problemas de rebose:*

```
   01       (+1)
+  01       (+1)
-------
  0|10  →   (-1)
```

Aquí hay error de rebose porque (+1)+(+1) es +2 y nos sale -1 (lógicamente en C-1). Esto es debido a que nos hace falta un bit más. Entonces, con tres bits, (010, +2 en decimal) sería correcta la suma. El bit de signo sería el 0 (+) y el de magnitud los otros dos, 10 (+2).

Así pues, al diseñar el circuito sumador en C-1 tenemos que detectar las configuraciones que generan este error de rebose. Para ello vamos a construir la tabla de verdad completa (*figura 5.14*).



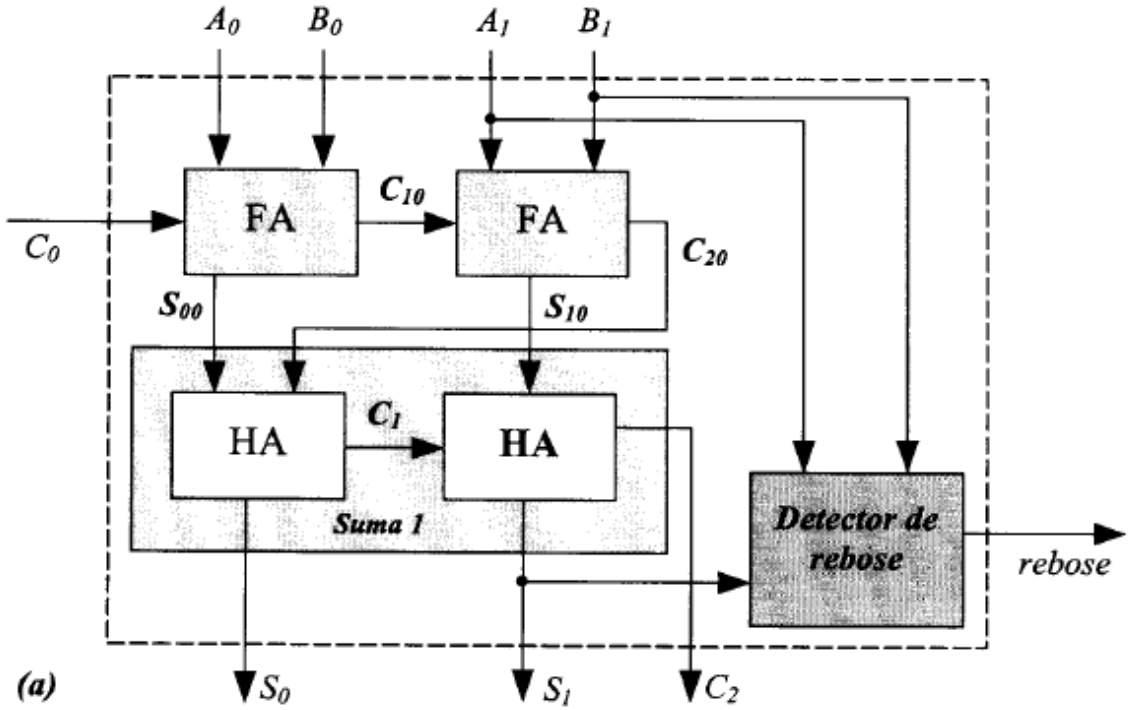
La ecuación del rebose es:

$$rebose = \overline{A}_1\ \overline{B}_1\ S_1 + A_1\ B_1\ \overline{S}_1 \qquad [5.12]$$

nº decimal	B_1	B_0	nº decimal	A_1	A_0	C_2	S_1	C_1	S_0	nº decimal
(+0)	0	0	(+0)	0	0	0	0	0	0	(+0)
	0	0	(+1)	0	1	0	0	0	1	(+1)
	0	0	(-1)	1	0	0	1	0	0	(-1)
	0	0	(-0)	1	1	0	1	0	1	(-0)
(+1)	0	1	(+0)	0	0	0	0	0	1	(+1)
	0	1	(+1)	0	1	0	1	1	0	(-1) ***rebose***
	0	1	(-1)	1	0	0	1	0	1	(-0)
	0	1	(-0)	1	1	1	0	1	0	
						+			1	
							0		1	(+1)
(-1)	1	0	(+0)	0	0	0	1	0	0	(-1)
	1	0	(+1)	0	1	0	1	0	1	(-0)
	1	0	(-1)	1	0	1	0	0	0	
						+			1	
							0		1	(+1) ***rebose***
	1	0	(-0)	1	1	1	0	0	1	
						+			1	
							1		0	(-1)
(-0)	1	1	(+0)	0	0	0	1	0	1	(-0)
	1	1	(+1)	0	1	1	0	1	0	
						+			1	
							0		1	(1)
	1	1	(-1)	1	0	1	0	0	1	
						+		1	1	
							1		0	(-1)
	1	1	(-0)	1	1	1	1	1	0	
						+			1	
							1		1	(-0)

