Facultad de Informática
Universidad Complutense de Madrid

EXAMEN PARCIAL DE FUNDAMENTOS DE COMPUTADORES

CURSO 2015-16, PRIMER PARCIAL (CONV. DE SEPTIEMBRE), 12 DE SEPTIEMBRE 2016

1. **(1 punto)** Dados los siguientes números: $A = +(50)_{10}$, $B = +(57)_8$, $C = +(4E)_{16}$ y $D = -(10100)_2$
   a) **(0,5 puntos)** Exprésélos en representación en complemento a 2 con 8 bits.
   b) **(0,5 puntos)** Efectúe las operaciones (A-B) y (C+D) indicando en cada caso si hay desbordamiento y/o acarreo y el por qué.

2. **(2,5 puntos)** Sea un sistema combinacional que tiene una entrada de 3 bits ($\underline{X}$), una entrada de 1 bit (C2), una salida de 3 bits ($\underline{Z}$) y una salida de 1 bit (E). La señal C2 indica el tipo de datos con los que opera el circuito:
   - Si C2=0, los valores de $\underline{X}$ y $\underline{Z}$ representan números enteros sin signo codificados en binario.
   - Si C2=1, los valores de $\underline{X}$ y $\underline{Z}$ representan números enteros con signo codificados en complemento a 2.

   En ambos casos el sistema debe realizar la siguiente operación:
   - Si $\underline{X} \leq 3$ entonces $\underline{Z} = 2\bullet\underline{X}$
   - Si $\underline{X} > 3$ entonces $\underline{Z} = \underline{X}/2$ (división entera)

   Si el resultado de la operación es representable con 3 bits en la codificación correspondiente, E valdrá '0' y $\underline{Z}$ tomará el valor de la operación. En caso contrario, E valdrá '1' y el valor que tome $\underline{Z}$ será irrelevante. Se pide:
   a) **(1 puntos)** Obtener la tabla de verdad del sistema.
   b) **(1,5 puntos)** Implementar el sistema utilizando el menor número de puertas NAND.

3. **(1 punto)** Usando registros de 4 bits y las puertas que considere necesarias, diseñe un sistema secuencial con una entrada de 4 bits ($\underline{X}$) por la que en cada ciclo de reloj recibe un dígito codificado en BCD y 3 salidas de 1 bit (C, N y M) tales que:
   - C vale 1, si los 3 últimos dígitos recibidos forman un número capicúa.
   - N vale 1, si los 3 últimos dígitos recibidos (suponiendo que se reciben comenzando por el dígito más significativo) forman un número mayor que 99.
   - M vale 1, si los 3 últimos dígitos recibidos (suponiendo que se reciben comenzando por el dígito más significativo) forman un número múltiplo de 5.

4. **(3 puntos)** Diseñar un generador de patrones con una entrada binaria, X, y una salida Z ∈ {a, b} que se comporte de la siguiente manera:
   - Si X = 0, la salida generará repetidamente la secuencia "baab"
   - Si X = 1, la salida generará repetidamente la secuencia "abbb"

   El sistema generará patrones completos (véase la figura), de manera que sólo se tendrá en cuenta el valor de la entrada cuando finaliza la generación de un patrón, es decir, en su último carácter. Inicialmente el sistema generará la secuencia "baab".

X(t)	0	0	0	**0**	0	0	1	**1**	1	1	1	**1**	1	0	0	**0**
Z(t)	b	a	a	b	b	a	a	b	a	b	b	b	a	b	b	b
	Secuencia "baab"				Secuencia "baab"				Secuencia "abbb"				Secuencia "abbb"			

   Se pide:
   a) **(1 punto)** Especificar el sistema mediante un diagrama de estados de tipo Moore.
   b) **(2 puntos)** Implementar el sistema con un contador binario módulo 8 y puertas lógicas.

5. **(2,5 puntos)** HDLC es un protocolo de comunicaciones que transmite los datos agrupados en tramas. Cada trama está delimitada en sus extremos inicial y final por la combinación de bits "01111110". El problema de este protocolo es que entre los datos que envía el emisor puede aparecer casualmente esa combinación y el receptor podría creer erróneamente que le ha llegado el fin de la trama. Por este motivo, el emisor cuando envía los datos en serie inserta un 0 siempre después de cinco 1 seguidos. Usando biestables D y una ROM, diseñe un sistema secuencial de tipo Mealy que, recibiendo en serie los datos de la trama a transmitir, indique cuándo debe introducirse un 0.

TC - Septiembre - 2016 (1)

(1) $A = +50_{10}$

$B = +57_8$

$C = +4E_{16}$

$D = -(10100)_2$

Expresarlos en C'2 con 8 bits.

Todos los nº están representados en magnitud y signo. Expresando el signo mediante el símbolo clásico (+,-) y la magnitud expresada en diferentes representaciones (decimal, octal, hexadecimal, binario)

⊗ $A = +50_{10}$

1º paso → calculo la magnitud en binario puro.

$|+50_{10}| = 50_{10}$ → división recursiva por la base

50 ⌊2
(0) 25 ⌊2
(1) 12 ⌊2
(0) 6 ⌊2
(0) 3 ⌊2
(1) (1)

menor peso ← → mayor peso

1100010bp

recordar q. la magnitud no tiene signo y q. (2)
al aplico la división por la base el bit más significativo siempre es 1 PERO este 1 no significa signo, es parte de la magnitud

2º paso añadir signo positivo (0) para convertirlo en C2

$$0110010_{C2,MS,C1} = +50_{10}$$

recordar q. las representaciones positivas de C2, C1 y MS coinciden

la representación obtenida utilizando 7 bit y el enunciado pide 8 ⇒ aplicar la extensión de signo. q. en C2 es replicar el bit de signo tantas veces como sea necesario.

$$\boxed{00110010_{C2} = A}$$

⊕ $B = +(57)_8$

1º paso calculo la magnitud en binario puro.

