Tema 2 : Códigos Binarios

Objetivo:

- Conocer diferentes códigos binarios
- Conocer algunos códigos de detección y corrección de errores.

Códigos Binarios

•No toda la información que maneja un sistema digital es numérica, e inclusive, para la información numérica a veces no es conveniente utilizar el sistema binario descrito en el capítulo anterior, por ejemplo:

- Cuando se busca una conversión sencilla decimal binario (Código BCD)
- Cuando se busca minimizar errores de sensado en Encoders (Gray)
- Cuando se quiere detectar/corregir errores en transmisiones (Hamming)

•Por todo ello es conveniente idear otras formas diferentes de codificar información usando ceros y unos

•Existen diferentes códigos binarios BCD natural, BCD XS3, GRAY, AIKEN, PARIDAD, HAIKEN etc. que poseerán ciertas ventajas frente al código natural binario

Códigos Binarios :Conceptos previos

- **Códigos ponderados**: Se dice que un código es ponderado cuando cada bit tiene un valor diferente dependiendo la posición que ocupe.
- **Código Continuo**: Es aquel código en el que si nos fijamos en dos cantidades contiguas, éstas sólo variaran en el valor de un bit.
- **Código continuo cíclico:** Un código continuo se dice que es cíclico cuando la primera y la última cantidad sólo se diferencian en el valor de un bit.
- **Código autocomplementario :** Un código es autocomplementario de 9 si el código de una cifra d cualquiera es igual al de la cifra 9-d cambiándole los 0 por 1 y los 1 por 0 (complementar)

Codificación decimal en Binario

- Los códigos BCD nos permiten representar cada uno de los dígitos decimales.
- Los dígitos decimales son 10, por tanto necesitamos códigos que al menos tengan 10 combinaciones diferentes.
  - Si usamos códigos de 3 bits podremos representar hasta 8 dígitos diferentes (000-111) por lo que es **insuficiente**
  - Si usamos códigos de 4 bits podremos representar hasta 16 dígitos diferentes (0000-1111) por lo que es **más que suficiente**

Codificación decimal en Binario

Decimal	BCD Natural				Biquinario BCD							BCD XS3 Autoc. 9				Gray (2^n) cont.cícl.refl				BCD Johnson (5bits)				
Peso	**8**	**4**	**2**	**1**	**5**	**0**	**4**	**3**	**2**	**1**	**0**													
0	0	0	0	0	0	1	0	0	0	0	1	0	0	1	1	0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	1	0	0	0	1	0	0	1	0	0	0	0	0	1	0	0	0	0	1
2	0	0	1	0	0	1	0	0	1	0	0	0	1	0	1	0	0	1	1	0	0	0	1	1
3	0	0	1	1	0	1	0	1	0	0	0	0	1	1	0	0	0	1	0	0	0	1	1	1
4	0	1	0	0	0	1	1	0	0	0	0	0	1	1	1	0	1	1	0	0	1	1	1	1
5	0	1	0	1	1	0	0	0	0	0	1	1	0	0	0	0	1	1	1	1	1	1	1	1
6	0	1	1	0	1	0	0	0	0	1	0	1	0	0	1	0	1	0	1	1	1	1	1	0
7	0	1	1	1	1	0	0	0	1	0	0	1	0	1	0	0	1	0	0	1	1	1	0	0
8	1	0	0	0	1	0	0	1	0	0	0	1	0	1	1	1	1	0	0	1	1	0	0	0
9	1	0	0	1	1	0	1	0	0	0	0	1	1	0	0	1	1	0	1	1	0	0	0	0

Decimal	BCD AIKEN Autoc. 9				BCD AIKEN				Gray 3 bits			Gray 2 bits	
Peso	2	4	2	1	5	4	2	1					
0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1	0	0	1	0	1
2	0	0	1	0	0	0	1	0	0	1	1	1	1
3	0	0	1	1	0	0	1	1	0	1	0	1	0
4	0	1	0	0	0	1	0	0	1	1	0		
5	1	0	1	1	1	0	0	0	1	1	1		
6	1	1	0	0	1	0	0	1	1	0	1		
7	1	1	0	1	1	0	1	0	1	0	0		
8	1	1	1	0	1	0	1	1					
9	1	1	1	1	1	1	0	0					

CÓDIGO BCD Natural o BCD_{8421}

- Es la codificación más corriente que se suele emplear.
- Consiste en asociar a cada dígito su valor en el sistema de numeración binario.
- Una cifra formada por varios dígitos decimales se forma por la concatenación de dichos dígitos expresados cada uno de ellos en binario y manteniendo el mismo orden relativo que la cifra inicial
- Ejemplo

$$256_{10} \rightarrow [0010][0101][0110] \rightarrow 001001010110_{BCD}$$

$$001110011000_{BCD} \rightarrow [0011][1001][1000] \rightarrow 398_{10}$$

CÓDIGO BCD XS3 (Exceso 3)