$$rebose = S_1\overline{A}_1\overline{B}_1 + \overline{S}_1 A_1 B_1$$

Figura 5.14. Descripción en extenso de las distintas situaciones posibles en la suma por *C-1* de dos palabras de 2 bits.



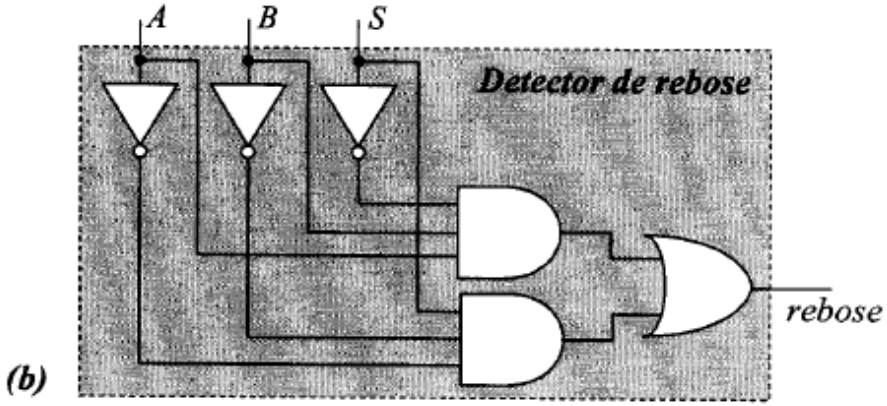


Figura 5.15. Suma por *C-1*. ***(a)*** Esquema del circuito a partir de sumadores y semisumadores. ***(b)*** Circuito detector del rebose.

