CÓDIGOS

Prof. L. F. Caram

CÓDIGOS BINARIOS

Ejercicios 1-2-3

Un Código Binario representa cualquier dato utilizando un sistema de dos símbolos (0 y 1, el sistema binario), esto se llama **BINARIO NATURAL**.

Con 4 bits puedo representar 16 combinaciones enteras del 0 al 15, binario natural de 4 bits. Es un código **pesado o ponderado** es decir, cada posición de una secuencia de dígitos o palabra del código tiene asociado un peso, una potencia de dos, así el número decimal equivalente es el resultado de sumar el valor de la posición de los dígitos binarios que constituyen el código multiplicado por el peso correspondiente. En este código los pesos de los dígitos son las potencias sucesivas de dos.

Un **código BCD (Decimal Codificado a Binario)** solamente tiene 10 combinaciones o posee **módulo 10 (cantidad de palabras de un código o combinaciones**) del 0 al 9. Es decir cualquier código de módulo 10 será BCD. En la tabla podemos ver el **BCD Natural** o **BCD 8421** y es un **código pesado**.

BINARIO NATURAL

N°	A	B	C	D
	2^3 8	2^2 4	2^1 2	2^0 1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

BCD NATURAL

N°	A	B	C	D
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

El módulo de código en general se calcula como 2^n, siendo n el número de bits o variables. Esto se cumple siempre salvo en el caso del código Johnson donde el módulo es $2n$.

CÓDIGOS BINARIOS

Ejercicios 6 y 7

EX – 3

N°	A	B	C	D
–3	0	0	0	0
–2	0	0	0	1
–1	0	0	1	0
0	0	0	1	1
1	0	1	0	0
2	0	1	0	1
3	0	1	1	0
4	0	1	1	1
5	1	0	0	0
6	1	0	0	1
7	1	0	1	0
8	1	0	1	1
9	1	1	0	0
10	1	1	0	1
11	1	1	1	0
12	1	1	1	1

BCD EX – 3

N°	A	B	C	D
0	0	0	1	1
1	0	1	0	0
2	0	1	0	1
3	0	1	1	0
4	0	1	1	1
5	1	0	0	0
6	1	0	0	1
7	1	0	1	0
8	1	0	1	1
9	1	1	0	0

BCD NATURAL

N°	A	B	C	D
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

En el código de exceso 3 EX-3, los números se representan como dígitos decimales, y cada dígito se representa con cuatro bits como el valor del dígito más 3 (la cantidad "en exceso”). Éste es un código no ponderado, para obtener la relación entre el código decimal y este binario, se suma a cada dígito tres y después se convierte el resultado obtenido en binario 8421. Ejemplo 18 = 0100 1011. Es decir 1 (0100) y 8 (1011).

Un código binario es **autocomplementado, autocomplementable o autocomplementario,** cuando el complemento a 9 del equivalente decimal de cualquier palabra del código se puede encontrar invirtiendo los valores de cada uno de los bits (complemento al módulo menos uno) y el resultado sigue siendo una palabra de ese código. Por ejemplo el BDC EX-3, es autocomplementado pero no es pesado.

CÓDIGOS BINARIOS

Ejercicios 8-10

La **Distancia de Hamming** se define como el número de bits que tienen que cambiarse para transformar una palabra código en otra. Es decir la distancia entre dos combinaciones es el número de bits que cambian de una a otra.
Con el concepto de distancia se puede definir la **distancia mínima de un código**, que es la distancia menor que haya entre dos de las palabras de ese código.

