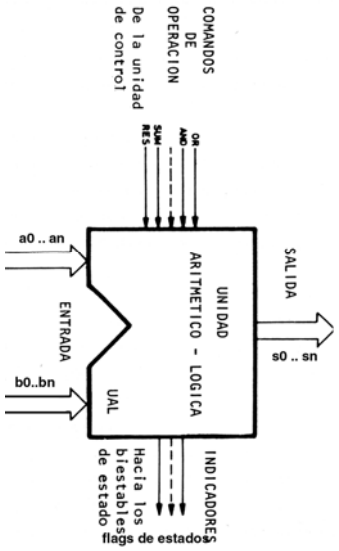
UNIDAD ARITMETICO-LOGICA

Conceptos

Unidad aritmético-lógica: Elemento que realiza las operaciones aritméticas y lógicas entre los datos

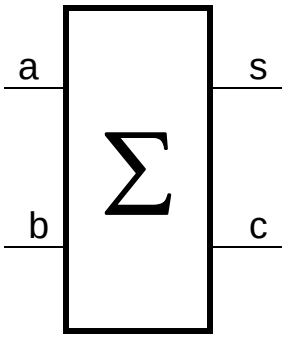


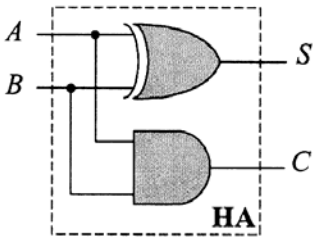
Operaciones típicas

- Sumar
- Restar
- Multiplicar
- Desplazamiento de registros
- Comparaciones

1. Sumadores y restadores

Semisumador:

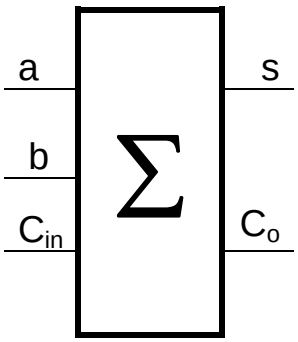




A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

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

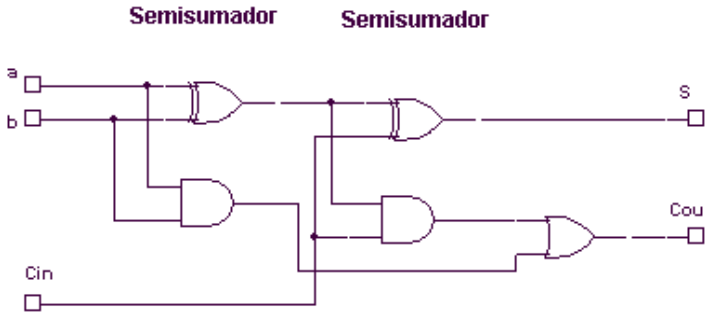
Sumador:



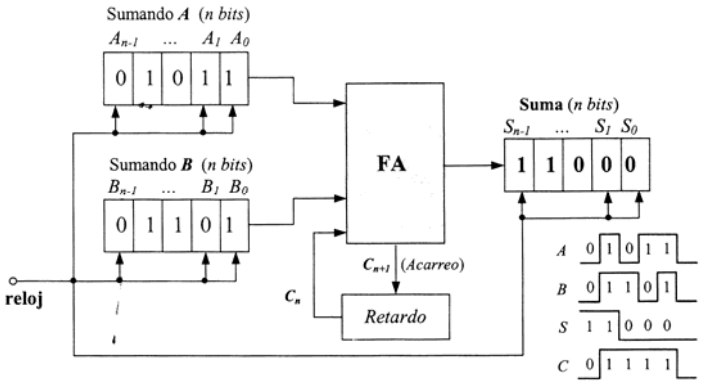
a	b	C_{in}	s	C_o
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

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

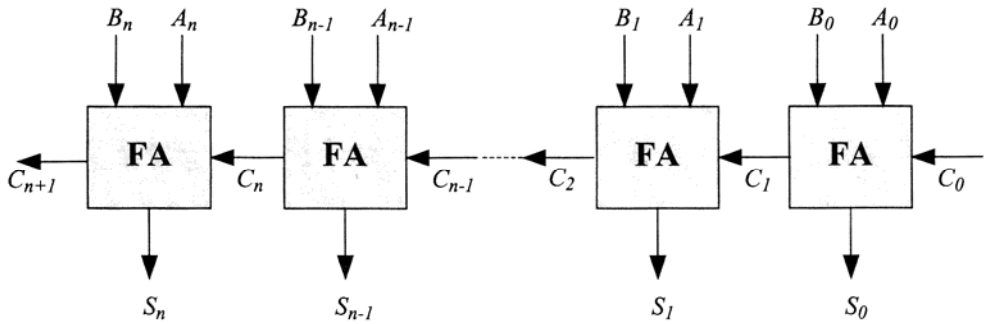
$$C_0 = \bar{a}bc + a\bar{b}c + ab\bar{c} + abc = ab + c(a \oplus b)$$



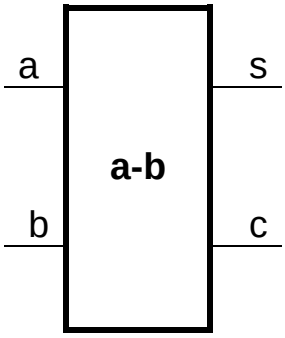
Sumador serie:

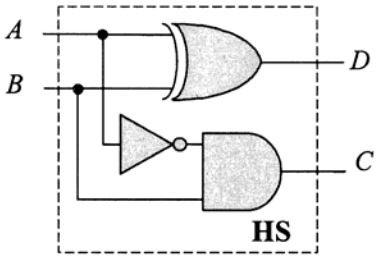


Sumador paralelo con propagación de arrastre:



Semirrestador:

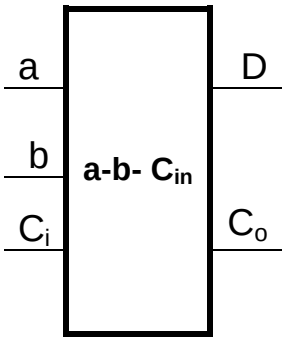




A	B	D	C
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

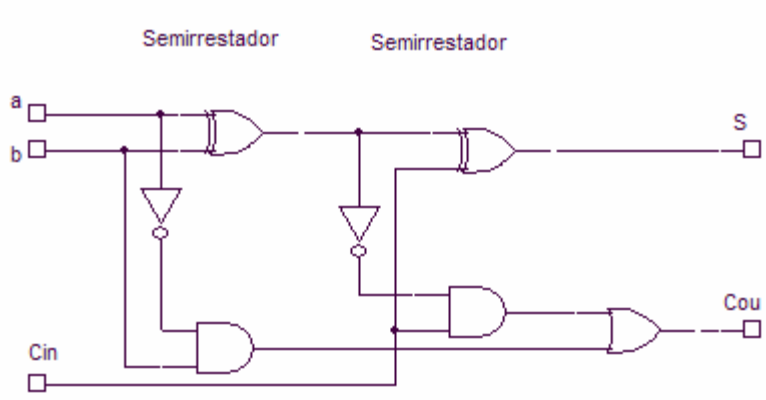
$$D = \overline{A}B + A\overline{B} = A \oplus B$$
$$C = \overline{A}B$$

Restador:

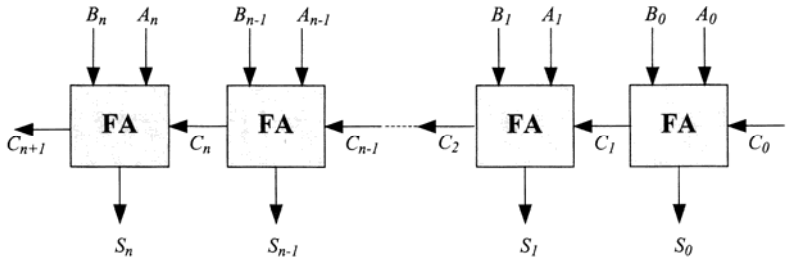


a	b	C_i	D	C_{i+1}
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

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

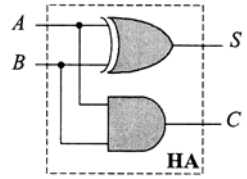


Restador paralelo con propagación de arrastre:



Convertir un semisumador en semirrestador

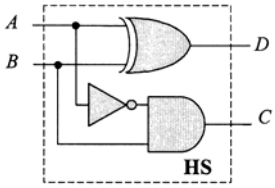
Semisumador:



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

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

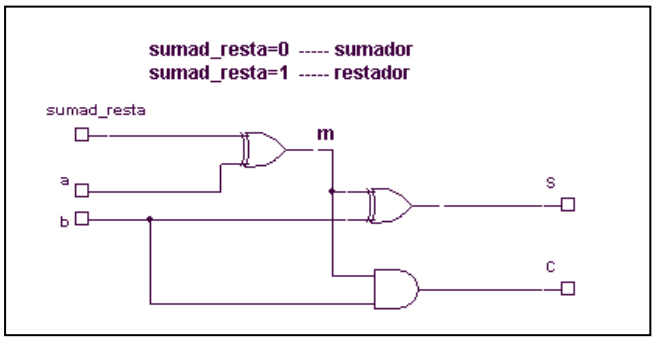
Semirrestador:



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

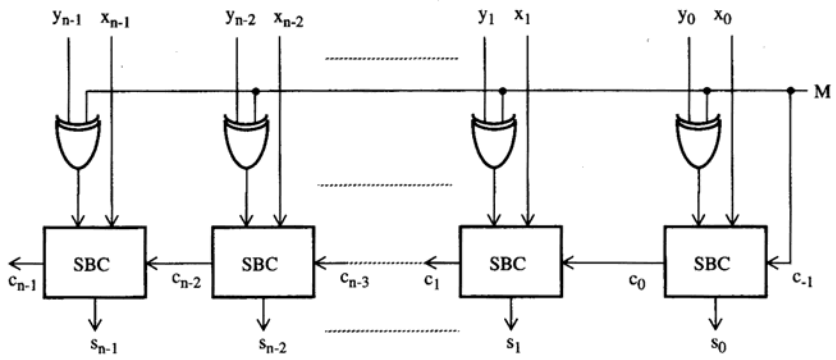
$$D = \overline{A}B + A\overline{B} = A \oplus B$$
$$C = \overline{A}B$$

Como se puede apreciar la diferencia consiste en que la entrada “A” en el semisumador es literal y en el semirestador 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



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

Sumador-Restador paralelo con propagación de arrastre:



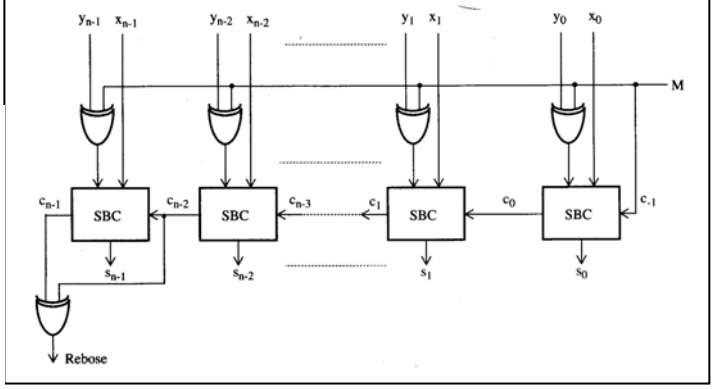
Detección del rebose en el sumador-restador con propagación de arrastre:

Rebose: Efecto que se produce cuando se realiza una operación aritmética entre dos o más números, cuyo resultado es mayor a la capacidad de representación del sistema, interpretando de esta manera un error en el resultado

Cuando se suman números con signo, la suma de dos números de diferente signo no produce nunca rebose. Sin embargo si se suman dos números del mismo signo, el resultado puede producir rebose. En la tabla siguiente se muestran las condiciones de rebose y el circuito detector del mismo.

x_{n-1}	y_{n-1}	s_{n-1}	Rebose (R)
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

Tabla 4.5: Condiciones de rebose



$$+\ \begin{array}{r} -1 \\ -1 \\ \hline -2 \end{array} \Rightarrow +\ \begin{array}{cc} 1 & 0 \\ 1 & 0 \\ \hline 0 & 0 \\ 1 & 0 \end{array}$$

$$R = C_n \overline{C_{n-1}} + \overline{C_n} C_{n-1} = C_n \oplus C_{n-1}$$

$$+\ \begin{array}{r} +1 \\ +1 \\ \hline +2 \end{array} \Rightarrow +\ \begin{array}{cc} 0 & 1 \\ 0 & 1 \\ \hline 1 & 0 \\ 0 & 1 \end{array}$$

