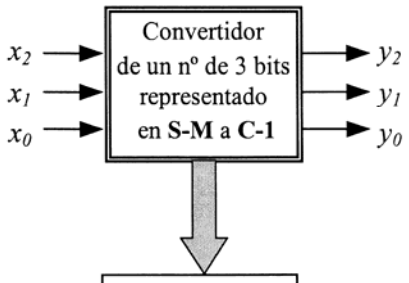
FUNCIONES ARITMÉTICO-LÓGICAS

Representación de números en binario

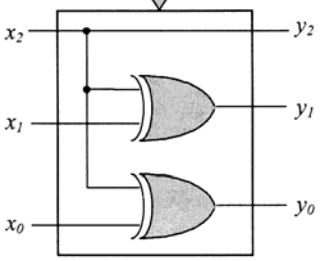
- Binario puro
- Magnitud + signo
- Complemento a 1
- Complemento a 2

Codificador de magnitud + signo a Complemento a 1

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

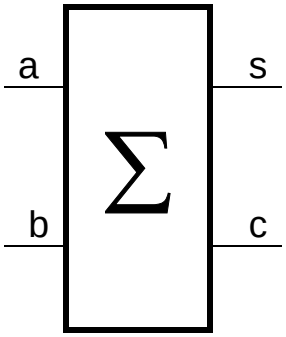


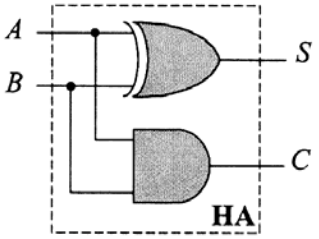
Complemento a 1 (C-1) $y_2\ y_1\ y_0$
0 1 1
0 1 0
0 0 1
0 0 0
1 1 1
1 1 0
1 0 1
1 0 0



2. Sumadores y restadores

Semisumador:

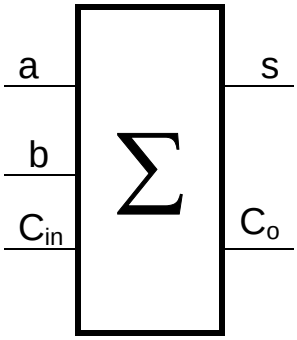




A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

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

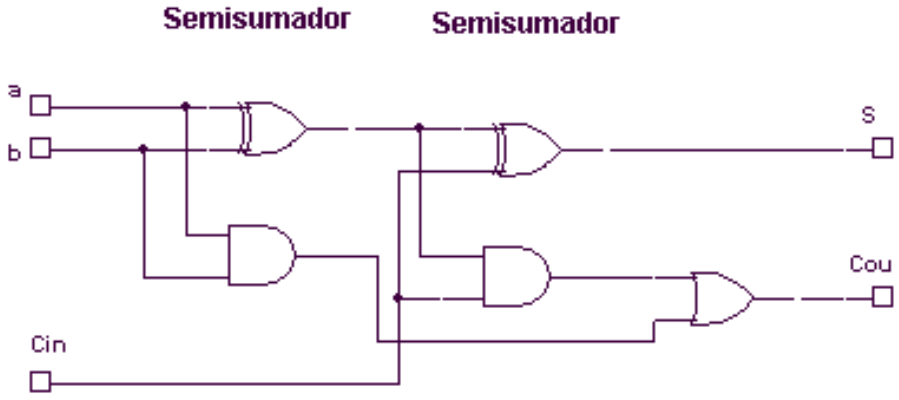
Sumador:



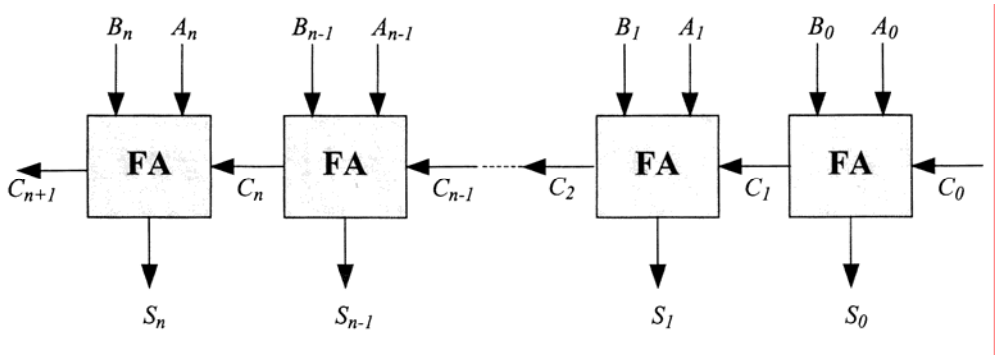
a	b	C_{in}	s	C_o
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$$S = \bar{a}\bar{b}c + \bar{a}b\bar{c} + a\bar{b}\bar{c} + abc = c(\bar{a}\bar{b} + ab) + \bar{c}(\bar{a}b + a\bar{b}) = c\overline{(a \oplus b)} + \bar{c}(a \oplus b) = c\bar{m} + \bar{c}m) = c \oplus m = c \oplus (a \oplus b)$$