5.4 Comparadores

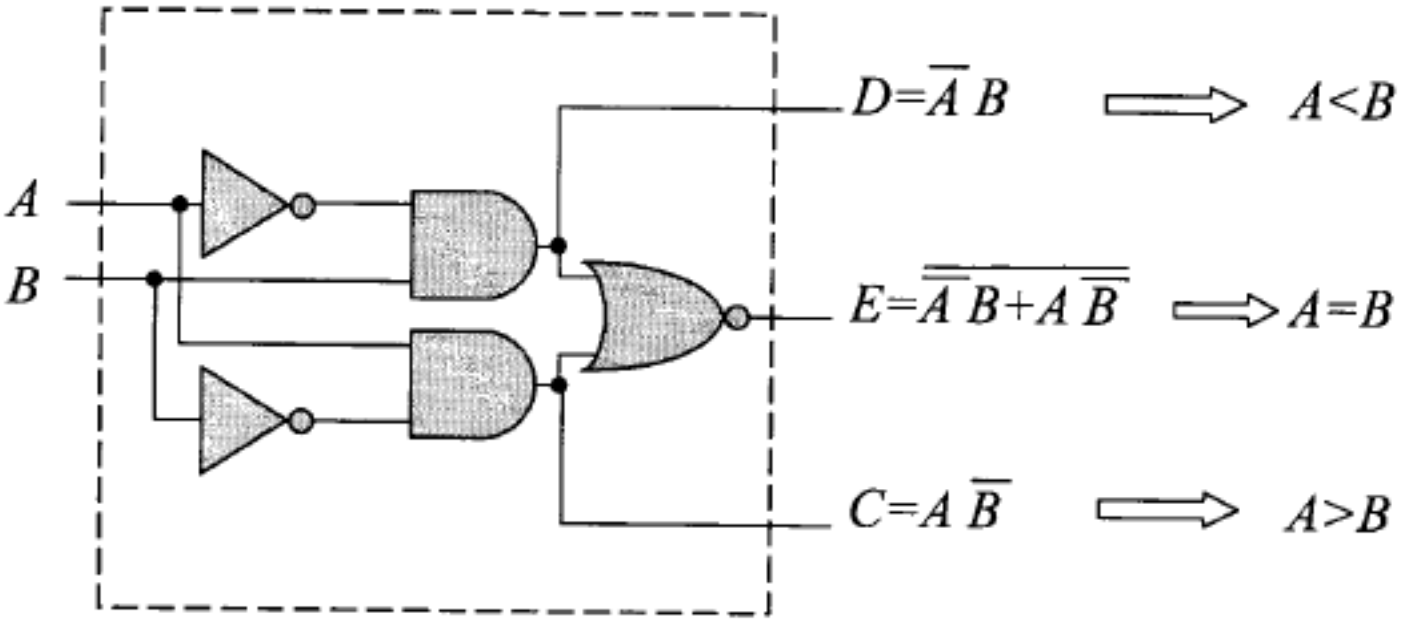


Figura 5.16. Circuito comparador de dos "palabras" de un bit.

Comparador de 4 bits

Si $A = B$ tendrá que ser: $A_3=B_3$, $A_2=B_2$, $A_1=B_1$ y $A_0=B_0$ y, por consiguiente, la condición de igualdad será:

$$E = E_3 \cdot E_2 \cdot E_1 \cdot E_0 = 1 \qquad [5.14]$$

Para que se cumpla la condición A>B pueden ocurrir dos de las siguientes situaciones:

$$A_3 > B_3 \Rightarrow A_3 \overline{B}_3$$

$$\acute{o} \quad A_3 = B_3 \;\; y \;\; A_2 > B_2 \Rightarrow E_3 A_2 \overline{B}_2$$

$$\acute{o} \quad A_3 = B_3 \;\; y \;\; A_2 = B_2 \;\; y \;\; A_1 > B_1 \Rightarrow E_3 E_2 A_1 \overline{B}_1$$

$$\acute{o} \quad A_3 = B_3 \;\; y \;\; A_2 = B_2 \;\; y \; A_1 = B_1 \;\; y \;\; A_0 > B_0 \Rightarrow E_3 E_2 E_1 A_0 \overline{B}_0 \qquad [5.15]$$

Finalmente, por exclusión, si $\overline{E} = 1 \;\; y \;\; C(A > B) = 0$, se tiene que cumplir que: $A < B$.

