Sumadores

En electrónica un **sumador** es un circuito lógico que calcula la operación suma. En los computadores modernos se encuentra en lo que se denomina Unidad aritmético lógica (ALU). Generalmente realizan las operaciones aritméticas en código binario decimal o BCD exceso 3, por regla general los sumadores emplean el sistema binario. En los casos en los que se esté empleando un complemento a dos para representar números negativos el sumador se convertirá en un sumador-substractor (*Adder-subtracter*).

Tipos de sumadores:

Half-adder.
Full-Adder.
Carry-Look-Ahead.
Carry-select.

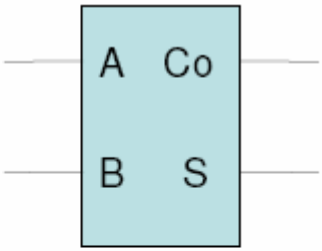
Half-Adder

Se denomina **semisumador** al circuito combinacional capaz de realizar la suma aritmética binaria de dos únicos bits A y B, proporcionando a su salida un bit resultado de suma S y un bit de acarreo C. En la siguiente figura se muestra la tabla de verdad de este circuito con sus funciones, acompañado de un esquema del Half-Adder.

A + B		Co	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

$S = A \oplus B$

$Co = A\ B$



Semisumador o "Half-Adder"

Full-Adder

Este dispositivo nos ofrece una mejora del semisumador al cual se le añade un acarreo de entrada.

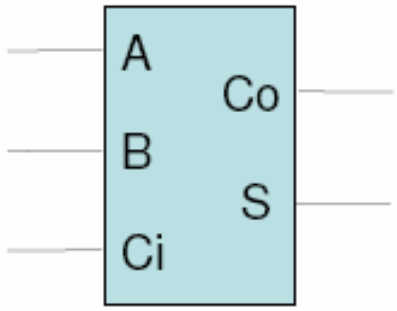
De está manera podemos afrontar sumas de mas de un bit para las cuales utilizaremos el acarreo de salida del anterior en el acarreo de entrada del siguiente. Así completamos la suma correctamente.

A continuación vemos la tabla de verdad y un esquema.

A +	B +	Ci	Co	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

$S = A \oplus B \oplus Ci$

$Co = A\,B + A\,Ci + B\,Ci$



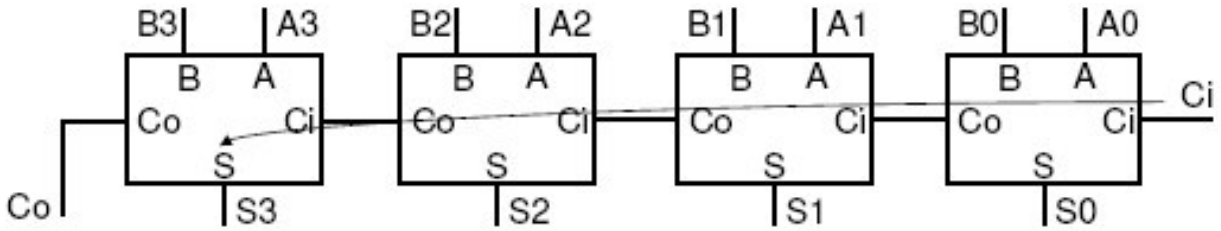
Sumador completo ó
"Full-Adder"

Método Ripple

Un sumador de dos informaciones binarias A+B de ***n*** bits necesita realizar ***n*** sumas parciales, empleando para ello ***n*** sumadores completos.

Esto nos hace conectar el acarreo de salida con el siguiente acarreo de entrada de manera que podamos realizar la suma del siguiente bit con acarreo.

Es un circuito muy simple e intuitivo pero presenta el serio inconveniente de tener que esperar un tiempo igual a ***n*** **tiempos de propagación** antes de obtener un resultado estable.



VHDL Half-Adder

Ejemplo: Desarrollo de un semisumador de 4 bits en base a sentencias generate

```
entity parasum is
generic (N: integer := 4);
port (A,B:  in bit_vector(N downto 1);
      S:    out bit_vector(N downto 1);
      Cout: out bit);
end parasum;

architecture genera of parasum is
signal C: bit_vector(N downto 1);
begin
G1: for i in 1 to N generate
  G2: if ( i = 1 ) generate
    S(i) <= A(i) xor B(i);
    C(i) <= A(i) and B(i);
  end generate G2;
  G3: if ( i /= 1 ) generate
    S(i) <= A(i) xor B(i) xor C(i-1);
    C(i) <= (A(i) and B(i)) or (A(i) and C(i-1)) or (B(i) and C(i-1));
  end generate G3;
end generate G1;
Cout <= C(4);
end genera;
```

Carry Look Ahead

Este sumador, llamado también sumador paralelo con acarreo anticipado, realiza la suma aumentando la velocidad de proceso sobre la conexión en serie. Lo logra mediante la generación de todos los bits de acarreo en el mismo proceso de calculo de las sumas parciales.

