Capítulo 3

Códigos

3.1. Introducción

Los bits no sólo sirven para representar números en un sistema de cómputo. También puden representar letras, colores, sonidos, etc.; cualquier tipo de información que pueda ser digitalizada. Por definición, un código es una representación alternativa de información. Un código binario es la representación alternativa de información en forma de unos y ceros. Se pueden utilizar códigos lo mismo para la representación de información que para incrementar la seguridad de dicha información, durante su transmisión o almacenamiento. Los códigos también evitan que la información sea entendida por entes (máquinas o personas) que no deban hacerlo.

Un número binario de n bits puede formar un total de 2^n combinaciones. Cada una de estas combinaciones puede representar números como se hizo en el Capítulo 3 o colores, letras. Resumiendo, puede representar *información*. Cuando se tiene un conjunto que consta de un número de elementos que no son una potencia de dos $(1, 2, 4, 16, 32, \ldots)$, existirán en el código binario combinaciones sin asignar. Por ejemplo, considerense los 10 dígitos decimales $(0, 1, 2, 3, 4, 5, 6, 7, 8, 9)$. Si se desea codificarlos en binario, se deben usar 4 bits, por lo que $2^4 = 16$, se utilizarán 10 de las combinaciones, pero 6 quedarán sin asignarse. Si se usaran sólo 3 bits, $2^3 = 8$, sólo se podrían asignar 8 dígitos, faltarían 2 dígitos por asignar código. Esta no es la única forma de codificar los 10 dígitos decimales, pueden usarse por ejemplo 10 bits, y prender sólo un bit a la vez, según la posición que se requiera. Para aclarar esto, considérese la tabla 3.1. En esta se muestran las dos formas previamente

discutidas para representar los diez dígitos decimales.

Dígito Decimal	Código de 4 bits	Código de 10 bits
0	0000	0000000001
1	0001	0000000010
2	0010	0000000100
3	0011	0000001000
4	0100	0000010000
5	0101	0000100000
6	0110	0001000000
7	0111	0010000000
8	1000	0100000000
9	1001	1000000000

Tabla 3.1: Dos códigos binarios para representar los dígitos decimales

Existen muchas clasificaciones de códigos, principalmente por su funcionalidad. Para códigos binarios, se utilizará la siguiente clasificiación:

- Códigos Binarios con Peso
- Códigos Binarios sin Peso
- Códigos Alfanuméricos
- Códigos Detectores de Errores
- Códigos Correctores de Errores

Dentro de cada división de la clasificación existen varios códigos distintos. En cada división se analizará al menos un código real que mostrará su comportamiento.

3.2. Códigos Binarios con Peso

Un código binario con peso es aquel en el que cada dígito, al ocupar una posición lleva implícito un determinado peso por ocuparla. Un ejemplo de este tipo de códigos es el binario natural, en el que sabemos que la posición menos signficativa representa el valor del bit multiplicado por $2^0 = 1$, el

siguiente va multiplicado por $2^1 = 2$, el tercero va multiplicado por $2^2 = 4$, etc. Cada posición tiene un peso determinado. Si se desea conocer el valor de dichos bits, mediante la suma de los productos de los bits por la potencia correspondiente, es posible calcular dicho valor. Otro código binario con peso es el código BCD.

3.2.1. BCD

El código BCD, Decimal Codificado en Binario (Binary-Coded Decimal), en este, cada dígito de un número decimal se encuentra representado por un equivalente binario. El dígito decimal mas grande es el 9 por lo que se necesitan 4 bits para representarlo. De esta forma, por cada dígito decimal, se tendrán 4 bits. Por ejemplo, considérese el número decimal 863. El equivalente binario del 8 es el 1000, el del 6 es el 0110 y el del 3 es 0011, acomodando todo junto tenemos que $863_{10} = 100001100011_{BCD}$. Es muy importante notar que cada dígito binario es convertido a su equivalente binario directo. Para dejar muy claro esto, el número 863 convertido a su equivalente binario es: $863_{10} = 1101011111_2$.

Dígito Decimal	BCD 8-4-2-1
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
10	0001 0000
11	0001 0001

Tabla 3.2: Tabla de conversión de BCD

El código BCD se muestra en la tabla 3.2.1. A esta versión del código también se le conoce como BCD 8-4-2-1, por el peso de cada bit en los nibble

(4 bits). También existen otras versiones de BCD, como la 4-2-2-1 o la 5-4-2-1. Para el caso de los números reales, se sigue el mismo principio, cada dígito decimal es reemplazado por 4 bits correspondientes al código. Considerese el número $842{,}7259_{10} = 100001000010{,}0111001001011001_{BCD}$.

Existen además 6 combinaciones de bits que no son parte del código BCD, estas son: 1010, 1011, 1100, 1101, 1110 y 1111. Ninguna de estas combinaciones pueden aparecer en un número codificado en BCD. Es importante resaltar que el BCD no es otro sistema numérico como el binario, octal o hexadecimal. Es de hecho el sistema decimal con cada dígito codificado en su equivalente binario. Es importante también entender que un número BCD no es lo mismo que un número binario natural. Un número binario natural se obtiene al convertir de forma complenta el número decimal a su equivalente binario; el código BCD convierte cada dígito decimal a su equivalente binario.

