ESTRUCTURA Y TECNOLOGIA DE COMPUTADORES II

TEMA 4: *Unidad Aritmético-Lógica*

SOLUCION A LOS PROBLEMAS PROPUESTOS EN EXAMEN

Curso 2008-2009
Jose Manuel Díaz Martínez
Tutor de la asignatura ETC-II

CONTENIDO

SOLUCION PROBLEMA 4.1

La solución a este problema se encuentra en el fichero `probT4_01.pdf`.

SOLUCION PROBLEMA 4.2

La solución a este problema se encuentra en el fichero `probT4_02.pdf`.

SOLUCION PROBLEMA 4.3

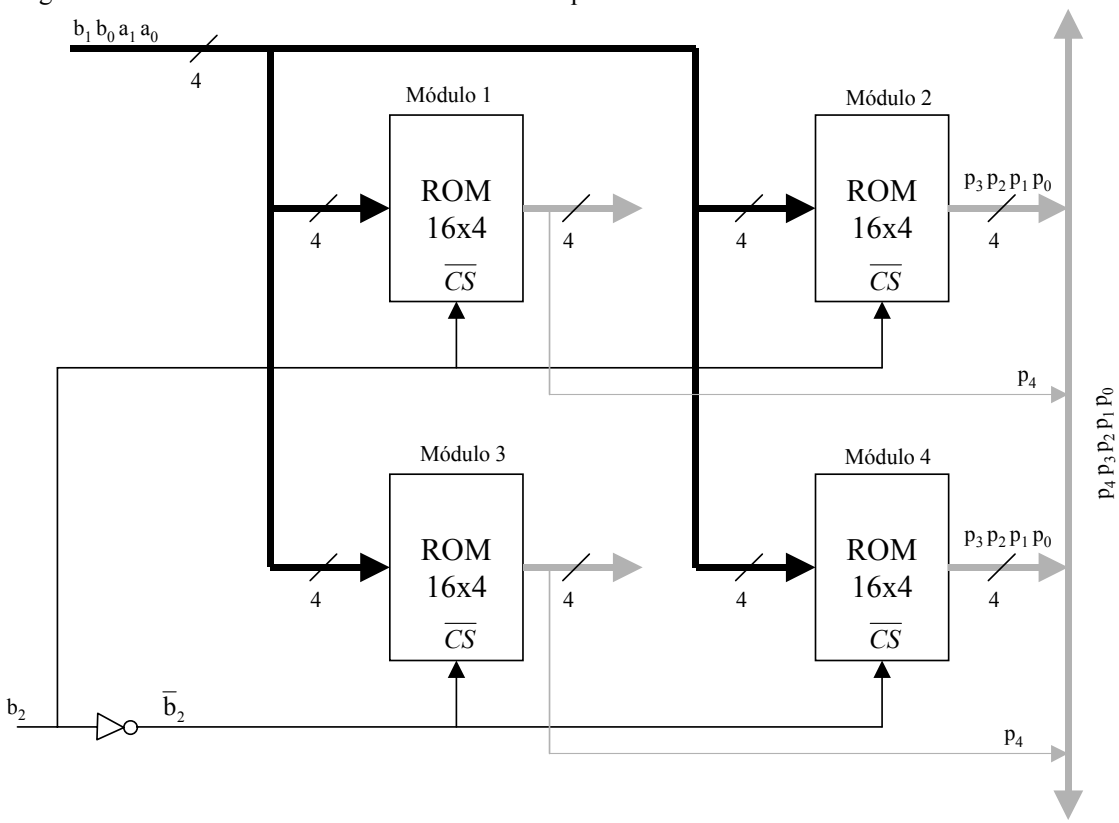
a) Para conocer los módulos de ROM necesarios, es necesario darse cuenta que el producto de un número binario $B(b_2b_1b_0)$ de 3 bits por otro número binario $A(A_1A_0)$ de 2 bits da como resultado un número binario $P(p_4p_3p_2p_1p_0)$ de de 5 bits.

$$\begin{array}{ccccc} & & b_2 & b_1 & b_0 \\ & x & & a_1 & a_0 \\ \hline & & \bullet & \bullet & \bullet \\ & \bullet & \bullet & \bullet & \\ \hline p_4 & p_3 & p_2 & p_1 & p_0 \end{array}$$

Por lo tanto en la ROM se deben almacenar todos los resultados posibles que pueden resultar al multiplicar un número de 3 bits por otro de 2 bits. El número de resultados posibles es 2^5. Por otro lado ya se ha visto que el resultado debe de tener 5 bits. Luego se necesitaría un módulo ROM de capacidad

$$C=2^5 \text{ pal x } 5 \text{ bits/pal}$$

Ahora bien los módulos de ROM de que se disponen tienen una capacidad de 2^4 pal x 4 bits/pal.
Luego para conseguir almacenar 2^5 (32) resultados necesitamos 2 módulos de este tipo ($2^4 + 2^4$)
Por otro lado como cada resultado tiene 5 bits necesitamos otros dos módulos: en uno se almacenará el bit más significativo (p_4) de cada resultado quedando tres bits ($r_2r_1r_0$) sin utilizar (x) y en el otro los cuatro bits restantes ($p_3p_2p_1p_0$).
Luego el **número de módulos necesarios es 4**. El esquema sería :



Comentar que con el bit b_2 se selecciona el par de módulos donde se encuentra almacenado el resultado. Y con los bits $b_1b_0a_1a_0$ ya se determina la posición de ROM que se va a leer.

b) A continuación se muestran la tablas de los contenidos de cada uno de los módulos.

Dirección					**Contenidos de los módulos ROM**							
b_2	**b_1**	**b_0**	**a_1**	**a_0**	**r_2**	**r_1**	**r_0**	**p_4**	**p_3**	**p_2**	**p_1**	**p_0**
0	0	0	0	0	x	x	x	0	0	0	0	0
0	0	0	0	1	x	x	x	0	0	0	0	0
0	0	0	1	0	x	x	x	0	0	0	0	0
0	0	0	1	1	x	x	x	0	0	0	0	0
0	0	1	0	0	x	x	x	0	0	0	0	0
0	0	1	0	1	x	x	x	0	0	0	0	1
0	0	1	1	0	x	x	x	0	0	0	1	0
0	0	1	1	1	x	x	x	0	0	0	1	1
0	1	0	0	0	x	x	x	0	0	0	0	0
0	1	0	0	1	x	x	x	0	0	0	1	0
0	1	0	1	0	x	x	x	0	0	1	0	0
0	1	0	1	1	x	x	x	0	0	1	1	0
0	1	1	0	0	x	x	x	0	0	0	0	0
0	1	1	0	1	x	x	x	0	0	0	1	1
0	1	1	1	0	x	x	x	0	0	1	1	0
0	1	1	1	1	x	x	x	0	1	0	0	1
					Módulo 1				Módulo 2			
1	0	0	0	0	x	x	x	0	0	0	0	0
1	0	0	0	1	x	x	x	0	0	1	0	0
1	0	0	1	0	x	x	x	0	1	0	0	0
1	0	0	1	1	x	x	x	0	0	1	1	0
1	0	1	0	0	x	x	x	0	0	0	0	0
1	0	1	0	1	x	x	x	0	0	1	0	1
1	0	1	1	0	x	x	x	0	1	0	1	0
1	0	1	1	1	x	x	x	0	1	1	1	1
1	1	0	0	0	x	x	x	0	0	0	0	0
1	1	0	0	1	x	x	x	0	0	1	1	0
1	1	0	1	0	x	x	x	0	1	1	0	0
1	1	0	1	1	x	x	x	1	0	0	1	0
1	1	1	0	0	x	x	x	0	0	0	0	0
1	1	1	0	1	x	x	x	0	0	1	1	1
1	1	1	1	0	x	x	x	0	1	1	1	0
1	1	1	1	1	x	x	x	1	0	1	0	1
					Módulo 3				Módulo 4			

c) El aprovechamiento de la memoria vendrá dado por:

$$R = \frac{bits\ \ utilizados}{bits\ \ totales} \cdot 100 = \frac{2^5 x5}{4 \cdot (2^4 x4)} \cdot 100 = \frac{2^5 \cdot 5}{2^8} \cdot 100 = \frac{5}{8} \cdot 100 = 62.5\%$$

Una forma de mejorar el aprovechamiento sería fijarse en la tabla de contenidos de cada uno de los 4 módulos del apartado b). En ella se observa que sólo en dos casos el bit p_4 es 1, y son cuando se multiplica 6x3=18 y cuando se multiplica 7 x 3=21. La función lógica del bit p_4 es:

$$p_4 = b_2 b_1 \bar{b}_0 a_1 a_0 + b_2 b_1 b_0 a_1 a_0 = b_2 b_1 a_1 a_0$$

Que puede generarse fácilmente con una puerta AND. Con lo que sólo se necesitarían 2 módulos de ROM con un aprovechamiento R=100 %.

