5. Codificación de información en binario

Introducción

Existen muchas formas distintas de expresar información en binario, es decir, usando sólo ceros y unos

Distintos problemas físicos podrían alterar el almacenamiento y la transmisión de esta información (radiación, temperatura…) y generar errores

Los objetivos de este tema son:

- Describir distintas maneras en que los sistemas digitales codifican información usando el sistema binario
- Describir técnicas usadas para añadir información redundante a estos códigos binarios, de forma que sea posible detectar y corregir errores que se puedan producir

Estructura del tema

- Introducción
- Códigos binarios para números decimales
  - El código BCD 8421
  - Otros códigos: BCD 2421, Exceso 3, Biquinario, Gray, Johnson
- Códigos alfanuméricos
  - El código ASCII
  - Otros códigos: ASCII extendido, EBCDIC, UNICODE
- Códigos para detección y corrección de errores
  - Distancia de Hamming
  - Códigos de detección de errores
  - Códigos de corrección de errores
- Resumen y bibliografía

Códigos binarios para números decimales

- La conversión de binario a octal o hexadecimal es muy fácil, pero no lo es tanto la conversión a decimal
- Para hacer más fácil la conversión podemos crear un código en que cada combinación de bits se corresponda exactamente con un número decimal
- En general, cualquier conjunto de objetos se puede representar mediante combinaciones de bits
  - El conjunto de las combinaciones es llamado código
  - Cada combinación de bits dentro del código es considerada como una palabra del código

Códigos binarios para números decimales

Por ejemplo, podemos usar 4 bits para representar los dígitos decimales

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

257 0010 0101 0111

La asignación de palabras del código puede hacerse mediante una expresión aritmética o una tabla

- No todas las palabras del código tienen que usarse
- Si la asignación se hace con una tabla, puede no haber una explicación lógica de por qué el código es de una manera

Estructura del tema

Introducción

Códigos binarios para números decimales
- El código BCD 8421
- Otros códigos: BCD 2421, Exceso 3, Biquinario, Gray, Johnson

Códigos alfanuméricos
- El código ASCII
- Otros códigos: ASCII extendido, EBCDIC, UNICODE

Códigos para detección y corrección de errores
- Distancia de Hamming
- Códigos de detección de errores
- Códigos de corrección de errores

Resumen y bibliografía

BCD – Código Decimal Binario

El código BCD (Binary Coded Digit) es una forma de expresar cada uno de los dígitos decimales con un código binario

La fácil conversión de decimal a BCD convierte este código en una excelente interfaz para sistemas binarios como teclados, salidas digitales…

Con cuatro dígitos se pueden representar $2^4 = 16$ combinaciones, pero solo necesitamos 10, por lo que hay muchos códigos BCD posibles

El código BCD 8421

La versión más usada del código BCD es la 8421, también llamada BCD natural

Cada dígito decimal se representa por medio de un código binario de cuatro bits

Este código es ponderado: la designación 8421 indica los pesos binarios de cada uno de los cuatro bits: $2^3\ 2^2\ 2^1\ 2^0$

0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	no valido
1011	no valido
1100	no valido
1101	no valido
1110	no valido
1111	no valido

Equivalencia – BCD 8421

Para convertir un número decimal a BCD 8421 basta con sustituir cada dígito por la combinación de cuatro bits correspondiente

25718 0010 0101 0111 0001 1000

Para convertir un número BCD 8421 a decimal hacemos el proceso contrario

- Se comienza por el bit más a la derecha
- Se divide el código en grupos de cuatro bits
- Se sustituye cada grupo por el dígito decimal correspondiente

Formato signo-magnitud

La representación de números BCD con signo puede realizarse usando el formato signo-magnitud

El signo vendrá dado por un grupo adicional de 4 bits

La representación de los signos positivo y negativo puede realizarse usando cualquier cadena de bits de entre las que no se usan en el código

Este formato no resulta interesante a causa de una desventaja: es necesaria una operación de resta

Formato complemento a 10

Los números negativos se representan usando el complemento a la base del correspondiente positivo

Para calcular el complemento a 10 de un número tenemos que restar el número a la potencia de 10 inmediatamente superior

789 1000 - 789 = 211

Un número BCD también necesita un bit de signo

789 0 0111 1000 1001

-789 1 0010 0001 0001

Suma en BCD 8421

La suma es la más importante de las operaciones aritméticas en BCD, ya que la resta, la multiplicación y la división pueden realizarse utilizando la suma