$|+(57)_8| = 57_8$

cada dígito octal se sustituye por su representación binaria con 3 bits:

101111_{BP}

2º se le añade el bit de signo positivo

$$0101111_{C2,C1,MS} = +57_8$$

como se pide q. se represente ~~con~~ 8 bits se aplica la operación extensión de signo

$$\boxed{B = 0010\,1111_{C2}}$$

c) $C = +(4E)_{16}$

1º paso - se calcula la magnitud en bp.

$|+4E_{16}| = 4E_{16}$

se sustituye cada dígito hexadecimal por su representación binaria de 4 bits

01001110_{bp}

2º se le añade el bit de signo ~~positivo~~. En este caso no hace falta añadirle el bit de signo positivo puesto q. la representación binaria empieza por cero y sabemos q. ese cero no tiene ningun valor:

$01001110_{bp} = 1001110_{bp}$

En este caso no hace falta aplicar la op. extensión de signo puesto q. ~~la~~ la representación ya tiene 8 bits

$D = -(10100)_2$

1º paso cálculo de la magnitud en binario puro

$|-(10100)_2| = 10100_{bp}$

2º paso añadir el signo positivo para representarlo en C2

~~010100~~ 010100_{C2}

3º paso, como el nº inicial es negativo, hay q. aplicar la operación cambio de signo.

$$010100_{C2} \longrightarrow \begin{array}{r} 101011 \\ +\ \ 1 \\ \hline 101100_{C2} \end{array}$$

101100_{C2}

4º paso como la representación sólo tiene 6 bits hay q. aplicar la operación extensión de signo de complemento a dos q consiste en repetir el bit de signo tantas veces como sea necesario

$$\boxed{11101100_{C2} = D}$$

1b A−B y C+D

- A−B → NUNCA realizamos restas en C2 directamente. hay q. convertir la resta en una suma

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

A lo conocemos = 00110010_{C2}

B lo conocemos = 00101111_{C2}

PERO −B NO lo conocemos, tenemos q. calcularlo aplicando la op. cambio de signo en C2

$$
\begin{array}{rr}
B = 00101111 \longrightarrow & 11010000 \\
& 1 \\
\hline
& 11010001
\end{array}
$$

$$
\begin{array}{rr}
A+(-B) = & 00110010 \\
& 11010001 \\
\hline
& 100000011
\end{array}
$$

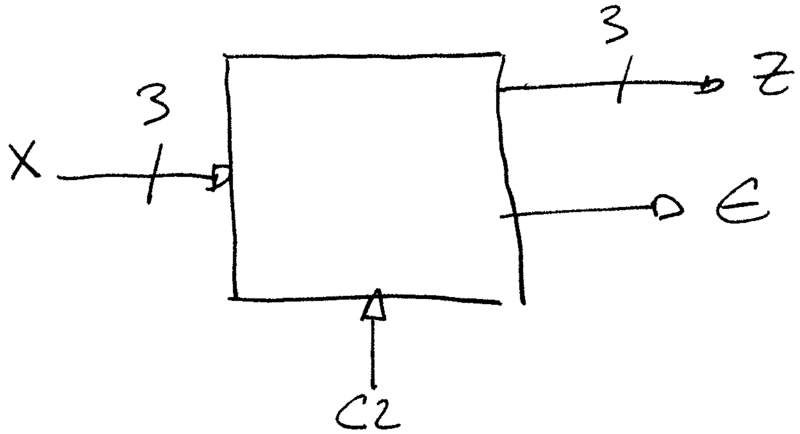
Existe acarreo puesto q. el resultado tiene un bit mas q. los operandos

NO Existe desbordamiento, la suma de un no positivo y uno negativo si puede dar un no positivo

1b $C+D$

$$
\begin{array}{rl}
 & 1 \\
C = & 01001110 \\
D = & 11101100 \\
\hline
 & 100111010
\end{array}
$$

2



Si $C_2 = \phi \Rightarrow x_y z$ enteros sin signo codificados en binario

Si $C_2 = 1 \Rightarrow x_y z$ enteros con signo en C2

Si $x \leq 3 \rightarrow z = 2 \cdot x$

Si $x > 3 \Rightarrow z = x/2$

respuesta → si $C2 = \phi$ se pueden representar nos comprendidos entre ϕ y z

X	Z
0	0
1	2
2	4
3	6
4	2
5	2
6	3
7	3

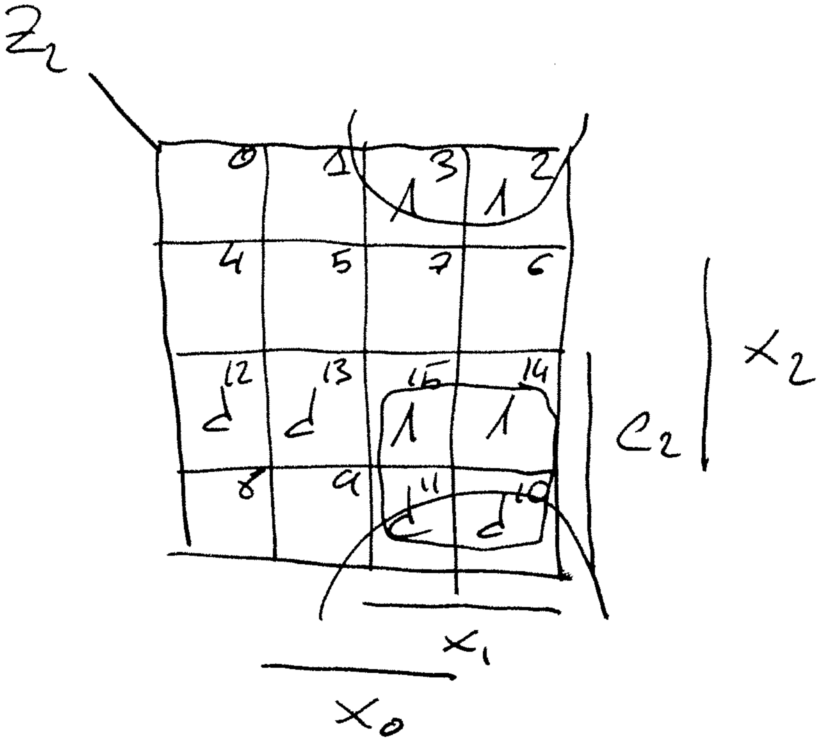
Si C2=1 se representan n.os comprendidos entre (-4,+3) (7)