3.3. Códigos Binarios sin Peso

Algunos códigos binarios no tienen peso. Cada bit por lo tanto no tiene un peso específico. A este tipo de códigos se les conoce como códigos binarios sin peso y tienen aplicaciones muy específicas. Algunos de estos códigos son el código Exceso 3 (XS3) y el código de Gray.

3.3.1. Exceso 3 (XS3)

El código XS3 esta relacionado con el código BCD 8-4-2-1 por su naturaleza decimal codificada en binario. Nuevamente cada dígito decimal equivale a 4 bits del código XS3. El código XS3 se muestra en la tabla 3.3. Como su nombre lo indica, cada dígito XS3 se encuentra excedido en 3 unidades al BCD. Por ejemplo, el código $0000_{BCD} = 0011_{XS3}$. En este código, las 6 combinaciones inválidas son: 0000, 0001, 0010, 1101, 1110 y 1111.

La característica más importante del código XS3 es la propiedad de autocomplemento. Esto es, que el complemento a la base -1 de un número codificado en XS3 es el código XS3 del complemento a la base -1 del número decimal correspondiente. Por ejemplo, considerese el número 6_{10}, cuyo complemento a la base -1 es 3. El código XS3 del número decimal 6_{10} es 1001_{BCD}. El complemento a la base -1 de este número es 0110, que equivale al código XS3 del número decimal 3_{10}. La principal aplicación del código XS3 se

Dígito Decimal	Código XS3
0	0011
1	0100
2	0101
3	0110
4	0111
5	1000
6	1001
7	1010
8	1011
9	1100
10	0100 0011
11	0100 0100

Tabla 3.3: Tabla de conversión del código XS3

encuentra en la sustracción por la propiedad autocomplementaria.

3.3.2. Código Gray

El código de Gray es otro código binario sin peso. El código de Gray se muestra en la tabla 3.4 para 4 bits. Si se observa cuidadosamente, es posible notar que cada incremento en la cuenta secuencial de la columna del Código de Gray, únicamente cambia el estado de un bit.

Al Código Gray además de ser un código sin peso, es no aritmético y es cíclico. El hecho de ser cíclico, indica que conserva su característica de cambio de un bit en secuencia, incluso pasando de la combinación más grande a la inicial, como se puede apreciar en la tabla 3.4. La combinación más alta es 1000 y la más baja es el 0000, únicamente cambia un bit. Existen dos formas para generar el código de Gray. La primera se utiliza cuando se va a emplear una tabla. La segunda es la conversión directa de un número binario a su equivalente Gray y viceversa.

Generación de Código de Gray para Tabla

El método para la generación de un código de $(n + 1)$ bits a partir de el código de n bits se ilustra en la tabla 3.5. Por ejemplo, el segundo bit se

Número Decimal	Binario Natural	Código Gray
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

Tabla 3.4: Tabla de conversión del Código Gray

genera reflejando los códigos del primer bit, en la parte superior del reflejo se colocan ceros antecediendo al reflejo y en la parte inferior se colocan unos antecediendo al reflejo. El código de 4 bits se genera a partir del código de 3 bits. Se genera el reflejo, y en la parte superior del reflejo se antecede la combinación con ceros, mientras que en la parte inferior se antecede con unos.

Conversión de un número binario natural a su equivalente Gray

Para encontrar el equivalente Gray de un número binario natural, no es necesario construir la tabla al tamaño del número de bits del número. Existe un procedimiento que convierte directamente el número binario a su equivalente Gray. Los pasos son los siguientes:

1. El bit más significativo es siempre el mismo en binario natural y en Gray

1 Bit	2 Bits	3 Bits	4 Bits
0	0 0	0 00	0 000
1	0 1	0 01	0 001
	1 1	0 11	0 011
	1 0	0 10	0 010
		1 10	0 110
		1 11	0 111
		1 01	0 101
		1 00	0 100
			1 100
			1 101
			1 111
			1 110
			1 010
			1 011
			1 001
			1 000

Tabla 3.5: Tabla de generación del código Gray

2. Se suma el bit más significativo con el bit situado a su derecha inmediata y se registra la suma, despreciando siempre el acarreo, en la línea donde aparecerá el número en Gray

3. Se continúa la suma de cada bit con el bit a su derecha inmediata y se registran las sumas, hasta llegar al bit menos significativo

Es importante mencionar que un número en Gray siempre tendrá el mismo número de bits que su equivalente binario. Como ejemplo, considérese el número binario 110011001_2 que se desea convertir a su equivalente Gray. El bit más significativo es igual en binario natural que en Gray:

1	1	0	0	1	1	0	0	1	Binario Natural
1									Código Gray

En seguida, se suma el bit más significativo con el bit inmediatemente ubicado a su derecha. La suma se hace sin acarreos:

		1								
1	+	1	0	0	1	1	0	0	1	Binario Natural
1		0								Código Gray

Ahora el segundo bit más significativo se suma con el bit inmediato a la derecha:

			1							
1	1	+	0	0	1	1	0	0	1	Binario Natural
1	0		1							Código Gray

Esto se repite hasta llegar al bit menos significativo:

									0	
1	1	0	0	1	1	0	0		1	Binario Natural
1	0	1	0	1	0	1	0	+	1	Código Gray

El resultado se encuentra en el renglón del Código Gray, esto es: $110011001_2 = 101010101_{Gray}$. Si se desea corroborar, se puede hacer la conversión del número siguiente en código binario natural y demostrar que sólo varía un bit entre las dos combinaciones subsecuentes. Ahora el número que se debe convertir es: 110011010_2:

1	1	0	0	1	1	0	1	0	Binario Natural
1	0	1	0	1	0	1	1	1	Código Gray

El resultado es entonces $110011010_2 = 101010111_{Gray}$. Si se analizan las dos combinaciones en Código Gray, se puede apreciar que únicamente cambia un bit entre las dos combinaciones.

