SUMADORES

Rodrigo Nevado Anton
Sergio Gomez Rodriguez

(Grupo5)

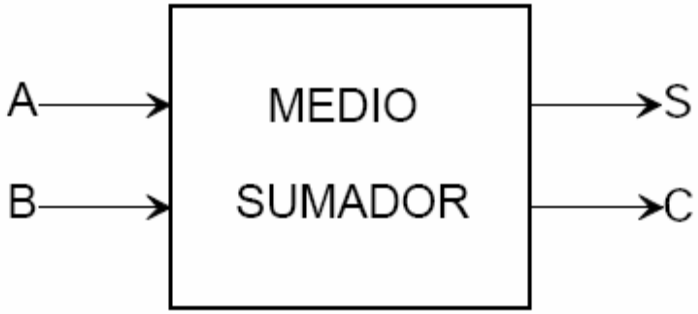
Sumadores

- Los circuitos aritméticos posibilitan las operaciones de cálculo en la tecnología digital y representan la base para el desarrollo de los sistemas computacionales.
- En nuestro caso los circuitos digitales sumadores realizan la suma aritmética de dos números enteros positivos, aunque se pueden desarrollar para otros formatos de descripción numérica.
- No solo existe un tipo de estructura de circuitos digitales sumadores sino que existen muchas diferenciándose en su tamaño y velocidad.
- Para operandos de entrada de un bit A y B existen dos estructuras el full-adder (sumador completo) o half-adder (semi-sumador) que no contiene acarreo de entrada
- Existirán dos bits a la salida de cada suma que será S (suma) y el acarreo de salida (Co) a partir de la tabla que se genera se pueden sacar sus funciones lógicas siendo las siguientes:

sumadores

Medio sumador

Un medio sumador es un sumador capaz de sumar dos datos de un sólo bit y producir un bit de acarreo de salida. Como se muestra en el siguiente diagrama de bloques

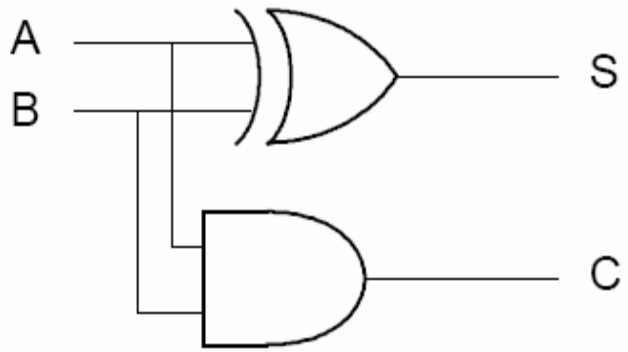


La manera como realiza la suma y produce el acarreo el medio sumador se desglosa en la siguiente tabla de verdad

sumadores

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

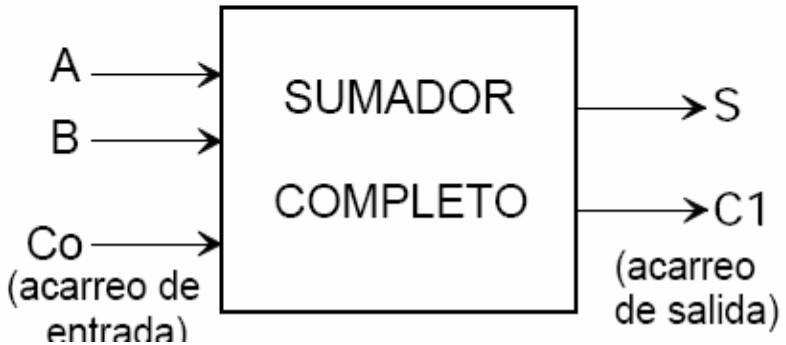
De lo cual es evidente la expresión lógica para cada salida: C= A.B y S =A⊕B. Con lo cual, la implementación del medio sumador es como se muestra a continuación



sumadores

Sumador completo de un bit

El medio sumador no puede ser interconectado con otros medios sumadores para formar un sumador más grande, por ello es necesario diseñar un sumador que admita otra entrada aparte de los datos a sumar, es decir, un sumador de 3 datos de 1 bit, éste es denominado sumador completo y su diagrama de bloques es como se muestra a continuación