SOLUCION PROBLEMA 4.4

DATOS

- Dos números X e Y de 12 bits representados en binario puro:
- X= 0000 1100 0010
- Y= 0001 0111 0001

Para calcular la suma en BCD de estos dos números binarios puros X e Y, hay que realizar los siguientes pasos:

- Pasar X de binario a decimal.

  $X_{10}= 2^7+2^6+2^1 =128 + 64 +2=194$

- Pasar Y de binario a decimal.

  $Y_{10}= 2^8+2^6+2^5+2^4+2^0 =256 + 64 +32+16+1=369$

- Sumar ambos números en decimal.

  $X_{10}+Y_{10}=S_{10}= 563$

- Y finalmente pasar cada cifra de S_{10} de decimal a binario con lo que se obtiene la suma en BCD S_{BCD}

  $S_{10}= 563 \rightarrow$ **S_{BCD}= 0101 0110 0011**

SOLUCION PROBLEMA 4.5

El producto P $[p_{2n-1},...., p_0]$ de dos números binarios X $(x_{n-1},...,x_0)$ e $Y(y_{n-1},...,y_0)$ de n bits cada uno, posee un total de (n+n) bits, es decir, 2n bits. Por lo tanto la memoria que se requiere para implementar este multiplicador requiere un bus de direcciones de 2n bits $(x_{n-1},...,x_0,y_{n-1},...,y_0)$, para direccionar 2n posibles productos P con que tendrán un tamaño de 2n bits $[p_{2n-1},...., p_0]$ cada uno. Es decir, la capacidad de memoria necesaria es:

$$C=2^{2n} \text{ palabras x } 2n \text{ bits/palabra}$$

Puesto que se tienen módulos de capacidad $C_0=2^n$ palabras x n bits/palabra, el número N de módulos de capacidad C_0 a utiliza sería:

$$N = \frac{C_T}{C_0} = \frac{2^{2n}}{2^n} \times \frac{2n}{n} = 2^n \times 2 = 2^{n+1} \, módulos$$

SOLUCION PROBLEMA 4.6

En el libro de teoría se explica que cuando se implementa un multiplicador binario mediante el *algoritmo de lápiz y papel*, para dos números X de n bits e Y de m bits, con n>m, se requieren n·m puertas AND y n·(m-1) sumadores binarios completos (SBC). Puesto que nos dicen en el enunciado que hay que multiplicar dos números de n=m 8 bits . Entonces el número de puertas AND necesarias será 8·8= **64 puertas AND**.

SOLUCION PROBLEMA 4.7

Del libro de teoría se comprueba en un sumador binario completo SBC que las expresiones para el bit de suma s_i y para el bit de acarreo c_i, vienen dado por las siguientes expresiones:

$$s_i = x_i \oplus y_i \oplus c_{i-1}$$
$$c_i = x_i y_i + (x_i \oplus y_i) c_{i-1}$$

Para el caso i=1 se obtiene:

$$s_1 = x_1 \oplus y_1 \oplus c_0$$
$$c_1 = x_1 y_1 + (x_1 \oplus y_1) c_0$$

Mientras que para el caso i=0 se obtiene

$$s_0 = x_0 \oplus y_0 \oplus c_{-1}$$
$$c_1 = x_0 y_0 + (x_0 \oplus y_0) c_{-1}$$

SOLUCION PROBLEMA 4.8

Afirmación I: Se tienen dos números de n=4 bits y m= 3 bits, luego el módulo de memoria ROM que se necesitaría tendría una capacidad de

$$C=2^{n+m} \text{ palabras x } (n+m) \text{ bits/palabra}$$

Es decir, $C= 2^7$ palabras x 7 bits/palabras, luego la afirmación I es **verdadera.**

Afirmación II: Por el mismo razonamiento de la solución del problema 4.6, se necesitarían $n \cdot m = 4 \cdot 3 = 12$ puertas AND y $(n \cdot (m-1)) = (4 \cdot (3-1)) = 8$ sumadores binarios completos. Luego la afirmación II es **verdadera**.

SOLUCION PROBLEMA 4.9

De acuerdo con lo explicado en el libro de teoría sobre las características de un sumador binario serie.

Afirmación I: El tiempo de calculo para sumar dos números de n bits viene dado por

$$t=n \cdot (\delta + \Delta)$$

siendo δ el tiempo en sumar 1 bits y Δ el retardo del elemento de memoria. Por lo tanto, se observa que el tiempo de calculo depende del número de bits n a sumar. Por lo tanto, la afirmación es **Falsa**.

Afirmación II: La estructura (complejidad) de un sumador binario serie es siempre la misma, independientemente del número de bits a sumar. Luego la afirmación es **Falsa**.

SOLUCION PROBLEMA 4.10

DATOS

- Multiplicador binario X·Y
- X de 3 bits, Y de 6 bits
- Representados en binario puro sin signo
- Utilizando:
  - 2 memorias ROM de 64 palabras x 6 bits/palabra.
  - 2 sumadores binarios paralelos de 3 bits cada uno.

a) Sea el número de 6 bits $Y(y_5y_4y_3y_2y_1y_0)$ y el número de 3 bits $X(x_2x_1x_0)$ el producto de ambos números es un número de 9 bits $P(p_8p_7p_6p_5p_4p_3p_2p_1p_0)$. La multiplicación se puede representar esquemáticamente como se muestra a continuación.

			y_5	y_4	y_3	y_2	y_1	y_0
		x				x_2	x_1	x_0
			#	#	#	•	•	•
		#	#	#	•	•	•	
	#	#	#	•	•	•		
			p_{05}	p_{04}	p_{03}	p_{02}	p_{01}	p_{00}
p_{15}	p_{14}	p_{13}	p_{12}	p_{11}	p_{10}			
p_8	p_7	p_6	p_5	p_4	p_3	p_2	p_1	p_0

Si se denota como Y^1a $(y_5y_4y_3)$ y como Y^0 a$(y_2y_1y_0)$, en el esquema anterior se puede observar que:

- Con <u>una ROM (ROM1)</u> de 64 pal x 6 bits/pal se pueden almacenar los resultados del producto binario

$$Y^0 \text{ x } X=(p_{05}\ p_{04}\ p_{03}\ p_{02}\ p_{01}\ p_{00}).$$

- Con <u>otra ROM (ROM2)</u> de 64 pal x 6 bits/pal se pueden almacenar los resultados del producto binario de

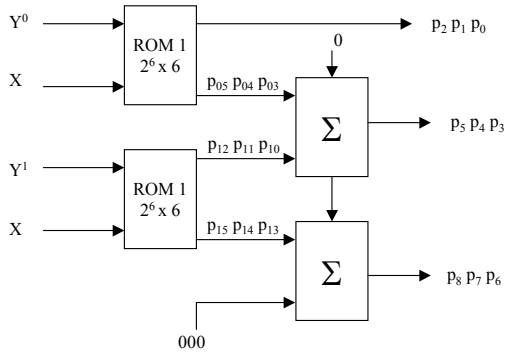
$$Y^1 \text{ x } X=(p_{15}\ p_{14}\ p_{13}\ p_{12}\ p_{11}\ p_{10}).$$

- Los bits $p_2\ p_1\ p_0$ del producto final son directamente: $p_{02}\ p_{01}\ p_{00}$

- Los bits $p_5\ p_4\ p_3$ del producto final son la suma de $(p_{05}\ p_{04}\ p_{03})+ (p_{12}\ p_{11}\ p_{10})$, y además se generará un bit de acarreo de salida. Para implementar esta operación se necesita <u>un sumador binario paralelo</u> de 3 bits (sumador 1).

- Los bits $p_8\ p_7\ p_6$ del producto final son la suma de $(p_{15}\ p_{14}\ p_{13})+ (0\ 0\ 0)+$ acarreo de salida del sumador 1, y además se generará un bit de acarreo de salida. Para implementar esta operación se necesita <u>un sumador binario paralelo</u> de 3 bits (sumador 2).

-Luego el diseño del sumador con los elementos propuestos es el que se muestra en la figura



SOLUCION PROBLEMA 4.11

De acuerdo con las expresiones generales para el resultado de la suma y el acarreo de salida de un SBC deducidas en el libro de teoría(ver problema 4.7):
Afirmación I: **Falsa**.
Afirmación II: V**erdadera**.

SOLUCION PROBLEMA 4.12

La solución a este problema se encuentra en el fichero `probT4_12.pdf`.