Conversión de un número Gray a su equivalente binario natural

Para convertir un número en Código Gray a su equvalente binario, se siguen los siguientes pasos:

1. El bit más significativo es siempre el mismo en binario natural y en Gray por lo que es transferido a la línea binaria

2. Se suma el bit recien transferido a la línea binaira con el bit situado a su derecha inmediata en código de Gray y se registra la suma en la línea binaria, despreciando siempre el acarreo

3. Se continúa la suma de cada bit binario con el bit a su derecha inmediata en Gray y se registran las sumas sin acarreo en el renglón binario, hasta llegar al bit menos significativo

Para ilustrar esto, considerese el número 101101010_{Gray}. Para encontrar su equivalente binario, se siguen los pasos descritos anteriormente. El bit más significativo es igual en Gray y en binario:

1	0	1	1	0	1	0	1	0	Código Gray
1									Binario Natural

El bit que se ha colocado en el renglón del Binario Natural, se suma ahora con el siguiente bit a la derecha del código de Gray. Es importante recordar que el acarreo se ignora.

		1								
1	+	0	1	1	0	1	0	1	0	Código Gray
1		1								Binario Natural

El bit que se acaba de colocar en el renglón del Binario se suma ahora con el siguiente bit a la derecha en el código de Gray.

			1							
1	0	+	1	1	0	1	0	1	0	Código Gray
1	1		0							Binario Natural

El cero que se colocó en el rengón del Binario, se suma ahora con el siguiente bit a la derecha del código de Gray.

				0						
1	0	1	+	1	0	1	0	1	0	Código Gray
1	1	0		1						Binario Natural

Se continúa el proceso hasta llegar al bit menos significativo.

									1	
1	0	1	1	0	1	0	1	+	0	Código Gray
1	1	0	1	1	0	0	1		1	Binario Natural

Al final de la conversión, obtenemos que $101101010_{Gray} = 110110011_2$.

3.4. Códigos Alfanuméricos

Hasta el momento, sólo se han utilizado los dígitos binarios para representar números. Sin embargo, los bits también pueden ser utilizados para representar las letras del alfabeto, los números, los signos de puntuación y algunos caracteres especiales. Los sistemas de comunicación requieren de estos códigos para enviar las letras del alfabeto, además, de los dígitos del 0 al 9 y los signos de puntuación más comunes como mínimo. Si a esto le agregamos mayúsculas y minúsculas, los signos de puntuación y de matemáticas y otros caracteres empleados día a día en los sitemas electrónicos, como la arroba obtenemos un número considerable de elementos. Además se deben considerar algunas combinaciones que producen caracteres no imprimibles y que se utilizan en los protocolos de información.

Algunos de los códigos alfanuméricos más comunes son el EBCDIC (Extended Binary-Coded-Decimal Interchange Code), el Hellerith de 12 bits (utilizado en las tarjetas perforadas) o el Selectric de algunas máquinas de escribir con cabeza giratoria, pero ninguno es tan utilizado ni tan importante como el código ASCII (American Standard Code for Information Interchange). El ASCII es un código alfanumérico internacionalmente aceptado utilizado en la gran mayoría de los sistemas de comunicaciones, cómputo y otros sistemas electrónicos. El código ASCII dispone de 128 caracteres que se representan mediante un código binario de 7 bits. La tabla 3.4 muestra los caracteres, el código binario equivalente, el hexadecimal y el decimal de estós códigos. Como se puede apreciar, los primeros 32 elementos del código son caracteres de control y los 96 restantes son símbolos gráficos que pueden imprimirse o mostrarse en pantalla.

Símbolo	Dec.	Bin.	Hex.	Símbolo	Dec.	Bin.	Hex.
NUL	0	0000000	00	@	64	1000000	40
SOH	1	0000001	01	A	65	1000001	41
STX	2	0000010	02	B	66	1000010	42
ETX	3	0000011	03	C	67	1000011	43
EOT	4	0000100	04	D	68	1000100	44
ENQ	5	0000101	05	E	69	1000101	45
ACK	6	0000110	06	F	70	1000110	46
BEL	7	0000111	07	G	71	1000111	47
BS	8	0001000	08	H	72	1001000	48
HT	9	0001001	09	I	73	1001001	49