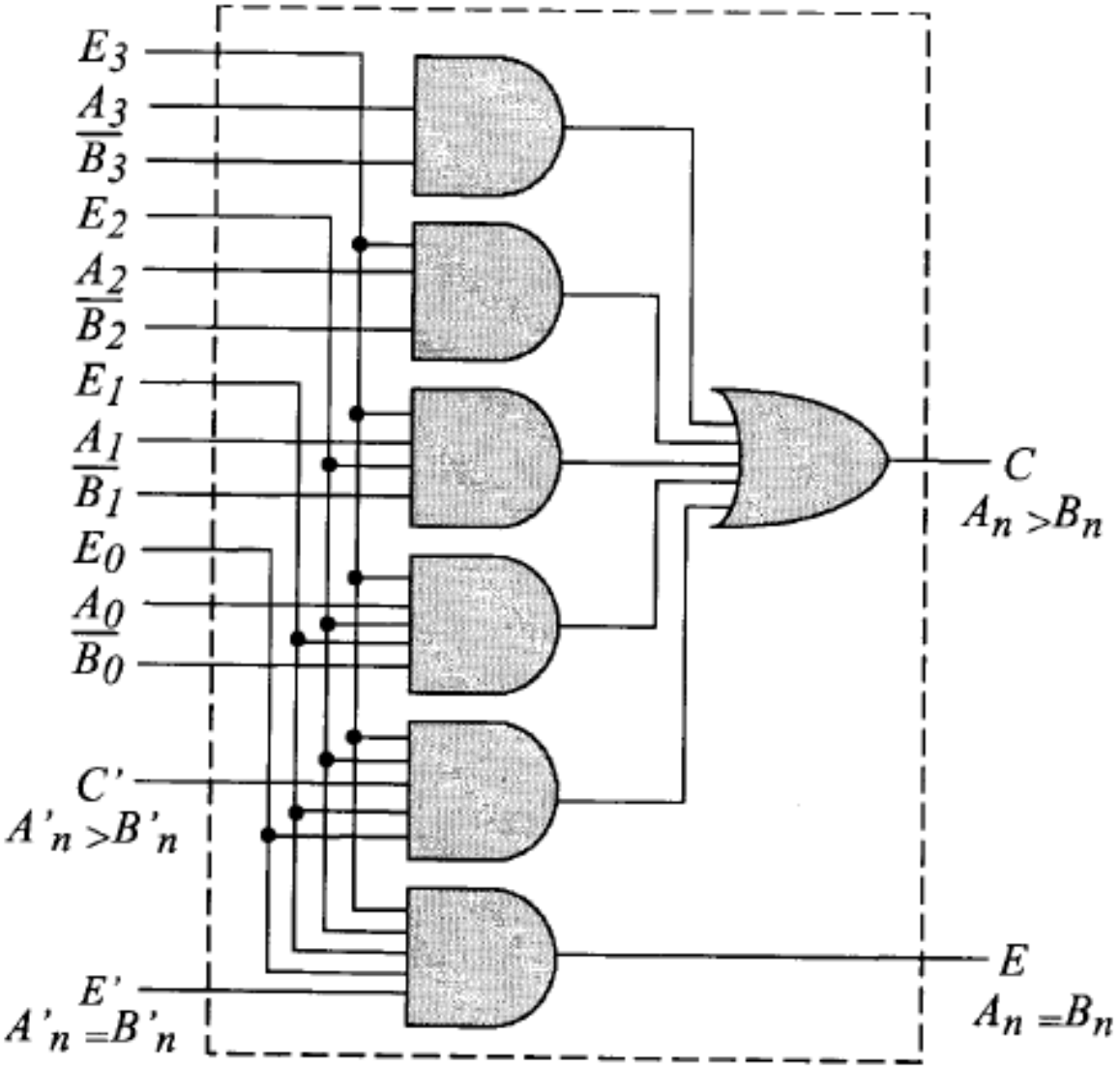


Figura 5.17. Extensión del circuito comparador a palabras de cuatro bits.

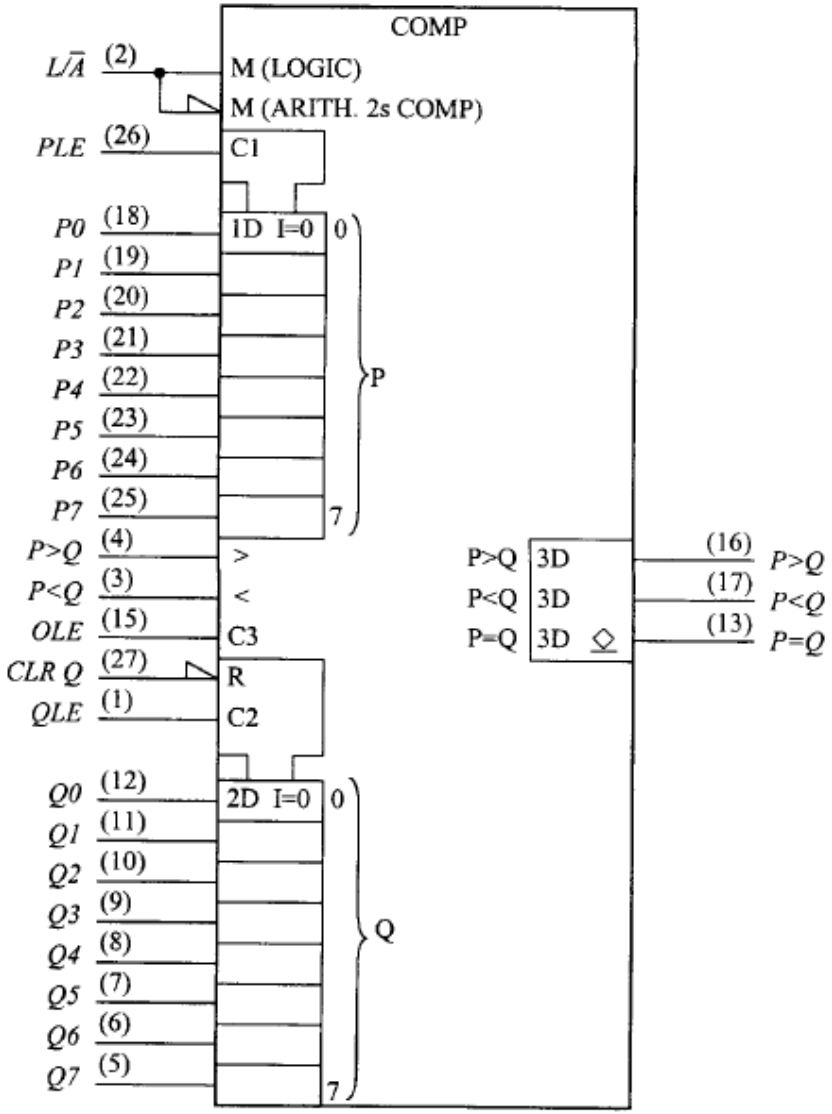


Figura 5.18. Circuito comparador de 8 bits, SN74AS866A. Símbolo lógico de acuerdo con las normas ANSI/IEEE.