2. Sumadores de alta velocidad

Características de los arrastres:

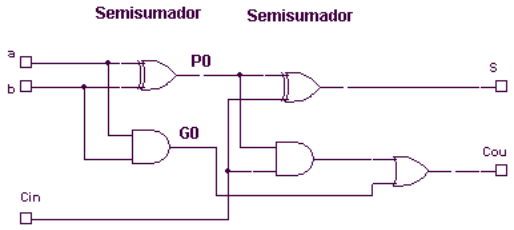
- Un arrastre se generará en la posición *i-ésima* si $(x_i+y_i)>1$
- Un arrastre se propagará de la posición *i-ésima* a la *(i+1)-ésima* si $(x_i+y_i)=1$

SECUENCIAS DE ARRASTRE

- De acuerdo a lo expuesto las secuencias de arrastre que se iniciarán simultáneamente en una suma será aquellas etapas cuyos valores de entradas sean $x_i=y_i=1$
- Continuarán a través de las etapas en las que $x_i \neq y_i$
- Pararán cuando lleguen a una etapa en la que $x_i=y_i$

	Inico de secuencia							Fin de secuencia					
Nº Secuencias			2					4		1		2	
x=	0	0	1	0	0	1	0	1	0	1	0	1	1
y=	0	1	1	0	1	0	1	1	0	1	0	0	1

Sumador paralelo con acarreo adelantado:



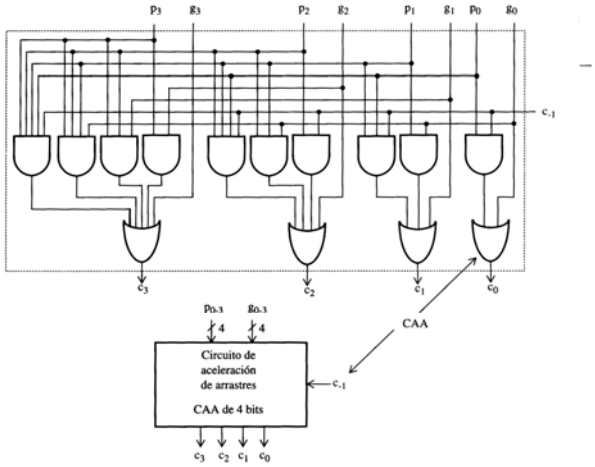
$$P_i = a_i \oplus b_i$$
$$G_i = a_i \square b_i$$

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

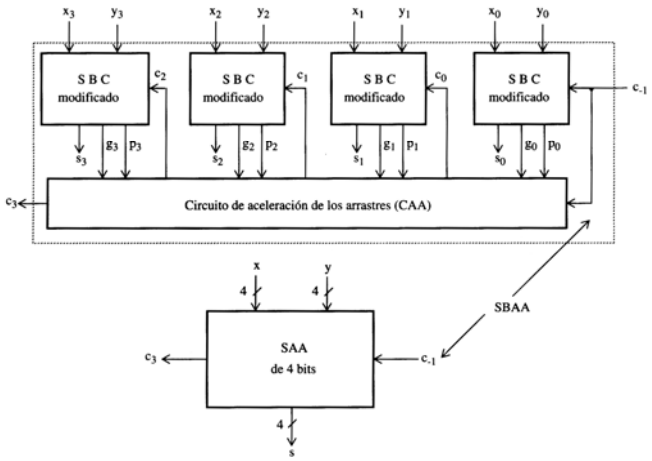
$$C_1 = G_0 + P_0C_0$$
$$C_2 = G_1 + P_1C_1 = G_1 + P_1(G_0 + P_0C_0) = G_1 + P_1G_0 + P_1P_0C_0$$
$$C_3 = G_2 + P_2C_2 = G_2 + P_2(G_1 + P_1G_0 + P_1P_0C_0) = G_2 + P_2G_1 + P_2P_1G_0 + P_2P_1P_0C_0$$
$$C_4 = G_3 + P_3C_3 = G_3 + P_3(G_2 + P_2G_1 + P_2P_1G_0 + P_2P_1P_0C_0) = G_3 + P_3G_2 + P_3P_2G_1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

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

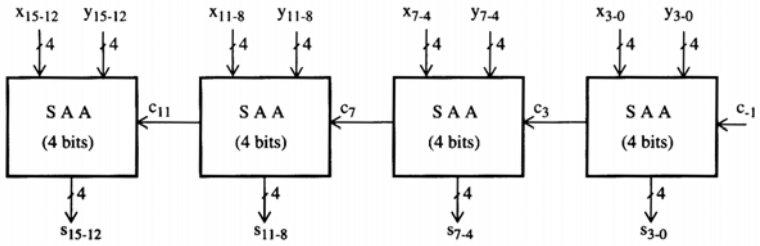
Circuito de aceleración de arrastres de 4 bits



Sumador con aceleración de los 4 bits



Sumador de 16 bits construido con 4 SAA de 4 bits

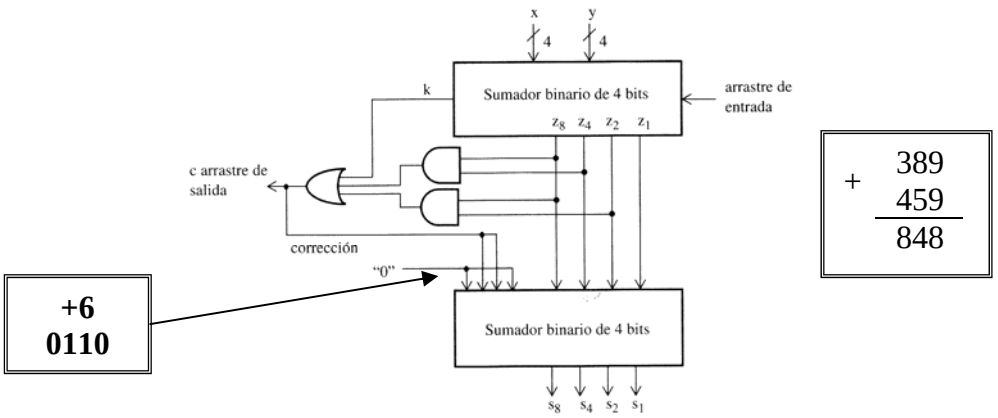


3. Sumadores en código BCD

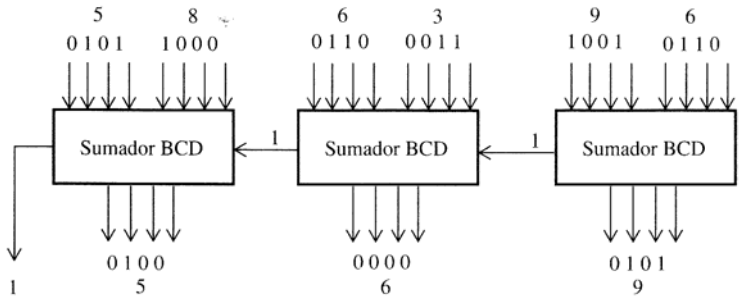
Se realizan como sumadores binarios naturales, añadiéndoles unos circuitos de corrección que garanticen la codificación de los resultados cuando el resultado de la suma sea mayor que 9.