X	Z
0	0
1	2
2	4 → no representable
3	6 → no representable
−4	−8 → no representable
−3	−6 → no representable
−2	−4
−1	−2

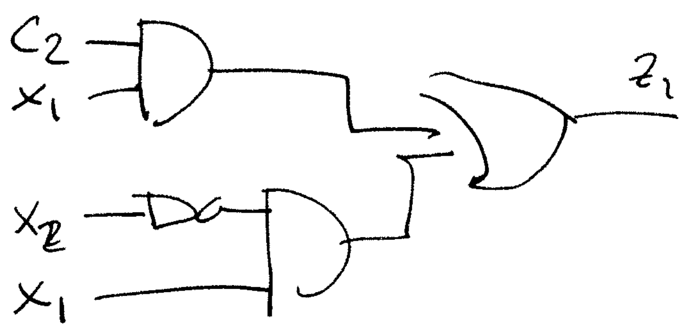
C_2 x_2 x_1 x_0	z_2 z_1 z_0 E
0000	0 0 0 0
0001	0 1 0 0
0010	1 0 0 0
0011	1 1 0 0
0100	0 1 0 0
0101	0 1 0 0
0110	0 1 1 0
0111	0 1 1 0
1000	0 0 0 0
1001	0 1 0 0
1010	d d d 1
1011	d d d 1
1100	d d d 1
1101	d d d 1
1110	1 0 0 0
1111	1 1 0 0

como el enunciado indica q. se utilicen puertas NAND (el menor n.º posible) hay q. aplicar los mapas de K.

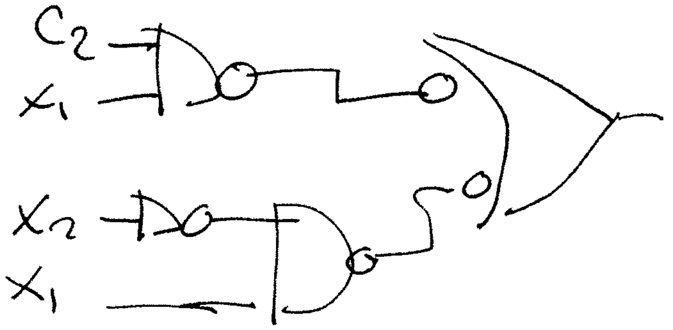
solamente voy a hacer 2 ejemplos, para z_2 y $\bar{E}$



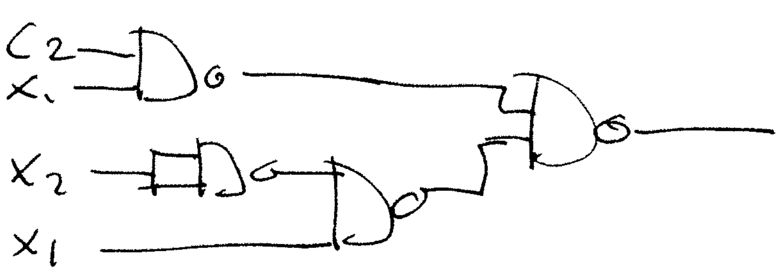
$$z_2 = C_2 x_1 + \bar{x}_2 x_1$$

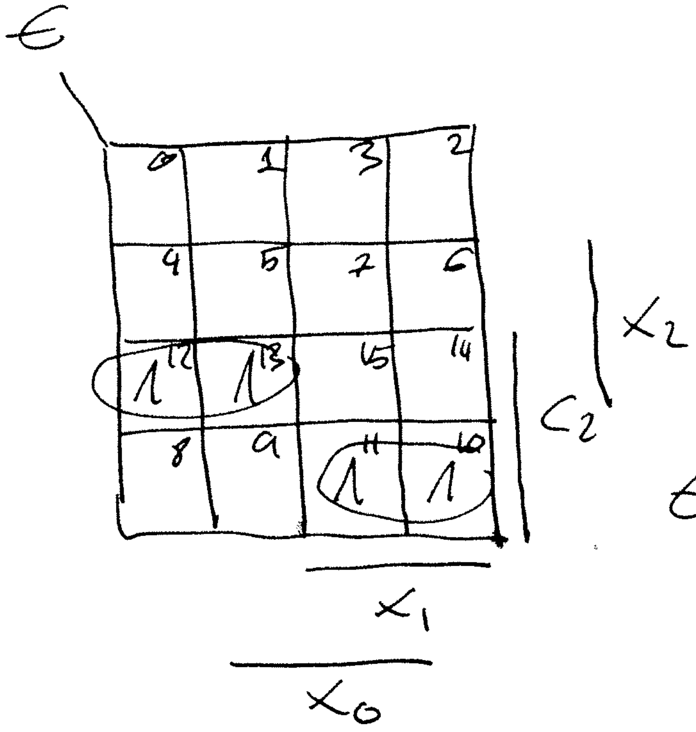


poniendo doble inversión a las salidas de las and y/o a las entradas de las or

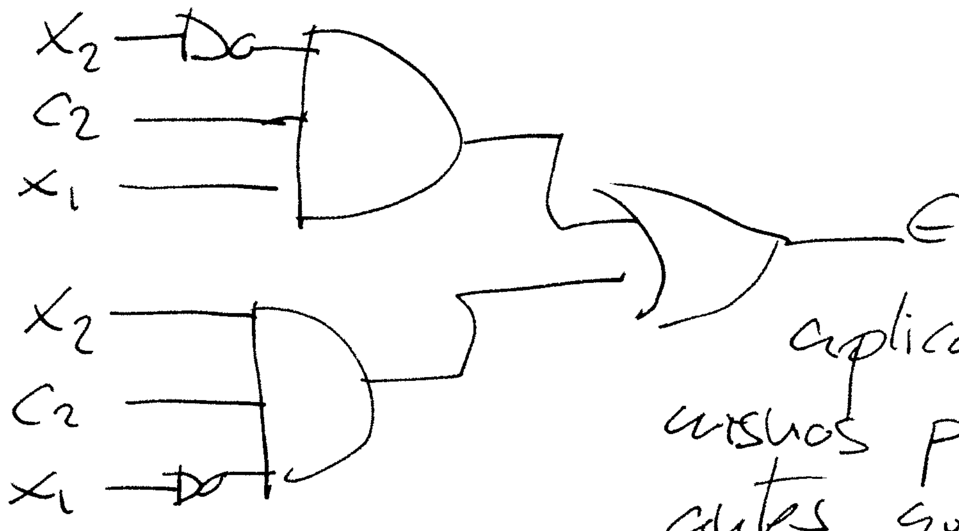


sustituyendo por su otra representación y cambiando el inverso por su representación nand (≡)