TABLA DE FUNCIÓN DEL COMPARADOR							
Comparación	L/$\overline{A}$	***Datos de Entrada*** *P0-P7, Q0-Q7*	***ENTRADAS***		***SALIDAS***		
			P>Q	*P<Q*	***P>Q***	***P<Q***	***P=Q***
Lógica	H	P>Q	x	x	***H***	***L***	***L***
Lógica	H	P<Q	x	x	***L***	***H***	***L***
Lógica	H	P=Q	L	L	***L***	***L***	***H***
Lógica	H	P=Q	L	H	***L***	***H***	***L***
Lógica	H	P=Q	H	L	***H***	***L***	***L***
Lógica	H	P=Q	H	H	***H***	***H***	***L***
Aritmética	L	P AG Q	x	x	***H***	***L***	***L***
Aritmética	L	Q AG P	x	x	***L***	***H***	***L***
Aritmética	L	P=Q	L	L	***L***	***L***	***H***
Aritmética	L	P=Q	L	H	***L***	***H***	***L***
Aritmética	L	P=Q	H	L	***H***	***L***	***L***
Aritmética	L	P=Q	H	H	***H***	***H***	***L***

Figura 5.19. Tabla de selección de función en el comparador aritmético-lógico 74866. Con $L/\overline{A}$ se selecciona el tipo de comparación (lógica o aritmética).