- Esta codificación es igual a la BCD natural excepto que el código de cada cifra es el de 3 unidades superior al correspondiente en el sistema binario de numeración.
- Una cifra formada por varios dígitos decimales se forma por la concatenación de dichos dígitos expresados cada uno de ellos en XS3 y manteniendo el mismo orden relativo que la cifra inicial
- El XS3 es un código Autocomplementario
- Ejemplo

$$256_{10} \rightarrow [0101][1000][1001] \rightarrow 010110001001_{XS3}$$

$$001110011000_{XS3} \rightarrow [0011][1001][1000] \rightarrow 065_{10}$$

CÓDIGO BCD AIKEN 2421

- El código AIKEN 2421 es también autocomplementario y su nombre 2421 indica la ponderación de sus bits.
- Una cifra formada por varios dígitos decimales se forma por la concatenación de dichos dígitos expresados cada uno de ellos en Aiken 2421 y manteniendo el mismo orden relativo que la cifra inicial
- Ejemplo

$256_{10} = [0010][1011][1100] = 001010111100_{Aiken2421}$

$001010111100_{Aiken2421} = [0010][1011][1100] = 256_{10}$

CÓDIGO BIQUINARIO o código 2 de 5

•En este tipo de codificación usamos los dos bits de mayor peso para indicar si el dígito es mayor o menor que 5.

	Biquinario						
DECIMAL	**5**	**0**	**4**	**3**	**2**	**1**	**0**
0	**0**	**1**	**0**	**0**	**0**	**0**	**1**
1	**0**	**1**	**0**	**0**	**0**	**1**	**0**
2	**0**	**1**	**0**	**0**	**1**	**0**	**0**
3	**0**	**1**	**0**	**1**	**0**	**0**	**0**
4	**0**	**1**	**1**	**0**	**0**	**0**	**0**
5	**1**	**0**	**0**	**0**	**0**	**0**	**1**
6	**1**	**0**	**0**	**0**	**0**	**1**	**0**
7	**1**	**0**	**0**	**0**	**1**	**0**	**0**
8	**1**	**0**	**0**	**1**	**0**	**0**	**0**
9	**1**	**0**	**1**	**0**	**0**	**0**	**0**

$$894_{10} \rightarrow [1001000][1010000][0110000] \rightarrow 100100010100000100000_{biquinario}$$

$$010010010001000100100_{biquinario} \rightarrow [0100100][1000100][0100100] \rightarrow 272_{10}$$

CÓDIGO BIQUINARIO o código 2 de 5

•El código biquinario es un código no ponderado, es decir, los códigos no pueden obtenerse aplicando una expresión polinomial.

• El nombre código 2 de 5 indica que en la codificación de cada cifra decimal, solamente 2 de los 5 bits puden estar a uno

$$894_{10} \rightarrow [1001000][1010000][0110000] \rightarrow 100100010100000100000_{biquinario}$$

$$0100100100010001001 00_{biquinario} \rightarrow [0100100][1000100][0100100] \rightarrow 272_{10}$$

CÓDIGO BIQUINARIO

•En este tipo de codificación usamos los dos bits de mayor peso para indicar si el dígito es mayor o menor que 5.

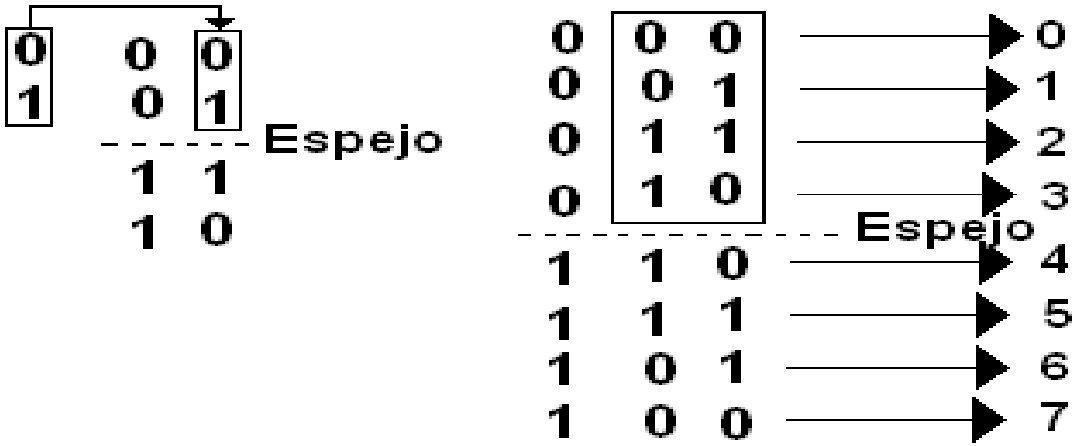
	Biquinario						
DECIMAL	**5**	**0**	**4**	**3**	**2**	**1**	**0**
0	**0**	**1**	**0**	**0**	**0**	**0**	**1**
1	**0**	**1**	**0**	**0**	**0**	**1**	**0**
2	**0**	**1**	**0**	**0**	**1**	**0**	**0**
3	**0**	**1**	**0**	**1**	**0**	**0**	**0**
4	**0**	**1**	**1**	**0**	**0**	**0**	**0**
5	**1**	**0**	**0**	**0**	**0**	**0**	**1**
6	**1**	**0**	**0**	**0**	**0**	**1**	**0**
7	**1**	**0**	**0**	**0**	**1**	**0**	**0**
8	**1**	**0**	**0**	**1**	**0**	**0**	**0**
9	**1**	**0**	**1**	**0**	**0**	**0**	**0**

$$894_{10} \rightarrow [1001000][1010000][0110000] \rightarrow 100100010100000100000_{biquinario}$$

$$010010010001000100100_{biquinario} \rightarrow [0100100][1000100][0100100] \rightarrow 272_{10}$$