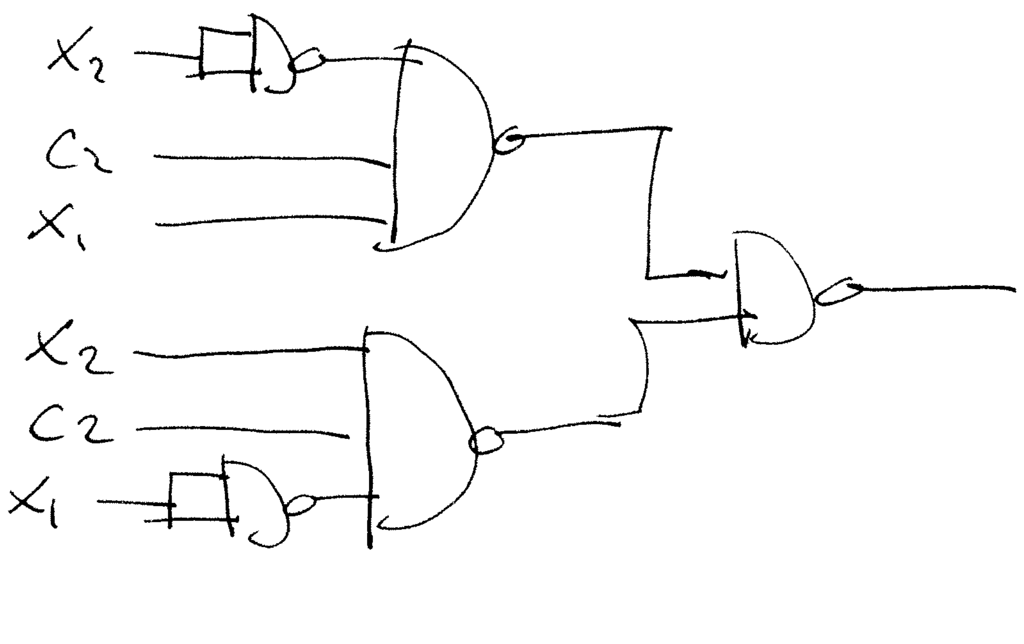




$$\epsilon = \overline{x_2}\, c_2\, x_1 + x_2\, c_2\, \overline{x_1}$$



aplicando los mismos pasos de antes queda



3

codificación BCD ⇒ cada dígito representado por 4 bits

	BCD
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

vamos a ver las condiciones que se tienen que cumplir

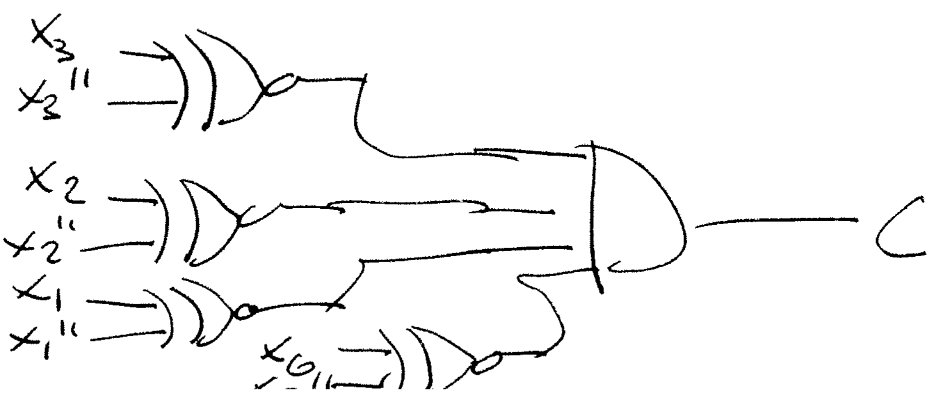
$C = 1$ ⇒ si los 3 últimos nos forman un nº capicúa ⇒ primer nº y último nº iguales

⇒ cada uno de los bits del nº BCD tienen que ser iguales. Siendo X un bit del primer dígito BCD y x'' un bit del tercer dígito BCD, tienen q. cumplir

X	x''	iguales
0	0	1
0	1	0
1	0	0
1	1	1

→ Puerta XNOR.

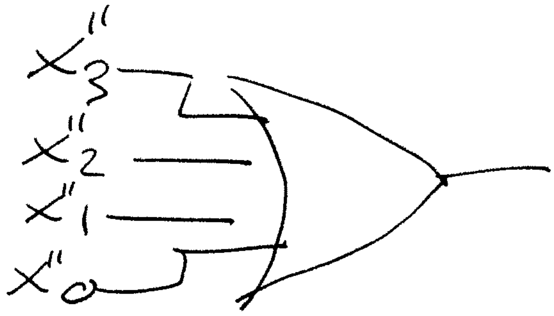
como tienen q. ser iguales los 4 bits de los 2 dígitos BCD