GRAY

N°	A	B	C	D
0	0	0	0	0
1	0	0	0	1
2	0	0	1	1
3	0	0	1	0
4	0	1	1	0
5	0	1	1	1
6	0	1	0	1
7	0	1	0	0
8	1	1	0	0
9	1	1	0	1
10	1	1	1	1
11	1	1	1	0
12	1	0	1	0
13	1	0	1	1
14	1	0	0	1
15	1	0	0	0

El algoritmo para construir un **código de Gray** es el siguiente, se coloca un 0 y debajo un 1, luego se traza una línea o espejo donde se refleja la columna de bits de la mitad superior, es decir 1 y 0; por último se rellena la siguiente columna mas significativa, del peso siguiente, la mitad superior con 0's y la segunda con 1's. Así se obtiene un Gray de 2-bits, de esta forma iterando el mecanismo, se obtienen los sucesivos Gray de 3, 4, y n bits. El **código GRAY es Reflejado o Espejado** por la manera en que se construye y además la distancia entre una palabra cualquiera del código y la siguiente o anterior es 1, a eso se le llama **Progresivo o Continuo** y si también la distancia entre la primera y la última palabra es 1, entonces el código es **Cerrado o Cíclico**.
De esta forma el código Gray es un código reflejado o espejado, progresivo o continuo y cerrado o cíclico y no es pesado.

CÓDIGOS BINARIOS

Ejercicio 9

GRAY

N°	A	B	C	D
0	0	0	0	0
1	0	0	0	1
2	0	0	1	1
3	0	0	1	0
4	0	1	1	0
5	0	1	1	1
6	0	1	0	1
7	0	1	0	0
8	1	1	0	0
9	1	1	0	1
10	1	1	1	1
11	1	1	1	0
12	1	0	1	0
13	1	0	1	1
14	1	0	0	1
15	1	0	0	0

BCD GRAY

N°	A	B	C	D
0	0	0	0	0
1	0	0	0	1
2	0	0	1	1
3	0	0	1	0
4	0	1	1	0
5	1	1	1	0
6	1	0	1	0
7	1	0	1	1
8	1	0	0	1
9	1	0	0	0

GRAY

N°	A	B	C	D
0	0	0	0	0
1	0	0	0	1
2	0	0	1	1
3	0	0	1	0
4	0	1	1	0
5	0	1	1	1
6	0	1	0	1
7	0	1	0	0
8	1	1	0	0
9	1	1	0	1
10	1	1	1	1
11	1	1	1	0
12	1	0	1	0
13	1	0	1	1
14	1	0	0	1
15	1	0	0	0

BCD GRAY EX – 3

N°	A	B	C	D
0	0	0	1	0
1	0	1	1	0
2	0	1	1	1
3	0	1	0	1
4	0	1	0	0
5	1	1	0	0
6	1	1	0	1
7	1	1	1	1
8	1	1	1	0
9	1	0	1	0

Para poder obtener un código BCD Gray y un BCD Gray EX-3, debo lograr códigos de módulo 10, mientras que el Gray de 4-bits tiene modulo 16, es decir debo quitar 6 palabras, pero considerando que los nuevos códigos deben seguir teniendo las propiedades del Gray, es decir reflejado, progresivo y cerrado. Entonces no puedo quitar las últimas 6 palabras como hago con el BCD 8421, debo quitar 6 palabras en forma simétrica y tengo dos opciones, si saco las 6 del medio como se muestra en rojo, obtengo el código BCD Gray; mientras que si saco las 3 primeras y las 3 últimas palabras como se muestra en verde, entonces obtengo el código BCD Gray EX-3.

CÓDIGOS BINARIOS

<u>Ejercicio 12</u> Un **<u>código BCD JOHNSON</u>** se construye desplazando unos de derecha a izquierda hasta completar el registro (ancho de palabra) y luego con ceros.

El módulo del código Johnson es $2n$ siendo n el número de bits o variables.

BCD JOHNSON

N°	A	B	C	D	D
0	0	0	0	0	0
1	0	0	0	0	1
2	0	0	0	1	1
3	0	0	1	1	1
4	0	1	1	1	1
5	1	1	1	1	1
6	1	1	1	1	0
7	1	1	1	0	0
8	1	1	0	0	0
9	1	0	0	0	0