Símbolo	Dec.	Bin.	Hex.	Símbolo	Dec.	Bin.	Hex.
LF	10	0001010	0A	J	74	1001010	4A
VT	11	0001011	0B	K	75	1001011	4B
FF	12	0001100	0C	L	76	1001100	4C
CR	13	0001101	0D	M	77	1001101	4D
SO	14	0001110	0E	N	78	1001110	4E
SI	15	0001111	0F	O	79	1001111	4F
DLE	16	0010000	10	P	80	1010000	50
DC1	17	0010001	11	Q	81	1010001	51
DC2	18	0010010	12	R	82	1010010	52
DC3	19	0010011	13	S	83	1010011	53
DC4	20	0010100	14	T	84	1010100	54
NAK	21	0010101	15	U	85	1010101	55
SYN	22	0010110	16	V	86	1010110	56
ETB	23	0010111	17	W	87	1010111	57
CAN	24	0011000	18	X	88	1011000	58
EM	25	0011001	19	Y	89	1011001	59
SUM	26	0011010	1A	Z	90	1011010	5A
ESC	27	0011011	1B	[	91	1011011	5B
FS	28	0011100	1C	\	92	1011100	5C
GS	29	0011101	1D	]	93	1011101	5D
RS	30	0011110	1E	^	94	1011110	5E
US	31	0011111	1F	_	95	10111111	5F
space	32	0100000	20	‘	96	1100000	60
!	33	0100001	21	a	97	1100001	61
”	34	0100010	22	b	98	1100010	62
#	35	0100011	23	c	99	1100011	63
$	36	0100100	24	d	100	1100100	64
%	37	0100101	25	e	101	1100101	65
&	38	0100110	26	f	102	1100110	66
	39	0100111	27	g	103	1100111	67
(	40	0101000	28	h	104	1101000	68
)	41	0101001	29	i	105	1101001	69
*	42	0101010	2A	j	106	1101010	6A
+	43	0101011	2B	k	107	1101011	6B

Símbolo	Dec.	Bin.	Hex.	Símbolo	Dec.	Bin.	Hex.
,	44	0101100	2C	l	108	1101100	6C
-	45	0101101	2D	m	109	1101101	6D
.	46	0101110	2E	n	110	1101110	6E
/	47	0101111	2F	o	111	1101111	6F
0	48	0110000	30	p	112	1110000	70
1	49	0110001	31	q	113	1110001	71
2	50	0110010	32	r	114	1110010	72
3	51	0110011	33	s	115	1110011	73
4	52	0110100	34	t	116	1110100	74
5	53	0110101	35	u	117	1110101	75
6	54	0110110	36	v	118	1110110	76
7	55	0110111	37	w	119	1110111	77
8	56	0111000	39	x	120	1111000	78
9	57	0111001	39	y	121	1111001	79
:	58	0111010	3A	z	122	1111010	7A
;	59	0111011	3B	{	123	1111011	7B
¡	60	0111100	3C	—	124	1111100	7C
=	61	0111101	3D	}	125	1111101	7D
¿	62	0111110	3E	˜	126	1111110	7E
?	63	0111111	3F	Del	127	1111111	7F

Por ejemplo, considerese el mensaje: `... y sin embargo se mueve!` el cual se desea codificar en ASCII. La secuencia de bits que lo representan está dada por (se insertaron espacios para facilitar su lectura): 0101110 0101110 0101110 0100000 1111001 0100000 1110011 1101001 1101110 0100000 1100101 1101101 1100010 1100001 1110010 1100111 1101111 0100000 1110011 1100101 0100000 1101101 1110101 1100101 1110110 1100101 0100001 y en hexadecimal es: 2E 2E 2E 20 79 20 73 69 6E 20 65 6D 62 61 72 67 6F 20 73 65 20 6D 75 65 76 65 21.

Cuándo IBM puso en el mercado la computadora personal o PC, le incorporó al ASCII otros 128 símbolos y formó lo que se conoce como el ASCII Extendido. Debido a la popularidad de las PC, el ASCII extendido se ha convertido en un standard no oficial. En esta adición al código se encuentran: caracteres alfabéticos no ingleses, como las vocales con acentos o diérisis, símbolos no ingleses como el del Yen, letras griegas, símbolos matemáticos como la integral o la raíz cuadrada, y algunos caracteres con los que se pueden formar gráficos sencillos. Estos caracteres se representan con el rango de

valores hexadecimales que va desde el 80_h hasta el FF_h.

3.5. Códigos Detectores y Correctores de Errores

Existen tres funciones fundamentales que se realizan con sistemas digitales: almacenamiento de información, transmisión de información y procesamiento de información. Al mover los bits de un lugar a otro para cualquiera de estas funciones, se pueden producir errores en la información. La principal causa de estos errores es la presencia del ruido electrico, que consiste en fluctuaciones en el voltaje o corriente en un sistema eléctrico y se presenta a manera de picos. Este ruido eléctrico se encuentra presente en todos los sistemas eléctricos en mayor o menor medida.

Suponga un sistema de comunicación, en el que el transmisor envía a través de la línea una señal binaria sin ruido. Cuando dicha señal llega al receptor, la señal original tiene sumada una señal de error. En ocasiones, esta señal de error es lo suficientemente grande como para alterar el nivel lógico de la señal. Cuando esto ocurre el receptor interpreta incorrectamente un bit. Los sistemas digitales modernos se encuentran diseñados para trabajar en forma relativamente libre de errores, sin embargo, es necesario recordar que muchos de estos sistemas transmiten o procesan miles e incluso millones de bits por segundo; aún cuando la ocurrencia de errores sea extremadamente baja, a estas velocidades es factible que aparezcan errores de vez en cuando. Por esta razón, muchos sistemas digitales utilizan métodos para detectar o corregir errores.