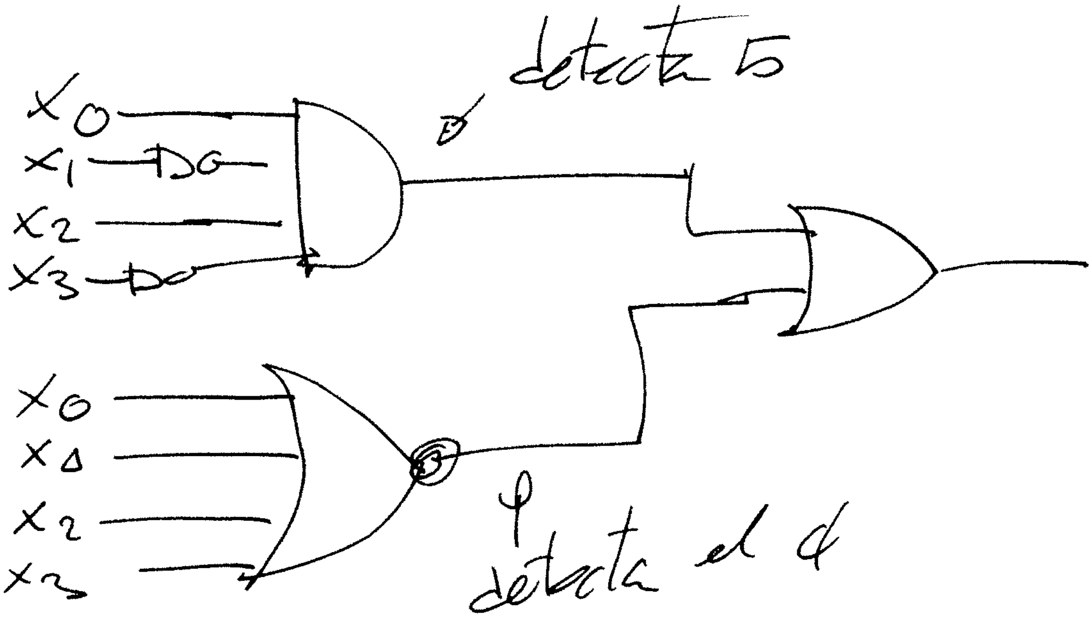
mayor de 99.

los n^os representables con 3 dígitos van del 000 → 999

los n^os mayores de 99 son todos aquellos en los q. el dígito más significativo es diferente de ϕ, es decir aquellos en los q. en su representación ~~binaria~~ BCD tienen al menos un 1. ⟹

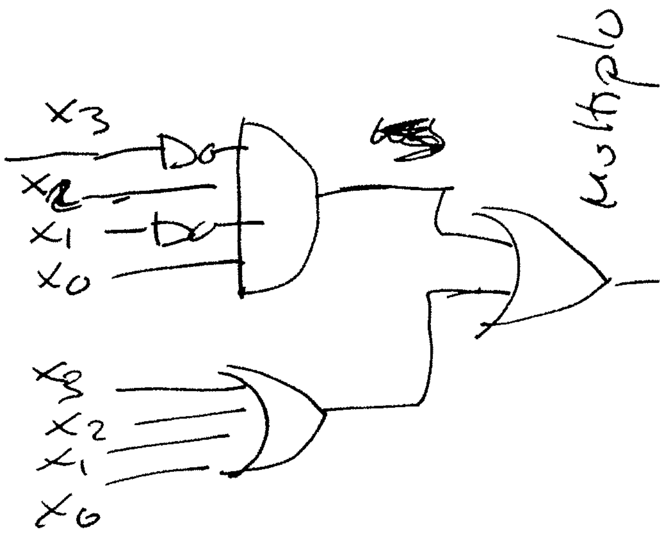


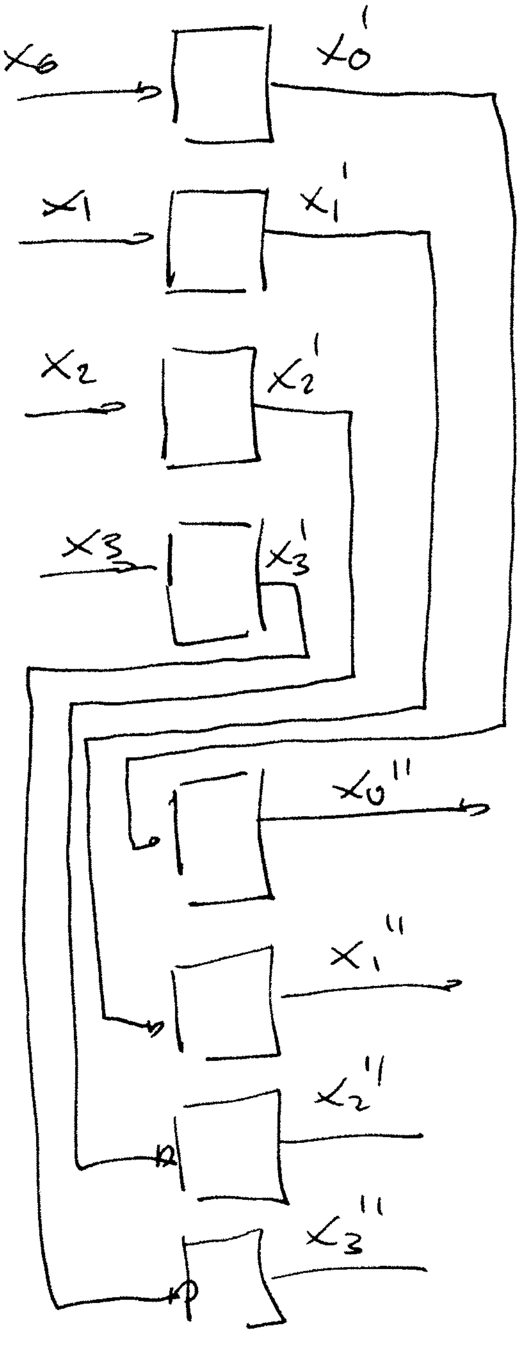
multiplo de 5.- son aquellos en los q. el ultimo dígito acaba en 5 o ϕ