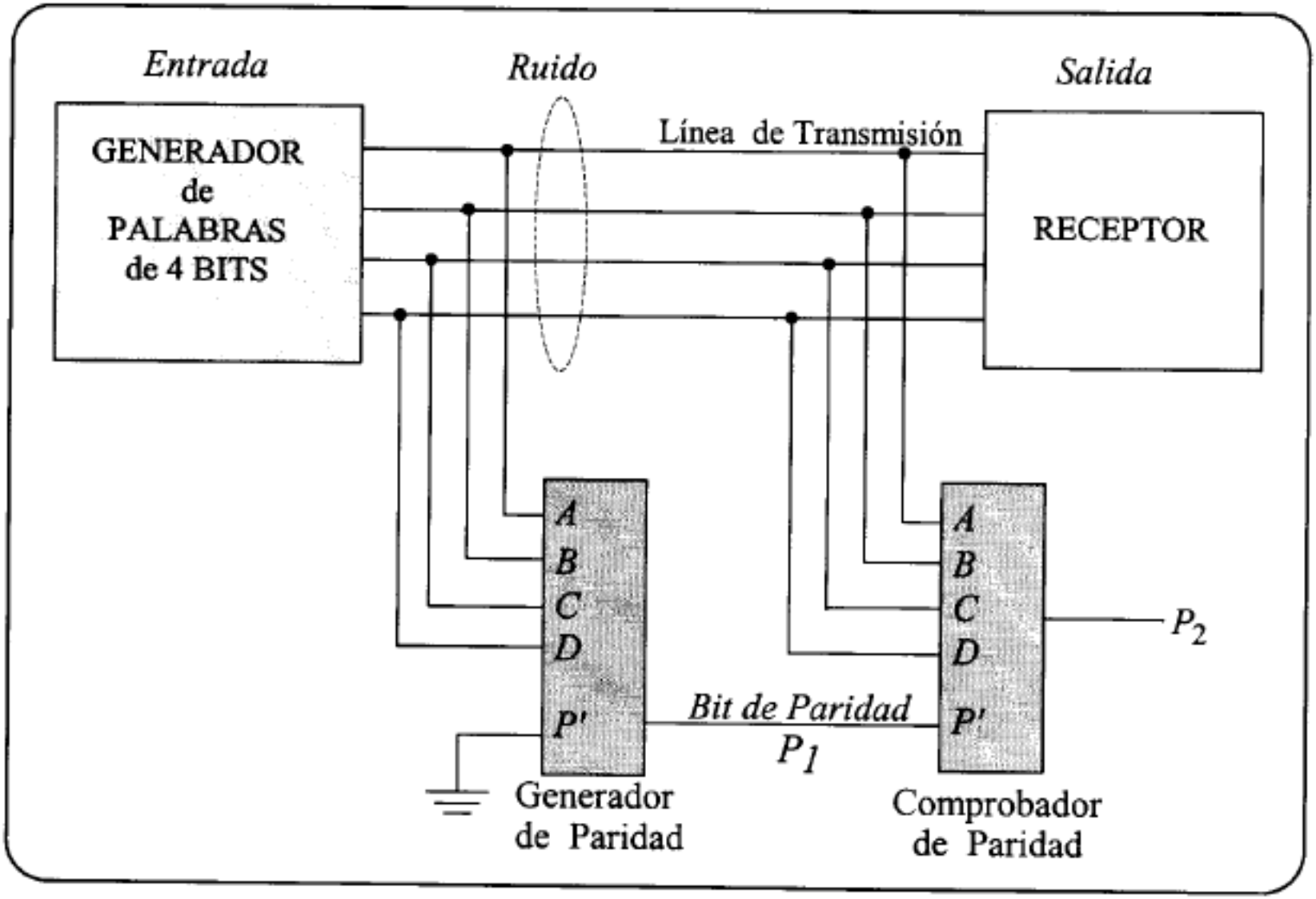


Figura 5.20. Circuitos detectores/generadores de paridad. Ilustración de su uso en transmisión.

Detector de paridad

a	b	Paridad par
0	0	0
0	1	1
1	0	1
1	1	0

$$paridad\_par = a\bar{b} + \bar{a}b = a \oplus b$$

Los generadores de paridad PAR son aquellos circuitos que generan un “0” cuando el número de “1” a la entrada es par y un “1” cuando es impar.

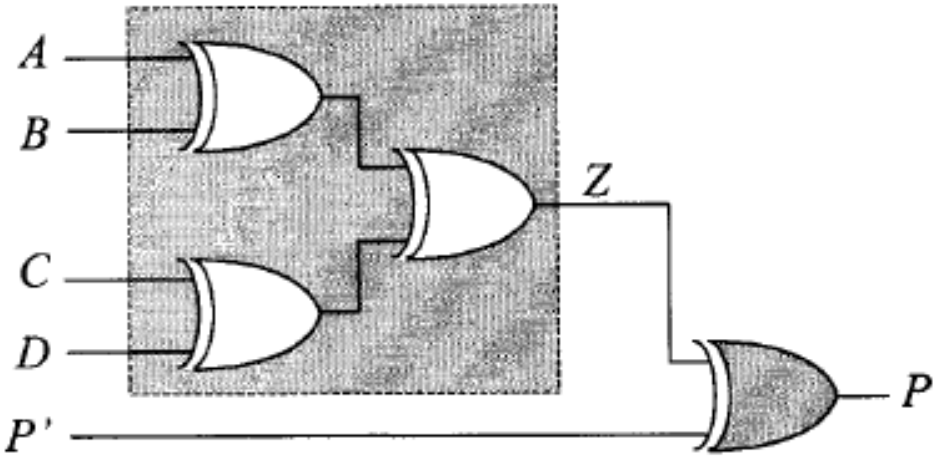
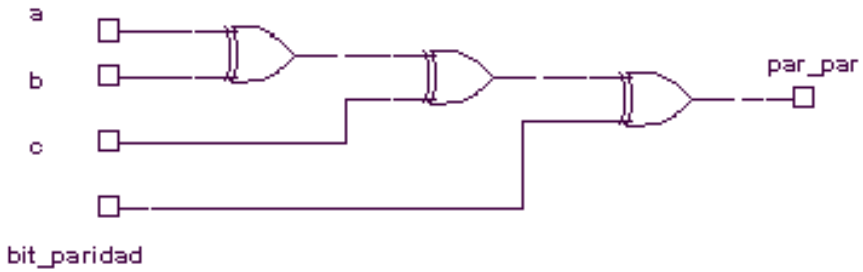
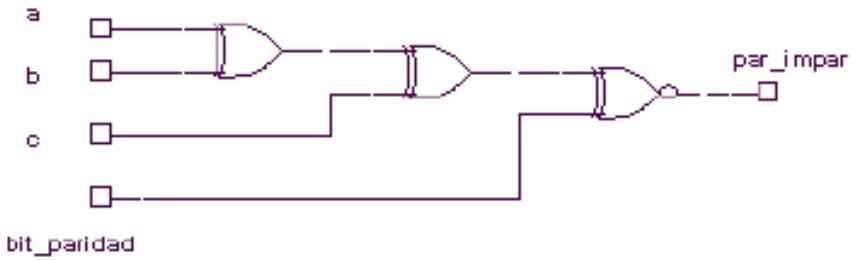


Figura 5.21. Síntesis de un detector de paridad para 4 bits.

