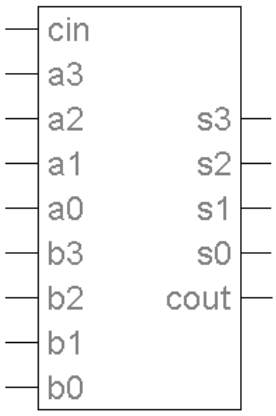
Un Sumador de 4 bits

La siguiente descripción de un sumador binario de 4 bits ilustra el uso de los operadores de concatenación para formar vectores a partir de bits. El símbolo & sirve para formar los vectores.



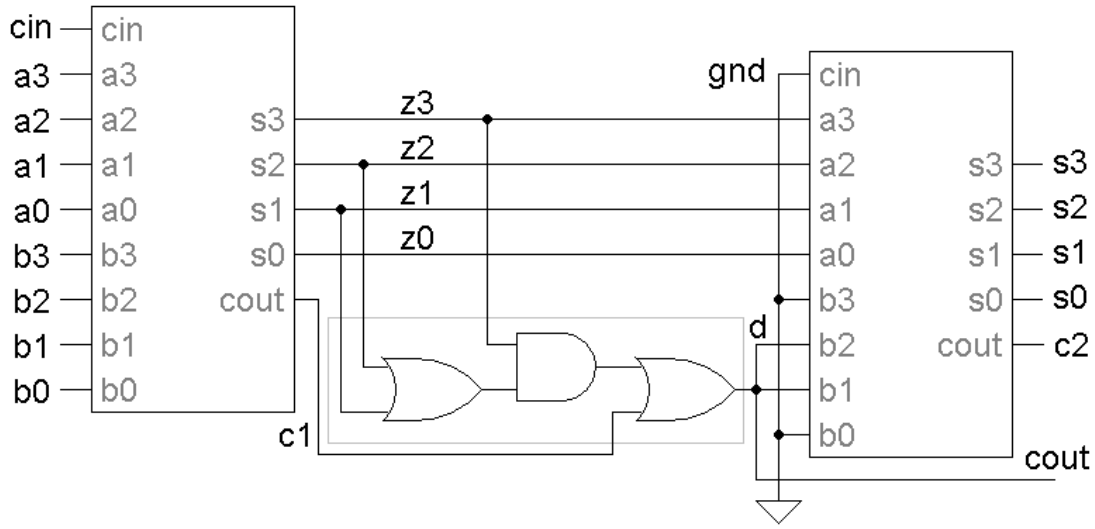

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;
USE ieee.std_logic_unsigned.all;

ENTITY Sumador4bits IS
PORT(
    cin             : IN    STD_LOGIC;
    a3, a2, a1, a0  : IN    STD_LOGIC;
    b3, b2, b1, b0  : IN    STD_LOGIC;
    s3, s2, s1, s0  : OUT   STD_LOGIC;
    cout            : OUT   STD_LOGIC);
END Sumador4bits;

ARCHITECTURE abstracta OF Sumador4bits IS
SIGNAL z: STD_LOGIC_VECTOR(4 DOWNTO 0);
BEGIN

z <= (a3 & a2 & a1 & a0) +
     (b3 & b2 & b1 & b0) + cin;

s0 <= z(0);
s1 <= z(1);
s2 <= z(2);
s3 <= z(3);

cout <= z(4);

END abstracta;
```

Figura 15. Descripción de un sumador de 4 bits.

La entidad del sumador define todas las entradas y salidas como señales de un solo bit. En la arquitectura los bits de las entradas se agrupan y luego se suman para guardar el resultado en la señal z.

Sumador BCD

La figura 16 muestra un circuito para sumar números en formato BCD. Este circuito emplea dos sumadores binarios. Si la suma de los números de entrada es mayor que 9 entonces se agrega 6 a la suma original para realizar el ajuste decimal. La señal **z** es de cinco bits para agrupar la suma y acarreo del primer sumador. La función lógica para la señal **d** determina si la primera suma es mayor que 9. La señal **y** es una entrada del segundo sumador binario que puede ser "0000" (sin ajuste decimal) ó "0110" (con ajuste decimal). Note que debe agregar la biblioteca ieee y sus componentes para realizar la suma de vectores.

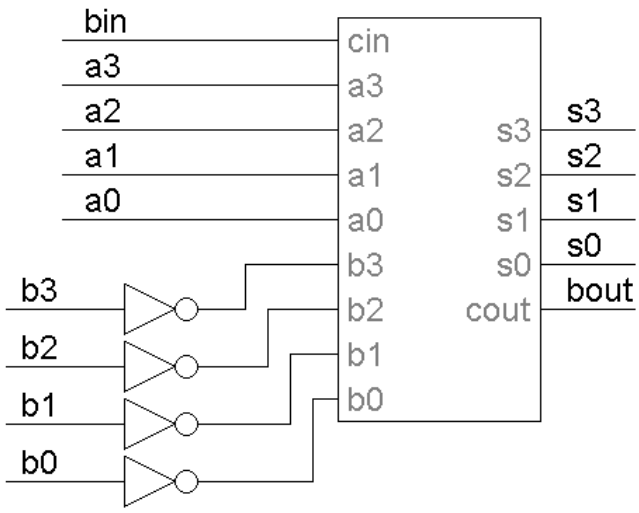