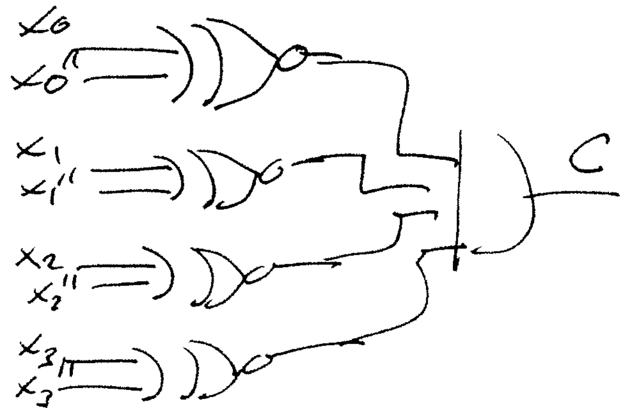


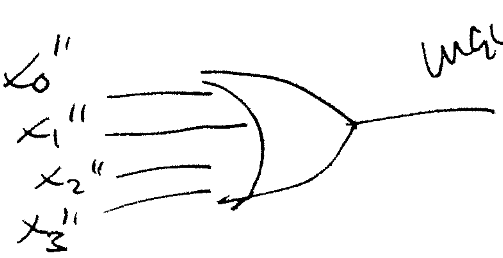
vamos a suponer q. el 1er digito q. llega sera el de mayor peso. es decir si llega ~~el~~ el 123, el 1º es el primer digito en llegar y el 3 el ultimo. visto esto el circuito queda

$x \rightarrow$ el ultimo digito
$x' \rightarrow$ segundo digito
$x'' \rightarrow$ primer digito







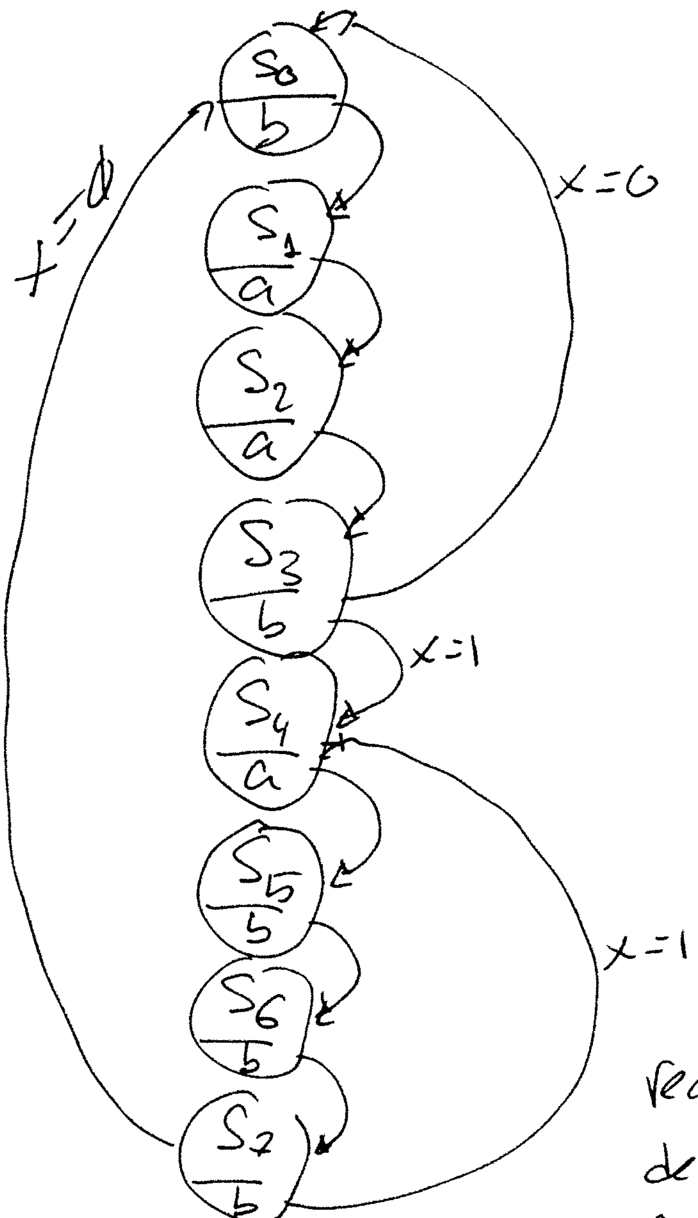


(4) Implementar un generador de patrones con una máquina de Moore es sencillo.

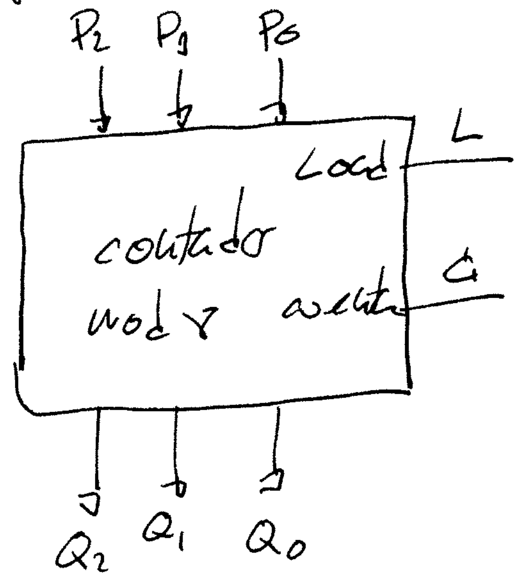
cada estado genera como salida un elemento del patrón. En este caso el patrón depende de una señal de control