El caso del detector es similar al del generador, solo que el bit de parida forma parte de la entrada en la recepción, convirtiéndose de esta manera en otro bit de datos y la salida que antes era el bit generado es ahora el bit indicador de error.



Para el caso del detector de paridad impar lo único que hay que hacer es sustituir la última puerta por una NOR-EXCLUSIVE.



5.5 Unidades Aritmético-Lógicas (ALUS)

- Tabla de verdad de la unidad aritmético-lógica comercial SN74181.

SELECCIÓN				DATO ACTIVO EN ALTA		
				M=H	M=L Operaciones Aritméticas	
S_3	S_2	S_1	S_0	Funciones Lógicas	$\overline{c_n}$=H (sin acarreo)	$\overline{c_n}$=L (con acarreo)
L	L	L	L	$F = \overline{A}$	$F = A$	$F = A$ PLUS 1
L	L	L	H	$F = \overline{A + B}$	$F = A + B$	$F = (A + B)$ PLUS 1
L	L	H	L	$F = \overline{A}B$	$F = A + \overline{B}$	$F = (A + \overline{B})$ PLUS 1
L	L	H	H	$F = 0$	F = MINUS 1 (comp. a 2)	$F = Cero$
L	H	L	L	$F = \overline{AB}$	$F = A$ PLUS $A\overline{B}$	$F = A$ PLUS $A\overline{B}$ PLUS 1
L	H	L	H	$F = \overline{B}$	$F = (A + B)$ PLUS $A\overline{B}$	$F = (A + B)$ PLUS $A\overline{B}$ PLUS 1
L	H	H	L	$F = A \oplus B$	$F = A$ MINUS B MINUS 1	$F = A$ MINUS B
L	H	H	H	$F = A\overline{B}$	$F = A\overline{B}$ MINUS 1	$F = A\overline{B}$
H	L	L	L	$F = \overline{A} + B$	$F = A$ PLUS AB	$F = A$ PLUS AB PLUS 1
H	L	L	H	$F = \overline{A \oplus B}$	$F = A$ PLUS B	$F = A$ PLUS B PLUS 1
H	L	H	L	$F = B$	$F = (A + \overline{B})$ PLUS AB	$F = (A + \overline{B})$ PLUS AB PLUS 1
H	L	H	H	$F = AB$	$F = AB$ MINUS 1	$F = AB$
H	H	L	L	$F = 1$	$F = A$ PLUS A	$F = A$ PLUS A PLUS 1
H	H	L	H	$F = A + \overline{B}$	$F = (A + B)$ PLUS A	$F = (A + B)$ PLUS A PLUS 1
H	H	H	L	$F = A + B$	$F = (A + \overline{B})$ PLUS A	$F = (A + \overline{B})$ PLUS A PLUS 1
H	H	H	H	$F = A$	$F = A$ MINUS 1	$F = A$

Entradas de control:

- M: distingue entre operación aritmética o lógica.
- $S_3S_2S_1S_0$: selección de operación.

- **Significado de símbolos:**
- + significa OR lógico.
- ⊕ significa OR exclusivo lógico.
- El producto lógico se sobreentiende.
- **PLUS** significa suma aritmética.
- **MINUS** significa resta aritmética mediante suma en complemento a 1.