Sumador completo de un bit

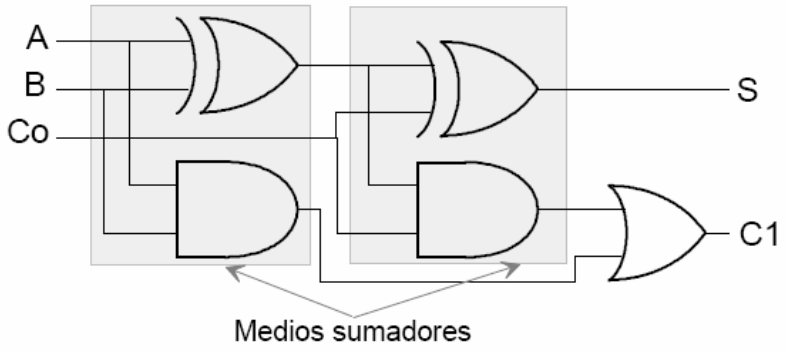
En la siguiente tabla de verdad se muestra la manera como este sumador realiza su función

A B Co	C_1 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

Un análisis de esta tabla de verdad y el uso de Mapas de Karnaugh nos lleva a las siguientes expresiones para C_1 y S:

$$S = A \oplus B \oplus C_0, \quad C_1 = AB + (A \oplus B)C_0$$

Con lo cual la implementación del sumador completo es como se muestra en la siguiente figura



Sumadores

Sumador completo

En la practica, los circuitos sumadores manejan informaciones binarias con una longitud de palabra superior a un bit, por lo que es preciso ampliar la funcionalidad del semisumador a un dispositivo capaz de realizar sumas binarias de n bits.

Desde un punto de vista genérico sean A y B dos informaciones binarias de n bits:

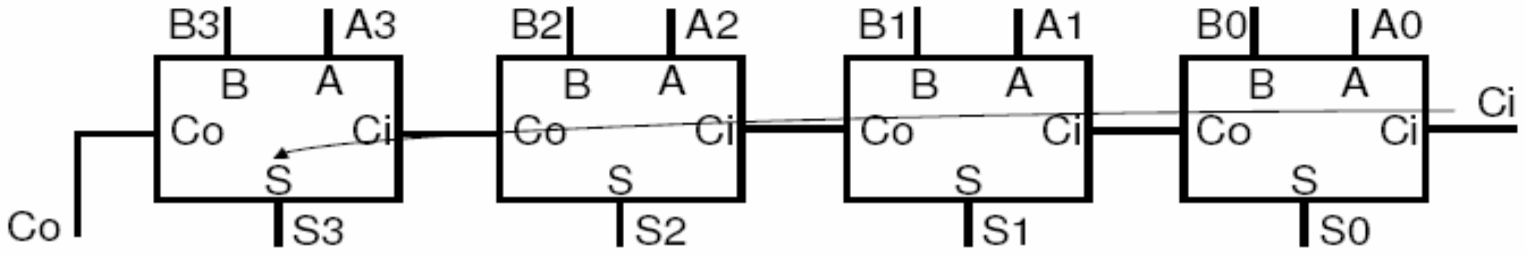
$A=A_{n-1}\ A_{n-2}\ \ldots A_2\ A_1\ A_0$ $B=B_{n-1}\ B_{n-2}\ \ldots.B_2\ B_1\ B_0$

La suma A + B será un proceso de n sumas parciales, comenzando por los dos bits de peso menor. Se comienza con A_0 Y B_0 dando esto un S_0 y un bit de acarreo C_0 que se propagará como elemento integrante de la suma, en los siguientes dos bits en orden ascendente de peso. A partir de entonces el resultado de todas las operaciones incorporara un bit de suma S_i y un bit de acarreo C_i que se propagara como elemento integrante de la suma en los dos bits siguientes y así sucesivamente.