La suma de dos números BCD se realiza dígito a dígito

La suma de cada dígito es una suma binaria de 4 bits

El resultado es válido si es menor o igual a 9

Si la suma es mayor que 9 ó se genera un acarreo

- Se suma 6 al resultado para saltar los 6 valores no válidos
- Si se genera un acarreo al sumar 6, se suma al dígito siguiente

Suma en BCD 8421

Ejemplo de suma: 23 + 15 = 38

```
  0 0 1 0   0 0 1 1
+ 0 0 0 1   0 1 0 1
-----------------------
  0 0 1 1   1 0 0 0
```

Ejemplo de suma: 16 + 15 = 31

```
  0 0 0 1   0 1 1 0
+ 0 0 0 1   0 1 0 1
-----------------------
  0 0 1 0   1 0 1 1     1011 no es valido
+             0 1 1 0   sumamos 6
-----------------------
0 0 1 1 1 0 0 0 1       el acarreo del resultado se suma al siguiente dígito
```


Suma en BCD 8421

Ejemplo de suma: 9 + 9 = 18

```
    1 0 0 1
  + 1 0 0 1
-------------
0 0 0 1   0 0 1 0   acarreo en 10010
        + 0 1 1 0   sumamos 6
-------------------
  0 0 0 1   1 0 0 0
```

Ejemplo de suma: 69 + 39 = 108

```
    0 1 1 0   1 0 0 1
  +   0 0 1 1   1 0 0 1
-----------------------
    1 0 0 1 1 0 0 1 0     acarreo en 10010
-----------------------
    1 0 1 0   0 0 1 0     1010 no es valido
+   0 1 1 0   0 1 1 0     sumamos 6
-----------------------
0 0 0 1 1 0 0 0 0 1 0 0 0 acarreo en 10000
```

Resta en BCD 8421

Ejemplo: 789 – 359 = 430

789 - 359 = 789 + (- 359)
complemento a 10 1000 - 359 = 641

		0	0 1 1 1	1 0 0 0	1 0 0 1
	+	1	0 1 1 0	0 1 0 0	0 0 0 1
los 3 dígitos son invalidos		1	1 1 0 1	1 1 0 0	1 0 1 0
sumamos 6	+		0 1 1 0	0 1 1 0	0 1 1 0
los 3 dígitos producen acarreo		1	1 0 0 1 1	1 0 0 1 0	1 0 0 0 0
el acarreo del signo se descarta		1 0	0 1 0 0	0 0 1 1	0 0 0 0

Estructura del tema

- Introducción
- Códigos binarios para números decimales
  - El código BCD 8421
  - Otros códigos: BCD 2421, Exceso 3, Biquinario, Gray, Johnson
- Códigos alfanuméricos
  - El código ASCII
  - Otros códigos: ASCII extendido, EBCDIC, UNICODE
- Códigos para detección y corrección de errores
  - Distancia de Hamming
  - Códigos de detección de errores
  - Códigos de corrección de errores
- Resumen y bibliografía

El código Aiken: BCD 2421

Este código es ponderado: la designación 2421 indica los pesos binarios de cada uno de los cuatro bits: $2^1\ 2^2\ 2^1\ 2^0$

Decimos que es un código autocomplementado, es decir, cambiando 0's por 1's obtenemos el complemento a 9 del número correspondiente

0000	0
0001	1
0010	2
0011	3
0100	4
0101	no valido
0110	no valido
0111	no valido
1000	no valido
1001	no valido
1010	no valido
1011	5
1100	6
1101	7
1110	8
1111	9

El código Exceso 3

Este código no es ponderado, es decir, los bits no tienen un peso asociado

Cada palabra del código se obtiene sumando 3 a la palabra correspondiente del BCD 8421

También se trata de un código autocomplementado, es decir, cambiando 0's por 1's obtenemos el complemento a 9 del número correspondiente

0000	no valido
0001	no valido
0010	no valido
0011	0
0100	1
0101	2
0110	3
0111	4
1000	5
1001	6
1010	7
1011	8
1100	9
1101	no valido
1110	no valido
1111	no valido

El código Biquinario

El código biquinario es un código no ponderado de 7 bits

El primer bit indica si el número está dentro del rango 5 – 9

El segundo bit indica si el número está dentro del rango 0 – 4

Cada uno de los bits restantes corresponde a un número del rango

0100001	0
0100010	1
0100100	2
0101000	3
0110000	4
1000001	5
1000010	6
1000100	7
1001000	8
1010000	9