CÓDIGOS BINARIOS

CÓDIGO DE HAMMING

Ejercicio 14: para palabras de datos de 8 bits

El **Código de Hamming es un método para detectar y corregir errores** en una transmisión. En particular **este método permite solamente detectar y corregir solamente 1 error** o solo detectar 2 errores pero no corregirlos. Por el contrario el código de paridad simple no puede corregir errores y solo puede detectar un número impar de bits en error. Para poder lograr esto, Hamming tenía que **aumentar la distancia del código**, es decir debía tener una distancia mayor o igual a 3, la forma de conseguirlo era agregando bits redundantes o de paridad e intercalarlos en las posiciones potencias de 2 dentro de la palabra bloque a enviar, es decir bits de datos o información intercalados con los bits de paridad. Asi se debe cumplir lo siguiente:

$$\left.\begin{array}{r} \mathbf{D \leq d_{min} - 1} \\ \mathbf{2C \leq d_{min} - 1} \end{array}\right\} \boxed{\mathbf{D + C \leq d_{min} - 1}}$$

$\mathbf{d_{min}}$: DISTANCIA MÍNIMA

D : Cantidad de Errores a Detectar

C : Cantidad de Errores a Corregir

Para entero $\mathbf{p}$ hay una código con largo de palabra bloque $\mathbf{n = 2^p - 1}$ y largo del mensaje $\mathbf{k = 2^p - p - 1}$. Por lo tanto la cantidad de bits redundantes o de paridad necesarios según la cantidad de bits de información será:

CANTIDAD DE BITS DE PARIDAD p NECESARIOS

PARA ENVIAR k BITS DE INFORMACIÓN

$$\boxed{\mathbf{2^p \geq p + k + 1}}$$

p : Cantidad de Bits de Paridad o Redundantes

k : Cantidad de Bits de Información

CÓDIGOS BINARIOS

CÓDIGO DE HAMMING: GENERADOR DE PARIDAD

La idea principal del algoritmo es elegir los bits de paridad o de corrección de errores, de modo que la operación EX-OR (el XOR de todas las posiciones de bits que contengan un 1) sea 0. Utilizamos las posiciones 1, 10, 100, etc. (en binario) como bits de corrección de errores o bits de paridad, lo que garantiza que es posible generar los bits de paridad de manera tal que la operación XOR de todo el mensaje sea 0.

Es decir para generar p_1 debo realizar el XOR de todos los bits de información que terminan en 1 (conjunto de bits menos significativo en 1) en la columna de p_1, es decir todos los impares que finalizan en 1. Luego para generar p_2 haré el XOR de todos los bit de información que tengan 1 en la columna de p_2, (segundo conjunto de bit menos significativo en 1) y asi sucesivemente.

CÓDIGO DE HAMMING: DETECTOR DE PARIDAD

Para detectar si hubo o no error debo realizar el cálculo de las c_i, asi para calcular c_1, haré el XOR de todos los bits que terminan en 1 (conjunto de bits menos significativo en 1) en la columna de c_1 esta vez incluyendo el bit de paridad. Para calcular c_2 haré el XOR de todos los bits el 1 en la columna de c_2 y asi sucesivamente. Luego si la combinación de todos los c_i es igual a 0, entonces no hubo error, mientras que si es distinta de 0, la combinación de los c_i nos dirá el índice del bit dañado.

CÓDIGOS BINARIOS

CÓDIGO DE HAMMING: GENERADOR DE PARIDAD

Ejercicio 14: ejemplo para palabras de datos de 8 bits

p_8	p_4	p_2	p_1	
0	0	0	0	
0	0	0	1	p_1
0	0	1	0	p_2
0	0	1	1	m_3
0	1	0	0	p_4
0	1	0	1	m_5
0	1	1	0	m_6
0	1	1	1	m_7
1	0	0	0	p_8
1	0	0	1	m_9
1	0	1	0	m_{10}
1	0	1	1	m_{11}
1	1	0	0	m_{12}