CÓDIGO GRAY O ESPEJO

- Primero veremos como se forma el código gray:
  1. Partiremos del 0 y del 1
  2. Haremos el espejo de los datos anteriores.
  3. Colocaremos 0 por encima del último espejo y unos por debajo.
  4. El proceso continuará indefinidamente repitiendo los pasos 2 y 3



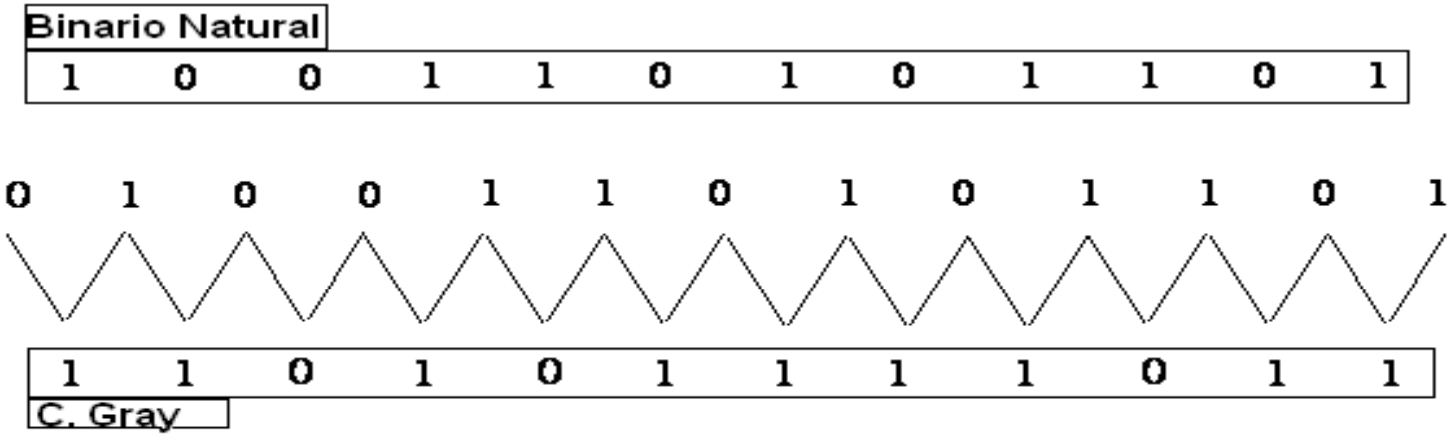
CÓDIGO GRAY O ESPEJO

Dígito Decimal	GRAY 4 bits				Dígito Decimal	GRAY 4 bits			
0	0	0	0	0	**8**	1	1	0	0
1	0	0	0	1	**9**	1	1	0	1
2	0	0	1	1	**10**	1	1	1	1
3	0	0	1	0	**11**	1	1	1	0
4	0	1	1	0	**12**	1	0	1	0
5	0	1	1	1	**13**	1	0	1	1
6	0	1	0	1	**14**	1	0	0	1
7	0	1	0	0	**15**	1	0	0	0

- Observar que en el código Gray una cifra no se forma por la concatenación de sus dígitos básicos traducidos a Gray. Este código es no ponderado, contínuo, cíclico y reflejado.

CONVERSIONES DEL C. GRAY (1/4)

- CONVERSIÓN DE SISTEMA BINARIO NATURAL A CÓDIGO GRAY (2 métodos):
- MÉTODO 1
- Seguiremos los pasos:
  1. Añadimos un 0 a la izquierda de la cifra en binario natural.
  2. Comparamos cada pareja de bits adyacentes:
     1. Si son iguales el resultado de la comparación es cero.
     2. Si son diferentes el resultado de la comparación es uno.