CORRECCIÓN

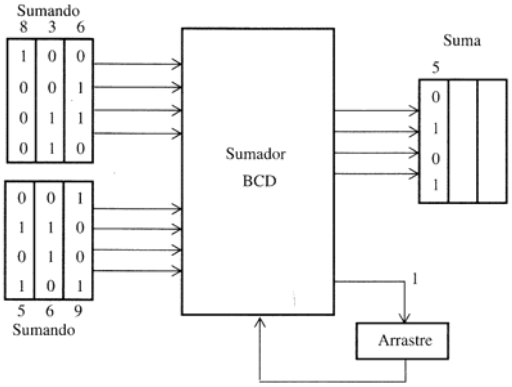
- Si hay arrastre de salida en el primer resultado (k=1). Esto sucede cuando la suma de los dígitos BCD es mayor de 15.
- Cuando la suma está comprendida entre 10 y 15 (c_1=1).



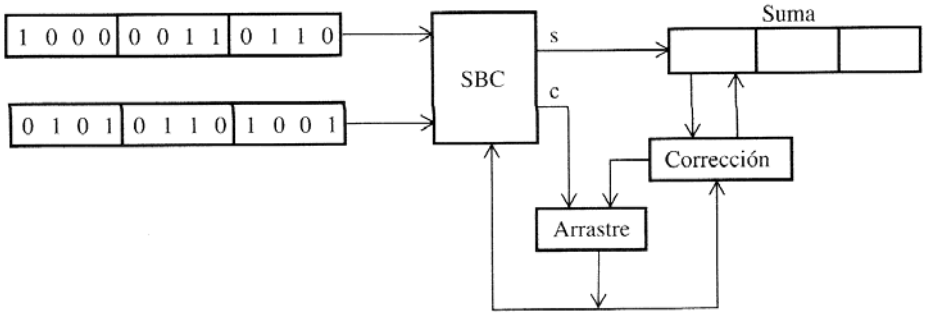
Sumador de dos dígitos en código BCD



Sumador BCD paralelo



Sumador BCD dígito-serie, bit-paralelo



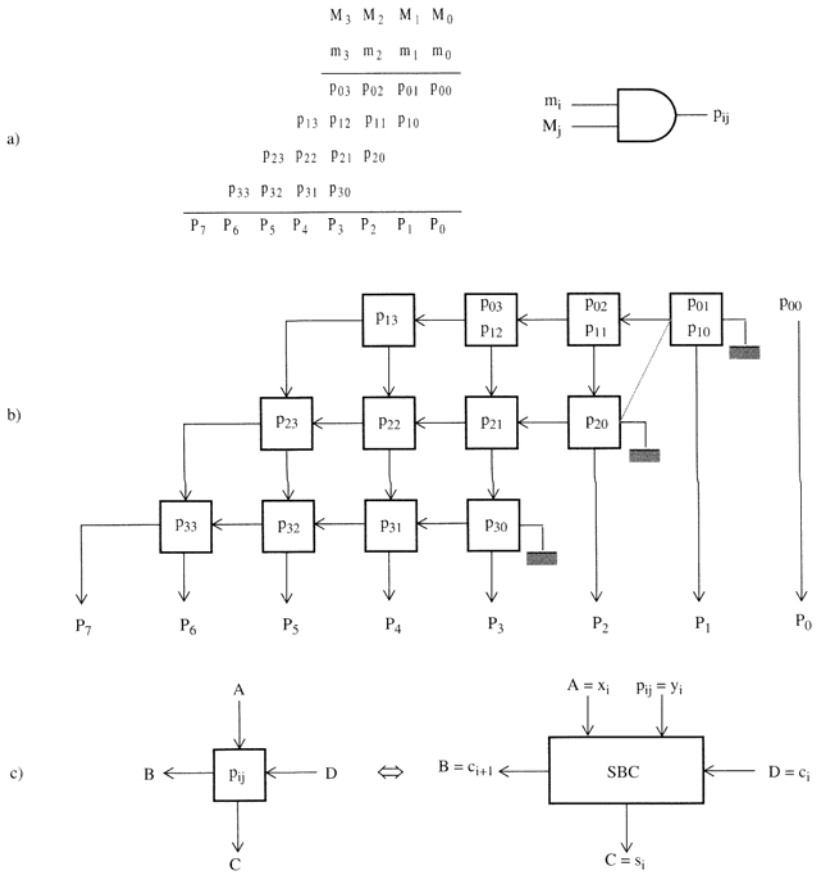
Sumador BCD dígito-serie, bit-serie

4. Multiplicadores binarios

Multiplicación de lápiz y papel

M =	1	0	0	1		Multiplicando (9)
m =	1	0	1	1		Multiplicador (11)
	1	0	0	1		
1	0	0	1			Productos parciales
0	0	0	0			
1	0	0	1			
1	1	0	0	0	1	1	Producto (99)

Multiplicación de dos números binarios sin signo

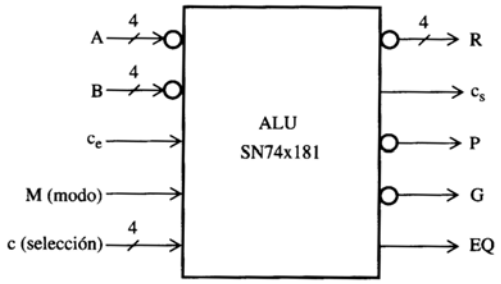


Multiplicador combinacional de 4 × 4 bits compuesto por 12 SBC's

5. Estructura de la unidad lógica aritmética

La estructura básica de una unidad lógica aritmética suele consistir en utilizar multiplexores con tantas entradas como operaciones queremos que realice dicha ALU y en cada entrada colocar el circuito que ha de realizar la operación correspondiente