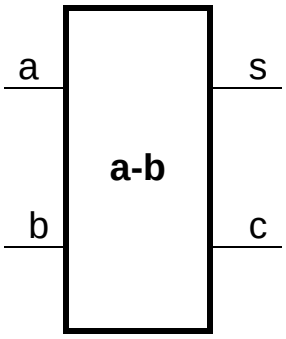
$$C_0 = \bar{a}bc + a\bar{b}c + ab\bar{c} + abc = ab + c(a \oplus b)$$

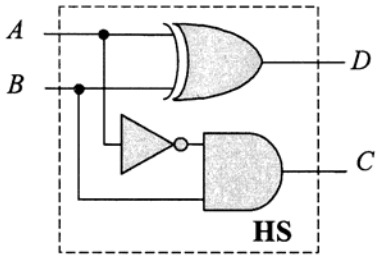


Sumador paralelo:



Semirrestador:

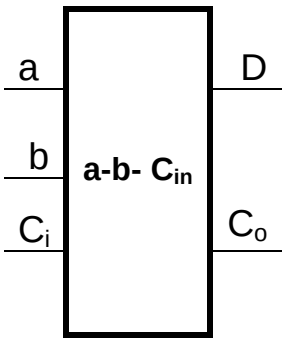




A	B	D	C
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

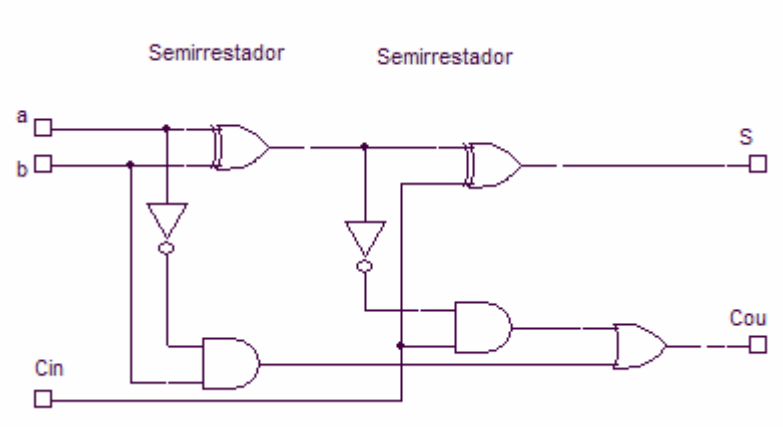
$$D = \overline{A}B + A\overline{B} = A \oplus B$$
$$C = \overline{A}B$$

Restador:

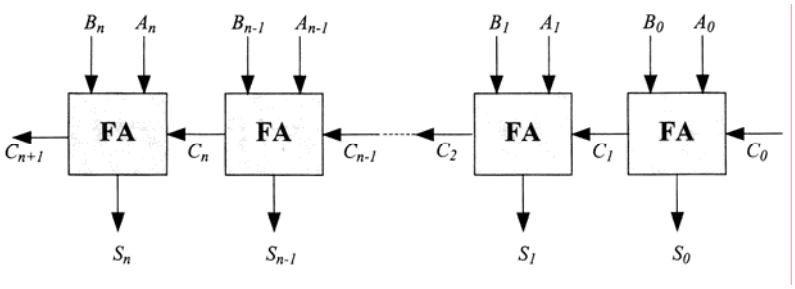


a	b	C_i	D	C_{i+1}
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

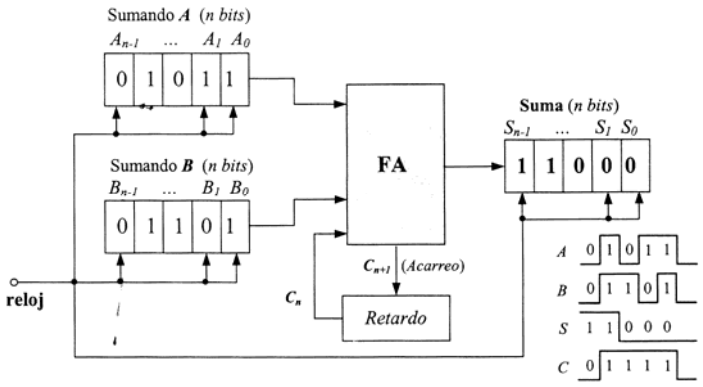
$$S = \overline{a}\overline{b}c + \overline{a}b\overline{c} + a\overline{b}\overline{c} + abc = c(\overline{a}\overline{b} + ab) + \overline{c}(\overline{a}b + a\overline{b}) = c\overline{(a \oplus b)} + \overline{c}(a \oplus b) = c\overline{m} + \overline{c}m) = c \oplus m = c \oplus (a \oplus b)$$
$$C_{i+1} = \overline{a}\overline{b}c + \overline{a}b\overline{c} + \overline{a}bc + abc = \overline{a}b + c_i\overline{(a \oplus b)}$$