SOLUCION PROBLEMA 4.13

De acuerdo con las expresiones generales deducidas en el libro de teoría para un comparador de números de tamaño n>1 implementado mediante un circuito combinacional:

$$M = M_{n-1} + I_{n-1}M_{n-2} + ... + I_{n-1}I_{n-2}..I_1M_0$$
$$I = I_{n-1}I_{n-2}...I_1I_0$$
$$m = m_{n-1} + I_{n-1}m_{n-2} + ... + I_{n-1}I_{n-2}..I_1m_0$$

En el caso que se propone n=4,

$$\boxed{\begin{aligned} M &= M_3 + I_3M_2 + I_3I_2M_1 + I_3I_2I_1M_0 \\ I &= I_3I_2I_1I_0 \\ m &= m_3 + I_3m_2 + I_3I_2m_1 + I_3I_2I_1m_0 \end{aligned}}$$

Por lo tanto:

Afirmación I: **Falsa**
Afirmación II: **Falsa**

SOLUCION PROBLEMA 4.14

De acuerdo con las expresiones generales deducidas en el libro de teoría para el semisumador binario SSB de dos números binarios de un bit., x e y

$$s = \bar{x}y + x\bar{y} = x \oplus y$$
$$c = xy$$

Por lo tanto:
Afirmación I: Desarrollando la expresión lógica dada en el enunciado s=

$$s = (x+y)\overline{xy} = (x+y)(\bar{x}+\bar{y}) = x\bar{x} + \bar{x}y + \bar{y}y + x\bar{y} = \bar{x}y + x\bar{y} = x \oplus y$$

Luego la afirmación es **verdadera**.

Afirmación II: **Falsa**

SOLUCION PROBLEMA 4.15

DATOS

- Dos números binarios de 12 bits representados en código BCD
- X_{BCD}= 0101 1000 1001 Y_{BCD}= 0011 0100 0111

El método más rápido y sencillo para resolver este problema, es pasar cada cifra BCD codificada en binario a decimal y realizar la suma:
Así:
X_{BCD}= 0101 1000 1001 = 5 8 9
Y_{BCD}= 0011 0100 0111 = 3 4 7

La suma $S_{BCD}=X_{BCD} + Y_{BCD}$= 9 3 6 = **1001 0011 0110**

SOLUCION PROBLEMA 4.16

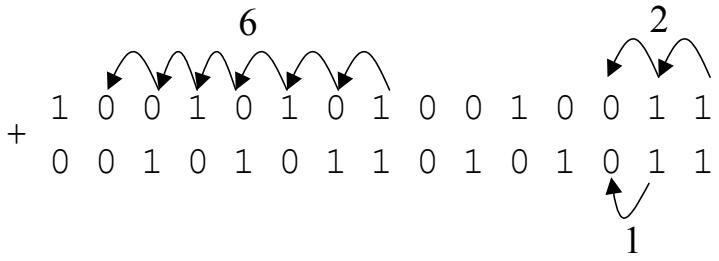
La solución a este problema se encuentra en el fichero `probT4_16.pdf`.

SOLUCION PROBLEMA 4.17

El número de secuencias de arrastre viene dada por el número de veces que el bit $x_i=y_i=1$, en la figura se puede apreciar que se generarán **3 secuencias de arrastre**:

```
  100 1010 1001 0011
+ 001 0101 1010 1011
```

Por otro lado dichas secuencias de arrastre se propagarán hasta que $x_i=y_i=0$. En la Figura inferior se puede observar como **la longitud máxima de las secuencias de arrastre es 6.**



SOLUCION PROBLEMA 4.18

Para expresar la suma de un número binario X=[X_7 X_6 X_5 X_4 X_3 X_2 X_1 X_0] de 8 bits y de otro número binario Y=[Y_3 Y_2 Y_1 Y_0] de 4 bits, se requieren de 9 bits [C S_7 S_6 S_5 S_4 S_3 S_2 S_1 S_0] (el resultado S de 8 bits más un bit de acarreo C).
Por lo tanto para implementar todas los resultados posibles, se necesita una ROM de $2^{(8+4)}$ palabras, cada palabra debe poseer 9 bits para almacenar el resultado y el acarreo. Luego la capacidad de la memoria ROM necesaria es:

$$C = 2^{12}\ palabras\ \ x\ \ 9\ (bits / palabra)$$

El bus de direcciones tendrá 12 bits con el siguiente significado: [X_7 X_6 X_5 X_4 X_3 X_2 X_1 X_0 Y_3 Y_2 Y_1 Y_0], es decir, los operandos X e Y.
El bus de datos tendrá 9 bits con el siguiente significado: [C S_7 S_6 S_5 S_4 S_3 S_2 S_1 S_0], es decir, el bits de acarreo y el resultado de la suma.

SOLUCION PROBLEMA 4.19

Afirmación I
Si se quiere implementar con una ROM un multiplicador binario de dos números binarios sin signo, X=[$X_2X_1X_0$] de n= 3 bit ee Y=[Y_1Y_0] de m=2 bits. La memoria ROM deberá tener tantas palabras como como combinaciones de X ·Y se puedan producir, es decir, $2^{(n+m)} = 2^{(3+2)}=2^5$ palabras.
Por otro lado la memoria ROM debe tener tantos bit por palabra como bits tenga el resultado del producto de X·Y, en general, el producto P de un número de n bits por otro de m bits posee n+m bits, es decir, en nuestro caso, 3+2+1=6 bits/palabra.
Luego la capacidad de la memoria ROM necesaria es:

$$C = 2^{5}\ palabras\ \ x\ \ 5\ (bits / palabra)$$

Luego la afirmación I, es **FALSA**.

Afirmación II:
En el libro de teoría se explica que cuando se implementa un multiplicador binario mediante el *algoritmo de lápiz y papel*, para dos números X de n bits e Y de m bits, con n>m, se requieren n·m puertas AND y n·(m-1) sumadores binarios completos (SBC).
En nuestro caso n=3, m=2, se requerirían 3·2= 6 puertas AND y 3(2-1)= 3 SBC

Luego la afirmación II, es **VERDADERA**.

SOLUCION PROBLEMA 4.20

Por el mismo razonamiento realizado para comprobar la afirmación I, del problema 4.19 con n=24 y m=16, la memoria ROM que se necesitaría para implementar el multiplicador debe tener una capacidad de

$$C_T = 2^{(24+16)}\ x\ (24+16) = 2^{40}\ palabras\ \ x\ 40\ bits / palabra$$

Afirmación I: La capacidad total de los módulos ROM disponible es C=4·(2^8 x 16). Como $C<C_T$, la afirmación es **FALSA**.

Afirmación II: La capacidad total de los módulos ROM disponible es C=6·(2^8 x 16) s. Como $C<C_T$, la afirmación es **FALSA**.

SOLUCION PROBLEMA 4.21

En el libro de teoría se explica que cuando se implementa un multiplicador binario mediante el *algoritmo de lápiz y papel*, para dos números X de n bits e Y de m bits, con n>m, se requieren n·m puertas AND y n·(m-1) sumadores binarios completos (SBC).
En nuestro caso n=4, m=4, se requerirían 4·4= 16 puertas AND y 4(4-1)= 12 SBC

Afirmación I : Puesto que un sumador binario con aceleración de arrastre de 4 bits es equivalente a 4 SBC. Se tendrían 4+8 =12 SBC, que junto con las 16 puertas AND disponibles permiten construir el multiplicador binario pedido. **VERDADERA**.

Afirmación II: Es **FALSA**, ya que sólo nos dan 8 SBC, faltarían 4, y no nos dan las 16 puertas AND necesarias.

SOLUCION PROBLEMA 4.22

De acuerdo con la sección 4.2.2 del libro de teoría dedicada a los sumadores con anticipación de arrastre, la expresión general del bit de arrastre de la etapa c_i , en función de los bits de propagación de arrastre (p_i, $p_{i-1},...,p_0$) y de los bits de generación de arrastre (g_i, $g_{i-1},...,g_0$), es:

$$c_i = g_i + p_i g_{i-1} + p_i p_{i-1} g_{i-2} + ... + p_i p_{i-1} .... p_1 g_0 + p_i p_{i-1} .... p_0 c_{-1}$$

La expresión que dan en el enunciado es:

$$g_2 + p_2 g_1 + p_2 p_1 g_0 + p_2 p_1 p_0 c_{-1}$$

que de acuerdo con la expresión general se corresponde con $\mathbf{c_2}$, luego la respuesta correcta es la **B**.

SOLUCION PROBLEMA 4.23

El circuito que se quiere implementar tiene 2 entradas de 4 bits cada una para los operandos X [$x_3x_2x_1x_0$] e Y [$y_3y_2y_1y_0$]. Y una entrada adicional M de 1 bit para indicar si la operación es de suma o de resta. Luego el número total de bits de entrada de este circuito es 9.
Por otro lado, este circuito debe tener dos salidas, una de 4 bits para indicar el resultado de la resta o de la suma y un bits de acarreo. Luego el número total de bits de salida del circuito es 5.

La tabla de verdad o tabla de funcionamiento del circuito constaría de 9 bits de entradas y de 5 bits de salida. Esta tabla puede ser implementada por una memoria ROM que disponga de una capacidad mínima :

$$C_T = 2^9 \text{ palabras x 5 bits/palabra}$$

Afirmación I: **FALSA.** La capacidad de la ROM es inferior a C_T .
Afirmación II: **VERDADERA.** La capacidad de la ROM es superior a C_T .