```
ENTITY SumadorBCD IS
    PORT
    (   cin     : IN    STD_LOGIC;
        a, b    : IN    STD_LOGIC_VECTOR(3 DOWNTO 0);
        s       : OUT   STD_LOGIC_VECTOR(3 DOWNTO 0);
        cout    : OUT   STD_LOGIC);
END SumadorBCD;

ARCHITECTURE rtl OF SumadorBCD IS
    SIGNAL z: STD_LOGIC_VECTOR(4 DOWNTO 0);
    SIGNAL y: STD_LOGIC_VECTOR(3 DOWNTO 0);
    SIGNAL d: STD_LOGIC;
BEGIN

    z  <= a + b + cin;
    d <= (z(3) AND (z(2) OR z(1))) OR z(4);

    y  <= '0' & d & d & '0';

    s  <= z(3 DOWNTO 0) + y;
    cout <= d;

END rtl;
```

Figura 16. Descripción de un sumador de 4 bits.

Restador Binario de 4 bits

El circuito de la figura 18 muestra una descripción de un restador binario de 4 bits sin signo empleando un sumador binario de 4 bits. La entrada de préstamo (bin) se activa en 0, al igual que la salida de préstamo (bout). El minuendo (a) se conecta directamente a una entrada del sumador y en la otra entrada se conecta el complemento a 1 del substraendo (b).



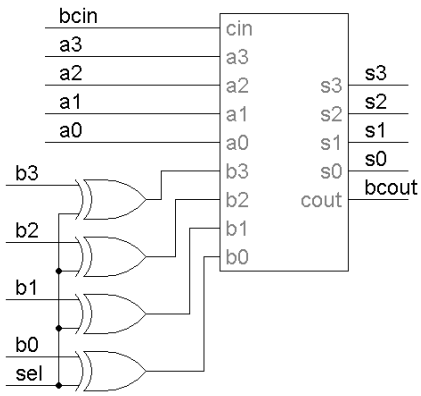

```
ENTITY Restador4bits IS
    PORT
    (   bin     : IN    STD_LOGIC;
        a, b    : IN    STD_LOGIC_VECTOR(3 DOWNTO 0);
        s       : OUT   STD_LOGIC_VECTOR(3 DOWNTO 0);
        bout    : OUT   STD_LOGIC);
END Restador4bits;

ARCHITECTURE rtl OF Restador4bits IS
    SIGNAL z: STD_LOGIC_VECTOR(4 DOWNTO 0);

BEGIN
    z <= a + (NOT b) + bin;
    s <= z(3 DOWNTO 0);
    bout <= z(4);

END rtl;
```

Figura 18. Restador binario de 4 bits

NOTA: Si en una aplicación los números negativos se representan en complemento a 2 entonces no se requiere crear un restador, basta con utilizar un sumador binario.

Sumador y Restador Binario de 4 bits

El circuito de la figura 19 suma o resta números binarios sin signo de 4 bits dependiendo de la entrada de selección. La entrada bcin y la salida bcout se activan en 1 para la suma y en 0 para la resta. Si sel es 0 el circuito realiza a + b + bcin, y si es 1 el circuito ejecuta a – b – (1 – bcin).




```
ENTITY SumResBIN IS
    PORT
    (   bcin        : IN    STD_LOGIC;
        a, b    : IN    STD_LOGIC_VECTOR(3 DOWNTO 0);
        sel     : IN    STD_LOGIC;
        s       : OUT   STD_LOGIC_VECTOR(3 DOWNTO 0);
        bcout   : OUT   STD_LOGIC);
END SumResBIN;

ARCHITECTURE estructural OF SumResBIN IS
    SIGNAL bn: STD_LOGIC_VECTOR(3 DOWNTO 0);
    COMPONENT SumadorNbits IS
    GENERIC( N: INTEGER:= 4);
    PORT(
        a, b    : IN    STD_LOGIC_VECTOR(N - 1 DOWNTO 0);
        cin     : IN    STD_LOGIC;
        s       : OUT   STD_LOGIC_VECTOR(N - 1 DOWNTO 0);
        cout    : OUT   STD_LOGIC);
    END COMPONENT;

BEGIN
    WITH sel SELECT
    bn <= NOT b WHEN '1',
          b     WHEN OTHERS;

    Sumador: SumadorNbits
        GENERIC MAP(N => 4)
        PORT MAP(a => a, b => bn, cin => bcin,
                 s => s, cout => bcout);