$x = 0 \rightarrow$ se genera ~~ba~~ aab

$x = 1 \rightarrow$ abbb



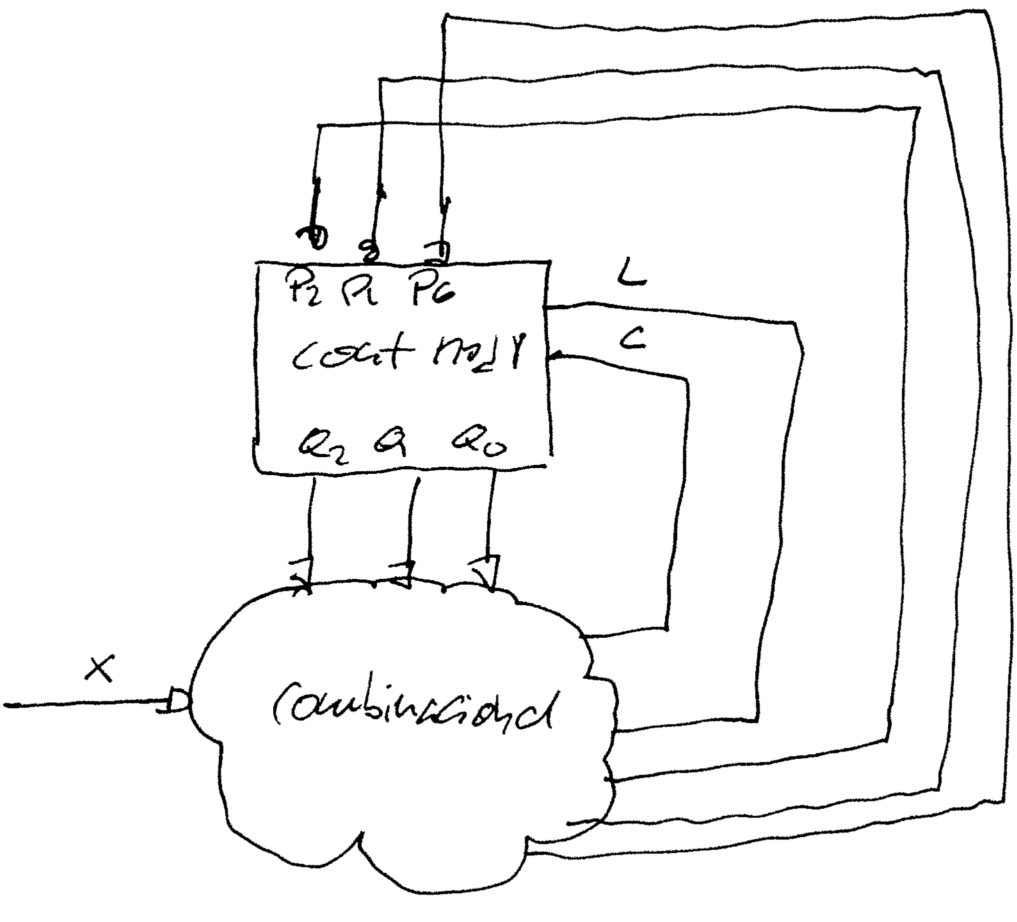
como los patrones se tienen q generar enteros solo importa el valor de x cuando se ha acabado de generar un patrón



recuerda → cuando se pasa de un estado a otro en una cuenta natural (p.e. $S_0 \rightarrow S_1 \rightarrow S_2$) se activa la señal de cuenta y se desactiva la de load

solo cuando se produce un salto (p.e. $S_7 \rightarrow S_0$) se activa la señal de load y se genera el valor de carga paralela $P_2 P_1 P_0$

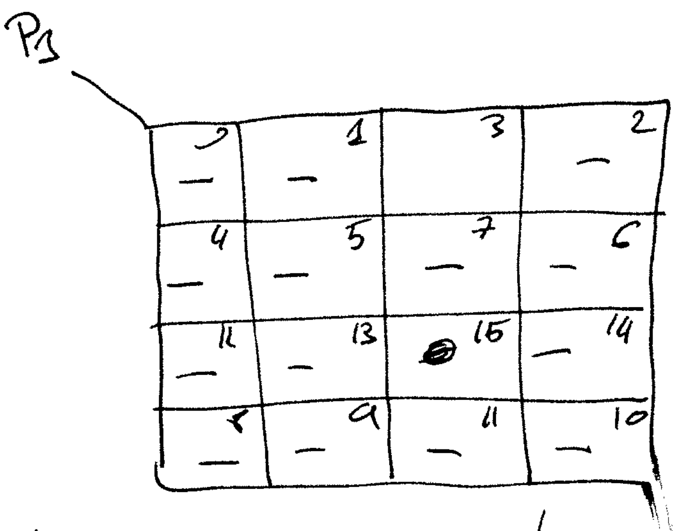
X	Q_2	Q_1	Q_0	P_2	P_1	P_0	L	C
0	0	0	0	–	–	–	0	1
0	0	0	1	–	–	–	0	1
0	0	1	0	–	–	–	0	1
0	0	1	1	0	0	0	1	–
0	1	0	0	–	–	–	0	1
0	1	0	1	–	–	–	0	1
0	1	1	0	–	–	–	0	1
0	1	1	1	–	–	–	0	1
1	0	0	0	–	–	–	0	1
1	0	0	1	–	–	–	0	1
1	0	1	0	–	–	–	0	1
1	0	1	1	–	–	–	0	1
1	1	0	0	–	–	–	0	1
1	1	0	1	–	–	–	0	1
1	1	1	0	–	–	–	0	1
1	1	1	1	1	0	0	1	–



Vamos a ver ahora como es (son) los circuitos combinacionales q generan las señales $P_2, P_1 P_0, L$ y C

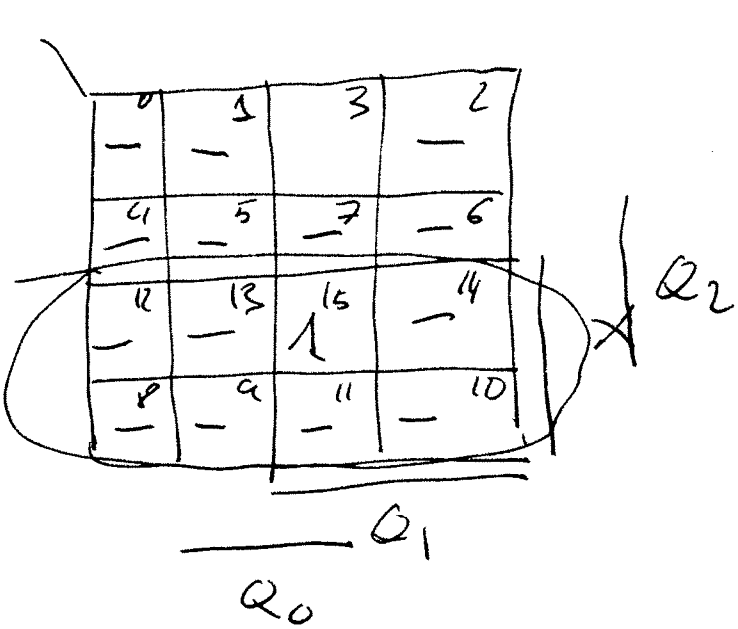
$C = 1 \rightarrow$ puesto q. casi todas sus valores son 1 salvo 2 don't care