Antes de analizar estos métodos de detección o corrección de errores, es necesario entender algunas definiciones. En primer instancia están los *Tipos de Palabra*. En términos generales, la palabra se refiere a la información que se desea codificar junto con el código que se esta utilizando. Se puede decir que la palabra es la informaición codificada.

Palabra de Código. Combinación de bits que representa un símbolo válido en el código

Palabra de Error. Combiandción de bits que representa un símbolo no válido en el código

Como se puede apreciar, sólo se tienen dos tipos de palabra, o bien esta pertenece al código o se trata de un error. El error se produce cuando el receptor interpreta incorrectamente algún bit, esto es que un bit cambió su valor. Al tratarse de bits, estos sólo pueden tener 2 valores, por lo que en ocasiones también se dice que el bit se volteó. Dentro de lo que son los errores existen 3 tipos:

Error Detectable. El cambio en los bits produce una palabra que no existe en el código

Error Indetectable. El cambio en los bits produce una palabra que si existen en el código

Error Corregible. El cambio en los bits produce una palabra que sólo puede asociarse con una palabra de entre todas las del código

Cuando se produjo un error en el que se formó una palabra que si existe en el código, el sistma leerá esa palabra como si no existiera el error. Un concepto que tiene gran relevancia en lo que son los códigos detectores y correctores de errores es el de la distancia:

Distancia entre dos combinaciones. Es el número de bits que se necesitan cambiar para convertir una combinación en la otra

Distancia Mínima, Natural o de Código. Es la distancia mínima de entre todas las distancias posibles entre 2 palabras del código. Se le conoce como d

Para calcular la distancia de código, es necesario calcular la distancia que existe entre todas las posibles combianciones de 2 en 2 en el código y después encontrar el mínimo. De hecho, es la distancia d la que determina el comportamiento del código en cuánto a los errores que se pueden detectar o corregir.

Código Detector. Un código puede detectar i errores si y sólo si su distancia es:

$$d \geq i + 1 \tag{3.1}$$

Código Corrector. Un código puede corregir i errores si y sólo si su distancia es:

$$d \geq 2i + 1 \tag{3.2}$$

Como se puede ver, entre mayor sea la distancia que presenta un código, mayor es el número de errores que se pueden detectar o corregir y a fin de cuentas, la distancia del código depende de los bits de paridad que se añadan.

Bit de Paridad. Es un bit adicional que se añade a la información para hacer que el número total de unos (o ceros) sea par (o impar). Su propósito es ampliar la distancia del código

3.5.1. Métodos de Paridad

El bit de paridad es un bit que se añade a la información para forzar a la palabra a tener un número par o impar de unos o ceros. Esto provoca que es que se aumente la distancia de código y por lo tanto que se puedan detectar y en algunos casos corregir errores. Por ejemplo, considérese el código binario natural de 3 bits, que permite $2^n = 2^3 = 8$ combinaciones que se muestra en la tabla 3.6. Esta tabla, se puede utilizar para codificar colores de una imagen.

c	b	a	Color
0	0	0	Negro
0	0	1	Amarillo
0	1	0	Cyan
0	1	1	Magenta
1	0	0	Rojo
1	0	1	Verde
1	1	0	Azul
1	1	1	Blanco

Tabla 3.6:

Si se comienza a analizar las combinaciones, no hay que avanzar mucho para descubrir que entre la combinación del Negro y la del Amarillo únicamente cambia un bit, por lo que la distancia de este código es $d = 1$. En la tabla 3.7, se muestra el mismo código y un bit de paridad par para los unos que se añade en la columna inicial de la tabla. Este bit de paridad provoca que el número de unos en la combinación sea siempre par.

Si se analizan ahora las combiniaciones se puede apreciar que entre cualquier par de combinaciones de la tabla, siempre existe por lo menos 2 bits

p	c	b	a	Color
0	0	0	0	Negro
1	0	0	1	Amarillo
1	0	1	0	Cyan
0	0	1	1	Magenta
1	1	0	0	Rojo
0	1	0	1	Verde
0	1	1	0	Azul
1	1	1	1	Blanco

Tabla 3.7:

de direrencia. Por ejemplo, compare el código para el color Negro, el 0000 y compárese contra cada uno de los demás colores. Para el caso del amarillo 1001, cyan 1010, magenta 0011, rojo 1100, verde 0101 y azul, hay 2 bits de diferencia y para el blanco hay 4. Ahora que se sabe que la distancia para este código es $d = 2$, es posible aplicar las ecuaciones 3.1 y 3.2. Para el caso de la ecuación 3.1, se tiene que $d \geq i + 1$, esto es, $2 \geq i + 1$, es decir que $i \leq 1$, lo que significa que este código es capaz de detectar un error. En el caso específico de la ecuación 3.2, la distancia mínima de un código capaz de corregir un error es $d = 3$, por lo que este código no es capaz de corregir errores.