El código Biquinario

La desventaja de este código es que hacen falta 7 bits para representar lo que los anteriores representaban con 4

La ventaja es la facilidad para detectar errores en un sólo bit

- Todas las palabras del código tienen dos 1
- Cualquier cambio accidental de un bit da lugar a una palabra que no pertenece al código

0100001	0
0100010	1
0100100	2
0101000	3
0110000	4
1000001	5
1000010	6
1000100	7
1001000	8
1010000	9

El código Gray

Se trata de un código no ponderado y no aritmético

El código Gray puede tener cualquier número de bits

Su característica más importante es que es continuo, es decir, sólo varía un bit desde una palabra del código hasta la siguiente

También es un código cíclico porque hay continuidad entre la primera y la última palabra

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

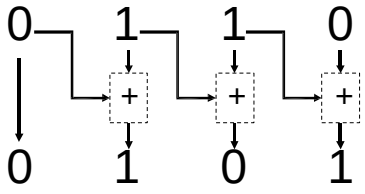
Conversión binario - Gray

El bit más significativo siempre es el mismo

De izquierda a derecha se debe sumar cada par adyacente de bits para obtener el siguiente bit en Gray

Los acarreos deben descartarse

Por ejemplo, ésta es la forma de convertir el número binario 0110 a Gray:



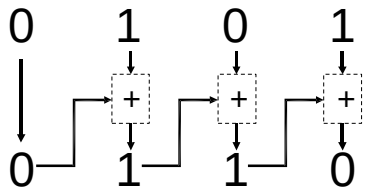
Conversión Gray - binario

El bit más significativo siempre es el mismo

De izquierda a derecha, cada bit del código binario se calcula sumando el correspondiente bit Gray con el bit anterior del código binario

Los acarreos deben descartarse

Por ejemplo, ésta es la forma de convertir el número binario 0110 a Gray:



El código Johnson

Al igual que el código Gray, se trata de un código
- No ponderado
- No aritmético
- Continuo
- Cíclico

Un código Johnson puede tener cualquier número de bits; con n bits se puede codificar 2^n valores

Resulta sencillo diseñar contadores muy rápidos usando este código

00000	0
00001	1
00011	2
00111	3
01111	4
11111	5
11110	6
11100	7
11000	8
10000	9

Estructura del tema

- Introducción
- Códigos binarios para números decimales
  - El código BCD 8421
  - Otros códigos: BCD 2421, Exceso 3, Biquinario, Gray, Johnson
- Códigos alfanuméricos
  - El código ASCII
  - Otros códigos: ASCII extendido, EBCDIC, UNICODE
- Códigos para detección y corrección de errores
  - Distancia de Hamming
  - Códigos de detección de errores
  - Códigos de corrección de errores
- Resumen y bibliografía

Códigos alfanuméricos

Los ordenadores no sólo necesitan representar números, sino también letras y otros símbolos

Los códigos alfanuméricos representan

- Números
- Caracteres alfabéticos
- Símbolos
- Instrucciones necesarias para la transferencia de información

Un código alfanumérico debe representar, como mínimo, los 10 dígitos decimales y las 26 letras (inglés)

Códigos alfanuméricos

Dado que estos 36 elementos no pueden representarse con 5 bits ($2^5 = 32$), como mínimo será necesario utilizar 6 bits ($2^6 = 64$)

Para representar los dígitos decimales y las letras del alfabeto se usarán 36 combinaciones

Las 28 combinaciones de 6 bits restantes podrán ser utilizadas para representar

- Símbolos (espacio, coma, punto, dos puntos…)
- Instrucciones para permitir que el emisor y el receptor se comuniquen entre ellos acerca de qué debe hacerse con la información (avance de línea, retorno de carro…)

El código ASCII

El código ASCII (American Standar Code for Information Interchange) es un código alfanumérico utilizado universalmente en la mayoría de ordenadores y equipos electrónicos

El código ASCII dispone de 128 caracteres que se representan mediante un código binario de 7 bits

La mayoría de los teclados usan el código ASCII para enviar información al ordenador cada vez que se pulsa una tecla o una combinación de teclas

El código ASCII

En realidad, el código ASCII se considera un código de 8 bits que siempre tiene a 0 el bit más significativo

H	o	l	a
$72_{(10)}$	$111_{(10)}$	$108_{(10)}$	$97_{(10)}$
$01001000_{(2)}$	$01101111_{(2)}$	$01101100_{(2)}$	$01100001_{(2)}$
$48_{(16)}$	$6F_{(16)}$	$6C_{(16)}$	$61_{(16)}$

Los primeros 32 caracteres son comandos de control y no son gráficos, es decir, nunca se imprimen o se muestran por pantalla

Los otros 96 caracteres son símbolos gráficos que pueden imprimirse o mostrarse por pantalla: dígitos decimales, letras mayúsculas y minúsculas, símbolos...