$P_1 = P_0 \rightarrow$



En este caso, si consideramos los don't care $= \phi$ todas las ~~sal~~ valores son ϕ por lo tanto $P_1 = P_0 = \phi$

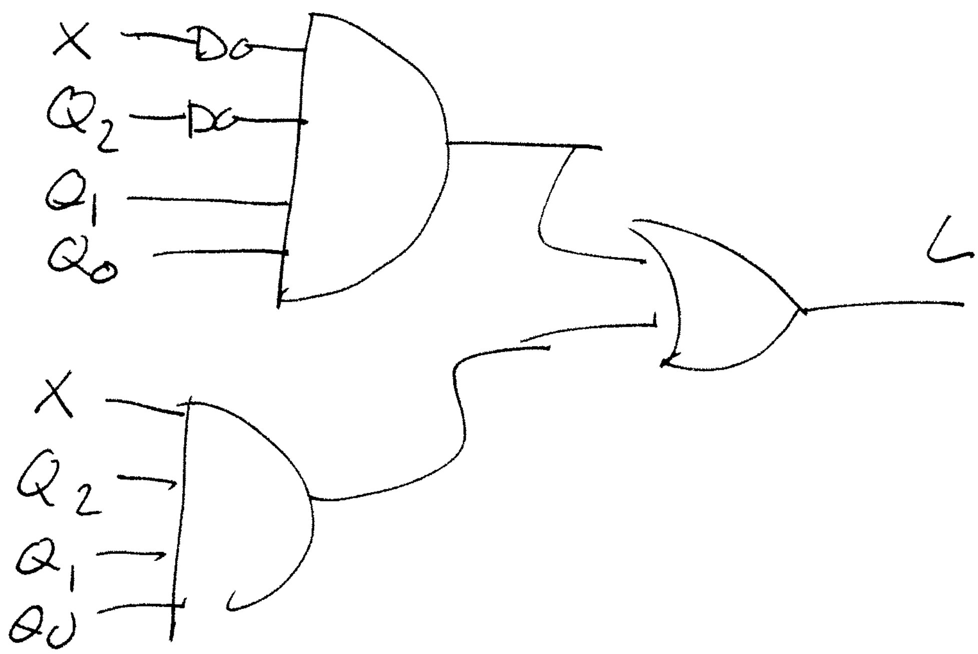
P_2



$P_2 = X$

(16)

L no se puede simplificar. será la suma de los minterminos 8 y 15



(b) El protocolo HDLC envía datos serie. se sabe q. al principio y al final de la información se insertan el código

01111110

El problema es q. esa combinación de 0's y 1's puede aparecer de manera casual en la información q. se está enviando por lo q. el receptor interpretará q. ha acabado la transmisión lo que da lugar a q. la información recibida sea incorrecta.

para evitar este problema se realiza la siguiente acción. Se controla la información q. se está enviando y si se detectan 5 1's seguidos se introduce un ϕ (q. después será eliminado en el receptor)

El problema pide implementar el detector de 5 1s seguidos. Es decir pide implementar el reconocedor de patrón "11111" mediante una máquina Mealy

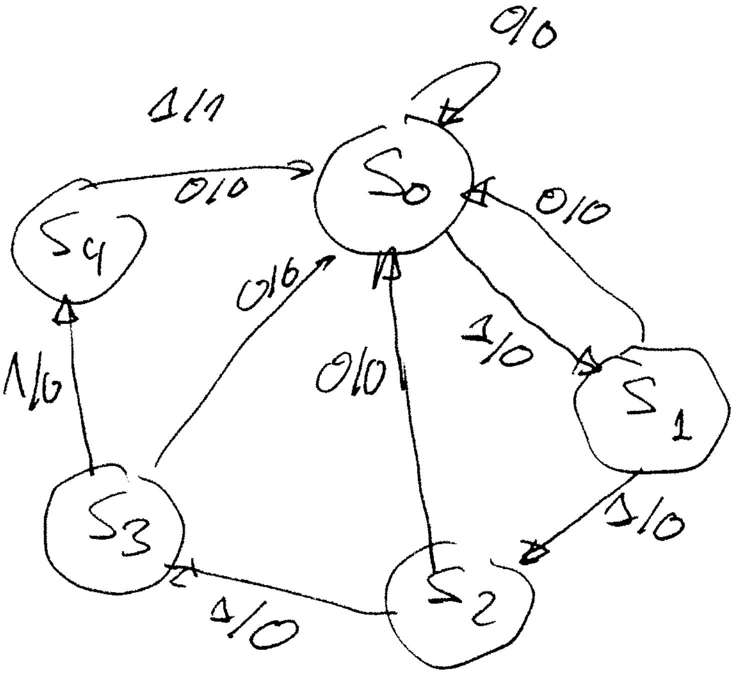
S_0 → No ha llegado el 1º elemento del patrón
↳ reset

S_1 → 1

S_2 → 11

S_2 → 111

S_4 − 1111



18

	$E_2E_1E_0$
S0	000
S1	001
S2	010
S3	011
S4	100

$(E_2E_1E_0)$ → estado actual
↳ el q. se almacena en los biestables

$(e_2e_1e_0)$ → siguiente estado

$E_2E_1E_0X$	$e_2e_1e_0Z$
0000	0000
0001	0010
0010	0000
0011	0100
0100	0000
0101	0110
0110	0000
0111	1000
1000	0000
1001	0001
1010	dddd
1011	dddd
1100	dddd
1101	dddd
1110	dddd
1111	dddd

como se utiliza una ROM no hace falta utilizar mapas de K.

puesto q. el don't care no se equivale con ningun valor real los sustituyo por 0s