CONVERSIONES DEL C. GRAY (2/4)

- CONVERSIÓN DE SISTEMA BINARIO NATURAL A CÓDIGO GRAY (2 métodos):
- MÉTODO 2
- Pasos:
  1. El bit Izquierdo (MSB) es el mismo para la cifra en binario natural y su correspondiente código Gray
  2. Sumamos el MSB al bit situado inmediatamente a su derecha despreciando el acarreo, el bit obtenido será el segundo más significativo de nuestro código Gray
  3. Seguimos el mismo método hasta acabar con todos los dígitos

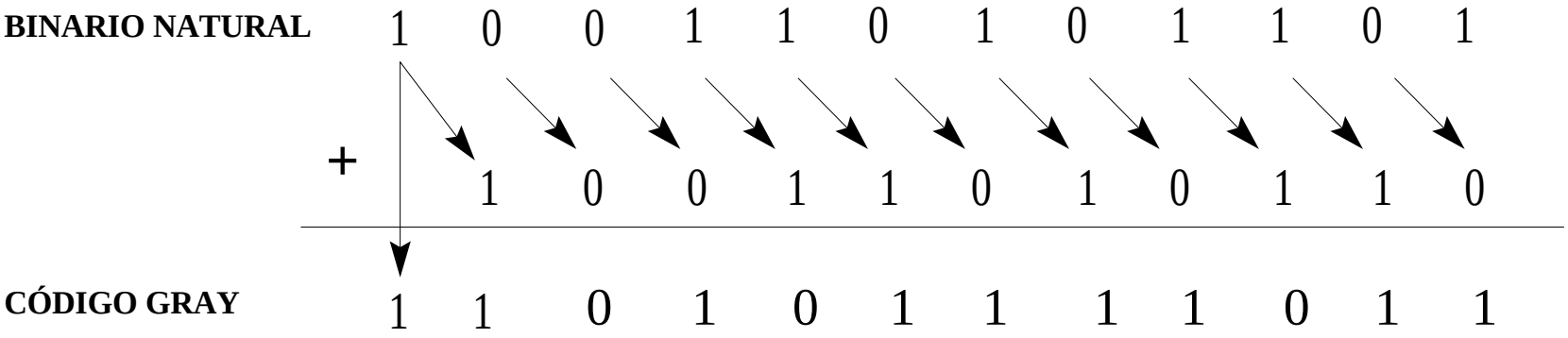
Nota: Aún no se ha estudiado la aritmética binaria, por lo que comentaremos brevemente como hacer la suma comentada en este tipo de conversiones, dejando la aritmética para un posterior tema.

CONVERSIONES DEL C. GRAY (3/4)

•Para realizar la suma de dos bits seguiremos la tabla:

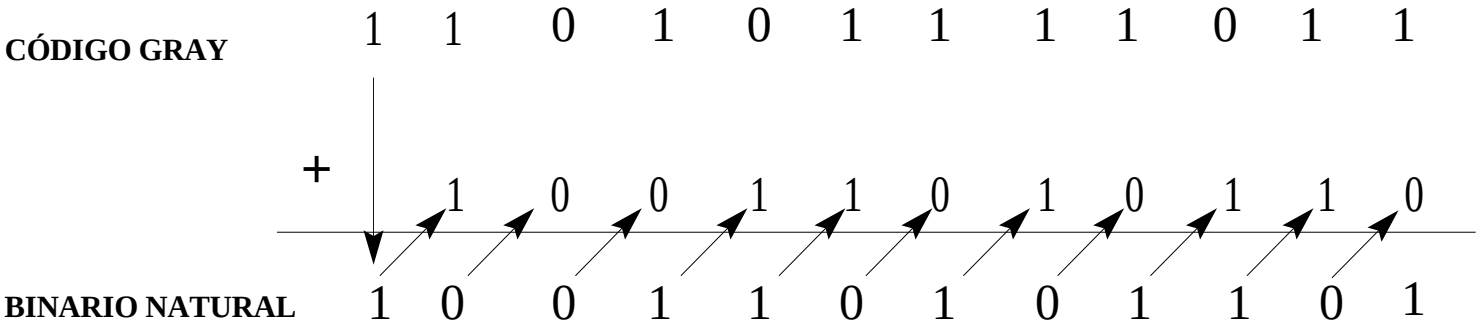
A	B	A+B	ACARREO (Llevada)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

•Ejemplo



CONVERSIONES DEL C. GRAY (4/4)

- CONVERSIÓN DE CÓDIGO GRAY A BINARIO NATURAL :
- Seguiremos los pasos:
  1. El bit Izquierdo (MSB) es el mismo para la cifra en binario natural y su correspondiente código Gray
  2. Sumamos el bit en binario natural recién obtenido (despreciando el acarreo) al siguiente bit Gray y así obtener el siguiente bit en binario natural
  3. Seguimos el mismo método hasta acabar con todos los dígitos



OTRAS CONVERSIONES

- Entre Sistema Binario Natural y BCD
  - S. Binario Natural ⟷ S. Decimal ⟷ BCD
- Entre Sistema Binario Natural y XS3
  - S. Binario Natural ⟷ S. Decimal ⟷ XS3
- Entre Sistema Octal y BCD
  - S. Octal ⟷ S. Decimal ⟷ BCD
- Entre Sistema Octal y XS3
  - S. Octal ⟷ S. Decimal ⟷ XS3
- Entre Sistema Hexadecimal y BCD
  - S. Hex. ⟷ S. Decimal ⟷ BCD
- Entre Sistema Hexadecimal y XS3
  - S. Hex. ⟷ S. Decimal ⟷ XS3
- Para códigos diferentes a los comentados o sistemas de numeración diferentes a los comentados, se seguirán los mismos pasos.