El código ASCII extendido

IBM adoptó 128 caracteres adicionales para usar el código ASCII en sus ordenadores personales (PCs)

Los nuevos caracteres tienen el bit más significativo a 1

Debido a la popularidad de los PCs, estos caracteres se han convertido en un estándar no oficial

- Caracteres alfabéticos no ingleses
- Símbolos de moneda no ingleses
- Letras griegas
- Símbolos matemáticos
- Caracteres para gráficos y para gráficos de barras
- Caracteres sombreados

El código EBCDIC

El código EBCDIC (Extended Binary Coded Decimal Interchange Code) es una extensión del código BCD desarrollada por IBM para normalizar el código utilizado por sus productos

Este código representa 256 caracteres de 8 bits, lo que permite ir más allá de los dígitos decimales del BCD

El código EBCDIC se utiliza poco en ordenadores, ya que la necesidad de utilizar 256 caracteres queda cubierta con el uso del código ASCII extendido

Aumentando el número de caracteres

Los 256 caracteres de los códigos ASCII extendido y EBCDIC no son suficientes para representar los caracteres usados por otras lenguas

Existen variantes ampliadas de EBCDIC que permiten representar más caracteres

El código que se está convirtiendo en el más popular es UNICODE, basado en ASCII, cuyos objetivos son

- Tratar de representar todos los caracteres existentes
- Facilitar la comunicación de datos entre máquinas de cualquier lugar del mundo

Estructura del tema

- Introducción
- Códigos binarios para números decimales
  - El código BCD 8421
  - Otros códigos: BCD 2421, Exceso 3, Biquinario, Gray, Johnson
- Códigos alfanuméricos
  - El código ASCII
  - Otros códigos: ASCII extendido, EBCDIC, UNICODE
- Códigos para detección y corrección de errores
  - Distancia de Hamming
  - Códigos de detección de errores
  - Códigos de corrección de errores
- Resumen y bibliografía

Detección y corrección de errores

En los sistemas digitales pueden producirse fallos físicos que causen una diferencia entre los datos transmitidos y los datos recibidos

- Una radiación podría cambiar el valor de un bit en memoria
- Una alta temperatura podría desconectar cables

Para detectar estos errores es necesario asegurar los datos usando los códigos adecuados

- Los códigos detectores permiten detectar errores
- Los códigos correctores permiten corregir errores

El concepto de n-cubo

Para comprender el concepto de distancia booleana es necesario definir el concepto de n-cubo

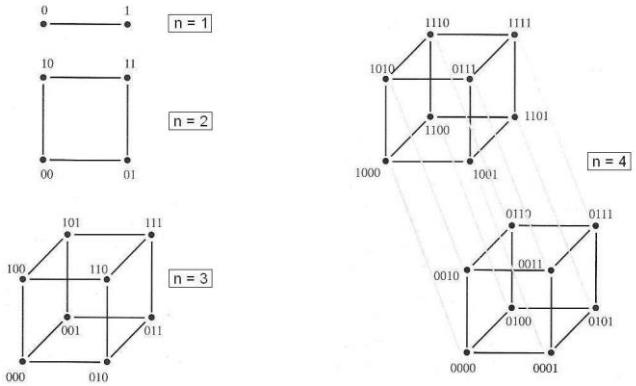
Un n-cubo binario es un cubo con un total de 2^n vértices, cada uno de los cuales se corresponde con una combinación particular de n bits

Dos vértices están unidos por una arista si y sólo si las combinaciones de bits correspondientes difieren en un único bit

El concepto de n-cubo

Estos son ejemplos de n-cubo para los siguientes cuatro valores de n: 1, 2, 3, 4

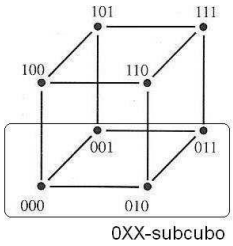
Un n-cubo con n mayor que 4 es difícil de visualizar