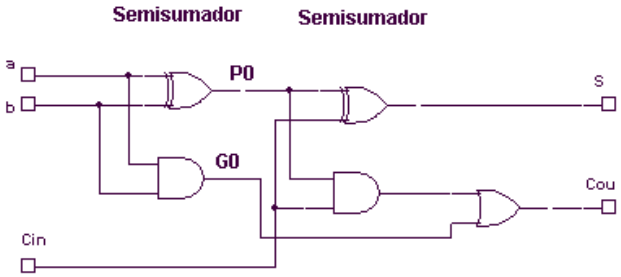
Restador paralelo:



Sumador serie:



Sumador paralelo con acarreo adelantado:

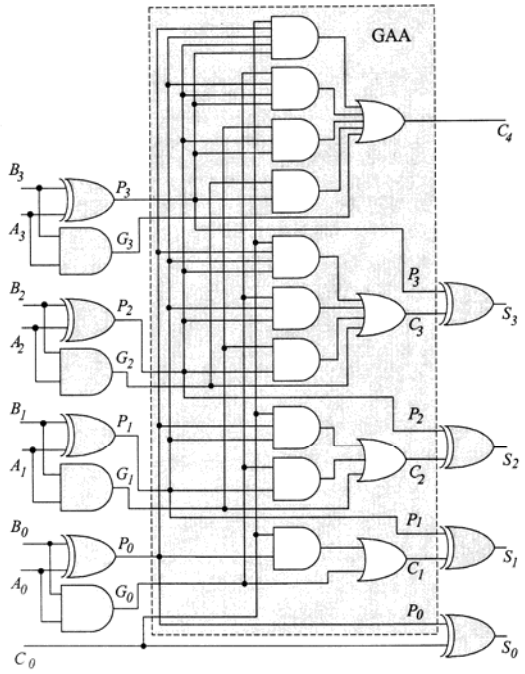


$$P_i = a_i \oplus b_i$$
$$G_i = a_i \bullet b_i$$

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

$$C_1 = G_0 + P_0C_0$$
$$C_2 = G_1 + P_1C_1 = G_1 + P_1(G_0 + P_0C_0) = G_1 + P_1G_0 + P_1P_0C_0$$
$$C_3 = G_2 + P_2C_2 = G_2 + P_2(G_1 + P_1G_0 + P_1P_0C_0) = G_2 + P_2G_1 + P_2P_1G_0 + P_2P_1P_0C_0$$
$$C_4 = G_3 + P_3C_3 = G_3 + P_3(G_2 + P_2G_1 + P_2P_1G_0 + P_2P_1P_0C_0) = G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0 + P_3P_2P_1P_0C_0$$

Se gestiona el acarreo desde el principio, suponiendo para cada salida 4 etapas de puertas lógicas, independientemente del orden de la salida S_n



3. Sumador en complemento a 1:

Cuando se opera en aritmética en "complemento a 1" el límite de representación está limitado al número de bits. De esta manera cuando el resultado de una suma o resta es superior al máximo de representación el resultado de la operación es errónea, dicho error queda recogido en el bit denominado de ***rebosamiento.***

Funciones que debe realizar el circuito

1. Sumar en binario puro sin signo ni magnitud.
2. Dar por válido el resultado si el acarreo=0 y no hay rebose.
3. Si acarreo=1 y no hay rebose ⇒ Sumar "1" al resultado.
4. Si hay rebose ⇒ Dar error

Situaciones:

No hay problemas en los casos

$$\begin{array}{rcc} & 0 & 0 \\ + & 0 & 1 \\ \hline & 0 & 1 \end{array} \text{ Decimal}\rightarrow \begin{array}{rc} & 0 \\ + & 1 \\ \hline & 1 \end{array} \qquad \begin{array}{rcc} & 0 & 1 \\ + & 1 & 0 \\ \hline & 1 & 1 \end{array} \text{ Decimal}\rightarrow \begin{array}{rc} & 1 \\ + & -1 \\ \hline & 0 \end{array}$$

Sumar "1" al resultado

$$\begin{array}{rcc} & 1 & 0 \\ + & 1 & 1 \\ \hline & 0 & 1 \end{array} \text{ Decimal}\rightarrow \begin{array}{rc} & -1 \\ + & -0 \\ \hline & -1 \end{array}$$

$$\begin{array}{rc} 1 & \\ & + \quad 1 \\ & 1 \quad 0 \end{array} \rightarrow -1$$