SOLUCION PROBLEMA 4.24

DATOS

- Dos números binarios de 16 bits representados en código BCD
- X_{BCD}=0011 1001 0101 0100 Y_{BCD}= 0011 1001 0000 0110

El método más rápido y sencillo para resolver este problema, es pasar cada cifra BCD codificada en binario a decimal y realizar la suma:
Así:
X_{BCD}= 0011 1001 0101 0100 = 3 9 5 4
Y_{BCD}= 0011 1001 0000 0110 = 3 9 0 6

La suma $S_{BCD}=X_{BCD} + Y_{BCD}$= 7 8 6 0 = **0111 1000 0110 0000**

SOLUCION PROBLEMA 4.25

El circuito que se quiere implementar tiene 1 entrada de 3 bits para el operando X [$x_2x_1x_0$] y una entrada de 4 bits para el operando Y [$y_3y_2y_1y_0$]. Luego el número total de bits de entrada de este circuito es 7.
Por otro lado, este circuito debe tener dos salidas, de 1 bits cada una, M (si x>y) e I(x=y). Luego el número total de bits de salida del circuito es 2.

La tabla de verdad o tabla de funcionamiento del circuito constaría de 7 bits de entrada y de 2 bits de salida. Esta tabla puede ser implementada por una memoria ROM que disponga de una capacidad mínima :

$$C_T = 2^7 \text{ palabras x 2 bits/palabra}$$

Afirmación I: **VERDADERA.** La capacidad de la ROM es superior a C_T.
Afirmación II: **VERDADERA.** La capacidad de la ROM es superior a C_T.

SOLUCION PROBLEMA 4.26

La solución a este problema se encuentra en el fichero `probT4_26.pdf`.

SOLUCION PROBLEMA 4.27

La expresión lógica $(x+y)\cdot\overline{xy}$ se puede desarrollar de la siguiente forma:

$$(x+y)\cdot\overline{xy} = (x+y)\cdot(\bar{x}+\bar{y}) = y\bar{x}+\bar{y}x = x \oplus y$$

Que se corresponde con la expresión lógica del bit de suma s= $x \oplus y$.
Por lo tanto la respuesta correcta es la **A**

SOLUCION PROBLEMA 4.28

De acuerdo con lo que se explica en el libro de teoría sobre las características de un sumador binario serie, este dispositivo solamente requiere de **1 SBC de 1bit**.

SOLUCION PROBLEMA 4.29

El circuito que se quiere implementar tiene 2 entradas de 8 bits cada una para los operandos X e Y. Además el circuito tiene 5 entradas de 1 bit, C, e_d, e_I, d_1 y d_0. Luego el número total de bits de entrada de este circuito es 21.
Por otro lado, este circuito debe tener 1 salidas de 8 bits para indicar el resultado. Luego el número total de bits de salida del circuito es 8.

La tabla de verdad o tabla de funcionamiento del circuito constaría de 21 bits de entradas y de 8 bits de salida. Esta tabla puede ser implementada por una memoria ROM que disponga de una capacidad mínima :

$$C_T = 2^{21} \text{ palabras x 8 bits/palabra}$$

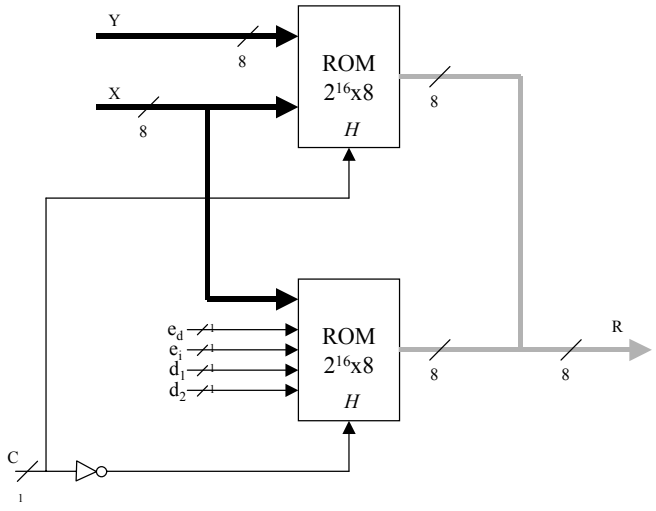
Afirmación I: La memoria total entre las dos ROM propuestas en la afirmación es $C=2^{17}$ palabras x8 bits/palabra. Por lo que podría pensarse que esta afirmación es falsa. Pero si se lee atentamente el enunciado se observa que se trata de dos circuitos independientes, por un lado un comparador y por otro un desplazador, que se puede implementar con las dos memorias ROM y con una puerta lógica NOT :

Comparador : Tiene 2 entradas de 8 bits cada una para los operandos X e Y.. Luego el número total de bits de entrada de este circuito es 16.
Por otro lado, este circuito debe tener 1 salidas de 8 bits para indicar el resultado. Luego el número total de bits de salida del circuito es 8.
La tabla de verdad o tabla de funcionamiento del circuito comparador constaría de 16 bits de entradas y de 8 bits de salida. Esta tabla puede ser implementada por una memoria ROM que disponga de una capacidad mínima :

$$C = 2^{16} \text{ palabras x 8 bits/palabra}$$

Desplazador: Tiene 1 entradas de 8 bits para el operando X y 4 entradas de 1 bit e_d, e_I, d_1 y d_0 Luego el número total de bits de entrada de este circuito es 12.
Por otro lado, este circuito debe tener 1 salidas de 8 bits para indicar el resultado. Luego el número total de bits de salida del circuito es 8.
La tabla de verdad o tabla de funcionamiento del circuito comparador constaría de 12 bits de entradas y de 8 bits de salida. Esta tabla puede ser implementada por una memoria ROM que disponga de una capacidad mínima :

$$C = 2^{12} \text{ palabras x 8 bits/palabra}$$



Con el inversor se utiliza para habilitar uno de los dos módulos de memoria ROM, de acuerdo con el valor de la señal C=1(Comparador) y C=0 (Desplazador). Se tiene por tanto el esquema que se muestra en la Figura

La afirmación I es **VERDADERA**

Afirmación II: La memoria total de la ROM que proponen es inferior a C_T, luego la afirmación es **FALSA**.

SOLUCION PROBLEMA 4.30

De acuerdo con lo que se explica en el libro de teoría sobre las características de un sumador binario completo SBC, este se puede construir de al menos dos modos:

1) Se puede construir con dos niveles de puertas lógicas, un nivel de puertas AND y un nivel de puertas OR.
2) También, se puede construir con 2 SSB y una puerta OR.

Afirmación I: **VERDADERA**.
Afirmación II: **VERDADERA**. Ya que si se construye del modo1, puesto que un SSB requiere de dos niveles de puertas AND-OR, habría que atravesar un total de 4 niveles como máximo, el retardo sería mayor que si se construye el SBC del modo 2.

SOLUCION PROBLEMA 4.31

En el libro de teoría se deducen las expresiones de los bits de suma y de acarreo en un sumador binario completo (ver Problema 4.7). Por lo tanto, las expresiones E_1 y E_2 son equivalentes a:

$$E_1 = x_1 \oplus y_1 \oplus c_0$$
$$E_2 = x_1 y_1 + (x_1 \oplus y_1) c_0$$

Es decir, E_1 y E_2 son respectivamente S_1 y C_1. Luego la respuesta correcta es la **A**

SOLUCION PROBLEMA 4.32

De acuerdo con lo que se explica en el libro de teoría sobre las características de un sumador binario serie (ver Problema 4.9)

Afirmación I: **FALSA**. Ya que la complejidad de un sumador binario serie es siempre la misma independientemente del número de bits que tenga que sumar.

Afirmación II: **VERDADERA**.

SOLUCION PROBLEMA 4.33

Afirmación I: Si se usa lógica combinacional de dos niveles y cada puerta tiene un retardo de 5 μs, el retardo total es t=2·5=10 μs. La afirmación es **FALSA**

Afirmación II: En un sumador binario paralelo (sumador con propagación de arrastre), en el caso más desfavorable el resultado no será efectivo hasta que no haya pasado un tiempo t=n·δ, donde n es el número de bits y δ es el tiempo que tarda un SBC en generar el arrastre para la etapa siguiente. Según el enunciado n=4 bits y δ=10 μs, luego t=40 μs. La afirmación es **VERDADERA**..

