

Problema 1

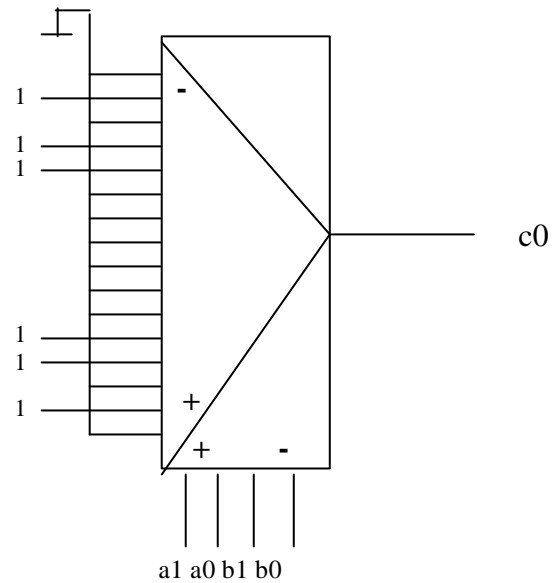
a.1) Tabla de Verdad del sistema

A	B	3(A-B)	OV
0	0	0	0
0	1	-3	0
0	-2	6	1
0	-1	3	0
1	0	3	0
1	1	0	0
1	-2	9	1
1	-1	6	1
-2	0	-6	1
-2	1	-9	1
-2	-2	0	0
-2	-1	-3	0
-1	0	-3	0
-1	1	-6	1
-1	-2	3	0
-1	-1	0	0

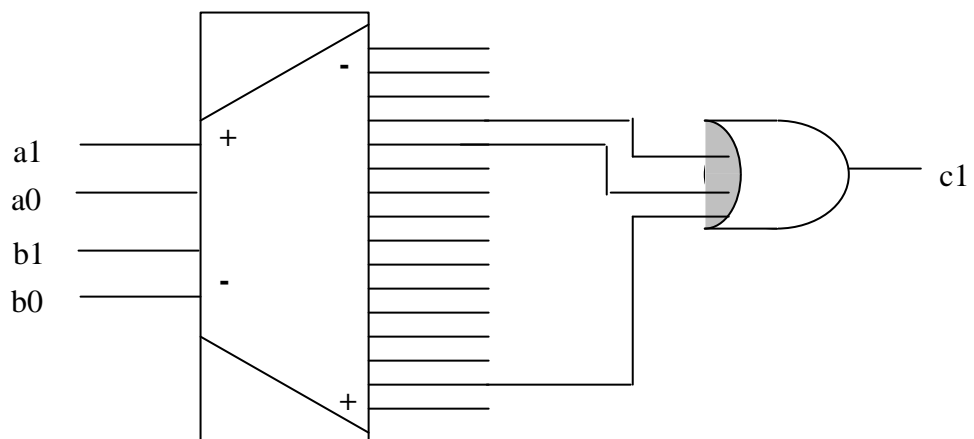
a1 a0	b1 b0	c2 c1 c0	OV
00	00	000	0
00	01	101	0
00	10	XXX	1
00	11	011	0
01	00	011	0
01	01	000	0
01	10	XXX	1
01	11	XXX	1
10	00	XXX	1
10	01	XXX	1
10	10	000	0
10	11	101	0
11	00	101	0
11	01	XXX	1
11	10	011	0
11	11	000	0

a.2)

Sintetizamos **c0** utilizando un **Multiplexor de 16 entradas de datos** (4 entradas de control):



Sintetizamos **c1** utilizando un **Decodificador 4 a 16** y puertas lógicas:

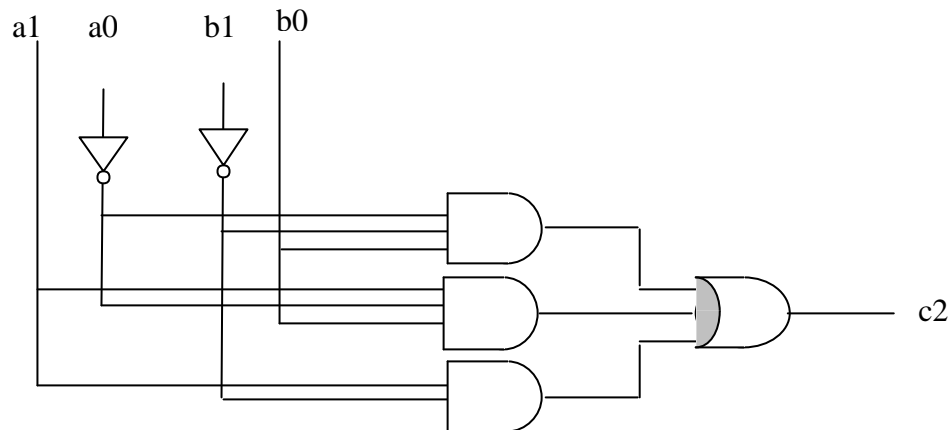


Para sintetizar **c2** a dos niveles de forma mínima, comenzamos obteniendo la expresión simplificada de c2 mediante un mapa de Karnaugh:

c2

a1 a0 \ b1 b0	00	01	11	10
00	0	0	1	X
01	1	0	X	X
11	0	X	0	1
10	X	X	0	0

$$c2 = a0' b1' b0 + a1 a0' b0 + a1 b1'$$



b.1) $D = 3*(A-B) \text{ MOD } 16$ suponiendo que A y B son números **enteros de 8 bits**

Recordamos que la operación **mod** calcula el Resto de una división (en este caso, el Resto de la división entre 16).

Por tanto, sea cual sea el valor que tome $3*(A-B)^i$, el resultado de calcular sobre éste el “mod 16”, será siempre un número menor que 16.

Así, el Rango del resultado (D) será: [0,15]

Para representar el número D necesitamos, pues, **4 bits**.

b.2) Para realizar el diseño a nivel de bloques, utilizaremos los siguientes elementos del chuletario:

Indicar claramente el tamaño de cada uno de los bloques utilizados.

Al plantearnos el diseño a nivel de bloques consideramos:

1. La siguiente igualdad, válida cuando los números están representados en Complemento a 2:

$$A - B = A + (-B) = A + B' + 1$$

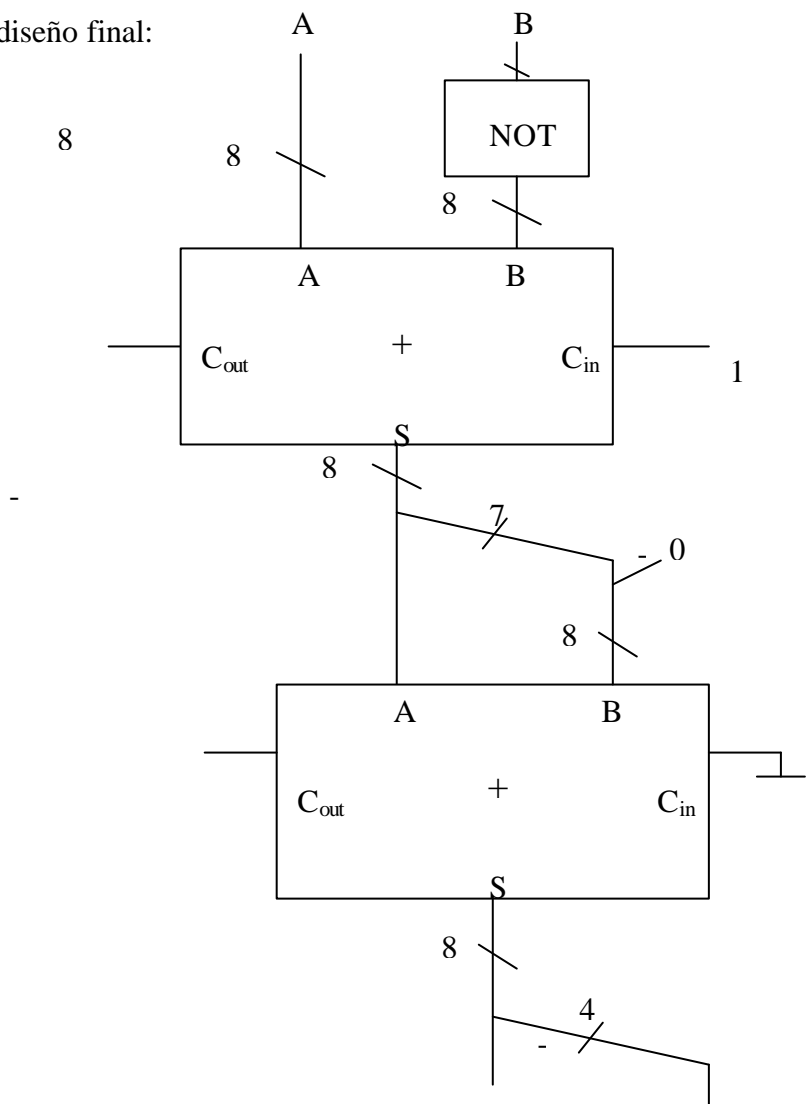
Para implementarlo necesitamos un bloque NOT de 8 bits para obtener B' y un sumador de números de 8 bits para hacer la suma $A+B'$. El 1 que todavía queda por sumar puede conectarse a la entrada de carry del sumador.