Rebose

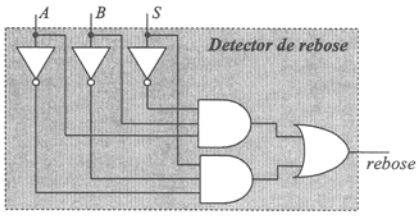
$$\begin{array}{rcc} & 0 & 1 \\ + & 0 & 1 \\ \hline & 1 & 0 \end{array} \text{ Decimal}\rightarrow \begin{array}{rc} & +1 \\ + & +1 \\ \hline & +2 \end{array}$$

$$-1$$

$$\begin{array}{rcc} & 1 & 0 \\ + & 1 & 0 \\ \hline & 0 & 0 \end{array} \text{ Decimal}\rightarrow \begin{array}{rc} & -1 \\ + & -1 \\ \hline & -2 \end{array}$$

$$1 \qquad -0$$

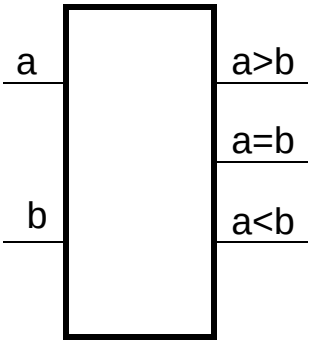
$$rebose = S_1\overline{A}_1\overline{B}_1 + \overline{S}_1A_1B_1$$



4. COMPARADORES

Comparador = elemento que compara dos datos de entrada (a, b) de n bits cada uno y activa una de entre tres salidas en función de que (a>b), (a=b), (a<b).

El número de bits de cada una de las entradas da nombre al comparador. Así un comparador de 4 bits es el que tiene cada una de sus entradas de 4 bits.



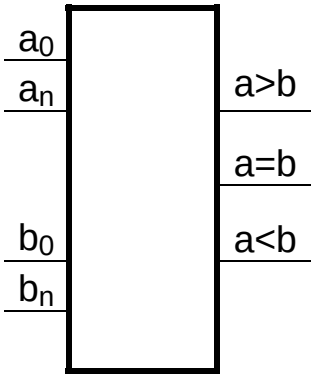
a	b	a>b	a=b	a<b
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

$$(a > b) = a\overline{b}$$

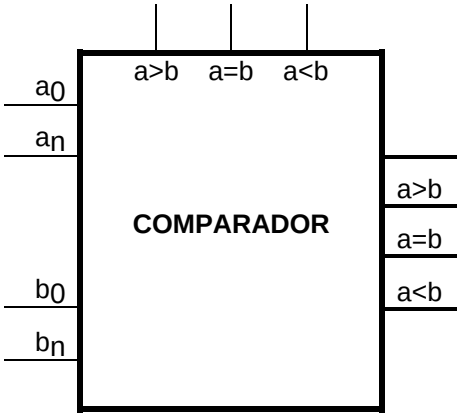
$$(a = b) = \overline{a}\overline{b} + ab = \overline{a \oplus b}$$