SOLUCION PROBLEMA 4.34

De acuerdo con las expresiones generales deducidas en el libro de teoría para un comparador de números de tamaño n>1 implementado mediante un circuito combinacional:

$$M = M_{n-1} + I_{n-1}M_{n-2} + ... + I_{n-1}I_{n-2}..I_1M_0$$
$$I = I_{n-1}I_{n-2}...I_1I_0$$
$$m = m_{n-1} + I_{n-1}m_{n-2} + ... + I_{n-1}I_{n-2}..I_1m_0$$

En el caso que se propone n=5, ya que el número Y de cuatro bits se puede considerar que tiene 5 bits suponiendo y_4=0, luego

$$M = M_4 + I_4M_3 + I_4I_3M_2 + I_4I_3I_2M_1 + I_4I_3I_2I_1M_0$$
$$I = I_4I_3I_2I_1I_0$$

Puesto que M_4 e I_4 son:

$$M_4 = x_4\bar{y}_4$$
$$I_4 = x_4y_4 + \bar{x}_4\bar{y}_4$$

y además y_4=0, entonces:

$$M_4 = x_4$$
$$I_4 = \bar{x}_4$$

Luego

$$M = x_4 + \bar{x}_4M_3 + \bar{x}_4I_3M_2 + \bar{x}_4I_3I_2M_1 + \bar{x}_4I_3I_2I_1M_0$$
$$I = \bar{x}_4I_3I_2I_1I_0$$

En la expresión de M, $\bar{x}_4 = 1$, por lo tanto se llega

$$\boxed{\begin{array}{l} M = x_4 + M_3 + I_3M_2 + I_3I_2M_1 + I_3I_2I_1M_0 \\ I = \bar{x}_4I_3I_2I_1I_0 \end{array}}$$

Afirmación I: **Verdadera**
Afirmación II: **Verdadera**

SOLUCION PROBLEMA 4.35

Puesto que un sumador binario paralelo de números de 8 bits, se puede implementar con 8 SBC. Como un SBC se puede implementar con 2 SSB y 1 puerta OR. Entonces el sumador pedido requeriría 2·8=**16 SSB** y 1·8=**8 puertas OR**.

SOLUCION PROBLEMA 4.36

DATOS

- Dos números binarios de 12 bits representados en código BCD
- X_{BCD}= 0101 0001 1001 Y_{BCD}= 0011 0100 0111
- X_{BCB}- Y_{BCD} expresado en código BCD?

El método más rápido y sencillo para resolver este problema, es pasar cada cifra BCD codificada en binario a decimal y realizar la resta:
Así:
X_{BCD}= 0101 0001 1001 = 519_{10}
Y_{BCD}= 0011 0100 0111 = 347_{10}

La resta X_{BCD} - Y_{BCD}= 172_{10} que expresada en código BCD resulta **0001 0111 0010**.

SOLUCION PROBLEMA 4.37

La solución a este problema se encuentra en el fichero `probT4_37.pdf`.

SOLUCION PROBLEMA 4.38

DATOS

- Circuito secuencial síncrono con dos estados (S_0, S_1).y con dos entradas (x,y)
- Q es la variable de estado, que puede tomar los valores 0 para referirse al estado S_0 y 1 para referirse al estado S_1.
- El circuito se diseña utilizando un elemento de memoria de tipo D.

FORMA 1:
La forma más adecuada de resolver este ejercicio consiste en construir la tabla de la verdad de cada una de las soluciones propuestas y compararlas con los valores de D(t) obtenidos de la tabla de estados del enunciado.

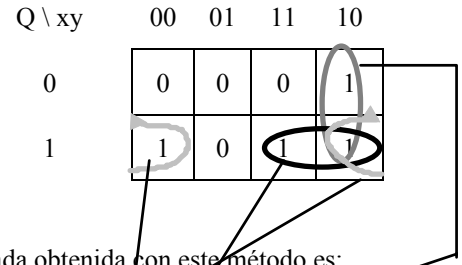
(Q(t), x , y)	De la Tabla D(t)	Solución A $D(t)=\overline{Q}\cdot x\cdot\overline{y}+Q\cdot\overline{x}\cdot y$	Solución B $D(t)=Q(\overline{y}+x)+\cdot x\cdot\overline{y}$	Solución C $D(t)=Q\cdot x\cdot\overline{y}$
0 0 0	0	0	0	0
0 0 1	0	0	0	0
0 1 0	1	1	1	0
0 1 1	0	0	0	0
1 0 0	1	1	1	0
1 0 1	0	1	0	0
1 1 0	1	0	1	1
1 1 1	1	0	1	0

Luego la respuesta correcta es la **B)**, ya que es la única que genera una D(t) igual a la indicada en el enunciado (columna 2).

Una misma función lógica puede expresarse de múltiples maneras, todas ellas equivalentes entre sí. Como se deduce a continuación, en este caso la solución B) coincide con la expresión simplificada de la función de entrada al elemento de memoria, D(t)=Q(t+1).

FORMA 2:

Otra forma alternativa de hacer este problema es obteniendo la función lógica D(t) a partir del método de reducción de Karnough. tal y como se muestra en la siguiente figura:



Luego la expresión simplificada obtenida con este método es:

$$D(t) = Q\cdot\overline{y} + Q\cdot x + x\cdot\overline{y} = Q\cdot(\overline{y} + x) + x\cdot\overline{y}$$

¡¡Atención!!: El que la expresión simplificada obtenida por el método de Karnough sea igual a la expresión B) es condición suficiente para afirmar que la solución correcta es la B), sin embargo no es una condición necesaria, ya que al plantear el enunciado del ejercicio, podría haberse sustituido la expresión B) por cualquier otra de sus formas equivalentes como por ejemplo:

$$D(t) = Q\cdot\overline{y}\cdot(x + \overline{x}) + Q\cdot x + x\cdot\overline{y} = Q\cdot\overline{y}\cdot x + Q\cdot\overline{y}\cdot\overline{x} + Q\cdot x + x\cdot\overline{y}$$

SOLUCION PROBLEMA 4.39

DATOS

- Dos números binarios de 12 bits representados en código BCD
- X_{BCD}= 1001 0011 0101 Y_{BCD}= 0001 0101 0001
- Resto división entera X_{BCB}/Y_{BCD} expresado en binario?

El método más rápido y sencillo para resolver este problema, es pasar cada cifra BCD codificada en binario a decimal y realizar la resta:
Así:
X_{BCD}= 1001 0011 0101 = 935_{10}
Y_{BCD}= 0001 0101 0001 = 151_{10}

El resto de la división entera X_{BCD}/Y_{BCD}= 029_{10} que expresado en binario resulta **0000 0001 1101**.

SOLUCION PROBLEMA 4.40

DATOS
- X= 010 1000 1110
- Secuencia de desplazamiento LICS, LDCS, LICS, LDCS, LICS, LDCS y LDCS,

Con la notación LICS se está haciendo referencia a un desplazamiento de un bit, de tipo Lógico, hacia la Izquierda, Cerrado y Simple. Luego aplicando un desplazamiento de este tipo a X se obtiene:

```
LICS(X)= 101 0001 1100
```

Mientras que con la notación LDCS se está haciendo referencia a un desplazamiento de un bit, de tipo Lógico, hacia la Derecha, Cerrado y Simple. Luego aplicando un desplazamiento de este tipo a X se obtiene:

```
LDCS(X)= 001 0100 0111
```

Si ahora se aplica LICS sobre LDCS(X) se obtiene

```
LICS(LDCS(X))= 010 1000 1110=X
```

Se observa que la aplicación de un desplazamiento LICS seguido por un desplazamiento LDCS se cancelan y se sigue manteniendo X.

En consecuencia, el resultado de la secuencia de desplazamientos:

```
(LICS LDCS) (LICS LDCS) (LICS LDCS) LDCS
```

es equivalente a la realización de una único desplazamiento LDCS (X). Es decir, el resultado es:

`LDCS(X)= 001 0100 0111`

SOLUCION PROBLEMA 4.41

DATOS
- C_0= 2^n palabras x 1bit/palabra
- Implementación de un sumador de dos números de n bits y 2n bits

La suma de un número X de n bits (X_{n-1}, X_{n-2},...,X_0) y de un número Y de 2n bits (Y_{2n-1}, Y_{2n-2},..., Y_0), da como resultado un número R de 2n bits (R_{2n-1}, R_{2n-2},..., R_0)además hay que considerar un bit de acarreo c, luego. Es decir el sumador combinacional debe poseer (n+2n) entradas y (2n+1) salidas.

En definitiva para implementar este sumador se necesita una memoria ROM de la siguiente capacidad C_T:

$$C_T = 2^{N^o de entradas}\ pal \times N^o salidas(bits/pal)$$
$$C_T = 2^{n+2n}\ pal \times (2n+1)(bits/pal)$$
$$C_T = 2^{3n}\ pal \times (2n+1)(bits/pal)$$

Luego el número de módulos N de capacidad C_0 que se requieren para implementar una memoria de capacidad C_T es:

$$N = \frac{C_T}{C_0} = \frac{2^{3n}\ pal}{2^n\ pal} x \frac{2n+1\ (bits/pal)}{1\ (bits/pal)} = 2^{2n} \times (2n+1)$$ módulos

SOLUCION PROBLEMA 4.42

DATOS

- Comparador secuencial de dos números X e Y de n bits sin signo.
- Los bits se reciben de forma serie empezando por los menos significativos

Sean dos números de n bits sin signo X =[x_{n-1}, x_{n-2},...,x_0] eY =[y_{n-1}, y_{n-2},...,y_0], los bits menos significativos son los x_0 e y_0 y los más significativos son los x_{n-1} e y_{n-1}.