Al sumar dos informaciones se obtendrá el acarreo por dos posibilidades:

- Se genera acarreo en la propia etapa del sumador. **Generado (A=B=1)**

$$Gj = Aj * Bj$$

- Proviene de la etapa anterior. **Propagado**

$$Pj = Aj \oplus Bj$$

Por tanto el acarreo producido en la etapa i-esima C_i será porque se genera o propaga y se expresará:

$$C_i = G_i + P_i\, C_{i-1} = A_i\, B_i + (A_i + B_i)\, C_{i-1}$$

Carry Look Ahead

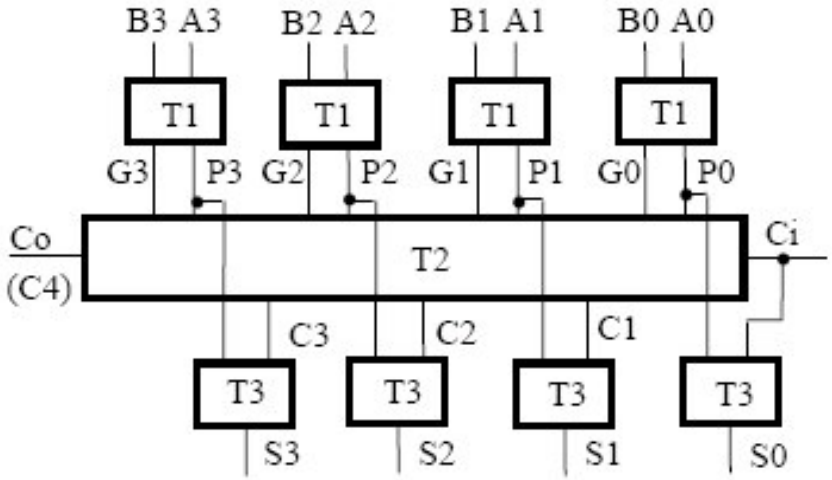
Para un sumador de cuatro bits (i=0, … , 3) se tiene que:

- Para 4 bits se generan los acarreos en paralelo:

T2

$C1 = G0 + P0Ci$
$C2 = G1 + P1\ C1 = G1 + P1\ (G0 + P0\ Ci) = G1 + P1G2 + P1P0Ci$
$C3 = G2 + P2\ C2 = G2 + P2G1 + P2P1G0 + P2P1P0Ci$
$Co = C4 = G3 + P3\ C3 = G3 + P3G2 + P3P2G1 + P3P2P1G0 +$
$+ P4P3P2P1Ci$

$Sj = Ai \oplus Bj \oplus Ci = Pj \oplus Ci$



VHDL Carry Look Ahead

```
ENTITY Sum_anticipado IS
        Generic (nbits: integer := 8);   -- nbits es el número de bits del sumador
        PORT(     sa_A, sa_B : in std_logic_vector(nbits-1 downto 0);
        sa_Cin : in std_logic;
        sa_S : out std_logic_vector(nbits-1 downto 0);
        sa_Cout : out std_logic);
END Sum_anticipado;

-- Descripción estructural

ARCHITECTURE estructural OF Sum_anticipado IS

  constant  tp: time := 10 ns;      -- retardo de propagación
  signal C: std_logic_vector(nbits-1 downto -1);
  signal G, P:std_logic_vector(nbits-1 downto 0);

  component Sum_completo
        PORT(     a,b : in std_logic;
                  Cin : in std_logic;
                  S : out std_logic;
                  Cout : out std_logic);
 END component;
```