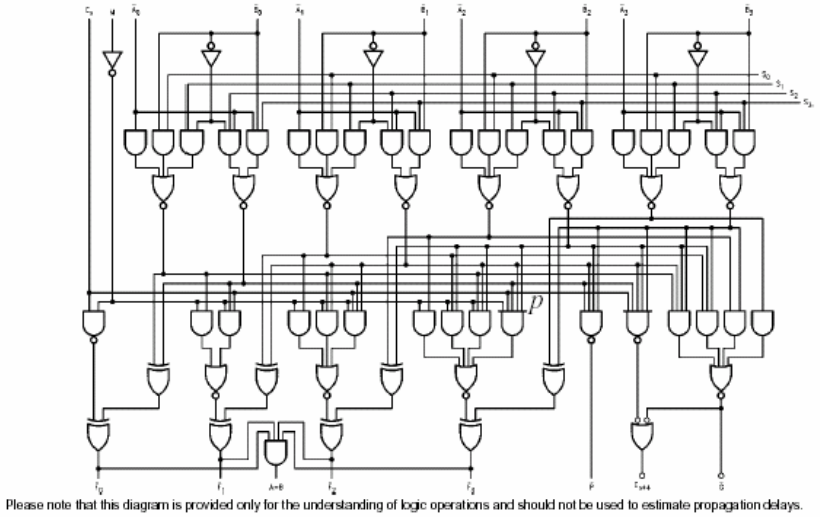
ALU SN74181



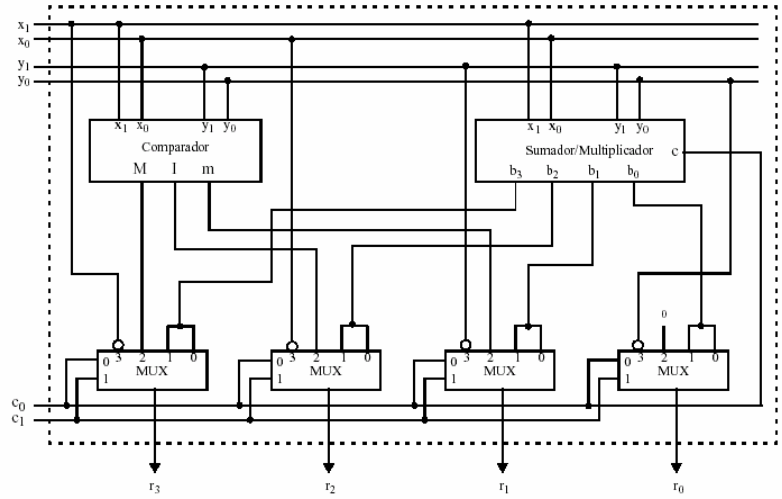
Selección $c_3c_2c_1c_0$	M = 1 Función lógica	M = 0 Función aritmética v(R) = s mod 16
0000	$R = \overline{A}$	$s = v(A) - 1 + c_e$
0001	$R = \overline{A} \vee \overline{B}$	$s = v(A \wedge B) - 1 + c_e$
0010	$R = \overline{A} \vee B$	$s = v(A \wedge \overline{B}) - 1 + c_e$
0011	R = 1111	$s = 1111 + c_e$
0100	$R = \overline{A} \wedge \overline{B}$	$s = v(A) + v(A \vee \overline{B}) + c_e$
0101	$R = \overline{B}$	$s = v(A \wedge B) + v(A \vee \overline{B}) + c_e$
0110	$R = A \oplus \overline{B}$	$s = v(A) - v(B) - 1 + c_e$
0111	$R = A \vee \overline{B}$	$s = v(A \vee \overline{B}) + c_e$
1000	$R = \overline{A} \wedge B$	$s = v(A) + v(A \vee B) + c_e$
1001	$R = A \oplus B$	$s = v(A) + v(B) + c_e$
1010	R = B	$s = v(A \wedge \overline{B}) + v(A \vee B) + c_e$
1011	$R = A \vee B$	$s = v(A \vee B) + c_e$
1100	R = 0000	$s = v(A) + v(A) + c_e$
1101	$R = A \wedge \overline{B}$	$s = v(A \wedge B) + v(A) + c_e$
1110	$R = A \wedge B$	$s = v(A \wedge \overline{B}) + v(A) + c_e$
1111	R = A	$s = v(A) + c_e$

Tabla 4.15: Funciones de la ALU SN74181

Logic Diagram



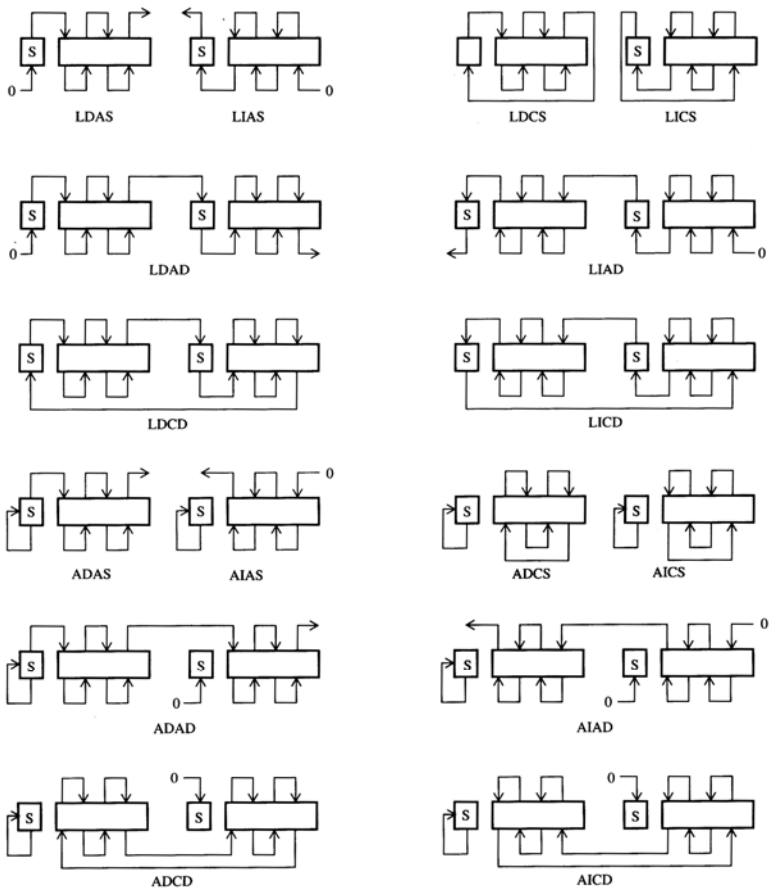
Please note that this diagram is provided only for the understanding of logic operations and should not be used to estimate propagation delays.