2. Sabemos que: $3*X = (2*X) + X$

También sabemos que podemos multiplicar por dos un número binario desplazando todos sus bits una posición a la izquierda y añadiendo un 0 como bit de menor peso. Por tanto, podemos calcular esta expresión utilizando un bloque sumador de números de 8 bits, cuyas entradas sean: el número que queremos multiplicar y ese mismo número decalado una posición a la izquierda.

3. En binario, la operación $X \bmod 2^n$ consiste en quedarnos con los n bits de menor peso de X.

La figura siguiente muestra el diseño final:



¹ Incluso cuando es un número negativo, puesto que los operadores cociente (div) y resto (mod), cumplen las siguientes relaciones: Dados $c := a \text{ div } b$, $r := a \text{ mod } b$, se tiene que:

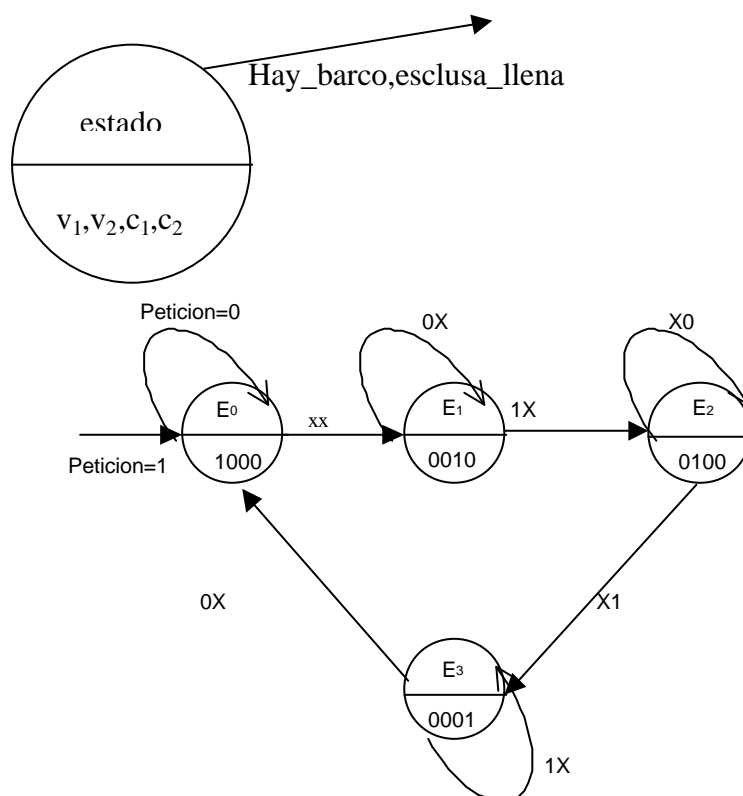
- (1) $a = c*b + r$
- (2) $|r| < |b|$
- (3) $|c| = |a| \text{ div } |b|$
- (4) $|r| = |a| \text{ mod } |b|$

Las ecuaciones (3) y (4) indican que cuando el dividendo y/o el divisor son negativos, los valores absolutos del cociente y el resto proporcionados son iguales a los obtenidos cuando el dividendo y el divisor son positivos, y los signos del cociente y el resto se eligen de manera que se cumpla la ecuación (1).

Problema 2

- a) Entradas: petición, hay_barco y esclusa_llena
Salidas: c_1 y c_2 : señales para abrir o cerrar compuertas
 v_1 y v_2 : controles válvulas

b)



c) **Tabla de salidas:**

Estado	v_1	v_2	c_1	C_2
E0: barco fuera	1	0	0	0
E1: barco entrando	0	0	1	0
E2: barco dentro (llenar esclusa)	0	1	0	0
E3: barco saliendo	0	0	0	1

Tabla de codificación de estados:

Estado	Codificación
E0	00
E1	01
E2	10
E3	11

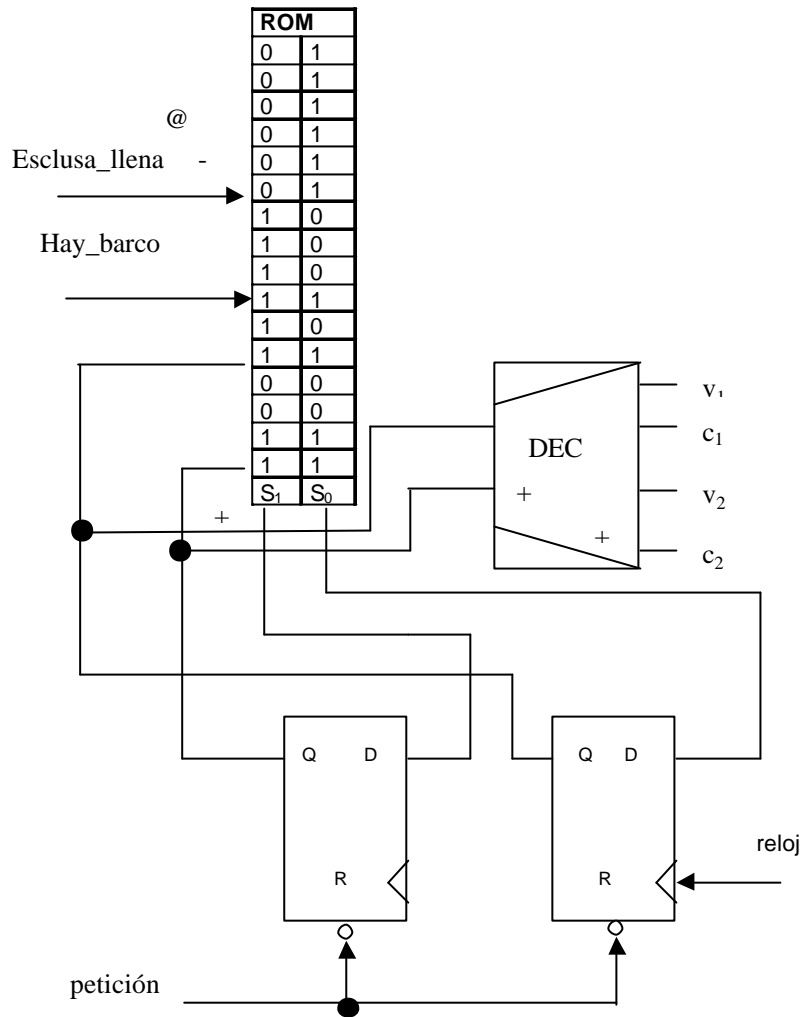
Tabla de transiciones entre estados:

Estado n	entradas	Estado n+1	salidas
	Barco,esclusa		v1,v2,c1,c2
00	XX	01	1000
01	0X	01	0010
01	1X	10	0010
10	X0	10	0100
10	X1	11	0100
11	1X	11	0001
11	0X	00	0001

Sustituyendo las X por sus valores para poder escribir la tabla completa. Las dos columnas del estado n+1 son los bits que escribiremos en la ROM.

Estado n	entradas	Estado n+1	salidas
	Barco,esclusa		v1,v2,c1,c2
00	00	01	1000
00	01	01	1000
00	10	01	1000
00	11	01	1000
01	00	01	0010
01	01	01	0010
01	10	10	0010
01	11	10	0010
10	00	10	0100
10	01	11	0100
10	10	10	0100
10	11	11	0100
11	00	00	0001
11	01	00	0001
11	10	11	0001
11	11	11	0001

d) La señal petición=0 resetea los biestables de forma que el sistema secuencial retorna al estado 0. La ROM tendrá cuatro entradas de direccionamiento y 2 salidas. Por tanto, será de $2^4 \times 2$ bits (16 posiciones o palabras de dos bits cada una).



Problema 3

Dirección	Instrucción en LM	M[@]: LM	M[@] en hexadecimal
00h		0000 0000 0110 0111	
01h		0000 0000 0000 0000	
02h		0000 0000 0000 0000	
03h	LOAD 0(R0),R1	0000 1000 0000 0000	0800h
04h	BNE 07h		
05h	STORE R2,76h(R1)	0101 0001 0111 0110	5176h
06h	LOAD 35h(R5),R1	0000 1101 0011 0101	0D35h
07h	BLE 05h	1001 1000 0000 0101	9805h
08h	ADD R1,R1,R1	1100 1001 0010 0100	C924h
09h	STORE R0,0(R1)	0100 0001 0000 0000	4100h
0Ah	BR 06h		
0Bh	STORE R3,2(R0)		

Estado	FETCH	DECO	LOAD	FETCH	DECO	BRANCH	DECO	FECTH
M@	03h	X	00h	04h	X	07h	X	08h
Mout	0800h	X	0067h	A807h	X	9805h	X	C924h
Min	X	0000h	X	X		X		X
Din	X	X	0067h	X	X	X	X	X
PC	03h	04h	04h	04h	05h	05h	08h	08h
IR	X	0800h	0800h	0800h	A807h	A807h	9805h	9805h
R@	X	X	00h	04h	00h			
RA	X	X	X	X	X			
RZ	X	X	X	0	0	0	0	0
RN	X	X	X	0	0	0	0	0
R1	X	X	X	0067h	0067h	0067h	0067h	0067