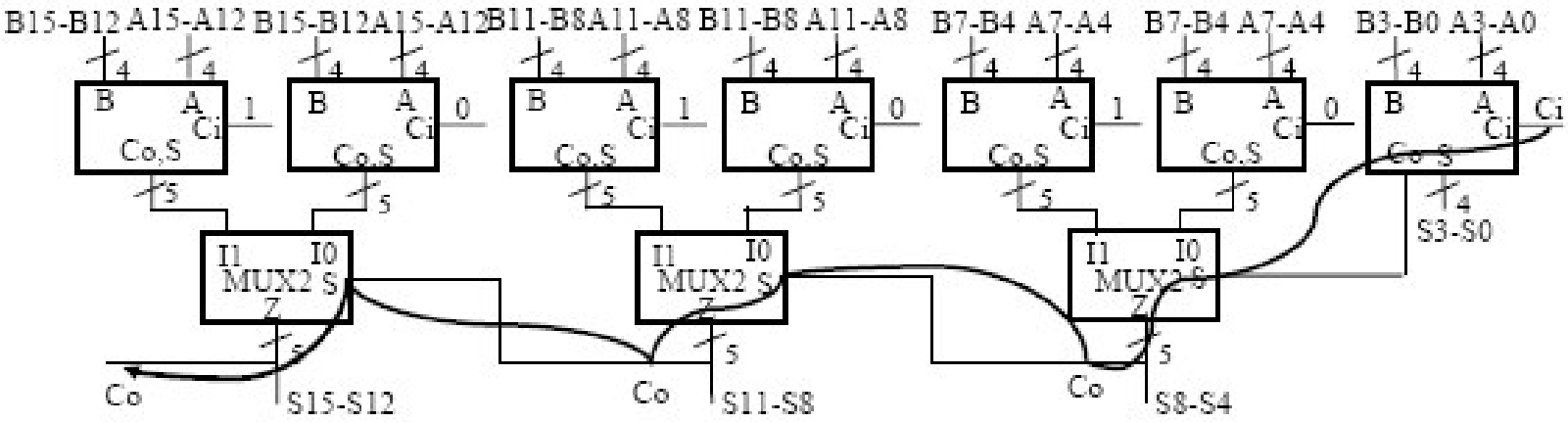
VHDL Carry Look Ahead

```
BEGIN
            C(-1) <= sa_Cin;
            G <= sa_A and sa_B after tp;
            P <= sa_A xor sa_B after tp;

              -- Generación de acarreos anticipados

acarreo:   for j in 0 to nbits-1 generate
                C(j) <= G(j) or (P(j) and C(j-1));   -- Expresión del acarreo anticipado
            end generate acarreo;

            sa_Cout <= C(nbits-1) after 2*tp;

sum_ant: for i in 0 to nbits-1 generate
              elemento_sumador:sum_completo port map(
                                a => sa_A(i),
                                 b => sa_B(i),
                                Cin => C(i-1),
                                S => sa_S(i),
                                Cout => open);
            End generate sum_ant;

END estructural;
```

Carry Select

En este tipo de sumador se realiza un acarreo mixto basado en sumadores y multiplexores, donde la generación de acarreo en cada sumador se realiza en paralelo y la propagación en cada multiplexor en serie.

El tiempo de propagación de este sumador depende del tiempo de propagación de la primera etapa, más el tiempo de propagación de los (M/N-1) multiplexores para propagación del acarreo. A cambio el circuito es bastante más grande que la estructura “ripple”.



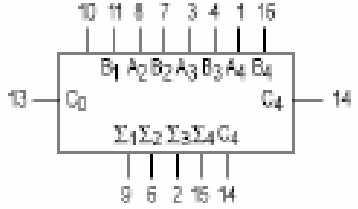
Circ. Comercial 83

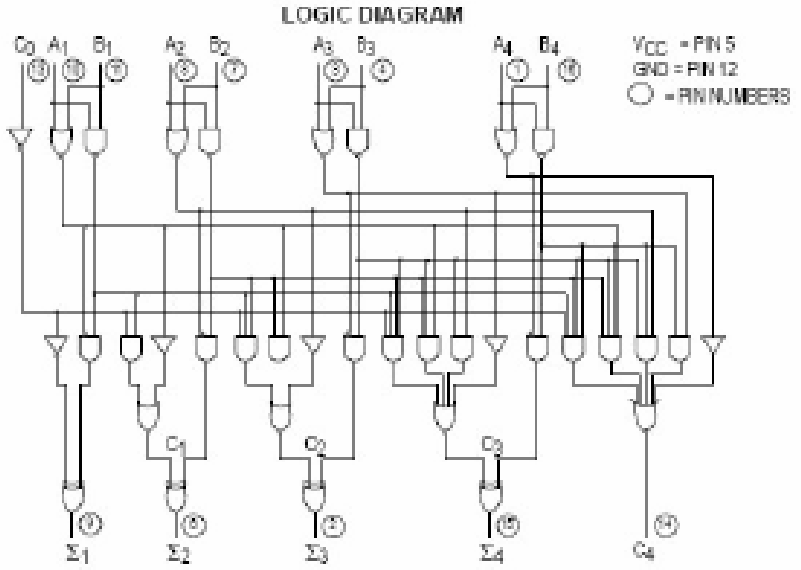
Circuitos comerciales: 74'83. Sumador de 4 bits, con estructura interna de carry look-ahead. El circuito opera como sumador suponiendo las entradas y salidas en polaridad positiva o en polaridad negativa.