$$\mathbf{S=C_{n-1}\ S_{n-1}\ S_{n-2}\ \ldots.\ S_2\ S_1\ S_0}$$

Sumadores

Para sumadores de N bits un método natural de realizar la suma es situar sumadores completos en modo "ripple" o en serie, de forma que desde el bit menos significativo hacia el más significativo el acarreo de entrada del bit j esté conectado al acarreo de salida del bit j-1.
El primer bit se puede construir con un semisumador mientras que los demás bits requieren un sumador completo, en este caso se tiene un semisumador de N bits.
Si en el primer bit se utiliza un sumador completo, el circuito dispone además de acarreo de entrada Ci y se tiene un sumador completo de N bits. Tener acarreo de entrada permite un mejor funcionalidad del circuito, como por ejemplo poner en serie dos (o más) sumadores de N bits para formar un sumador de 2N bits.
En cualquier caso se tienen N+1 bits de salida: N de suma S y un acarreo de salida Co.
El tiempo de propagación de este sumador es proporcional al número N de sumadores en serie.



Sumadores

- El sumador "ripple" es un sumador lento. Para construir sumadores rápidos hay que intentar paralelizar el cálculo de los acarreos. El método "carry look-ahead" genera los acarreos en paralelo obteniendo su expresión lógica de forma recursiva:

- Para cada bit j se obtiene de Aj y Bj en paralelo el "Carry-Propagate" Pj que indica las condiciones bajo las cuales el acarreo se propaga de la entrada a la salida, y el "Carry-Generate" Gj que indica las condiciones bajo las cuales se genera acarreo de salida independientemente del acarreo de entrada.

T1 $\boxed{\begin{array}{l} Pj = Aj \oplus Bj \\ Gj = Aj\ Bj \end{array}}$ hay un 1 en Aj ó en Bj, se propaga el acarreo
hay dos 1s entre Aj y Bj, se genera acarreo.

Co = G + P Ci, hay acarreo si se genera o si se propaga con Ci a 1.

- Para 4 bits se generan los acarreos en paralelo:

T2

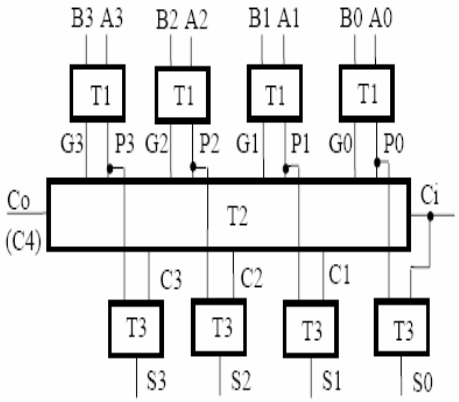
$$C1 = G0 + P0Ci$$
$$C2 = G1 + P1\ C1 = G1 + P1\ (G0 + P0\ Ci) = G1 + P1G2 + P1P0Ci$$
$$C3 = G2 + P2\ C2 = G2 + P2G1 + P2P1G0 + P2P1P0Ci$$
$$Co = C4 = G3 + P3\ C3 = G3 + P3G2 + P3P2G1 + P3P2P1G0 + P4P3P2P1Ci$$

Sumadores

- Una vez generados los acarreos se generan las salidas también en paralelo:

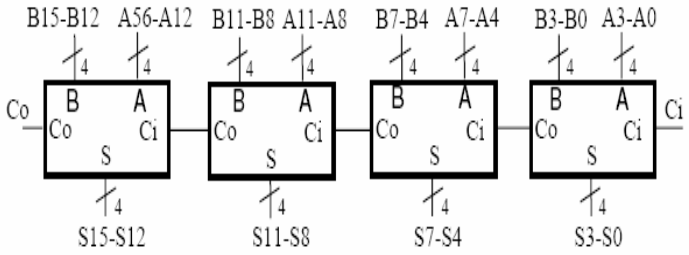
T3

$S0 = P0 \oplus Ci$
$Sj = Pj \oplus Cj$ para j = 0, j entre 1 y 3.