CÓDIGOS DETECTORES Y CORRECTORES DE ERRORES

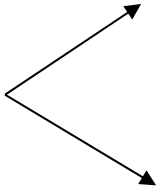
- Representaciones Redundantes:
  - Objetivo de la redundancia: Salvaguardar la información frente a posibles errores surgidos al almacenar, transmitir o manipular la información
  - Se añade cierta información adicional (redundancia) al dato manejado, de tal forma que el conjunto cumpla una cierta condición
  - Si tras la manipulación del dato redundante, éste deja de cumplir con la condición asignada, se podrá concluir que ha ocurrido algún tipo de error
  - Si la redundancia es suficiente, podremos ser capaces de detectar donde está el error e incluso corregirlo.
  - Algunos Códigos detectores y correctores :
    - Código de paridad
    - Código corrector de Hamming

CÓDIGOS DETECTORES Y CORRECTORES DE ERRORES

•CÓDIGO DE **PARIDAD SIMPLE O A NIVEL CARÁCTER**: En el código de paridad simple, se añade un bit adicional, haciendo que el número de bits a 1 en el conjunto de bits manipulados sea par o impar (según trabajemos con paridad par o impar).

•Ejemplo

– Información : 011010101



P. PAR 011010101 1

P. IMPAR 011010101 0

•De esta forma, cuando se quiera comprobar si un dato es correcto, debemos contar el número de bits a uno, si dicho número es par, y estamos trabajando con paridad par, no detectamos error, o si dicho número es impar, y estamos trabajando con paridad impar, no detectamos error.

•Este sistema no es muy fiable ya que si tenemos un numero par de bits modificados, el error no se detectará.

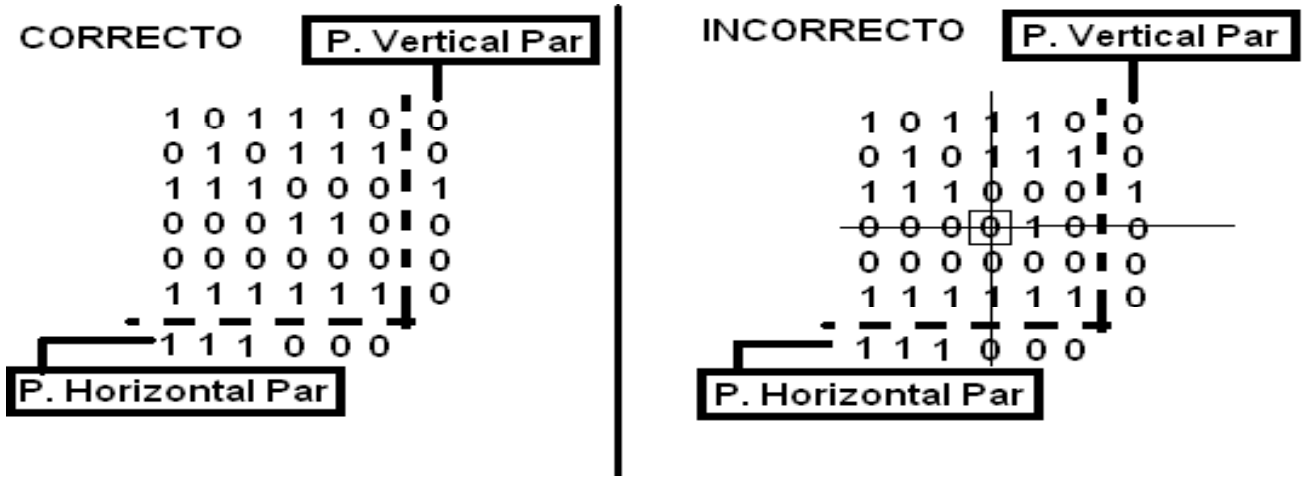
•Con este método sólo detectamos que ha ocurrido un error pero no sabemos en que bit, ni podemos corregirlo

CÓDIGOS DETECTORES Y CORRECTORES DE ERRORES

CÓDIGO DE **PARIDAD A NIVEL BLOQUE**:

–Trabajaremos con varios datos a la vez, cada uno de ellos con su bit de paridad, (para el ejemplo vamos a suponer paridad par).

–Generaremos un segundo bit de paridad según sea el número de unos de los bits que ocupan la misma posición relativa en el conjunto de datos manejado.



–El destino comprueba las paridades horizontales y verticales, donde sean erróneas detectará problemas, en el caso de ser un solo bit el afectado no habrá problemas en corregirlo, pero esto se complica si existe más de un bit afectado.

CÓDIGOS DETECTORES Y CORRECTORES DE ERRORES: Código de Hamming (1/6)

CÓDIGO DE **Hamming**:

–Los códigos de paridad no ofrecen mucha seguridad en la detección y corrección de errores.

–El código de Hamming consiste en intercalar bits de paridad en las posiciones que son potencias de dos del dato a tratar.