$C_0 + (A_1+B_1)+2(A_2+B_2)+4(A_3+B_3)+8(A_4+B_4) = \Sigma_1+2\Sigma_2+4\Sigma_3+8\Sigma_4+16C_4$

Where: (+) = plus

	C_0	A_1	A_2	A_3	A_4	B_1	B_2	B_3	B_4	Σ_1	Σ_2	Σ_3	Σ_4	C_4	
Logic Levels	L	L	H	L	H	H	L	L	H	H	H	L	L	H	
Active HIGH	0	0	1	0	1	1	0	0	1	1	1	0	0	1	(10+9 = 19)
Active LOW	1	1	0	1	0	0	1	1	0	0	0	1	1	0	(carry+5+6 = 12)



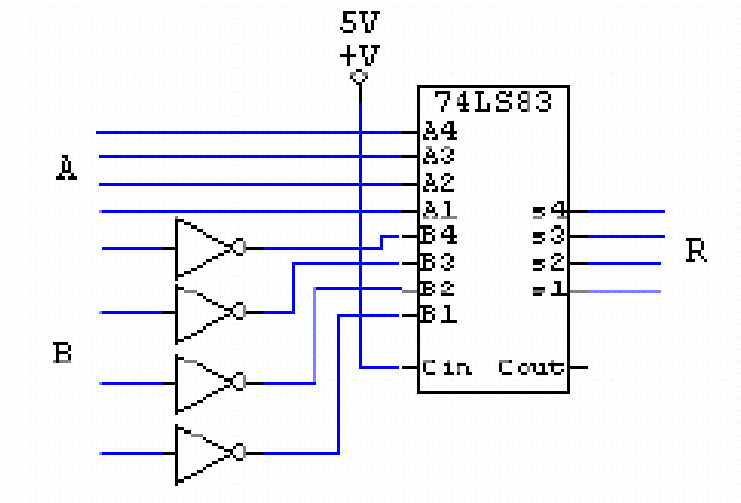


Otras Aplicaciones

A base de sumadores se pueden desarrollar otras aplicaciones como por ejemplo un restador o un multiplicador vamos a explicar el restador y un sumador/restador.

Este dispositivo puede verse como **A – B = A + (-B)** , para la conversión del operando B se emplea la codificación en complemento a dos.

A continuación vemos un restador con un 83.



Otras Aplicaciones

Un sumador/restador necesita una entrada de control C, que indique si se realiza la operación de suma o de resta. Para hacer la resta se requiere el c-a-2, luego los operandos X e Y, y la salida Z están en esta notación.

Si C = 0 => Z = X + Y + 0

Si C = 1 => Z = X – Y = X + Y + 1

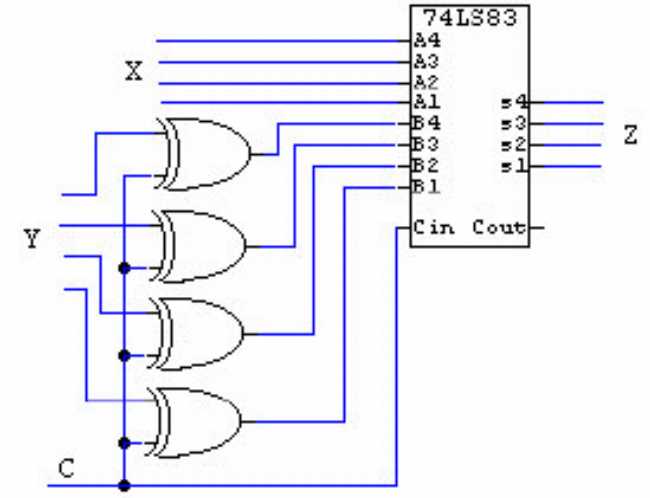
Para el operando A del sumador da igual el valor de C, A <= X

Para el operando B si C = 0 => B <= Y, si C = 1=> B <= Y, luego

B = C Y + C Y = C Å Y

Para Cin, si C = 0 => Cin <= 0, si C = 1=> Cin <= 1, luego Cin = C

Sumador/Restador de 4 bits usando el sumador 74LS83



Problemas propuestos

• Diseñar utilizando únicamente semisumadores y sumadores completos un circuito digital que realice la multiplicación de un números binario de dos bits por otro de tres bits.

• Realizar la suma de cuatro números de dos bits A (a2a1), B (b2b1), C (c2c1) y D (d2d1) y tres números de 1 bit, E (e1), F (f1) y G (g1) utilizando el menor número posible de sumadores completos (“fulladders”).

• Diseñar un circuito que realice la operación aritmética:
O = 5 X + 2 Y + Z
para operandos X (x1x0), Y (y1y0) y Z (z1z0) de dos bits, utilizando el menor número posible de semisumadores de dos bits de operandos de entradas A (a1a0) y B (b1b0).

• Realizar el diseño de un circuito que sume dos dígitos NBCD, dando el resultado en código NBCD, utilizando puertas lógicas cuando sea necesario. Indicar como puede utilizarse este circuito para sumar números NBCD de más de un dígito
Ayuda: Hay que sumar 6.