Operaciones de desplazamiento

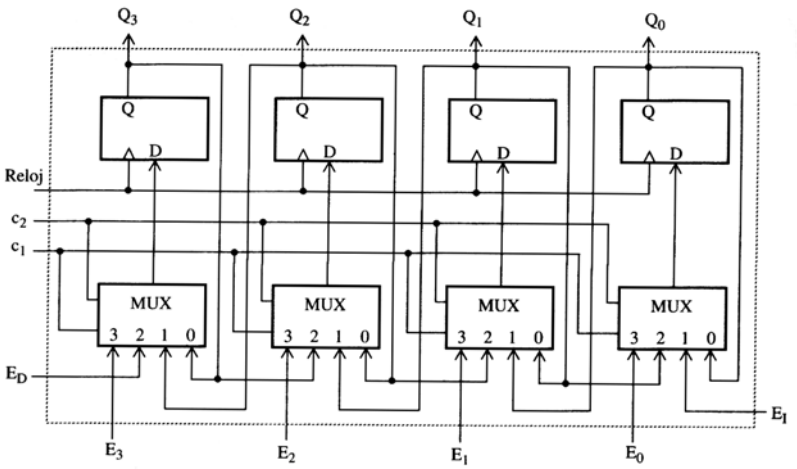
Clasificación de los desplazamientos

- Tratamiento del bit de signo
  - Aritméticos (A) → No afecta al signo
  - Lógicos (L) → Interviene el signo
- Sentido de desplazamiento
  - Dcha (D)
  - Izda (I)
- Tratami. Bits que rebosan
  - Abierto (A) → Se pierde le bit de rebose
  - Cerrado (C) → Interviene el bit de rebose
- Longitud de registros
  - Simples (S) → Registro único
  - Dobles (D) → Pareja de registros



Ejemplo: Diseñar un registro de desplazamiento de 4 bits que sea capaz de realizar los desplazamientos indicados en la siguiente tabla de la verdad.

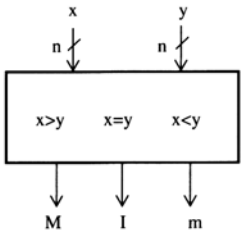
Operación	c_2	c_1
NOP	0	0
LIAS	0	1
LDAS	1	0
CARGA	1	1



Comparadores

Son elementos que en base a introducirle dos números de n bits (x , y) entregan a su salida mediante tres señales el valor de la comparación M (x>y), I (x=y) y m (x<y).

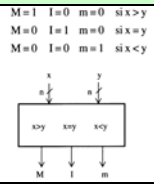
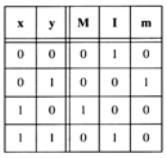
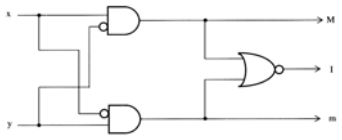
$$M=1 \quad I=0 \quad m=0 \quad si\ x>y$$
$$M=0 \quad I=1 \quad m=0 \quad si\ x=y$$
$$M=0 \quad I=0 \quad m=1 \quad si\ x<y$$



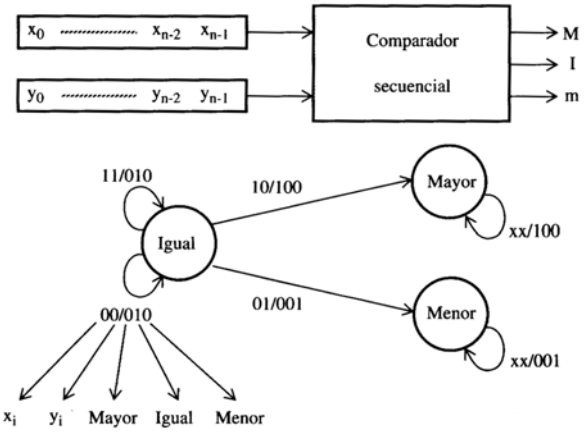
Tipos de comparadores

- Circuito combinacional
- Circuito secuencial
- Utilizando un sumador

Circuito combinacional

Símbolo	Tabla de la verdad	Ecuaciones	Circuito lógico
		$M = M_{n-1} + I_{n-1} M_{n-2} + I_{n-1} I_{n-2} M_{n-3} + ...+ I_{n-1} I_{n-2}...I_1 M_0$ $I = I_{n-1} I_{n-2}.....I_1 I_0$ $m = m_{n-1} + I_{n-1} m_{n-2} + I_{n-1} I_{n-2} m_{n-3} + ...+ I_{n-1} I_{n-2}...I_1 m_0$	

Circuito secuencial



Utilizando un sumador

Consiste en restar los dos números a comparar y analizar los bits de estados generados por la ALU.
La resta se efectúa sumando x más el complemento a 2 de y. x+(comp. a 2).
Los bits analizados son: el carry (C) o llevada, el cero (Z) , el de signo (N) y el rebosamiento o overflow (V).

Posibilidades

- 2 num positivos sin signo
  - $x = y \rightarrow Z = 1$
  - $x > y \rightarrow Z = 0$ y $C = 1$
  - $x < y \rightarrow Z = 0$ y $C = 0$
- 2 num con signo en complemento a 2
  - $x = y \rightarrow Z = 1$
  - $x > y \rightarrow Z = 0$ y $N \oplus V = 0$
  - $x < y \rightarrow N \oplus V = 1$

Problemas de UNIDAD ARITMETICO-LOGICA:

Problema 2ª semana de Junio de 2000

Se desea realizar una Unidad Aritmético Lógica (UAL) como la mostrada en la figura, con dos entradas de datos *X* (x_1x_0) e *Y* (y_1y_0), una entrada de control *C* (c_1c_0) y una salida de datos *R* ($r_3r_2r_1r_0$). El funcionamiento de la UAL viene descrito por la siguiente tabla:

Señal de control	Operación
$c_1c_0 = 0\ 0$: suma	R = X + Y
$c_1c_0 = 0\ 1$: producto	R = X * Y
$c_1c_0 = 1\ 0$: comparación	Si X > Y entonces R = 1 0 0 0 Si X = Y entonces R = 0 1 0 0 Si X < Y entonces R = 0 0 1 0
$c_1c_0 = 1\ 1$: sacar $\overline{X}$, $\overline{Y}$	R = $\overline{X}$, $\overline{Y}$ ($r_3 = \overline{x}_1$, $r_2 = \overline{x}_0$, $r_1 = \overline{y}_1$, $r_0 = \overline{y}_0$)

Tabla 2000-2-1: Tabla de funcionamiento de la UAL

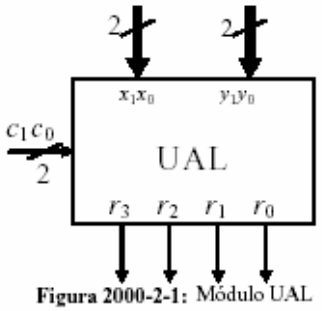


Figura 2000-2-1: Módulo UAL

Para resolver el problema, seguir obligatoriamente los siguientes apartados:

a) (1 punto) Diseñar un comparador de números de dos bits utilizando únicamente comparadores de números de un bit y puertas lógicas.

b) (2 puntos) Diseñar un sumador/multiplicador de dos números de 2 bits utilizando únicamente un módulo de memoria ROM. ¿Cuál es el tamaño necesario para este módulo de memoria ROM? Indique claramente el significado de cada una de sus entradas y cada una de sus salidas. Escriba **todo** el contenido de la memoria ROM en forma de tabla.

c) (1 punto) Utilizando únicamente los módulos diseñados en los apartados anteriores, los módulos combinacionales necesarios y puertas lógicas, diseñe la UAL pedida.

Solución

a) Para comparar X e Y es necesario comparar cada bit de uno de ellos con el bit de igual peso del otro. Para hacer estas comparaciones utilizamos los comparadores de 1 bit propuestos. Si el resultado de la comparación de x_i e y_i es: M_i, I_i y m_i, se cumple que:

$M_i = 1$ si $x_i > y_i$
$I_i = 1$ si $x_i = y_i$
$m_i = 1$ si $x_i < y_i$

A partir de M_i, I_i y m_i se generan las salidas M, I y m del comparador de 2 bits mediante las siguientes funciones booleanas:

M = M1 + I1 M0
I = I1 I0
m = m1 + I1 m0

La explicación de estas expresiones es inmediata:

x > y si ($x_1 > y_1$) **o** (($x_1 = y_1$) **y** ($x_0 > y_0$))
x = y si ($x_1 = y_1$) **y** ($x_0 = y_0$)
x < y si ($x_1 < y_1$) **o** (($x_1 = y_1$) **y** ($x_0 < y_0$))

La Figura 2000-2-2 muestra el circuito lógico pedido en este apartado obtenido a partir de las expresiones anteriores.

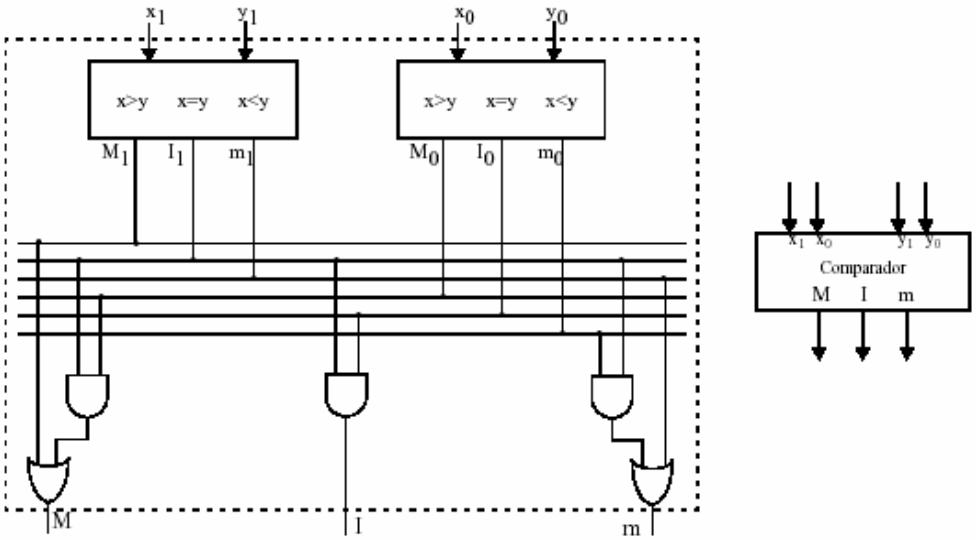


Figura 2000-2-2: Circuito lógico de un comparador de dos números de 2bits

b) Para calcular el tamaño de la memoria ROM necesaria hay que saber el número entradas y salidas del circuito.

- Entradas. Este circuito tiene **5 entradas**: 2 para la entrada de datos X (x_1x_0), 2 para la entrada de datos Y (y_1y_0) y una entrada de control (c) para la selección de la operación a realizar por el circuito. Esta última entrada de control va a tener el siguiente significado:
  c = 0: Suma
  c = 1: Producto
- Salidas: Este circuito tiene 4 salidas. El resultado de la suma de dos números de dos bits tiene tan sólo 3 bits pero, para el resultado del producto de dos números de dos bits se requieren 4 bits. Como el circuito a diseñar tiene que poder hacer ambas operaciones, su salida ha de tener, necesariamente, 4 bits.

Por tanto, el tamaño de la memoria ROM necesario para implementar el sumador es de: 2^5 palabras × 4 bits/palabra. El significado de cada una de sus entradas y salidas se muestra claramente en la Figura 2000-2-3. Finalmente, en la Tabla 2000-2-2 se muestra el contenido que debería tener la memoria ROM.