$$(a < b) = \overline{a}b$$

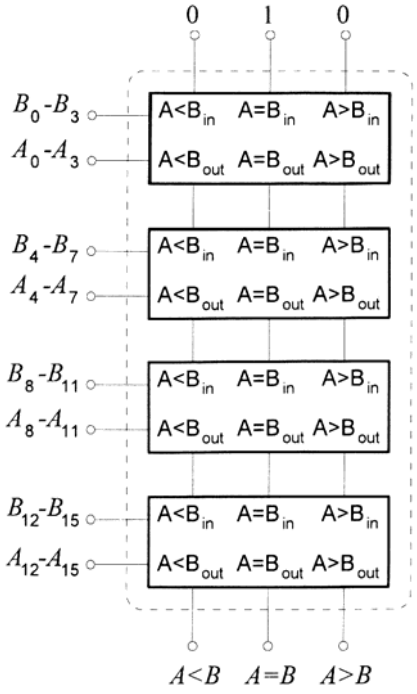
Comparador de n bits



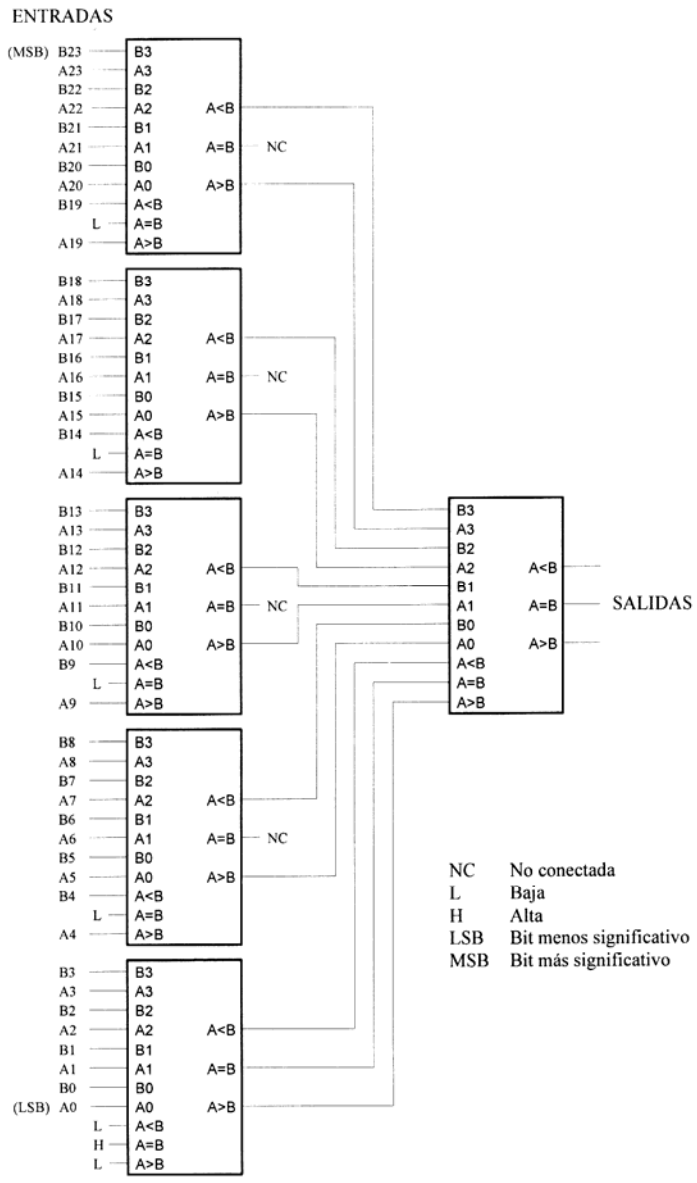
Comparador con entradas en cascada



En las entradas se meten las salidas de comparación de los bits inmediatamente inferiores. De esta manera siempre que haya una diferencia entre ai y bi la salida se posicionará en función de ello. Pero si $a_i=b_i$, la salida tomará el valor delas entradas dela comparación en cascada. Que resulta ser el valor de la comparación de a_{i-1} y b_{i-1}

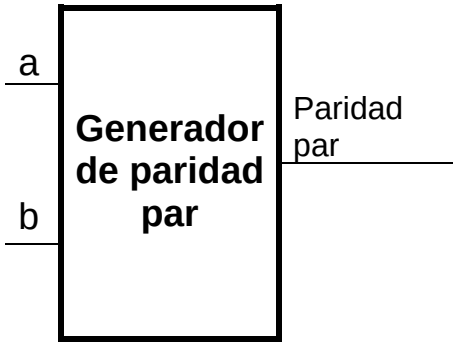


Comparador de 24 bits



Generadores / detectores de paridad

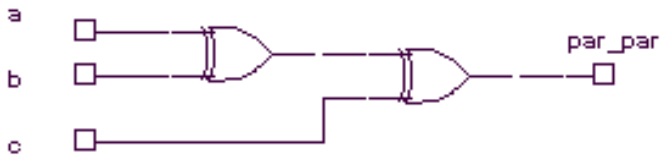
Los generadores de paridad PAR son aquellos circuitos que generan un “0” cuando el número de “1” a la entrada es par y un “1” cuando es impar.



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

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

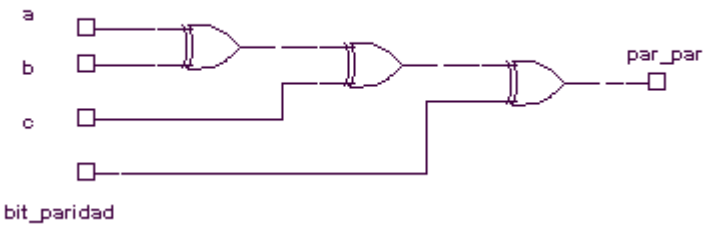
Para el caso de tres bits



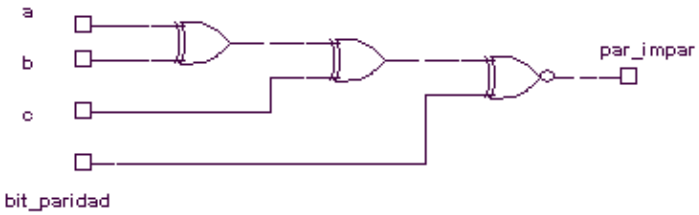
Como se puede observar lo único que hay que hacer para ampliar el número de bits es ir aumentando el número de puertas.

5. **Detector de paridad:**

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.