El concepto de m-subcubo

Dentro de un n-cubo puede haber varios m-subcubos

- Por ejemplo, los vértices 000, 001, 011 y 010 forman un 2-subcubo dentro de un 3-cubo
- Los m bits tomarán todas las 2^m posibles combinaciones
- Los restantes n – m bits siempre tienen el mismo valor



0XX-subcubo

Un m-subcubo puede especificarse con una combinación de n bits que contiene los n – m bits fijos y una X en las restantes m posiciones para indicar que el valor de esos bits es indiferente

Distancia booleana o de Hamming

El n-cubo proporciona una interpretación geométrica del concepto de distancia booleana, también llamada distancia de Hamming

La distancia entre dos vértices es igual al número de pasos del camino más corto entre ellos

- La distancia entre dos vértices adyacentes siempre es 1
- Dos vértices no adyacentes dentro de cualquier 2-subcubo siempre están a una distancia 2

La distancia entre dos combinaciones de bits es igual al número de bits en que difieren

Estructura del tema

- Introducción
- Códigos binarios para números decimales
  - El código BCD 8421
  - Otros códigos: BCD 2421, Exceso 3, Biquinario, Gray, Johnson
- Códigos alfanuméricos
  - El código ASCII
  - Otros códigos: ASCII extendido, EBCDIC, UNICODE
- Códigos para detección y corrección de errores
  - Distancia de Hamming
  - Códigos de detección de errores
  - Códigos de corrección de errores
- Resumen y bibliografía

Códigos de detección de errores

Cualquier código de n bits se puede considerar como un subconjunto de todas las combinaciones de n bits

- Las combinaciones de bits incluidas en este subconjunto son las palabras del código
- Las combinaciones no incluidas en el subconjunto son palabras que no pertenecen al código

Un código de detección de errores es aquel que tiene la propiedad de que ciertos tipos de errores pueden transformar palabras del código en palabras que no son del código, permitiendo la inmediata detección del error

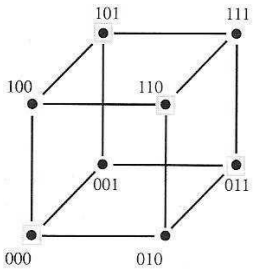
Deteccion de errores en un bit

Para detectar un error en un solo bit es necesario que la distancia mínima entre dos palabras del código sea 2

Dicho de otra manera, las palabras del código no pueden corresponder a vértices adyacentes del n-cubo correspondiente

Cualquier error en un bit dará lugar a una palabra que no pertenece al código

Si en una comunicación se recibe una palabra que no pertenece al código, es seguro que se ha producido un error



Bit de paridad

En general, para mantener la distancia 2 en cualquier código arbitrario es necesario utilizar sólo la mitad de las combinaciones posibles

n bits 2^n combinaciones posibles

2^n I 2 = 2^{n-1} solo quedan *n* -1 bits para codificar datos

Esto implica que para codificar un dato de n bits será necesario utilizar n + 1 bits

Los n bits son denominados bits de información, ya que contienen el dato, mientras que el bit adicional es comúnmente denominado bit de paridad

Codigos de paridad

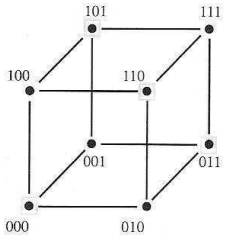
El bit de paridad puede tomar un valor tal que el número de unos de las palabras del código sea par y el número de unos de las palabras que no son del código sea impar, o viceversa

Estos códigos se denominan códigos de paridad de un bit para enfatizar el hecho de que se usa un único bit de paridad

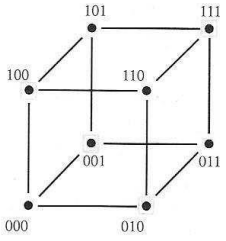
- Un código de paridad par es aquel en el que todas las palabras del código tienen un número par de unos
- Un código de paridad impar es aquel en el que todas las palabras del código tienen un número impar de unos

Codigos de paridad

Estos ejemplos muestran dos códigos de paridad de un bit, uno par y otro impar, con 2 bits de información



código de paridad par



código de paridad impar

El bit de paridad se puede añadir al principio o al final del código pero, al diseñar el circuito, es necesario determinar cuál es