El funcionamiento de los bits de paridad es muy sencillo. Primero, el transmisor debe calcular el bit de paridad par para los unos de tal forma que la palabra, antes de ser enviada, tenga un número par de unos. La palabra es transmitida por el medio que sea y al llegar al receptor, es verificada para encontrar si el número de unos es par. En caso afirmativo, se procede a eliminar el bit de paridad y se pasa la información al elemento siguiente. En caso negativo, esto es, que el número de unos sea impar, puede ocurrir una de varias alternativas. Una es que se envíe al transmisor una señal de retransmisión del dato que llegó alterado suponiendo que el error es aleatorio y no se repetirá.

Se llegaron a presentar más de un error, es posible que este se detecte o no. Si el número de errores es par, el error no se detectará pues para este caso particular, se producirá una palabra de código, esto es, una palabra con un número par de unos. Si en cambio el número de errores es impar, el sistema detectará que exisitío un error o mejor dicho que se generó una palabra de

error.

Existen algunas aplicaciones en las que no es posible pedir la retransmisión de los datos o resulta poco eficiente o incluso caro. En este tipo de aplicaciones se utilizan los códigos correctores de errores.

3.5.2. Códigos de Hamming

Los códigos correctores de errores, como su nombre lo indica, son códigos capaz de corregir los errores que se generaron durante la manipulación, transmisión o almacenamiento de dígitos binarios. Para que un código puede corregir un error, es necesario que la distancia del código sea de por lo menos $d = 3$. En abril de 1950, R. W. Hamming, quien trabajaba en la American Telephone and Telegraph Company publicó en "The Bell System Technical Journal", el artículo "Error Detecting and Error Correcting Codes", en el se presenta un código capaz de corregir errores sin necesidad de retransmisión de la información. Estos códigos se llamarían posteriormente Códigos de Hamming y han tenido un gran número de aplicaciones en soluciones de ingeniería.

En el Código de Hamming, no existe el bit 0, se empiezan a numerar a partir del bit 1. Los bits que son potencia de 2 (1, 2, 4, 8, 16, 32, etc.) se reservan para los bits de paridad. Los demás bits son utilizados como bits de informacion.

Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1
D10	D9	P8	D7	D6	D5	P4	D3	P2	P1

Los bits marcados con D son bits de datos, mientras que los marcados con P son de paridad. La información es vaciada sobre los bits de datos en el mismo orden en que será leido por el receptor. Los bits de paridad se calculan para que las series que se definen a continuación tengan siempre paridad par sobre los unos. Las series sobre las que se calcula la paridad siempre inician con el bit de paridad que se calcula. Para calcular el bit de paridad P1, se tiene la siguiente serie: P1, D3, D5, D7, D9, D11, ... En toda la serie, el único bit de paridad es el P1, los demás son bits de datos. Se comienza por el bit de paridad que se esta calculando, es decir P1 y se van saltando y escribiendo tantos bits como la paridad lo indique. En esta caso, la paridad es P1, entonces comenzando por P1, se escribe P1 y se salta uno, que sería P2, luego se escribe uno D3 y se salta uno P4, se escribe uno D5 y se salta

uno D6, continuando esta serie hasta llegar al bit de datos más grande que se tenga. Si se analiza la forma binaria de cada uno de los número de la serie, se tiene:

$$\begin{array}{rr} 1= & 1 \\ 3= & 11 \\ 5= & 101 \\ 7= & 111 \\ 9= & 1001 \\ 11= & 1011 \\ \vdots & \vdots \end{array}$$

donde el único bit que esta en uno en todas las combinaciones es el de la extrema derecha, esto es el bit 1.

Similarmente, para calcular el bit de paridad P2, se tiene que generar una serie que inicia justamente con el bit P2, se escriben 2 números, es decir, P2 y D3, se salta dos números, que son P4 y D5, y se escriben los dos siguientes: D6 y D7, se saltan los dos siguientes: P8 y D9 y se escriben los dos siguientes: D10 y D11, etc. La serie queda entonces P2, D3, D6, D7, D10, D11, D14, ... Si se analiza la representación binaria de cada uno de los bits que conforman la serie, es fácil notar que en todos ellos, se encuentra en uno el bit 2:

$$\begin{array}{rr} 2= & 1 \\ 3= & 11 \\ 6= & 101 \\ 7= & 111 \\ 10= & 1001 \\ 11= & 1011 \\ 14= & 1110 \\ 15= & 1111 \\ \vdots & \vdots \end{array}$$

Aplicando el mismo criterio que en los casos anteriores, para calcular el bit de paridad P4, se genera la siguiente serie: P4, D5, D6, D7, D12, D13, D14, D15, D20, ... Se comienza en el bit que se esta calculando, este es P4 y se escriben 4 bits consecutivos: P4, D5, D6 y D7; luego se brinca 4 bits consecutivos que son P8, D9, D10 y D11; se escriben los 4 siguientes: D12, D13, D14 y D15, etc. En todos las representaciones binarias de los bits

pertenecientes a la serie, se tiene que en todas, el tercer bit se encuentra en uno.

Para generar la serie que calcula el bit de paridad P8, se aplica el mismo criterio: se comienza en el bit que se esta calulando: P8 y se escriben 8 bits consecutivos: P8, D9, D10, D11, D12, D13, D14 y D15; se saltan 8 bits consecutivos: desde el D16 hasta el D23 y se escriben los 8 subsecuentes: D24, D25, D26, D27, D28, D29, D30 y D31. Nuevamente es fácil ver si se representan estos bits como sus combinaciones binarias que en todas ellas, el cuarto bit se encuentra en uno. La serie queda: P8, D9, D10, D11, D12, D13, D14, D15, D24, D25, ...