S3	S2	S1	S0	M	c_n
L	L	L	H	L	H

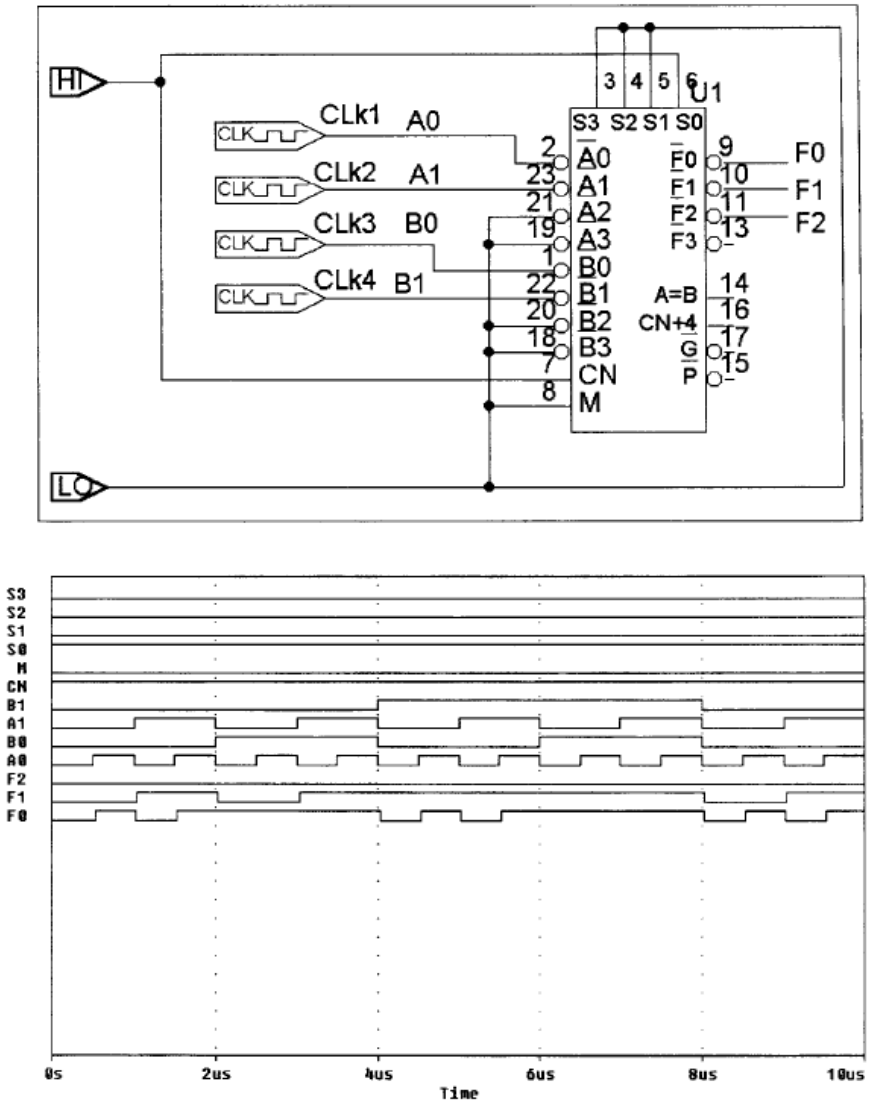


Figura 5.24. Esquema de conexión en el simulador de la ALU SN74181 y resultado correspondiente en forma de cronograma que muestra la evolución temporal de las entradas y salidas de la ALU manteniendo constantes los valores de las señales de control que seleccionan la operación.

B1	*A1*	*B0*	*A0*	*F2*	*F1*	*F0*
0	0	0	0	***0***	***0***	***0***
0	0	0	1	***0***	***0***	***1***
0	1	0	0	***0***	***1***	***0***
0	1	0	1	***0***	***1***	***1***
0	0	1	0	***0***	***0***	***1***
0	0	1	1	***0***	***0***	***1***
0	1	1	0	***0***	***1***	***1***
0	1	1	1	***0***	***1***	***1***
1	0	0	0	***0***	***1***	***0***
1	0	0	1	***0***	***1***	***1***
1	1	0	0	***0***	***1***	***0***
1	1	0	1	***0***	***1***	***1***
1	0	1	0	***0***	***1***	***1***
1	0	1	1	***0***	***1***	***1***
1	1	1	0	***0***	***1***	***1***
1	1	1	1	***0***	***1***	***1***

B1 A1 / *B0 A0*	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	1	1	1	1
10	1	1	1	1

$\overline{F0} = \overline{A0}\,\overline{B0}$ $F0 = A0 + B0$

B1 A1 / *B0 A0*	00	01	11	10
00	0	1	1	1
01	0	1	1	1
11	0	1	1	1
10	0	1	1	1

$F1 = A1 + B1$

Figura 5.25. Obtención de las expresiones de *F0, F1* y *F2*. Obsérvese que efectivamente, la función resultante es *A* OR *B*.