Método de paridad para la deteccion de errores

- Un determinado sistema puede trabajar con paridad par o con paridad impar, pero no con ambas
- Una palabra es errónea si no cumple la paridad
  - A pesar de que se detecta la existencia de un error, no se puede detectar cuál es el bit erróneo y, por tanto, no se puede corregir el error
  - Siempre que se reciba información con paridad incorrecta será necesario solicitar una retransmisión de la misma
- Un código de paridad de 1 bit permite detectar errores en un solo bit (o en un número impar de bits), pero nunca en dos bits (o en un número par de bits), ya que el resultado sería otra palabra del código

Estructura del tema

- Introducción
- Códigos binarios para números decimales
  - El código BCD 8421
  - Otros códigos: BCD 2421, Exceso 3, Biquinario, Gray, Johnson
- Códigos alfanuméricos
  - El código ASCII
  - Otros códigos: ASCII extendido, EBCDIC, UNICODE
- Códigos para detección y corrección de errores
  - Distancia de Hamming
  - Códigos de detección de errores
  - Códigos de corrección de errores
- Resumen y bibliografía

Codigos de correccion de errores

Un bit de paridad hace que la distancia mínima entre palabras del código sea 2, lo que permite detectar la aparición de un error en una palabra del código

Sin embargo, un único bit de paridad no indica dónde se produjo el error, es decir, cuál es el bit incorrecto

Si somos capaces de identificar el bit incorrecto, bastaría con cambiar su valor para corregir el error

Si queremos determinar dónde se produjo el error necesitaremos bits adicionales para aumentar la distancia mínima entre palabras a 3

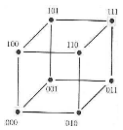
Ejemplo de codigo de correccion de errores

Supongamos que se va a enviar un bit de información

Usaremos un código de tres bits con distancia mínima 3

0 000

1 111



Este código nos permite corregir un error en un bit

Si hubiera un error en un bit del código 0, la palabra no formaría parte del código, pero tendría dos ceros, así que sabemos que el valor original era un 0

Si hubiera un error en un bit del código 1, la palabra no formaría parte del código, pero tendría dos unos, así que sabemos que el valor original era un 1

Ejemplo de codigo de correccion de errores

El código anterior también nos permite detectar errores en dos bits, ya que la distancia mínima es 3

Este código puede usarse para detectar dos errores o para corregir un error, pero nunca para las dos cosas al mismo tiempo

- Si recibimos 011, 101 ó 110 no sabríamos si el original era 000 y ha habido dos errores, o si el original era 111 y debemos corregir el error
- Si recibimos 001, 010 ó 100 no sabríamos si el original era 111 y ha habido dos errores, o si el original era 000 y debemos corregir el error

El método de Hamming

El método de Hamming es un método general para la construcción de códigos de corrección de errores usando una distancia mínima 3

Por cada valor entero m hay un código de Hamming con $2^m - 1$ bits que contiene:

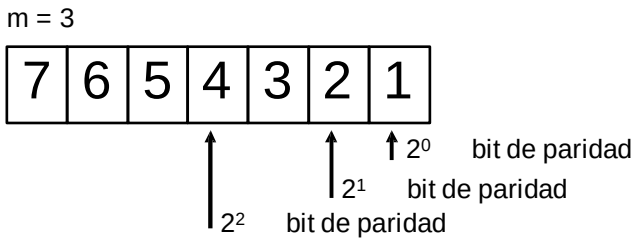
- m bits de paridad
- $2^m - 1 - m$ bits de información

Por ejemplo, para m = 3 se obtiene un código (7,4) con un total de 7 bits: 4 bits de información y 3 de paridad

Codigos de Hamming

Los bits de paridad de un código de Hamming están entremezclados con los bits de información

Si numeramos las posiciones de los bits desde 1 hasta $2^m - 1$ podemos decir que hay un bit de paridad en la posición 2^k, donde $0 \leq k \leq (m - 1)$



Codigos de Hamming

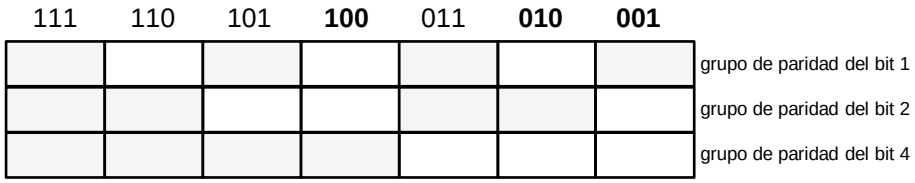
El valor de los bits de paridad se escoge de forma que el número total de unos en un grupo de bits sea par o impar (pero siempre la misma opción para todos los grupos)