Puesto que se trata de un comparador secuencial al comparar el bits x_i con el y_i existirán tres posibles estados, en función del resultado de la comparación

S_I (Igual) al que se entra si $x_i=y_i$, es decir, x_iy_i=00, 11
S_M (Mayor) al que se entra si $x_i>y_i$, es decir, x_iy_i=10
S_m (menor) al que se entra si $x_i<y_i$, es decir, x_iy_i=01

La comparación se realiza siguiendo los siguientes pasos:

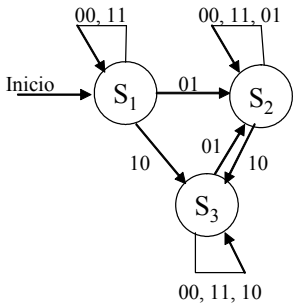
1) Se compara x_0 con y_o, del resultado de la comparación se entrará en alguno de los tres estados S_I, S_M o S_m.
2) Se compara x_1 con y_1, se pueden producir las siguientes transiciones de estado:
   2.1) Si se está en el estado S_I (Igual)en función del resultado de la comparación de x_1 con y_1 el próximo estado será:
   - S_I si $x_1=y_1$, es decir se tiene x_1y_1=11 o 00. Se permanece en el mismo estado.
   - S_M si $x_1>y_1$, es decir se tiene x_1y_1=10.
   - S_m si $x_1<y_1$, es decir se tiene x_1y_1=01.

   2.2) Si se está en el estado S_M (Mayor) en función del resultado de la comparación de x_1 con y_1 el próximo estado será:
   - S_M si $x_1= y_1$ o $x_1>y_1$, es decir se tiene x_1y_1=00,11 o10. Se permanece en el mismo estado.
   - S_m si $x_1<y_1$, es decir se tiene x_1y_1=01.

   2.3) Si se está en el estado S_m (Menor) en función del resultado de la comparación de x_1 con y_1 el próximo estado será:
   - S_m si $x_1= y_1$ o $x_1<y_1$, es decir se tiene x_1y_1=00,11 o 01. Se permanece en el mismo estado.
   - S_M si $x_1>y_1$, es decir se tiene x_1y_1=10.

3) Se repite el paso 2) para los bits i=2,...,n-1

Se puede observar que al comenzar la comparación por los bits menos significativos de X e Y solamente se tendrá un resultado final al llegar al bit más significativo. Es decir, es necesario comparar los n bits de cada número.



Si se observa la figura, de acuerdo con los que se acaba de explicar, es posible asociar a que estados {S_I, S_M, S_m} se corresponden los estados {S_1, S_2, S_3}. Se obtiene que:

$S_1 = S_I$
$S_2 = S_m$
$S_3 = S_M$.

Luego la respuesta correcta es la **A)**.

SOLUCION PROBLEMA 4.43

DATOS

- Se dispone de conexiones a "0" lógico y a "1" lógico.

Lo primero que hay que hacer es estudiar las cuatro funciones que piden implementar:

$$f1 = x \oplus y = \bar{x}y + x\bar{y}$$

f1 es la función XOR.

$$f2 = \overline{x \oplus y}$$

f2 es la función XOR complementada.

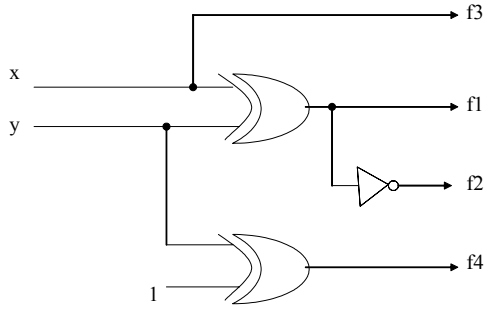
$$f3 = x\bar{y} + x = x \cdot (\bar{y} + 1) = x$$

f3 es simplemente la entrada *x*.

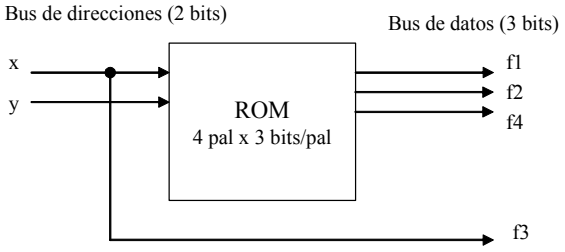
$$F4 = \bar{y}$$

f4 es simplemente el complemento de la entrada *y*.

Afirmación I: Nos dan dos puertas XOR y una puerta NOT para implementar las cuatro funciones. Con una puerta XOR se implementa la función f1, conectando la salida de esta puerta a una puerta NOT se obtiene f2. La función f3 es la entrada *x*. Finalmente la función f4 se obtiene con otra puerta XOR cuyas entradas deben ser una señal "1" lógico y la entrada *y*. Se tendría el circuito que se muestra en la figura. En conclusión, esta afirmación es **VERDADERA.**



Afirmación II: Nos dan una ROM de capacidad C_T =4 palabras x 3 (bits/palabras). Con esta ROM es posible implementar un circuito digital que posea 2 entradas y 3 salidas. En este problemas las entradas son *x* e *y*, además sólo hay que generar tres salidas: f1, f2 y f4, ya que f3 es la entrada *x*. Se tendría el esquema de la figura adjunta. En conclusión, esta afirmación es **VERDADERA**.



SOLUCION PROBLEMA 4.44

La solución a este problema se encuentra en el fichero `probT4_44.pdf`.

SOLUCION PROBLEMA 4.45

DATOS

- X= `111 1000 1100`
- A X se le aplican 5 operaciones LICS seguidas de 12 operaciones LDCS

Con la notación LICS se está haciendo referencia a un desplazamiento de un bit, de tipo Lógico, hacia la Izquierda, Cerrado y Simple. Luego aplicando un desplazamiento de este tipo a X se obtiene:

`LICS(X)= 111 0001 1001`

Mientras que con la notación LDCS se está haciendo referencia a un desplazamiento de un bit, de tipo Lógico, hacia la Derecha, Cerrado y Simple. Luego aplicando un desplazamiento de este tipo a X se obtiene:

`LDCS(X)= 011 1100 0110`

Si ahora se aplica LICS sobre LDCS(X) se obtiene

`LICS(LDCS(X))= 111 1000 1100=X`

Se observa que la aplicación de un desplazamiento LICS seguido por un desplazamiento LDCS se cancelan y se sigue manteniendo X. En consecuencia, es equivalente aplicar 5 operaciones LICS seguidas de 12 operaciones LDCS, que aplicar únicamente 7 operaciones LDCS. Luego para resolver el problema bastará con aplicar 7 operaciones LDCS sobre X. A continuación se muestran los resultados parciales que se obtienen al aplicar el operador LDCS sobre X:

```
X=          111 1000 1100
LDCS(X)=    011 1100 0110
LDCS^2(X)=  001 1110 0011
LDCS^3(X)=  100 1111 0001
LDCS^4(X)=  110 0111 1000
LDCS^5(X)=  011 0011 1100
LDCS^6(X)=  001 1001 1110
LDCS^7(X)=  000 1100 1111
```

Luego el resultado final es **`000 1100 1111`**

SOLUCION PROBLEMA 4.46

La solución a este problema se encuentra en el fichero `probT4_46.pdf`.

SOLUCION PROBLEMA 4.47

DATOS
- Comparador de dos números binarios A(a_1,a_0) y B(b_1,b_0)
- Salidas del comparador M, I y m.
- Implementar con una ROM y una puerta NOR

En principio cualquier circuito digital puede ser implementado con una memoria ROM. La capacidad total C_T de dicha ROM vendría dada por la expresión:

$$C_T = 2^{\text{Nº de entradas}} \text{ (palabras) x (Nº de salidas) (bits/palabras)}$$

El comparador descrito en el enunciado posee 4 entradas (a_1 a_0 b_1 b_0) y 3 salidas (M, I,m). Pero del libro de teoría se sabe que la salida *I* está relacionada con las salidas *M* y *m* a través de la siguiente función lógica:

$$I = \overline{M + m}$$

Es decir, *I* se puede obtener como la salida de una puerta NOR cuyas entradas sean *M* y *m*. Luego en realidad en la ROM sólo hay que almacenar los posibles valores de 2 salidas M y N, ya que en el enunciado nos proporcionan una puerta NOR. Luego la capacidad mínima de la ROM necesaria es:

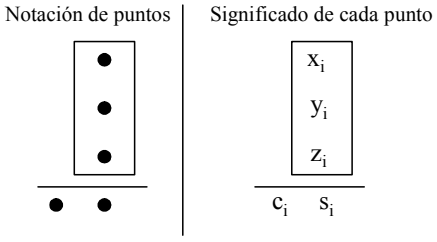
$C_T = 2^4$ (palabras) x 2 (bits/palabras)

SOLUCION PROBLEMA 4.48

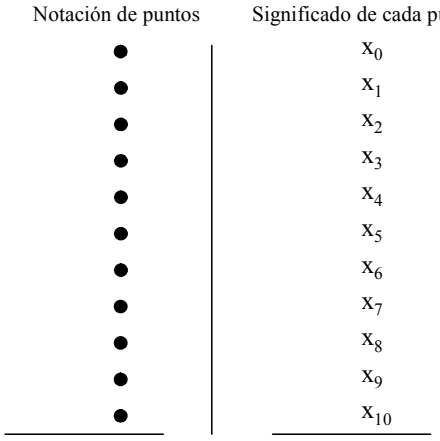
DATOS
- Circuito combinacional
- Sumador de 11 números binarios de 1 bit cada uno
- Implementarlo con módulos SBC

Nota: Este problema es similar al problema 4.13 del libro de problemas de la asignatura.