–Como regla diremos que el número “p” de bits de paridad a incluir en una cadena de “n” bits de información debe cumplir: $2^p \geq n+p+1$

Es decir, si quiero que n=64 deberé de intercalar al menos 7 bits de paridad : Código Hamming 64+7

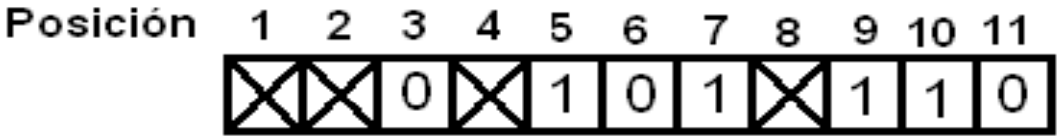
P	2^p	n+p+1
1	2	66
2	4	67
3	8	68
4	16	69
5	32	70
6	64	71
7	128	72

CÓDIGOS DETECTORES Y CORRECTORES DE ERRORES: Código de Hamming (2/6)

- Algoritmo:
  1. Todos los bits cuya posición es potencia de dos se utilizan como bits de paridad (posiciones 1, 2, 4, 8,16…)
  2. El resto de posiciones se utilizan como bits de datos.
  3. Cada bit de paridad se obtiene calculando paridades de ciertos bits de la cadena de datos de la siguiente forma:
     1. Posición 1 : salta 1, comprueba 1, salta 1, comprueba 1…
     2. Posición 2 : comprueba 1, salta 2, comprueba 2, salta 2, comprueba 2…
     3. Posición 4 : comprueba 3, salta 4, comprueba 4, salta 4, comprueba 4…
     4. Posición 8 : comprueba 7, salta 8, comprueba 8, salta 8, comprueba 8…
     5. …etc…

CÓDIGOS DETECTORES Y CORRECTORES DE ERRORES: Código de Hamming (3/6)

- Ejemplo: HAMMING 7+4 (7datos + 4 paridad)
  - **Consideremos el dato de 7 bits "0101110":**
    1. **Los colocamos en las posiciones que no son potencias de**



    2. **<u>Cálculo el bit de paridad de la posición 1</u>(salto 1, comprueba 1,salto 1, comprueba 1), luego debo comprobar las posiciones 3,5,7,9,11,13,15…, esto hace que tenga 3 bits a uno, luego el bit de paridad par será 1 POSICION 1 = 1**
    3. **<u>Cálculo el bit de paridad de la posición 2</u>(Comprueba 1, salto 2, comprueba 2,salto 2 comprueba 2), luego debo comprobar las posiciones 3,6,7,10,11,14,15…, esto hace que tenga 2 bits a uno, luego el bit de paridad par será 0 POSICION 2 = 0**

CÓDIGOS DETECTORES Y CORRECTORES DE ERRORES: Código de Hamming (4/6)

- Ejemplo: HAMMING 7+4 (7datos + 4 paridad)

4. **Cálculo el bit de paridad de la posición 4(Comprueba 3, salto 4, comprueba 4,salto 4, comprueba 4), luego debo comprobar las posiciones 5,6,7,12,13,14,15…, esto hace que tenga 2 bits a uno, luego el bit de paridad par será 0 POSICION 4 = 0**
5. **Cálculo el bit de paridad de la posición 8(Comprueba 7, salto 8, comprueba 8,salto 8, comprueba 8), luego debo comprobar las posiciones 9,10,11,12,13,14,15, 24,25,26,27,28,29,30,31…, esto hace que tenga 1 bit a uno, luego el bit de paridad par será 1**

POSICION 8 = 0

7. **Los bits de paridad quedan 10001010110**
8. **Por lo que el dato modificado que queda es 10001010110**
9. **Si quiero comprobar que el dato es correcto, bastará con que compruebe las paridades pares que se han realizado al introducir el Hamming, es decir:**
   - **Si cogemos las posiciones 1,3,5,7 deberíamos tener paridad par.**
   - **Lo mismo para 2,3,6,7,10,11. y para el resto de bits de paridad**

CÓDIGOS DETECTORES Y CORRECTORES DE ERRORES: Código de Hamming (5/6)

- Ejemplo: HAMMING 7+4 (7datos + 4 paridad)

10. Supongamos que ha ocurrido un error y el bit que ocupa la posición 3 (dato) ha cambiado de 0 a 1, por lo que el dato es: 10101010110:

Posición	BIT	Posición	BIT	Posición	BIT	Posicion	BIT
1	1	2	0	4	0	8	0
3	1	3	1	5	1	9	1
5	1	6	0	6	0	10	1
7	1	7	1	7	1	11	0
9	1	10	1	12	0	12	0
11	0	11	0	13	0	13	0
Paridad	1		1		0		0

ERROR **ERROR**

CÓDIGOS DETECTORES Y CORRECTORES DE ERRORES: Código de Hamming (6/6)

- Ejemplo: HAMMING 7+4 (7datos + 4 paridad)

11. **Localización del error:**

    La última fila de la tabla anterior nos muestra que el error se ha producido en el bit 0011, es decir, en el tercer bit.