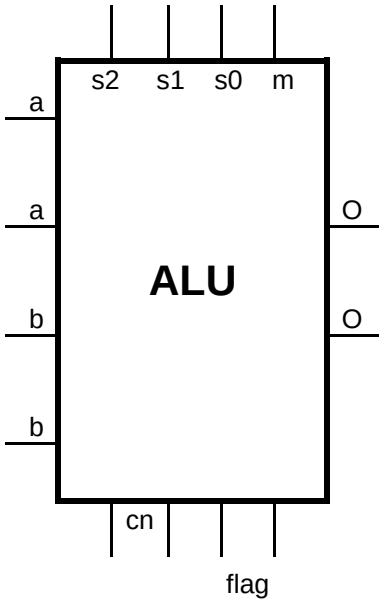


Para el caso del detector de paridad impar lo único que hay que hacer es sustituir la última puerta por una NOR-EXCLUSIVE.



6. UNIDADES ARITMÉTICO-LÓGICAS

Basados En multiplexores que seleccionan las funciones implementadas en las entradas



Arithmetic logic unit 74F181

FEATURES

- Provides 16 arithmetic operation: add, subtract, compare, and double; plus 12 other arithmetic operations
- Provides all 16 logic operations of two variables: Exclusive-OR, Compare, AND, NAND, NOR, OR, plus 10 other logic operations
- Full look-ahead carry for high speed arithmetic operation on long words

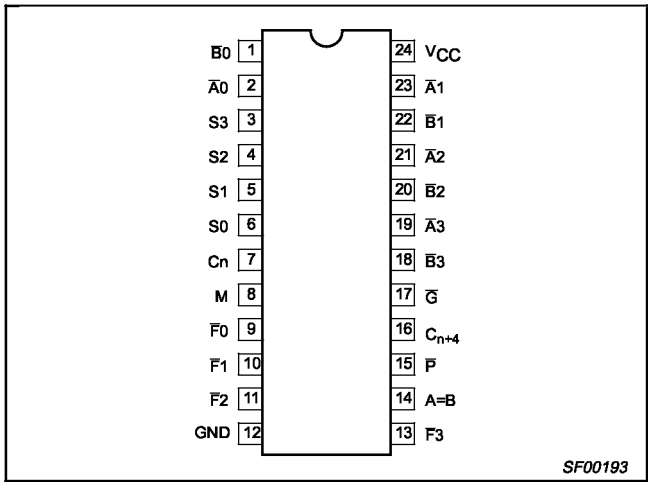
  40% faster than ’S181 with only 30% ’S181 power consumption

  Available in 300mil-wide Slim 24-pin Dual In-Line package

DESCRIPTION

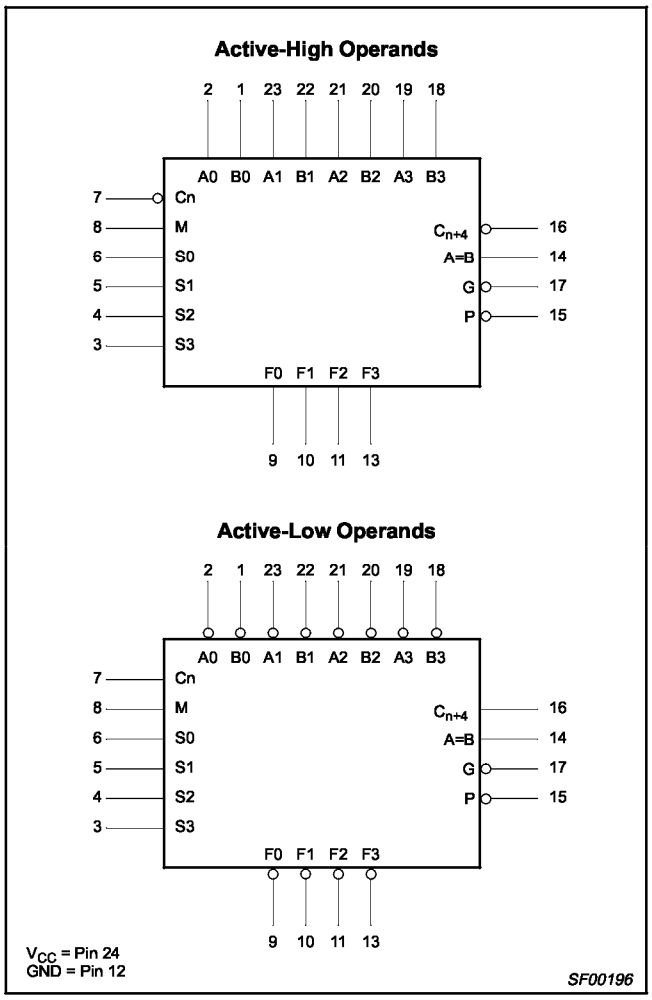
The 74F181 is a 4-bit high-speed parallel Arithmetic Logic Unit (ALU). Controlled by the four Function Select inputs (S0–S3) and the Mode Control input (M), it can perform all the 16 possible logic operations or 16 different arithmetic operations on active-High or active-Low operands. The Function Table lists these operations.