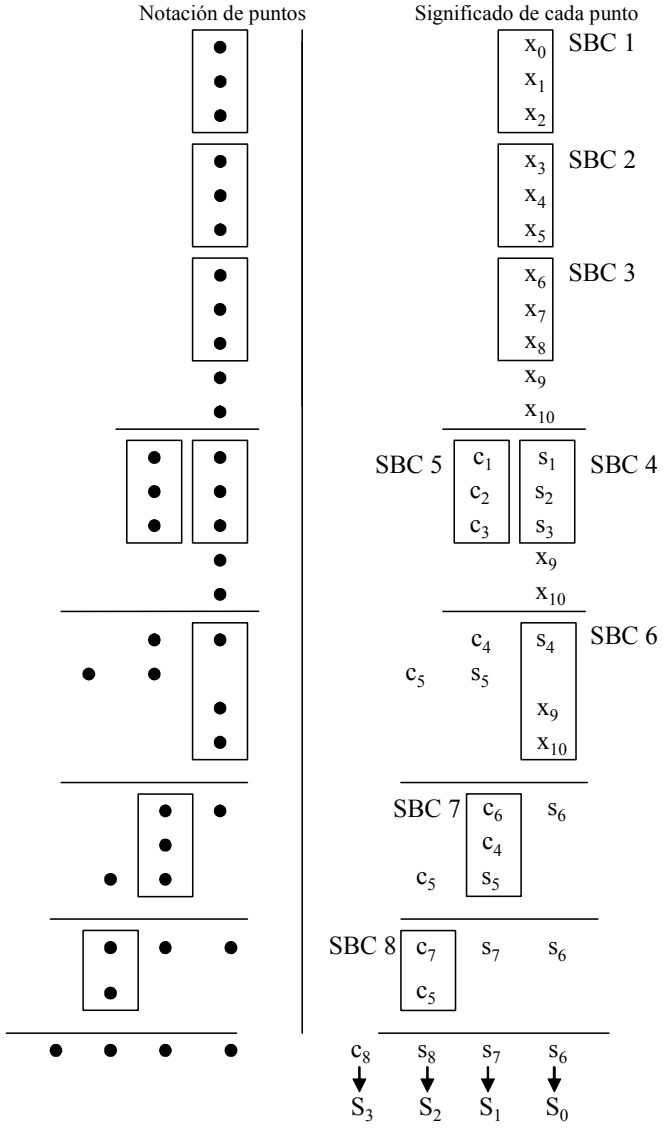
De forma general un Sumador Binario Completo (SBC) posee tres entradas de 1 bit cada una (sumando x_i, sumando y_i, sumando z_i) y dos salidas de un bit cada una (acarreo c_i, suma s_i). El funcionamiento de un SBC en notación de puntos (cada punto representa 1 bit) sería (esquema de la izquierda):



Se desean sumar 11 números binarios de 1 bit cada uno. Denotemos a estos números de la siguiente forma: x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}. Como máximo la suma de estos números daría 11 (1011), luego el resultado de la suma se debe expresar con tres bits $S_3S_2S_1S_0$. La suma de estos 11 números en notación de puntos se expresaría de la siguiente forma:



La resolución de esta operación usando SBCs genera el siguiente diagrama de puntos:



Luego se requieren cómo mínimo **8 SBCs** para diseñar un circuito combinacional que sea un sumador de 11 números de 1 bit de longitud.

SOLUCION PROBLEMA 4.49

DATOS
- Circuito de comparación secuencial de 2 números binarios de 8 bits de forma serie.
- Se implementa con biestables tipo D.

De acuerdo con el libro de teoría un comparador secuencial se puede define mediante tres estados: Mayor, Igual o Menor. Asimismo se necesitan $3 \leq 2^n \Rightarrow n=2$ variables de estado (Q_1, Q_0) para codificar estos tres estados. Luego habrá que asociar un biestable tipo D para almacenar cada variable. En consecuencia el número mínimo de biestables tipo D necesarios es **2**.

SOLUCION PROBLEMA 4.50

DATOS
- X_{BCD}=0011 0011 0001
- Y_{BCD}=0101 0100 1001
- Suma expresada en BCD?

En primer lugar hay que expresar ambos números en decimal, lo cual es sencillo al estar expresados en BCD: X_{10}=331 y Y_{10}=549.

En segundo lugar se suman $S_{10}= X_{10}+Y_{10}$=880

Y finalmente se pasa la suma a BCD: **S_{BCD}=1000 1000 0000**

SOLUCION PROBLEMA 4.51

DATOS
- X_{C1}=0 1000 1100
- Y_{C1}=1 1111 1011
- Calcular su suma.

Esta pregunta se puede resolver de varias formas, pero como lo que interesa es el resultado de la operación, un método sencillo es pasar los dos números de complemento a 1 a decimal, realizar la operación en decimal y, final mente, convertir de nuevo el resultado a complemento a 1. En complemento a 1, la representación de un número negativo se obtiene invirtiendo los bits en su representación binaria; esto es, los 0's se cambian por 1's y los 1's por 0's. El bit situado más a la izquierda indica el signo del número, que es 0 si el número es positivo y 1 si es negativo.

X_{C1} = 010001100 $\Rightarrow$ X_{10} = 140
Y_{C1} = 111111011 $\Rightarrow$ Y_{10} = -4
$X+Y = 136_{10} \Rightarrow X+Y =$ **010001000**

Otra forma de resolverlo es sumar los dos números y el acarreo volverlo a sumar, puesto que en este caso los dos números X e Y tienen distinto signo [Véase el Problema 4.6].

Finalmente, otra forma de resolverlo es:

1) Complementar X_{C1} => $\overline{X_{C1}}$ = 101110011
2) Complementar Y_{C1} => $\overline{Y_{C1}}$ = 000000100
3) Sumar $\overline{X_{C1}}$ y $\overline{Y_{C1}}$ => $\overline{X_{C1}} + \overline{Y_{C1}}$ = 101110111
4) Complementar la suma: $\overline{\overline{X_{C1}} + \overline{Y_{C1}}}$ = 010001000

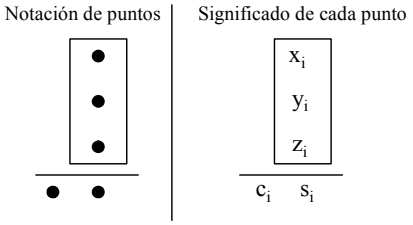
SOLUCION PROBLEMA 4.52

DATOS

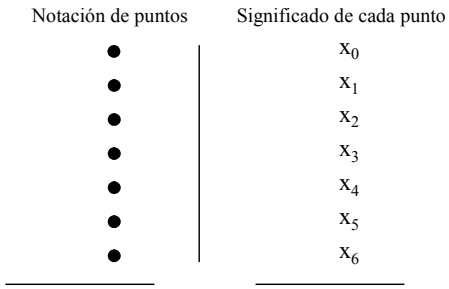
- Circuito combinacional
- I) Sumador de 7 números binarios de 1 bit cada uno
- II) Sumador de 5 números binarios de 2 bit cada uno
- Implementarlo con módulos SBC de 1 bit

Nota: Este problema es similar al problema 4.13 del libro de problemas de la asignatura.