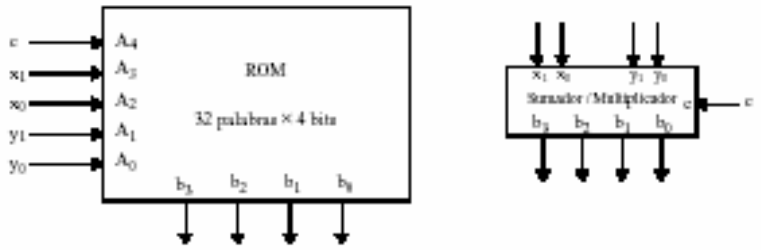


Figura 2000-2-3: Sumador/Multiplicador diseñado con memoria ROM

Dirección memoria c x_1 x_0 y_1 y_0	Contenido	Dirección memoria c x_1 x_0 y_1 y_0	Contenido
A_4 A_3 A_2 A_1 A_0	b_3 b_2 b_1 b_0	A_4 A_3 A_2 A_1 A_0	b_3 b_2 b_1 b_0
0 0 0 0 0	0 0 0 0	1 0 0 0 0	0 0 0 0
0 0 0 0 1	0 0 0 1	1 0 0 0 1	0 0 0 0
0 0 0 1 0	0 0 1 0	1 0 0 1 0	0 0 0 0
0 0 0 1 1	0 0 1 1	1 0 0 1 1	0 0 0 0
0 0 1 0 0	0 0 0 1	1 0 1 0 0	0 0 0 0
0 0 1 0 1	0 0 1 0	1 0 1 0 1	0 0 0 1
0 0 1 1 0	0 0 1 1	1 0 1 1 0	0 0 1 0
0 0 1 1 1	0 1 0 0	1 0 1 1 1	0 0 1 1
0 1 0 0 0	0 0 1 0	1 1 0 0 0	0 0 0 0
0 1 0 0 1	0 0 1 1	1 1 0 0 1	0 0 1 0
0 1 0 1 0	0 1 0 0	1 1 0 1 0	0 1 0 0
0 1 0 1 1	0 1 0 1	1 1 0 1 1	0 1 1 0
0 1 1 0 0	0 0 1 1	1 1 1 0 0	0 0 0 0
0 1 1 0 1	0 1 0 0	1 1 1 0 1	0 0 1 1
0 1 1 1 0	0 1 0 1	1 1 1 1 0	0 1 1 0
0 1 1 1 1	0 1 1 0	1 1 1 1 1	1 0 0 1

Tabla 2000-2-2: Contenido de la memoria ROM

c) En la Figura 2000-2-4 se muestra el diseño de la UAL pedida en el problema donde se han utilizado 4 multiplexores para seleccionar el valor adecuado a colocar en la salida, gobernados por las entradas de control c_1 y c_0. También la entrada de control de la UAL c_0 se utiliza para seleccionar la operación a realizar por el Sumador/Multiplicador.

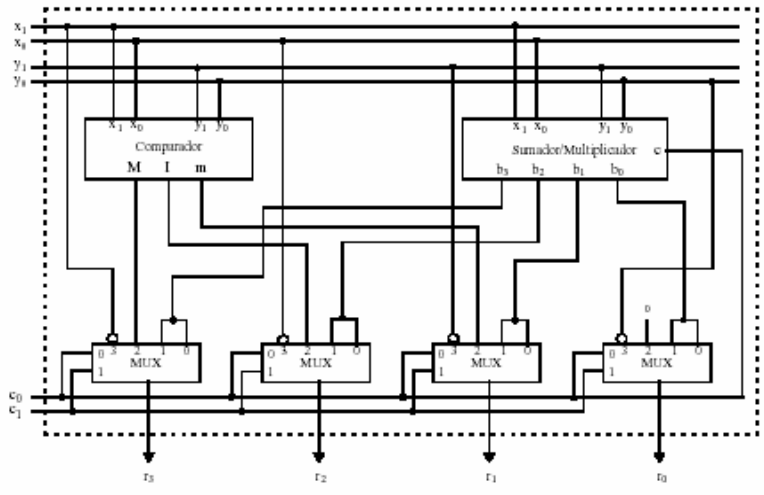


Figura 2000-2-4: UAL pedida en el problema

2001 Septiembre

2.- Decir cuál es el retardo que se obtiene al calcular la suma de dos números de 4 bits cada uno:

I. Si el sumador se ha implementado usando lógica combinacional con 2 niveles y cada puerta tarda 5 μs en generar el resultado, el retardo es de 20 μs.

II.Si el sumador se ha implementado usando 4 SBC's (Sumador Binario Completo) y cada uno de ellos tarda 10 (s en generar sus salidas si y ci, el retardo es de 40 μs.

A) I: sí, II: sí. B) I: sí, II: no. C) I: no, II: sí. D) I: no, II: no.

3.- Se desea comparar el valor de dos números binarios, uno de 5 bits: $X= x_4x_3x_2x_1x_0$, y otro de 4 bits: $Y=y_3y_2y_1y_0$. Utilizando comparadores de un bit se compara cada pareja de bits, x_i con y_i, i=0..3, obteniéndose: M_i ($x_i>y_i$), I_i ($x_i=y_i$) y m_i ($x_i<y_i$). Entonces:

I. La función lógica M (X>Y) es $M = x_4 + M_3 + I_3M_2 + I_3I_2M_1 + I_3I_2I_1M_0$.

II. La función lógica I (X=Y) es $I = \overline{x_4}I_0I_1I_2I_3$

A) I: sí, II: sí. B) I: sí, II: no. C) I: no, II: sí. D) I: no, II: no.

2000 Exámenes Septiembre de 2000

4.- ¿Cuántos SBC de 1 bit harían falta para construir un sumador binario serie capaz de sumar dos números binarios de *n* bits?

a) n B) 2 n C) $\log_2 n$ D) Ninguna de las anteriores

2000 Exámenes de Junio -1ª semana

4.- Se desea diseñar un circuito sumador/restador de dos números de cuatro bits cada uno, $x_3x_2x_1x_0$ e $y_3y_2y_1y_0$, con una señal de control M adicional para indicar la operación a realizar.

I. Con una memoria ROM de 2^8 palabras con 5 bits por palabra se podría construir.

II. Con una memoria ROM de 2^{10} palabras con 9 bits por palabra se podría construir.

A) I: sí, II: sí. B) I: sí, II: no. C) I: no, II: sí. D) I: no, II: no.

Junio 2003 Reserva

7.- Si se suman 0011100000110101 y 0100100110010011 en un sumador binario paralelo con propagación de arrastres:

I. El número de secuencias de arrastre que comienza simultáneamente es 3.

II. La longitud de la secuencia de arrastre más grande es 3.

A) I: sí, II: sí. B) I: sí, II: no. C) I: no, II: sí. D) I: no, II: no.

PROBLEMAS

En la sección de problemas se han puesto los correspondientes a los resueltos en el libro de problemas:

4.40.....Junio del 2001 – 1ª semana