END estructural;
```

Figura 19. Sumador restador binario de 4 bits

Restador BCD

Un diseño típico de un restador BCD utiliza un complemento a 9 de la entrada B (figura 20).

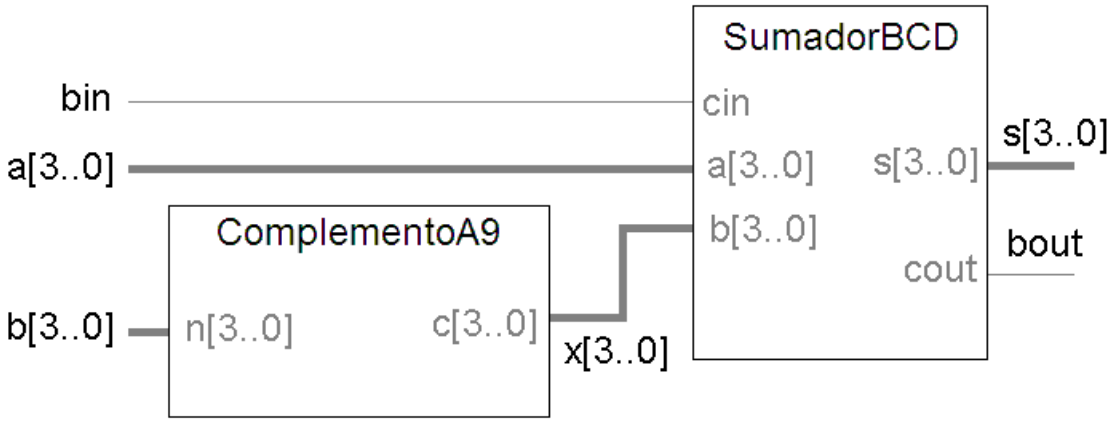


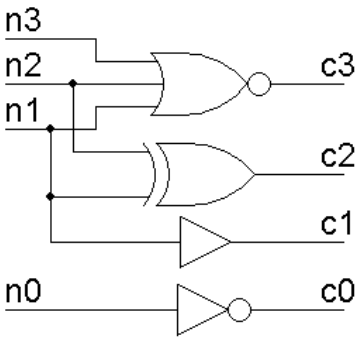
Figura 20. Restador BCD

La figura 21 muestra cuatro descripciones para la función complemento a 9. La señal n es la entrada BCD y la señal c es su complemento a 9 (c = 9 – n).

```
WITH n SELECT
c <= "1001" WHEN "0000",
     "1000" WHEN "0001",
     "0111" WHEN "0010",
     "0110" WHEN "0011",
     "0101" WHEN "0100",
     "0100" WHEN "0101",
     "0011" WHEN "0110",
     "0010" WHEN "0111",
     "0001" WHEN "1000",
     "0000" WHEN "1001",
     "0000" WHEN OTHERS;
```

```
WITH n SELECT
c <= "1001" WHEN "0000",
     "1000" WHEN "0001",
     "0111" WHEN "0010",
     "0110" WHEN "0011",
     "0101" WHEN "0100",
     "0100" WHEN "0101",
     "0011" WHEN "0110",
     "0010" WHEN "0111",
     "0001" WHEN "1000",
     "0000" WHEN "1001",
     "1111" WHEN OTHERS;
```

```
WITH n SELECT
c <= "1001" WHEN "0000",
     "1000" WHEN "0001",
     "0111" WHEN "0010",
     "0110" WHEN "0011",
     "0101" WHEN "0100",
     "0100" WHEN "0101",
     "0011" WHEN "0110",
     "0010" WHEN "0111",
     "0001" WHEN "1000",
     "0000" WHEN "1001",
     "----" WHEN OTHERS;
```




```
c(3) <= NOT (n(3) OR n(2) OR n(1));
c(2) <= n(2) XOR n(1);
c(1) <= n(1);
c(0) <= NOT n(0);
```

Figura 21. Descripciones VHDL para el complemento a 9

Una descripción estructural del restador BCD se indica en la figura 21. Esta descripción utiliza los componentes ComplementoA9 y SumadorBCD4bits.

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY RestadorBCD IS
    PORT(
        a, b    : IN    STD_LOGIC_VECTOR(3 DOWNTO 0);
        ci      : IN    STD_LOGIC;
        s       : OUT   STD_LOGIC_VECTOR(3 DOWNTO 0);
        co      : OUT   STD_LOGIC
        );
END RestadorBCD;

ARCHITECTURE estructural OF RestadorBCD IS

    COMPONENT SumadorBCD4bits
        PORT(
            a, b    : IN    STD_LOGIC_VECTOR(3 DOWNTO 0);
            ci      : IN    STD_LOGIC;
            s       : OUT   STD_LOGIC_VECTOR(3 DOWNTO 0);
            co      : OUT   STD_LOGIC
            );
    END COMPONENT;

    COMPONENT ComplementoA9
        PORT(
            n       : IN    STD_LOGIC_VECTOR(3 DOWNTO 0);
            c       : OUT   STD_LOGIC_VECTOR(3 DOWNTO 0)
            );
    END COMPONENT;

    SIGNAL x: STD_LOGIC_VECTOR(3 DOWNTO 0);

BEGIN

    U1: ComplementoA9
    PORT MAP (b, x);

    U2: SumadorBCD4bits
    PORT MAP (a, x, ci, s, co);

END estructural;
```

Figura 21. Descripción estructural del restador BCD