BITS DE INFORMACIÓN : $m_3\ m_5\ m_6\ m_7\ m_9\ m_{10}\ m_{11}\ m_{12}$

0 1 1 0 0 1 0 1

CÁLCULO DE BITS DE PARIDAD

$$p_1 = m_3 \oplus m_5 \oplus m_7 \oplus m_9 \oplus m_{11} = 0 \oplus 1 \oplus 0 \oplus 0 \oplus 0 = 1$$

$$p_2 = m_3 \oplus m_6 \oplus m_7 \oplus m_{10} \oplus m_{11} = 0 \oplus 1 \oplus 0 \oplus 1 \oplus 0 = 0$$

$$p_4 = m_5 \oplus m_6 \oplus m_7 \oplus m_{12} = 1 \oplus 1 \oplus 0 \oplus 1 = 1$$

$$p_8 = m_9 \oplus m_{10} \oplus m_{11} \oplus m_{12} = 0 \oplus 1 \oplus 0 \oplus 1 = 0$$

PALABRA BLOQUE A ENVIAR

$p_1\ p_2\ m_3\ p_4\ m_5\ m_6\ m_7\ p_8\ m_9\ m_{10}\ m_{11}\ m_{12}$

1 0 0 1 1 1 0 0 0 1 0 1

CÓDIGOS BINARIOS

CÓDIGO DE HAMMING: DETECTOR DE PARIDAD

c_8	c_4	c_2	c_1	
0	0	0	0	$\nexists$ **error**
0	0	0	1	$\mathbf{p_1}$
0	0	1	0	$\mathbf{p_2}$
0	0	1	1	m_3
0	1	0	0	$\mathbf{p_4}$
0	1	0	1	m_5
0	1	1	0	m_6
0	1	1	1	m_7
1	0	0	0	$\mathbf{p_8}$
1	0	0	1	m_9
1	0	1	0	m_{10}
1	0	1	1	m_{11}
1	1	0	0	m_{12}

SI SE RECIBE LA PALABRA BLOQUE

$p_1\ p_2\ m_3\ p_4\ m_5\ m_6\ m_7\ p_8\ m_9\ m_{10}\ m_{11}\ m_{12}$

1 0 0 1 1 1 0 0 0 1 0 1

NO HAY ERROR

$c_8\ c_4\ c_2\ c_1 = 0\ 0\ 0\ 0$

DETECCIÓN DE LA POSICIÓN DEL ERROR

$c_1 = p_1 \oplus m_3 \oplus m_5 \oplus m_7 \oplus m_9 \oplus m_{11} = 1 \oplus 0 \oplus 1 \oplus 0 \oplus 0 \oplus 0 = 0$

$c_2 = p_2 \oplus m_3 \oplus m_6 \oplus m_7 \oplus m_{10} \oplus m_{11} = 0 \oplus 0 \oplus 1 \oplus 0 \oplus 1 \oplus 0 = 0$

$c_4 = p_4 \oplus m_5 \oplus m_6 \oplus m_7 \oplus m_{12} = 1 \oplus 1 \oplus 1 \oplus 0 \oplus 1 = 0$

$c_8 = p_8 \oplus m_9 \oplus m_{10} \oplus m_{11} \oplus m_{12} = 0 \oplus 0 \oplus 1 \oplus 0 \oplus 1 = 0$

SI SE RECIBE LA PALABRA BLOQUE (BIT 10 ERRADO)

$p_1\ p_2\ m_3\ p_4\ m_5\ m_6\ m_7\ p_8\ m_9\ m_{10}\ m_{11}\ m_{12}$

1 0 0 1 1 1 0 0 0 0 0 1

ERROR EN EL BIT 10

$c_8\ c_4\ c_2\ c_1 = 1\ 0\ 1\ 0$

DETECCIÓN DE LA POSICIÓN DEL ERROR

$c_1 = p_1 \oplus m_3 \oplus m_5 \oplus m_7 \oplus m_9 \oplus m_{11} = 1 \oplus 0 \oplus 1 \oplus 0 \oplus 0 \oplus 0 = 0$

$c_2 = p_2 \oplus m_3 \oplus m_6 \oplus m_7 \oplus m_{10} \oplus m_{11} = 0 \oplus 0 \oplus 1 \oplus 0 \oplus 0 \oplus 0 = 1$