- El circuito se construye en paralelo a a partir de las ecuaciones de T1, T2 y T3.

Las ecuaciones que definen los acarreos son cada vez más grandes, luego no es rentable continuar aumentando el circuito. Una solución es poner sumadores "carry look-ahead" de 4 bits en serie.



Otra solución es llevar la estructura de "carry look-ahead" a más niveles, la expresión de Co también se puede poner de la forma:

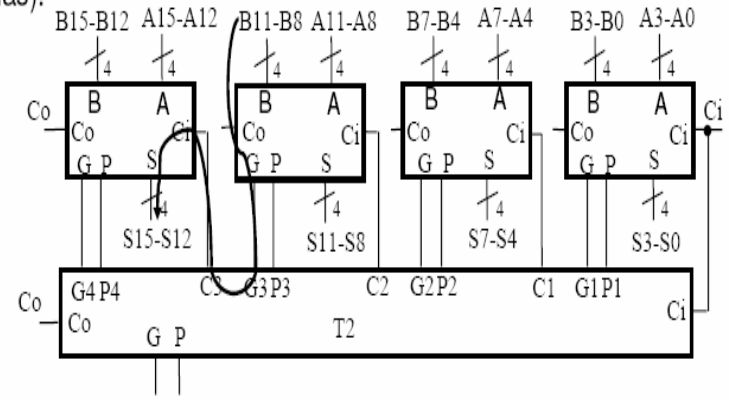
Co = G + P Ci, siendo

Co = G3 + P3G2 + P3P2G1 + P3P2P1G0 + P4P3P2P1Ci, luego

G = G3 + P3G2 + P3P2G1 + P3P2P1G0

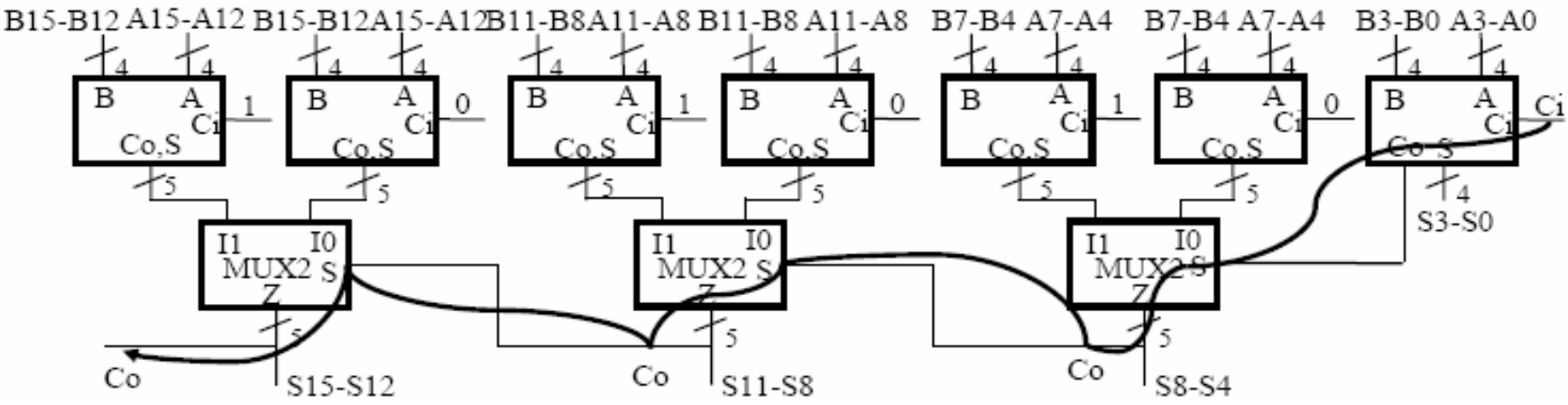
P = P4P3P2P1

donde G y P son el "carry-generate" y el "carry-propagate" de una suma de 4 bits. Si se añaden las ecuaciones de G y P a T2 se pueden realizar sumadores de 16 bits con esta estructura, que se puede llevar a más bits utilizando más niveles de T2 (64 bits, 4 grupos de 16 con un nivel más).