12. **En la comprobación de la paridad no se tienen en cuenta los bits de paridad. Si el error se produjera en uno de ellos , en la comprobación sólo se detectaría un error, justo el correspondiente al bit de paridad causante del mismo.**

13. **Si el número de bits cambiantes es superior a uno, en la localización del error se obtendría una cifra mayor que 11 (en este ejemplo), por lo que sólo podríamos decir que hay error pero no corregirlo**

CÓDIGOS ALFANUMÉRICOS

- En informática no es suficiente con manejar datos numéricos, sino que es igual de importante manejar otros tipos de datos como pueden ser : Letras mayúsculas, Letras minúsculas, símbolos especiales …etc…

- Para ello se propone la codificación esos símbolos mediante cadenas de ceros y unos y de esta forma facilitar su manejo en un ordenador.

CÓDIGOS ALFANUMÉRICOS

- CARACTERÍSTICAS:
  - Se debe intentar que el reconocimiento de los códigos que representen a los números sea muy sencillo, dada la repetición de las operaciones de traducción desde/a otros métodos de representación numérica
  - Se debe intentar que los códigos de las letras mayúsculas y minúsculas difieran en un solo bit
  - Se debe intentar tener bien diferenciados los caracteres numéricos, alfanuméricos y los de control.
  - Las cadenas de caracteres alfanuméricos pueden ser:
    - ***De Longitud Fija***: Cada dato ocupa un determinado número de bits, por lo tanto es sencillo saber donde comienza y donde acaba cada dato.
    - ***De Longitud Variable***: Será necesario poder encontrar el final de cada dato, para ello podemos optar por dos métodos:
      - Cada dato tiene en su inicio un campo que indica su longitud
      - Cada dato se separa del siguiente mediante un símbolo fijo

CÓDIGOS ALFANUMÉRICOS

- BREVE HISTORIA:
  - En lo años 50 se definieron sistemas de codificación empleando 6 bits por carácter, ello permite 2^6=64 códigos para diferentes caracteres : 26 Letras (A … Z), 10 Números (0 … 9), 28 Caracteres especiales (+,-,*,/ …).
  - Ejemplos: Fieldata, X3, BCDIC.
  - Debido a la necesidad de manejar tanto letras mayúsculas como minúsculas así como la de manejar diferentes periféricos, dio lugar a los códigos de 7 bits, como es el código ASCII (American Standard Code For Information Interchange) en el que tendremos 2^7=128 códigos para diferentes caracteres

ASCII	Hex	Símbolo
0	0	NUL
1	1	SOH
2	2	STX
3	3	ETX
4	4	EOT
5	5	ENQ
6	6	ACK
7	7	BEL
8	8	BS
9	9	TAB
10	A	LF
11	B	VT
12	C	FF
13	D	CR
14	E	SO
15	F	SI

ASCII	Hex	Símbolo
16	10	DLE
17	11	DC1
18	12	DC2
19	13	DC3
20	14	DC4
21	15	NAK
22	16	SYN
23	17	ETB
24	18	CAN
25	19	EM
26	1A	SUB
27	1B	ESC
28	1C	FS
29	1D	GS
30	1E	RS
31	1F	US

ASCII	Hex	Símbolo
32	20	(espacio)
33	21	!
34	22	"
35	23	#
36	24	$
37	25	%
38	26	&
39	27	'
40	28	(
41	29	)
42	2A	*
43	2B	+
44	2C	,
45	2D	-
46	2E	.
47	2F	/

ASCII	Hex	Símbolo
48	30	0
49	31	1
50	32	2
51	33	3
52	34	4
53	35	5
54	36	6
55	37	7
56	38	8
57	39	9
58	3A	:
59	3B	;
60	3C	<
61	3D	=
62	3E	>
63	3F	?

ASCII	Hex	Símbolo
64	40	@
65	41	A
66	42	B
67	43	C
68	44	D
69	45	E
70	46	F
71	47	G
72	48	H
73	49	I
74	4A	J
75	4B	K
76	4C	L
77	4D	M
78	4E	N
79	4F	O

ASCII	Hex	Símbolo
80	50	P
81	51	Q
82	52	R
83	53	S
84	54	T
85	55	U
86	56	V
87	57	W
88	58	X
89	59	Y
90	5A	Z
91	5B	[
92	5C	\
93	5D	]
94	5E	^
95	5F	_

ASCII	Hex	Símbolo
96	60	`
97	61	a
98	62	b
99	63	c
100	64	d
101	65	e
102	66	f
103	67	g
104	68	h
105	69	i
106	6A	j
107	6B	k
108	6C	l
109	6D	m
110	6E	n
111	6F	o

ASCII	Hex	Símbolo
112	70	p
113	71	q
114	72	r
115	73	s
116	74	t
117	75	u
118	76	v
119	77	w
120	78	x
121	79	y
122	7A	z
123	7B	{
124	7C	|
125	7D	}
126	7E	~
127	7F	□

CÓDIGOS ALFANUMÉRICOS

–Con el objetivo de incluir letras acentuadas , “ñ”, y muchos otros símbolos, se introducen los códigos alfanuméricos de 8 bits (2^8=256), como el EBCDIC introducido por IBM en 1964 y el ASCII extendido.