PIN CONFIGURATION

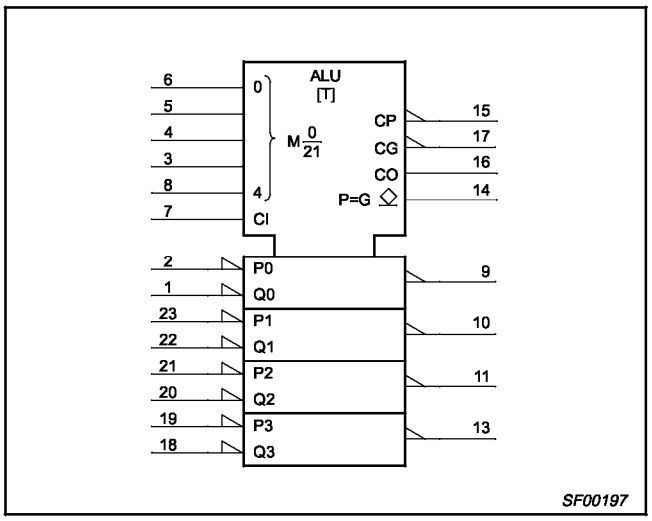


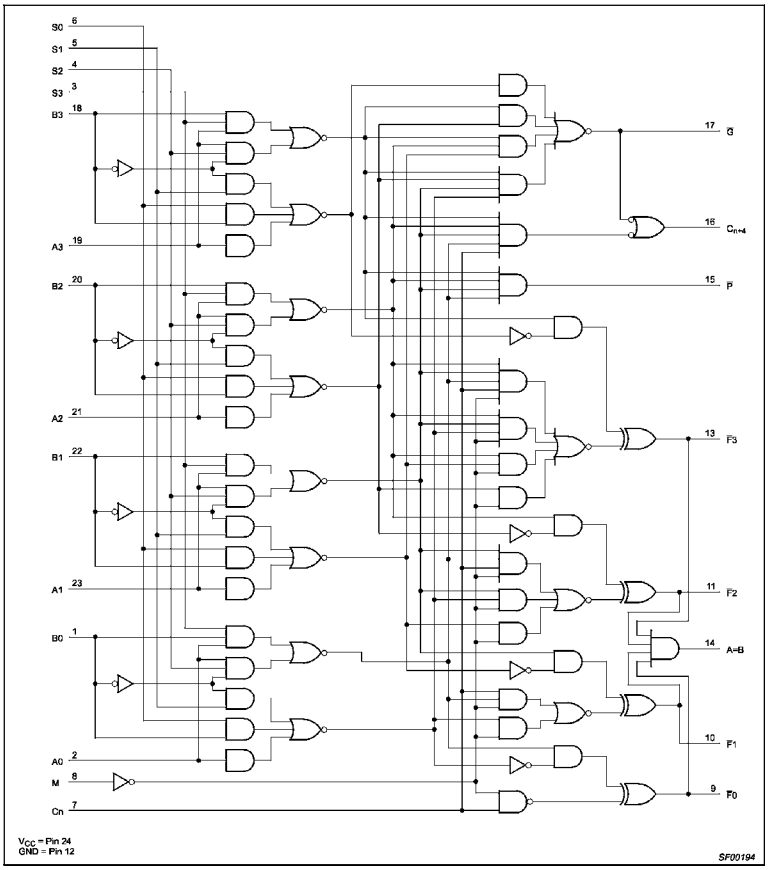
TYPE	TYPICAL PROPAGATION DELAY	TYPICAL SUPPLY CURRENT (TOTAL)
74F181	7.0ns	43mA

LOGIC SYMBOL



IEC/IEEE SYMBOL





When the Mode Control input (M) is High, all internal carries are inhibited and the device performs logic operations on the individual bits as listed. When the Mode control input is Low, the carries are enabled and the device performs arithmetic operations on the two 4-bit words. The device incorporates full internal carry look-ahead and provides for either ripple carry between device using the C_{n+4} output, or for carry look-ahead between packages using the signals P (Carry Propagate) and G (Carry Generate). P and G are not affected by carry in. When speed requirements are not stringent, it can be used in a simple ripple carry mode by connecting the Carry output (C_{n+4}) signal to the Carry input (Cn) of the next unit. For high-speed operation, the device is used in conjunction with the 74F182 carry look-ahead circuit. One carry look-ahead package is required for each group of four 74F181 devices. Carry look-ahead can be provided at various levels and offers high speed capability over extremely long word lengths.