Sumadores

- Otra estructura de suma rápida es el "carry-select" basada en sumadores y multiplexores. Para una suma de M bits en M/N etapas de N bits, en cada etapa de suma, menos la primera, se calcula la suma en paralelo con acarreo de entrada a 0 ó a 1, y luego, cuando se obtiene el acarreo real, se seleccionan con los multiplexores las salidas correctas.
  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".



Descripcion en VHDL de un sumador completo

```
Library ieee;
Entity Sumador is
Port ( a, b, co: in bit ;s , cout : out bit);
End sumador;
Architecture caso1 of sumador is
Begin
  Process(a,b,cin)[poner las entradas]
    Begin
      If( a='1' and b='1' or
          a='1' and co='1' or
          b='1' and co='1')
       Then  cout <='1' ;
       Else   cout  <='0';
      End if;
      If (  a='0' and b='0' and c='1'  or
            a='0' and b='1' and c='0' or
            a='1' and b='0' and c='0' or
            a='1' and b='1' and c='1')
       Then s<='1';
       Else  s<='0';
     End if;
  End process;
End caso1;
```

- Tabla de verdad del sumador completo

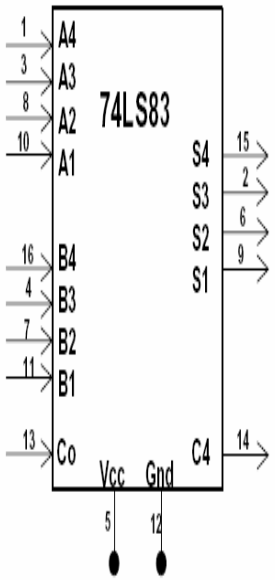
A B Co	C_1 S
0 0 0	0 0
0 0 1	0 1
0 1 0	0 1
0 1 1	1 0
1 0 0	0 1
1 0 1	1 0
1 1 0	1 0
1 1 1	1 1

Sumador completo de 4 bits

```
library ieee;
use.ieee.std_logic_ll64.all;
 use ieee.std_logic_signed.all;
-- Estas sentencias reconocen el tipo std_logic y las operaciones aritméticas sobre él.
entity addr4 is
port (A, B: in std_Iogic_vector(4 downto 1);
        Cin: in std_logic;
        S: out std_Iogic_vector(4 downto 1);
        Cout: out std_Iogic);
end addr4;
architecture comportamiento of addr4 is
signal sum: std_Iogic_vector(5 downto 1); -- Variable intermedia para acumular la suma
begin
-- Expande A a 5 bits por concatenación, y realiza la suma en 5 bits
Sum <= ('O' & A) + B + Cin;
S <= Sum(4 downto 1); - Los bits de suma son los cuatro bits menos significativos.
Cout <= Sum(5); -- El bit de acarreo es bit más significativo
end comportamiento;
```

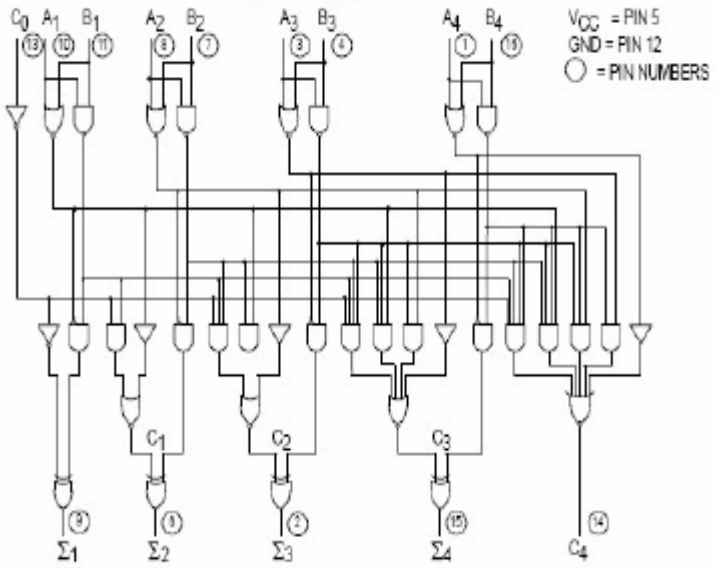
Sumador completo 74LS83

En la siguiente figura se muestra el diagrama funcional del 74LS83 (sumador binario de 4 bits)