–Dado el gran número de caracteres que requieren los distintos países, existen distintas alternativas de ASCII extendido, cada una de ellas denominada página de códigos. En este código:

- Es fácil comprobar si un carácter es numérico ya que todos ellos están seguidos y bastaría comprobar si se encuentra en el rango correspondiente.
- En ASCII para conocer el valor de un carácter numérico, basta con restar 48d
- Las letras mayúsculas y minúsculas tienen una codificación que solo difiere en un bit

CÓDIGOS ALFANUMÉRICOS

ASCII	Hex	Símbolo
0	0	NUL
1	1	SOH
2	2	STX
3	3	ETX
4	4	EOT
5	5	ENQ
6	6	ACK
7	7	BEL
8	8	BS
9	9	TAB
10	A	LF
11	B	VT
12	C	FF
13	D	CR
14	E	SO
15	F	SI

ASCII	Hex	Símbolo
16	10	DLE
17	11	DC1
18	12	DC2
19	13	DC3
20	14	DC4
21	15	NAK
22	16	SYN
23	17	ETB
24	18	CAN
25	19	EM
26	1A	SUB
27	1B	ESC
28	1C	FS
29	1D	GS
30	1E	RS
31	1F	US

ASCII	Hex	Símbolo
32	20	(espacio)
33	21	!
34	22	"
35	23	#
36	24	$
37	25	%
38	26	&
39	27	'
40	28	(
41	29	)
42	2A	*
43	2B	+
44	2C	,
45	2D	-
46	2E	.
47	2F	/

ASCII	Hex	Símbolo
48	30	0
49	31	1
50	32	2
51	33	3
52	34	4
53	35	5
54	36	6
55	37	7
56	38	8
57	39	9
58	3A	:
59	3B	;
60	3C	<
61	3D	=
62	3E	>
63	3F	?

ASCII	Hex	Símbolo
64	40	@
65	41	A
66	42	B
67	43	C
68	44	D
69	45	E
70	46	F
71	47	G
72	48	H
73	49	I
74	4A	J
75	4B	K
76	4C	L
77	4D	M
78	4E	N
79	4F	O

ASCII	Hex	Símbolo
80	50	P
81	51	Q
82	52	R
83	53	S
84	54	T
85	55	U
86	56	V
87	57	W
88	58	X
89	59	Y
90	5A	Z
91	5B	[
92	5C	\
93	5D	]
94	5E	^
95	5F	_

ASCII	Hex	Símbolo
96	60	`
97	61	a
98	62	b
99	63	c
100	64	d
101	65	e
102	66	f
103	67	g
104	68	h
105	69	i
106	6A	j
107	6B	k
108	6C	l
109	6D	m
110	6E	n
111	6F	o

ASCII	Hex	Símbolo
112	70	p
113	71	q
114	72	r
115	73	s
116	74	t
117	75	u
118	76	v
119	77	w
120	78	x
121	79	y
122	7A	z
123	7B	{
124	7C	|
125	7D	}
126	7E	~
127	7F	⌂

128	Ç	144	É	160	á	176	░	193	┴	209	╤	225	ß	241	±
129	ü	145	æ	161	í	177	▒	194	┬	210	╥	226	Γ	242	≥
130	é	146	Æ	162	ó	178	▓	195	├	211	╙	227	π	243	≤
131	â	147	ô	163	ú	179	|	196	─	212	╘	228	Σ	244	⌠
132	ä	148	ö	164	ñ	180	┤	197	┼	213	╒	229	σ	245	⌡
133	à	149	ò	165	Ñ	181	╡	198	╞	214	╓	230	µ	246	÷
134	å	150	û	166	ª	182	╢	199	╟	215	╫	231	τ	247	≈
135	ç	151	ù	167	º	183	╖	200	╚	216	╪	232	Φ	248	°
136	ê	152	ÿ	168	¿	184	╕	201	╔	217	┘	233	Θ	249	∙
137	ë	153	Ö	169	⌐	185	╣	202	╩	218	┌	234	Ω	250	·
138	è	154	Ü	170	¬	186	║	203	╦	219	█	235	δ	251	√
139	ï	156	£	171	½	187	╗	204	╠	220	▄	236	∞	252	ⁿ
140	î	157	¥	172	¼	188	╝	205	═	221	▌	237	φ	253	²
141	ì	158	₧	173	¡	189	╜	206	╬	222	▐	238	ε	254	■
142	Ä	159	ƒ	174	«	190	╛	207	╧	223	▀	239	∩	255	
143	Å	192	└	175	»	191	┐	208	╨	224	α	240	≡		

CÓDIGOS ALFANUMÉRICOS

–Debido a la infinidad de caracteres existentes en el mercado internacional, que deben cubrir los diferentes alfabetos (latino, árabe, griego, hebreo, cirílico …) se usan otras alternativas como son:

–***UNICODE***:

- Código para el intercambio de información internacional
- Usa dos octetos por carácter (16 bits).
- El estándar UNICODE tiene definidos 34.168 caracteres correspondientes a idiomas escritos de América, Europa, Oriente Medio, Africa, India, Asia y Pacífico.