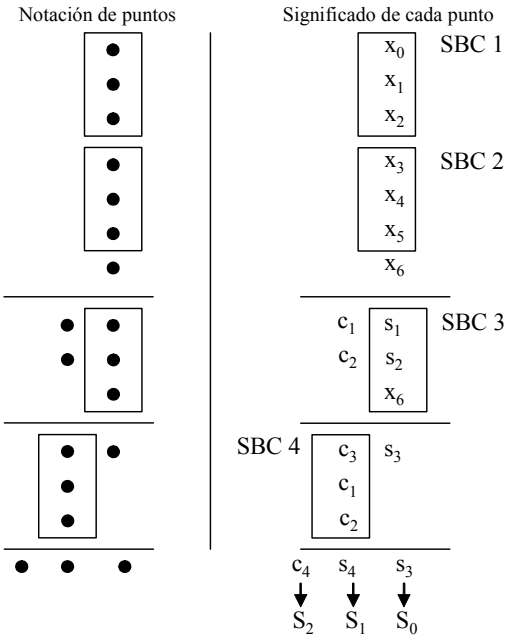
De forma general un Sumador Binario Completo (SBC) posee tres entradas de 1 bit cada una (sumando x_i, sumando y_i, sumando z_i) y dos salidas de un bit cada una (acarreo c_i, suma s_i). El funcionamiento de un SBC en notación de puntos (cada punto representa 1 bit) sería (esquema de la izquierda):



I) Se desean sumar 7 números binarios de 1 bit cada uno. Denotemos a estos números de la siguiente forma: x_0, x_1, x_2, x_3, x_4, x_5, x_6. Como máximo la suma de estos números daría 7 (111), luego el resultado de la suma se debe expresar con tres bits $S_2S_1S_0$. La suma de estos 7 números en notación de puntos se expresaría de la siguiente forma:

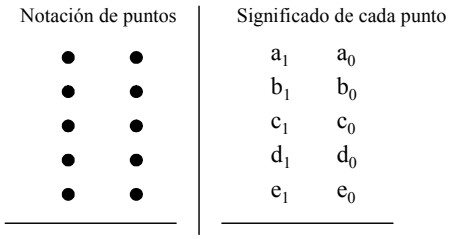


La resolución de esta operación usando SBCs genera el siguiente diagrama de puntos:

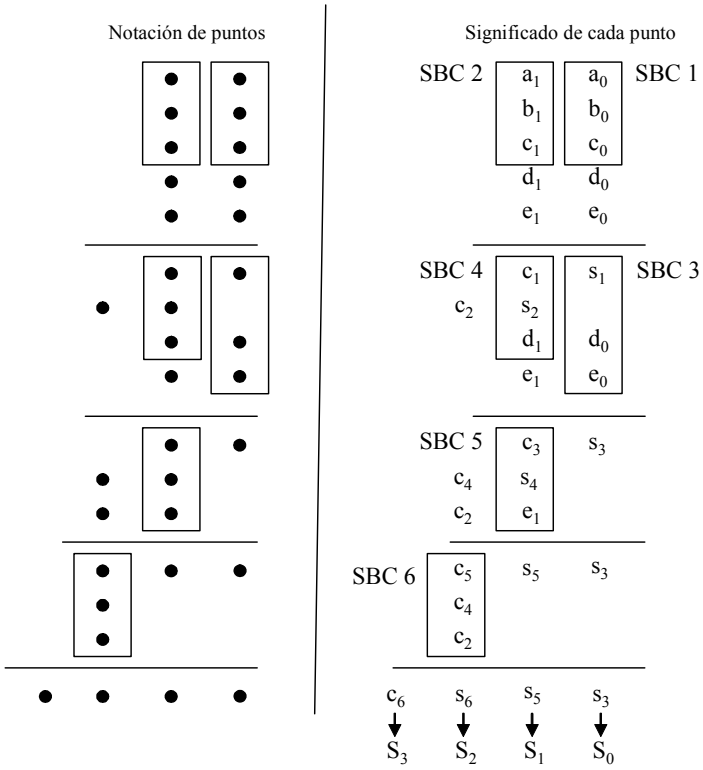


Luego se requieren cómo mínimo **4 SBCs** para diseñar un circuito combinacional que sea un sumador de 7 números de 1 bit de longitud.

II) Se desean sumar 5 números binarios de 2 bits cada uno. Denotemos a estos números de la siguiente forma: $A(a_1,a_0)$, $B(b_1,b_0)$, $C(c_1,c_0)$, $D(d_1,d_0)$ y $E(e_1,e_0)$. Como máximo la suma de estos números podría dar 15 (1111), luego el resultado de la suma se debe expresar con cuatro bits $S_3S_2S_1S_0$. La suma de estos 5 números en notación de puntos se expresaría de la siguiente forma:



La resolución de esta operación usando SBCs genera el siguiente diagrama de puntos:



Luego se requieren cómo mínimo **6 SBCs** para diseñar un circuito combinacional que sea un sumador de 5 números de 2 bits de longitud.

SOLUCION PROBLEMA 4.53

DATOS

- X_{BCD}=0001 1001 0101 0100
- Y_{BCD}=0111 1000 0010 0101
- Calcular su suma expresada en código exceso-3

Se debe pasar cada cifra BCD codificada en binario a decimal y realizar la suma:
Así:
X_{BCD}= 0001 1001 0101 0100 = 1 9 5 4
Y_{BCD}= 0111 1000 0010 0101 = 7 8 2 5

El resultado de la suma de estos dos números en decimal es S_{10}= 9779 y en BCD es:

$$S_{BCD}= 1001\ 0111\ 0111\ 1001$$

El código exceso-3 se obtiene sumándole 3, es decir 0011, a cada cifra expresada en BCD. Procediendo de esta forma se obtiene que la suma expresada en código exceso-3 es:

$$\mathbf{S_{E3}=1100\ 1010\ 1010\ 1100}$$

SOLUCION PROBLEMA 4.54

DATOS

- Módulos ROM $C_0=2^n$ palabras x n bits/palabra
- Circuito combinacional que eleve al cuadrado un número *X* de $2{\cdot}n$ bits

Se puede comprobar que un posible circuito combinacional que eleve al cuadrado un número X de $2{\cdot}n$ bits requiere como $2{\cdot}n$ entradas correspondientes al número X, y $2{\cdot}n + 2{\cdot}n = 4{\cdot}n$ salidas correspondientes al número X^2.

Si se desea implementar este circuito combinacional con una memoria ROM, la capacidad de esta memoria debería ser como mínimo de:

$$C_T=2^{N^o\ entradas}\ palabras\ x\ N^o\ salidas\ \ bits/palabra= 2^{2n}\ palabras\ x\ 4{\cdot}n\ bits/palabra$$

Para obtener el número de modulos ROM de capacidad C_0 que se necesitan para conseguir esta capacidad C_T hay que dividir C_T entre C_0:

$$N=\frac{C_T}{C_0}=\frac{2^{2\cdot n}}{2^n}x\frac{4\cdot n}{n}=2^n x4=2^{n+2}\ \ módulos$$

SOLUCION PROBLEMA 4.55

DATOS

- X= 1110 1011 0010
- A X se le aplican 14 operaciones LDCS seguidas de 2 operaciones LICS

Con la notación LDCS se está haciendo referencia a un desplazamiento de un bit, de tipo Lógico, hacia la Derecha, Cerrado y Simple. Luego aplicando un desplazamiento de este tipo a X se obtiene:

$$LDCS(X) = 0111\ 0101\ 1001$$

Con la notación LICS se está haciendo referencia a un desplazamiento de un bit, de tipo Lógico, hacia la Izquierda, Cerrado y Simple. Luego aplicando un desplazamiento de este tipo a X se obtiene:

$$LICS(X) = 1101\ 0110\ 0101$$

Si ahora se aplica LICS sobre LDCS(X) se obtiene

$$LICS(LDCS(X)) = 1110\ 1011\ 0010 = X$$

Se observa que la aplicación de un desplazamiento LICS seguido por un desplazamiento LDCS se cancelan y se sigue manteniendo X. En consecuencia, es equivalente aplicar 14 operaciones LDCS seguidas de 2 operaciones LICS, que aplicar únicamente 12 operaciones LDCS. Luego para resolver el problema bastará con aplicar 12 operaciones LDCS sobre X. A continuación se muestran los resultados parciales que se obtienen al aplicar el operador LDCS sobre X:

$X = 0111\ 0101\ 1001$
$LDCS(X) = 1011\ 1010\ 1100$
$LDCS^2(X) = 0101\ 1101\ 0110$
$LDCS^3(X) = 0010\ 1110\ 1011$
$LDCS^4(X) = 1001\ 0111\ 0101$
$LDCS^5(X) = 1100\ 1011\ 1010$
$LDCS^6(X) = 1110\ 0101\ 1101$
$LDCS^7(X) = 1111\ 0010\ 1110$
$LDCS^8(X) = 0111\ 1001\ 0111$
$LDCS^9(X) = 1011\ 1100\ 1011$
$LDCS^{10}(X) = 1101\ 1110\ 0101$
$LDCS^{11}(X) = 1110\ 1111\ 0010$
$LDCS^{12}(X) = 0111\ 0111\ 1001$

Luego el resultado final es **0111 0111 1001**.

De este resultado se concluye que la realización de *n* operaciones LDCS sobre un número de *n* bits deja al número inalterado. Es decir: $LDCS^n(X)=X$.