	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
Active LOW	1	1	0	1	0	0	1	1	0	0	0	1	1	0

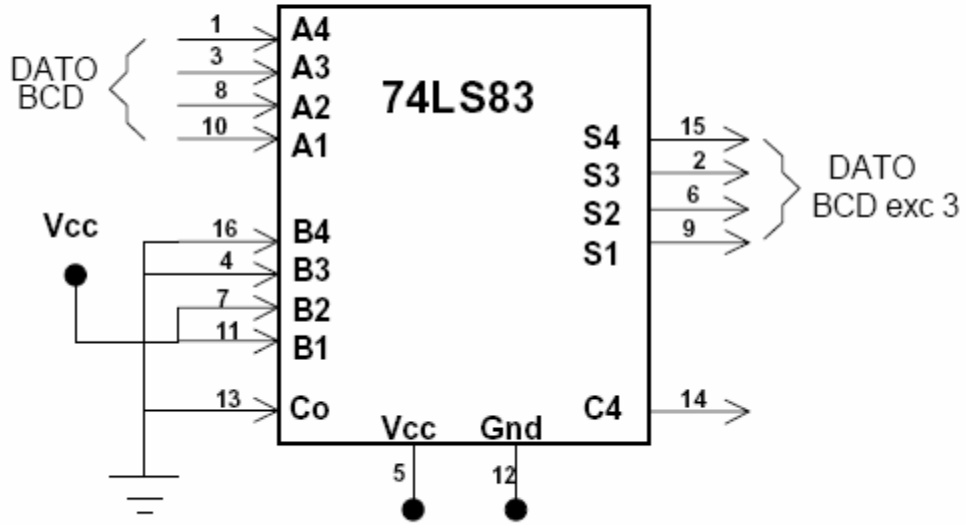
LOGIC DIAGRAM



Circuitos con sumadores

- 1

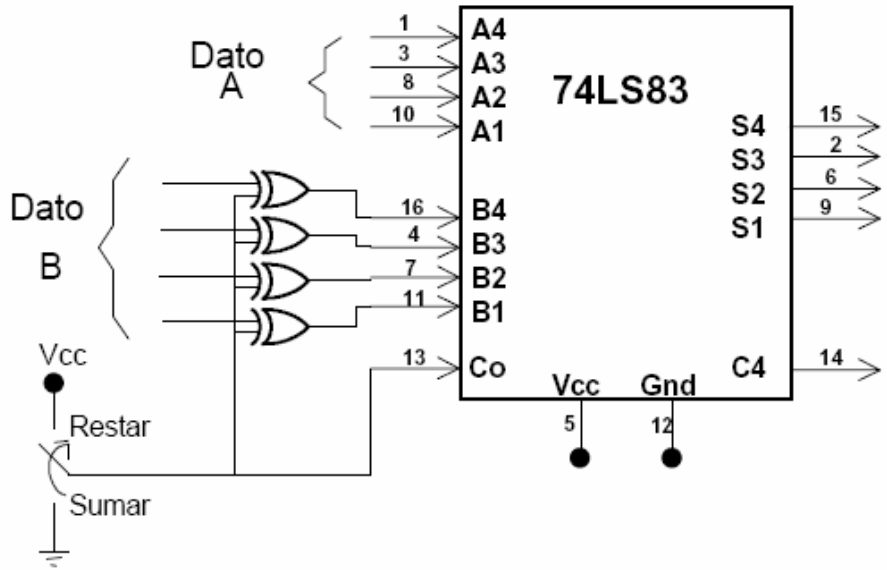
Convertidor BCD - BCD exceso tres.- Una aplicación directa de un sumador de cuatro bits como el 7483 es un convertidor de BCD a BCD exceso tres que se puede realizar sumado al dato de entrada A, una constante $B = 3 = (0011)_2$ como se muestra en la figura siguiente