$c_4 = p_4 \oplus m_5 \oplus m_6 \oplus m_7 \oplus m_{12} = 1 \oplus 1 \oplus 1 \oplus 0 \oplus 1 = 0$

$c_8 = p_8 \oplus m_9 \oplus m_{10} \oplus m_{11} \oplus m_{12} = 0 \oplus 0 \oplus 0 \oplus 0 \oplus 1 = 1$

CÓDIGOS BINARIOS

CÓDIGO DE HAMMING: DETECTOR DE PARIDAD

c_8	c_4	c_2	c_1	
0	0	0	0	$\nexists$ error
0	0	0	1	p_1
0	0	1	0	p_2
0	0	1	1	m_3
0	1	0	0	p_4
0	1	0	1	m_5
0	1	1	0	m_6
0	1	1	1	m_7
1	0	0	0	p_8
1	0	0	1	m_9
1	0	1	0	m_{10}
1	0	1	1	m_{11}
1	1	0	0	m_{12}

¿Qué pasa si cambian dos bits simultaneamente?
Veamos como el método falla

SI SE RECIBE LA PALABRA BLOQUE (BITS 3 Y 12 ERRADOS)

$$\begin{array}{cccccccccccc} p_1 & p_2 & m_3 & p_4 & m_5 & m_6 & m_7 & p_8 & m_9 & m_{10} & m_{11} & m_{12} \\ 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \end{array}$$

ERROR EN EL BIT 15

$c_8\ c_4\ c_2\ c_1 = 1\ 1\ 1\ 1$

DETECCIÓN DE LA POSICIÓN DEL ERROR

$$c_1 = p_1 \oplus m_3 \oplus m_5 \oplus m_7 \oplus m_9 \oplus m_{11} = 1 \oplus 1 \oplus 1 \oplus 0 \oplus 0 \oplus 0 = 1$$
$$c_2 = p_2 \oplus m_3 \oplus m_6 \oplus m_7 \oplus m_{10} \oplus m_{11} = 0 \oplus 1 \oplus 1 \oplus 0 \oplus 1 \oplus 0 = 1$$
$$c_4 = p_4 \oplus m_5 \oplus m_6 \oplus m_7 \oplus m_{12} = 1 \oplus 1 \oplus 1 \oplus 0 \oplus 0 = 1$$
$$c_8 = p_8 \oplus m_9 \oplus m_{10} \oplus m_{11} \oplus m_{12} = 0 \oplus 0 \oplus 1 \oplus 0 \oplus 0 = 1$$

Por lo tanto si cambian dos bits simultaneamente, el método falla, pues el error según el método esta en el bit 15 que no existe en la palabra bloque.

De todos modos el método es utilizado aún, ya que la probabilidad de que cambien dos o mas bits simultaneamente es extremadamente baja con la tecnología utilizada actualmente.

CÓDIGOS BINARIOS

CÓDIGO DE HAMMING: GENERADOR DE PARIDAD

Generalizando para 4 bits de paridad puedo enviar hasta 11 de información

p_8	p_4	p_2	p_1	
0	0	0	0	
0	0	0	1	$\mathbf{p_1}$
0	0	1	0	$\mathbf{p_2}$
0	0	1	1	m_3
0	1	0	0	$\mathbf{p_4}$
0	1	0	1	m_5
0	1	1	0	m_6
0	1	1	1	m_7
1	0	0	0	$\mathbf{p_8}$
1	0	0	1	m_9
1	0	1	0	m_{10}
1	0	1	1	m_{11}
1	1	0	0	m_{12}
1	1	0	1	m_{13}
1	1	1	0	m_{14}
1	1	1	1	m_{15}

BITS DE INFORMACIÓN : m_3 m_5 m_6 m_7 m_9 m_{10} m_{11} m_{12} m_{13} m_{14} m_{15}