Para aclarar esto, considerese el dato 1100. El bit 3, D3=0, el bit 5 D5=0, el bit 6 D6=1 y el bit 7 D7=1. Falta calcular los bits de paridad P1, P2 y P4. El bit 8 no se tiene que calcular, a no ser que se tenga otro bit de datos. Para calcular P1, se tiene la serie P1, D3, D5 y D7. De esta serie se conocen los bits de datos D3, D5 y D7. Entonces la serie queda de momento como: P1, 0, 0 y 1. Para lograr que la serie tenga paridad par en los unos, es necesario que el bit de paridad P1=1.

Para calcular el bit de paridad P2, se tiene la serie: P2, D3, D6 y D7. Se conocen los bits de datos D3, D6 y D7. La serie queda temporalmente como: P2, 0, 1 y 1. Para que la serie tenga paridad par es necesario que P2=0.

Similarmente, para calcular el bit P4, se tiene la serie: P4, D5, D6 y D7, se conocen los bits de datos D5, D6 y D7. La serie queda temporalmente como: P4, 0, 1 y 1. Para que la serie tenga paridad par es necesario que el bit P4=0.

El dato 1100 queda codificado en Hamming como 1100001. Supóngase que el bit 5 se invierte por cuestión de ruido y nos queda entonces la palabra: 1110001. Cuando se recibe la palabra, lo primero que debe verificarse es la paridad de las series de bits. Se trata de las mismas series con las que se generaron los bits de paridad. Estas series se conocen como las Series de Corrección C_1, C_2 y C_4. Las series son:

$$C_1 = P1 \oplus D3 \oplus D5 \oplus D7$$
$$C_2 = P2 \oplus D3 \oplus D6 \oplus D7$$
$$C_4 = P4 \oplus D5 \oplus D6 \oplus D7$$

El símbolo $\oplus$ representa la operación or-exclusivo o xor que se estudia en la sección 4. Lo que hace esta operación en las ecuaciones es simplemente una operación de paridad. Si el número de unos es par el valor de la función

es 0 en caso de paridad impar el valor es 1. Sustituyendo los valores en las funciones se tiene:

$$C_1 = 1 \oplus 0 \oplus 1 \oplus 1 = 1$$
$$C_2 = 0 \oplus 0 \oplus 1 \oplus 1 = 0$$
$$C_4 = 0 \oplus 1 \oplus 1 \oplus 1 = 1$$

Acomodando en orden los bits C se tiene:

C_4	C_2	C_1
1	0	1

El equivalente en base 10 de este número es el 5. Este fue el bit que se invirtió. De la palabra que llegó: 1110001, se debe cambiar el bit 5: 1100001. Lo único que se requiere ahora es eliminar los bits de paridad P1, P2 y P4 para dejar sólamente los bits de datos: 1100 que fue el dato inicial.

Puede ocurrir que no se presenten errores en la transmisión, en ese caso, la palabra recibida es 1100001. En este caso se tiene:

$$C_1 = 1 \oplus 0 \oplus 0 \oplus 1 = 0$$
$$C_2 = 0 \oplus 0 \oplus 1 \oplus 1 = 0$$
$$C_4 = 0 \oplus 0 \oplus 1 \oplus 1 = 1$$

En este caso, el número en la base 10 es el 0. Esto significa que no existe ningún error en la palabra y lo único que resta es eliminar los bits de paridad para obtener el dato original: 1100.

Otro caso puede ser que se presenten dos o más errores en la transmisión. Basándose en el mismo caso se incorporarán dos errores. Ahora se considerará que la palabra recibida es: 1010001. Se han invertido los bits 6 y 5. Aplicando las funciones correctoras se tiene:

$$C_1 = 1 \oplus 0 \oplus 1 \oplus 1 = 1$$
$$C_2 = 0 \oplus 0 \oplus 0 \oplus 1 = 1$$
$$C_4 = 0 \oplus 1 \oplus 0 \oplus 1 = 0$$

El equivalente decimal de la combinación C_4, C_2, C_1 es un 3. Siguiendo la metodología se invierte el bit 3 y obtenemos 1010101. Después de esta operación, no sólo no se logró corregir ninguno de los dos errores existentes, sino que además se agregó un error más. Al decodificar la palabra, esto es, eliminar los bits de paridad y dejar sólo el dato obtenemos: 1011. Hay 3 bits de diferencia con le dato original 1100.

Después de ver los 3 casos que pueden ocurrir en el Código de Hamming con distancia $d = 3$ queda claro, que con esa distancia no se pueden corregir los 2 errores, pero podrían detectarse mediante la modificación de las funciones de paridad.

3.6. Ejercicios del Capítulo

1. Describa brevemente qué es un código.

2. ¿Cuáles son las principales diferencias entre el código binario natural y el código BCD?

3. ¿Represente el número decimal 178 en binario y en BCD.