The A=B output from the device goes High when all four F outputs are High and can be used to indicate logic equivalence over 4-bits when the unit is in the subtract mode. The A=B output is open-collector and can be wired-AND with other A=B outputs to give a comparison for more than 4 bits. The A=B signal can also be used with the C_{n+4} signal to indicate A>B and A<B. The Function Table lists the arithmetic operations that are performed without a carry in. An incoming carry adds a one to each operation. Thus select code LHHL generates A minus B minus 1 (two's complement notation) without a carry in and generates A minus B when a carry is applied. Because subtraction is actually performed by complementary addition (one's complement), a carry out means borrow; thus, a carry is generated when there is no underflow and no carry is generated when there is underflow. As indicated, this device can be used with either active-Low inputs producing active-Low outputs or with active-High inputs producing active-High outputs. For either case, the table lists the operations that are performed to the operands labeled inside the logic symbol.

MODE SELECT INPUTS				ACTIVE HIGH INPUTS & OUTPUTS		ACTIVE LOW INPUTS & OUTPUTS	
S3	**S2**	**S1**	**S0**	**Logic (M=H)**	**Arithmetic** (M=L) (Cn=H)**	**Logic (M=H)**	**Arithmetic** (M=L) (Cn=L)**
L	L	L	L	A	A	A	A minus 1
L	L	L	H	A+B	A+B	AB	AB minus 1
L	L	H	L	AB	A+B	A+B	AB minus 1
L	L	H	H	Logical 0	minus 1	Logical 1	minus 1
L	H	L	L	AB	A plus AB	A+B	A plus (A+B)
L	H	L	H	B	(A+B) plus AB	B	AB plus (A+B)
L	H	H	L	A B	A minus B minus 1	A B	A minus B minus 1
L	H	H	H	AB	AB minus 1	A+B	A+B
H	L	L	L	A+B	A plus AB	AB	A plus (A+B)
H	L	L	H	A B	A plus B	A B	A plus B
H	L	H	L	B	(A+B) plus AB	B	AB plus (A+B)
H	L	H	H	AB	AB minus 1	A+B	A+B
H	H	L	L	Logical 1	A plus A*	Logical 0	A plus A*
H	H	L	H	A+B	(A+B) plus A	AB	AB plus A
H	H	H	L	A+B	(A+B) plus A	AB	AB plus A
H	H	H	H	A	A minus 1	A	A

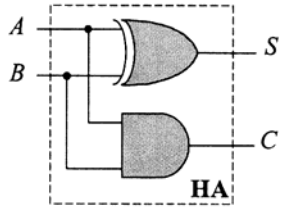
1999 2ª S

La función aritmética de sumar:
1.1 Semisumador, sumador completo y sumador serie.
1.2 ¿Cómo se convertiría un semisumador en semirrestador?.

1.1 Página 270 del libro de teoría.

1.2

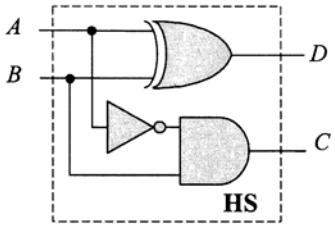
Semisumador:



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

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

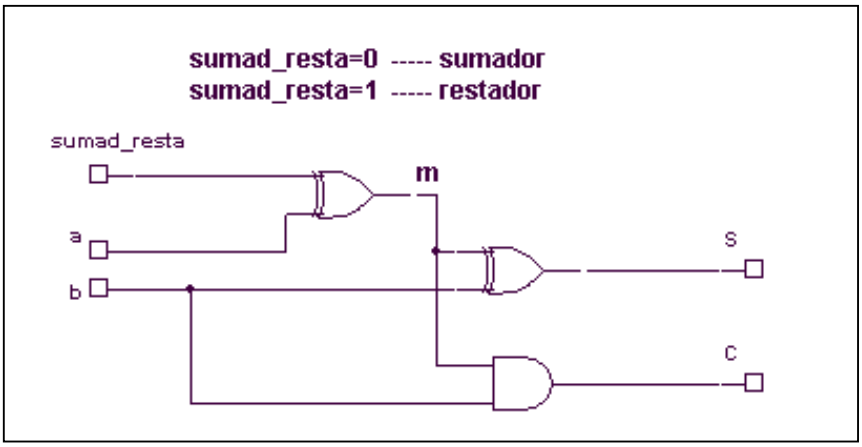
Semirrestador:



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

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

Como se puede apreciar la diferencia consiste en que la entrada "A" en el semisumador es literal y en el semirrestador es invertida.
Por lo tanto la solución es invertir dicha entrada.
Un circuito que podría servir para ambos propósitos consistiría en colocar una puerta que entregaría una variable literal o invertida según se seleccione con una patilla. Dicha puerta es una "O-exclusiva" en la que la variable entra en una de las entradas y la otra entrada se utiliza como selector de función



Sumad_resta	a	m
0	a	a
1	a	$\overline{a}$