CÁLCULO DE BITS DE PARIDAD

$$p_1 = m_3 \oplus m_5 \oplus m_7 \oplus m_9 \oplus m_{11} \oplus m_{13} \oplus m_{15}$$

$$p_2 = m_3 \oplus m_6 \oplus m_7 \oplus m_{10} \oplus m_{11} \oplus m_{14} \oplus m_{15}$$

$$p_4 = m_5 \oplus m_6 \oplus m_7 \oplus m_{12} \oplus m_{13} \oplus m_{14} \oplus m_{15}$$

$$p_8 = m_9 \oplus m_{10} \oplus m_{11} \oplus m_{12} \oplus m_{13} \oplus m_{14} \oplus m_{15}$$

PALABRA BLOQUE A ENVIAR

p_1 p_2 m_3 p_4 m_5 m_6 m_7 p_8 m_9 m_{10} m_{11} m_{12} m_{13} m_{14} m_{15}

CÓDIGOS BINARIOS

CÓDIGO DE HAMMING: DETECTOR DE PARIDAD

c_8	c_4	c_2	c_1	
0	0	0	0	**∄ error**
0	0	0	$\underline{1}$	$\mathbf{p_1}$
0	0	$\underline{1}$	0	$\mathbf{p_2}$
0	0	$\underline{1}$	$\underline{1}$	m_3
0	$\underline{1}$	0	0	$\mathbf{p_4}$
0	$\underline{1}$	0	$\underline{1}$	m_5
0	$\underline{1}$	$\underline{1}$	0	m_6
0	$\underline{1}$	$\underline{1}$	$\underline{1}$	m_7
$\underline{1}$	0	0	0	$\mathbf{p_8}$
$\underline{1}$	0	0	$\underline{1}$	m_9
$\underline{1}$	0	$\underline{1}$	0	m_{10}
$\underline{1}$	0	$\underline{1}$	$\underline{1}$	m_{11}
$\underline{1}$	$\underline{1}$	0	0	m_{12}
$\underline{1}$	$\underline{1}$	0	$\underline{1}$	m_{13}
$\underline{1}$	$\underline{1}$	$\underline{1}$	0	m_{14}
$\underline{1}$	$\underline{1}$	$\underline{1}$	$\underline{1}$	m_{15}

DETECCIÓN DE LA POSICIÓN DEL ERROR

$$c_1 = p_1 \oplus m_3 \oplus m_5 \oplus m_7 \oplus m_9 \oplus m_{11} \oplus m_{13} \oplus m_{15}$$
$$c_2 = p_2 \oplus m_3 \oplus m_6 \oplus m_7 \oplus m_{10} \oplus m_{11} \oplus m_{14} \oplus m_{15}$$
$$c_4 = p_4 \oplus m_5 \oplus m_6 \oplus m_7 \oplus m_{12} \oplus m_{13} \oplus m_{14} \oplus m_{15}$$
$$c_8 = p_8 \oplus m_9 \oplus m_{10} \oplus m_{11} \oplus m_{12} \oplus m_{13} \oplus m_{14} \oplus m_{15}$$

PALABRA BLOQUE RECIBIDA

$$p_1\ p_2\ m_3\ p_4\ m_5\ m_6\ m_7\ p_8\ m_9\ m_{10}\ m_{11}\ m_{12}\ m_{13}\ m_{14}\ m_{15}$$

Bits de Paridad	Bits Totales de la palabra bloque	Bits de información	Nombre	Tasa
p	$n = 2^p - 1$	$k = 2^p - 1 - p$	$H(2^p - 1, 2^p - 1 - p)$	$\frac{(2^p - 1 - p)}{(2^p - 1)}$
2	3	1	H(3,1)	1/3 ≈ 0.333
3	7	4	H(7,4)	4/7 ≈ 0.571
4	15	11	H(15,11)	11/15 ≈ 0.733
5	31	26	H(31,26)	26/31 ≈ 0.839
6	63	57	H(63,57)	57/63 ≈ 0.905
7	127	120	H(127,120)	120/127 ≈ 0.945
8	255	247	H(255,247)	247/255 ≈ 0.969
9	511	502	H(511,502)	502/511 ≈ 0.982
10	1023	1013	H(1023,1013)	1013/1023 ≈ 0.99