Los grupos de bits se escogen de manera que no haya dos bits que se cubran con la misma combinación de bits de paridad

En concreto, al bit de paridad en la posición 2^k le corresponden todos los bits situados en una posición cuya representación binaria tenga un 1 en la posición 2^k

Codigos de Hamming

Este ejemplo muestra los grupos de paridad de un código de Hamming (7,4)



La distancia mínima de este código es 3

Un error en un bit afecta al menos a un bit de paridad

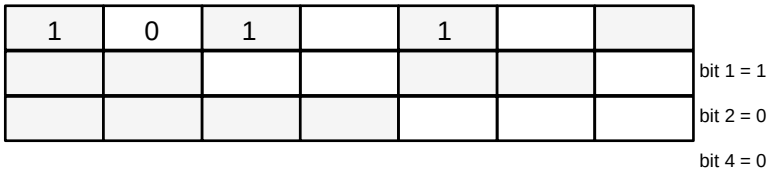
Un error en dos bits no afectaría a un bit de paridad si los dos estuvieran en el mismo grupo, pero siempre existirá un grupo de paridad que no los incluye a ambos

Calculo de la paridad

Dada una combinación binaria de 4 bits, es posible calcular el correspondiente código de Hamming de 7 bits

Por ejemplo, dada la combinación 1011 sabemos que se trata de los bits de información y que van en las posiciones 7, 6, 5 y 3

A partir de estos valores calculamos los bits de paridad

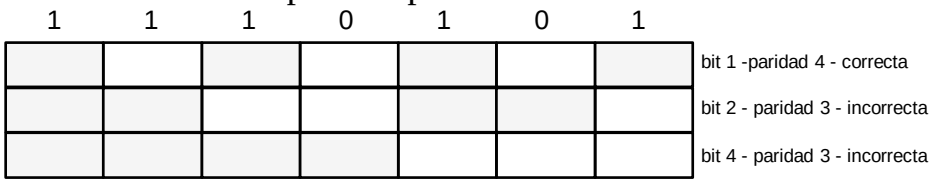


El código de Hamming de paridad par será: 1010101

Correccion de un error en un bit

Se calcula la paridad de cada grupo y la suma de las posiciones de los bits de paridad incorrectos nos dará la posición del bit incorrecto

Por ejemplo, si recibimos la palabra código 1110101 en un sistema de paridad par:



Los bits de paridad incorrectos son el 2 y el 4, lo que significa que el bit erróneo es el 6 y el número transmitido originalmente debió ser 1010101

Correccion y deteccion de errores

También resulta posible el diseño de códigos que permitan simultáneamente detectar y corregir errores

En general, para ser capaz de corregir x errores de un bit, un código necesita que la distancia mínima entre palabras sea de (2x + 1)

Un código que tenga distancia mínima (2x + y + 1) será capaz no sólo de corregir x errores de un bit, sino también de detectar hasta y errores adicionales

Estructura del tema

- Introducción
- Códigos binarios para números decimales
  - El código BCD 8421
  - Otros códigos: BCD 2421, Exceso 3, Biquinario, Gray, Johnson
- Códigos alfanuméricos
  - El código ASCII
  - Otros códigos: ASCII extendido, EBCDIC, UNICODE
- Códigos para detección y corrección de errores
  - Distancia de Hamming
  - Códigos de detección de errores
  - Códigos de corrección de errores
- Resumen y bibliografía

Resumen

- Existe una gran diversidad de tipos de código que permiten representar información en binario
  - Códigos aritméticos que permiten representar números y realizar operaciones con ellos
  - Códigos no aritméticos que permiten representar números pero sacrifican la posibilidad de operar a cambio de obtener otras ventajas
  - Códigos alfanuméricos que permiten representar no sólo números, sino también letras, símbolos…
- Estos códigos pueden utilizar técnicas para la detección y corrección de errores con el fin de mejorar la seguridad del almacenamiento y transmisión de datos

Bibliografía

Sistemas Digitales - Principios y Aplicaciones - Tocci - 10th