4. Realice las siguientes conversiones:

1934_{10} a BCD	11110001110001_{Gray} a base 2
101101110110_2 a Gray	110110110101_{Gray} a 2
2048_{10} a BCD	10111011110001_{Gray} a base 2
00111000.10011000_{BCD} a base 2	11101011010111_2 a Gray
0001100101100110_{BCD} a base 8	0001100101100110_{BCD} a base 4
0011100001010110_{BCD} a base 8	11101000110111_2 a Gray
1001010000111000_{BCD} a base 2	1934_{10} a XS3
0011100001010110_{BCD} a base 8	1001100001110001_{BCD} a Gray

5. ¿Cuántos bits se necesitan para representar un número decimal de 8 dígitos en BCD?

6. ¿Cuál es el característica más importante del código Gray?

7. Realice las siguientes operaciones:

$10011000_{BCD} + 00011000_{BCD}$
$10010010_{BCD} + 10000111_{BCD}$
$10001000_{BCD} + 00011000_{BCD}$
$10010010_{BCD} + 10000111_{BCD}$

8. El operador de una computadora ingresa la siguiente instrucción codificada en ASCII desde el teclado: $1010011101010010011111010000_{ASCII}$. Determine que le indica esta instrucción a la computadora

9. Codifique el siguiente mensaje en código ASCII utilizando la representación hexadecimal:

```
COST=$72
Hasta la vista
LOGIN
```

10. Agregue un bit de paridad par al código ASCII para cada uno de los siguientes símbolos y exprese el resultado en hexadecimal:

$	&
A	B
a	b

11. Agregue un bit de paridad impar al código BCD para los siguientes números decimales:

69	85
199	5
3902	70

12. Se recibieron las siguientes palabras codificadas en Hamming. Obtenga los datos de cada palabra ya corregidas si asi se requiere:

0001100
1000001
101111
0101001
0110100

13. Un astronauta, que se encontraba en una sonda espacial, fue rozado por un asteroide y su equipo de comunicaciones falló. El sistema recibe información pero no puede decodificarla. La información está codificada en Hamming y necesita decodificarla para poder encender la

nave. La información recibida es: I=100101010000. Decodifique el dato y verifique si la información que llegó es correcta o necesita alguna modificación. De ser asi corríjala.

14. Se colocó una nave en órbita. El método de comunicación utilizado con los chimpancés que se encuentran a bordo se basa en dibujos que le indican a los primates los botones que deben apretar. Cada dibujo está formado por blancos y negros en una matriz de 6 renglones por 12 columnas. Los negros tienen un valor de 0 y los blancos un valor de 1. Para asegurar la confiabilidad en el envío de la información, cada renglón (de 12 bits) es codificado en Hamming lo que da como resultado renglones de 17 bits. Los 6 renglones que forman la imagen pueden presentar errores. Revise la información, si hay errores corríjalos, decodifique el Hamming y haga el dibujo en una matriz. La información más reciente recibida en la nave es:

a)
0 0100 1100 1111 0011
1 1101 0000 1100 1101
0 1100 0010 0000 1110
0 0100 1100 1001 1110
0 0100 0000 0100 1110
0 0101 1110 1011 1001

b)
1 1001 1100 1111 1011
0 0110 1011 1010 0001
1 1100 1111 1001 0001
0 0110 1110 1111 0000
1 1001 1100 0111 1010
1 1011 1000 0011 0010

15. Reptia el ejercicio anterior pero esta vez cada dibujo está formado por blancos y negros en una matriz de 5 renglones por 10 columnas. Los negros tienen un valor de 0 y los blancos un valor de 1:

10 0111 1111 0101
00 1011 0101 0000
01 1000 0000 0001
01 1011 0110 0000
00 0111 0111 0101

16. Se colocó una nave en órbita. El método de comunicación utilizado con los astronautas se basa en el código alfabético de 5 bits mostrado a continuación.

A	00000	I	01000	P	10000	X	11000
B	00001	J	01001	Q	10001	Y	11001
C	00010	K	01010	R	10010	Z	11010
D	00011	L	01011	S	10011	0	11011
E	00100	M	01100	T	10100	1	11100
F	00101	N	01101	U	10101	2	11101
G	00110	Ñ	01110	V	10110	3	11110
H	00111	O	01111	W	10111	:	11111

Se agrupan las letras de dos en dos para formar un total de 10 bits de datos. Una vez unidas se codifican utilizando Hamming de tal forma que se añaden 4 bits de paridad y se envían. Verifique que las palabras no tengan errores; en caso de existir corríjalos; decodifique los datos y obtenga el mensaje orignal en texto. La nave recibió las siguientes palabras en el siguiente órden:

a)
01000001100001
00100111011100
01111010111001
10101010000000
00011110100001

b)
01011011100010
01111011100110
11101111110111
00010001100001
00100101101110

17. Repita el ejercicio anterior pero con la tabla mostrada a continuación y los bloques de información subsecuentes:

A	00000	I	01000	P	10000	X	11000
B	00001	J	01001	Q	10001	Y	11001
C	00010	K	01010	R	10010	Z	11010
D	00011	L	01011	S	10011	?	11011
E	00100	M	01100	T	10100	!	11100
F	00101	N	01101	U	10101	.	11101
G	00110	Ñ	01110	V	10110	_	11110
H	00111	O	01111	W	10111	:	11111

La nave recibió las palabras en el siguinete órden:

a)
10110001111101
00011001111110
11110001001111
00000010100011
01101101010111

b)
10110001111101
00011001111110
11110001001111
00000010100011
01101101010111