Circuitos con sumadores

- 2

Sumador - Sustractor de 4 bits.- Utilizando el método expuesto en el segundo capítulo para realizar la resta A-B usando la suma de A + complemento a dos de B podemos realizar un sumador/restador binario de cuatro bits como sigue



Obsérvese que el bloque de cuatro puertas EXOR realiza el complemento a uno del dato B cuando el switch está en la posición de restar y Co le suma 1 a este complemento a uno de B para obtener su complemento a dos.

Sumador BCD.- El problema de sumar dos datos BCD usando un sumador binario (como el 7483) ocurre cuando el resultado de la suma es mayor que 9, ya que entonces el sumador binario producirá un resultado erróneo en BCD.

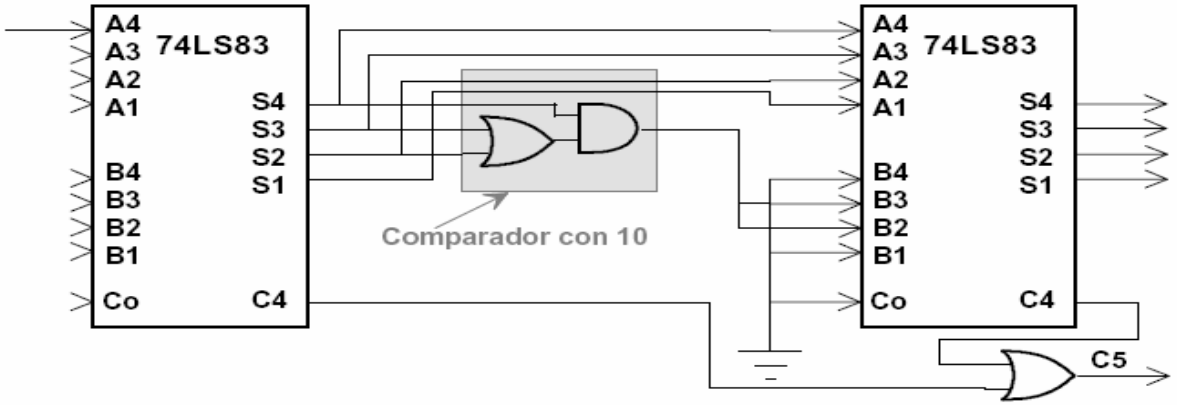
Por ejemplo, al sumar $4+7$ el resultado binario será $15=(1111)_2$ mientras que el resultado esperado en BCD es $15=(1\ 0101)_{BCD}$. Obsérvese que si al 15 producido por el sumador binario le sumáramos un 6: $15+6=21\ =(10101)_2$ ¡El resultado sería correcto en BCD!

Lo ilustrado en el caso de la suma $4+7$ se cumple en general, de manera que para realizar una suma de dos datos BCD se procederá de la siguiente manera:

i. Si el resultado es menor que 10 es correcto tanto en binario como en BCD
ii. Si el resultado es mayor o igual que 10, el resultado correcto en BCD es el resultado en binario más 6

Lo anterior se puede resolver usando un par de sumadores binarios: unao para realizar la primera suma y otro para realizar la corrección (sumar 6) en el caso necesario. Además se requiere un circuito lógico comparador para que active un indicador de que el resultado es mayor o igual que 10.

En la siguiente figura se muestra la implementación del sumador de dos dígitos BCD



Problemas propuestos

- 1• 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
- 2•Diseñar un circuito que realice la operación aritmética:
- O = 6X+Y
- 3•Realizar la suma de 2 números de 16 bits
- utilizando el menor número posible de sumadores completos ("74LS83").
- 4•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)