HAMMING BLOCK CODES

January 2000

Jose Micolau  y7325919@alumnes.eup.udl.es
Oscar Juvillà  p4370720@alumnes.eup.udl.es
David Rodriguez  b3698299@alumnes.eup.udl.es
Nuria Fornos  q7315742@alumnes.eup.udl.es
Joan Antoni Vidal  p4371771@alumnes.eup.udl.es
# Table of Contents

- **Introduction to Codes**
  - Historical Records
  - Introduction to Coding Theory
    - Basic Assumptions
    - Correcting and Detecting Error Patterns
  - Weight and Distance
  - Maximum Likelihood Decoding
  - Error-Detecting Codes
  - Error-Correcting Codes
- **Block & Linear Block Codes**
  - Introduction
  - Generator and Parity Check Matrix
  - Syndromes
- **The Hamming Code**
  - Encoding a (7,4) Hamming Code
  - Decoding a (7,4) Hamming Code
  - Example of coding and decoding
  - Example for a Hamming Code Real Application
- **Technical Applications**
- **Bibliography & Urls**
  - Bibliography
  - Urls
Introduction to Codes

Historical Records

Over the course of the last seven decades, research in the two areas has often been closely intertwined. The earliest researchers can be traced back to the 1930’s.

This was the time when the mathematical foundations of the theory of computing were being laid. In 1937, Claude Shannon showed how boolean algebra can be used for design and analysis of logic circuits. Later in 50’s, Edward Moore and George Mealy made fundamental contributions to automata theory.

Meanwhile, another mathematician George Stibitz, designed the first binary relay computer in 1937. It was around then that researchers began looking at error-correction as a way to build fault-tolerant computing hardware. Motivated by this, in 1948, Richard Hamming laid the foundations of error-correcting codes.

Around the same time, Edgar Gilbert came up with a lower bound on the size of an error-correcting code with a certain minimum distance (the famous Gilbert Bound), and David Slepian established underpinnings of algebraic coding theory. In 1968, Hamming was awarded the prestigious Turing award for his pioneering work. Some of the early combinatorialists at the Math Center include people such as Riordan, who wrote a celebrated text on combinatorial analysis in 1958, and Ron Graham who won George Polya prize for pathbreaking work in Ramsey theory.

Hamming is best known for his work on error-detecting and error-correcting codes. His fundamental paper on this topic appeared in 1950 and with this he started a new subject within information theory. Hamming codes are of fundamental importance in coding theory and are of practical use in computer design.

Work in codes is related to packing problems and error-correcting codes due to Hamming led to the solution of a packing problem for matrices over finite fields.

In 1956 Hamming worked on the early computer, the IBM 650. His work here led to the development of a programming language which has evolved into the High-level computer languages used to program computers today.

Hamming also worked on numerical analysis, integrating differential equations, and the Hamming spectral window which is much used in computation for smoothing data before Fourier analysing it.

His major works include Numerical Methods for Scientists and Engineers (1962), Introduction to applied numerical analysis (1971), Digital filters (1977), Coding and information theory (1980), Methods of mathematics applied to calculus, probability, and statistics (1985), Introduction to applied numerical analysis

The error correcting codes (Golay, Hamming, Shanon) for their side probabilistic and Algebraic, used in the telecommunications engineering for their importance in the data transmission and in the computer engineering for their use in the digital supports, to be able to conserve the quality of the information and the communications against the noise, give the distortions and the lost of the chanel by mathematical tecnis (from the theory of probability to the calculate combination or linear algebra until the arithmetic, the bodies theory or the algebraic geometry).

First of all Shanon study the probabilistic focus (Shanon Theory), later some people studied a more algebraic focus about this problem with practical examples (that is using at the moment) such as Golay or Hamming Codes, the Cyclic Codes, BCH or Reed-Solomon and Reed-Muller Codes.

Finally at 70s, Goppa studied new codes construction starting from flat algebraic curves (Goppa’s Geometric Codes or algebraic-Geometrics Codes)[Algebraic-Geometric formula such as Riemann-Roch Theorem].

This Theories allows the explicit construction of code families (80s) the parameters of which surpass assintheticlly the Varshamov-Gilbert quota and consequently he gave one effective solution (with polynomial complexity) to the initial problem from codes theory thank from Shanon in probabilistic terms. But without a constructive idea.

The first efficient algorithms at end of 80s were from Justessen Et Al, Skorobogatov and Vladut and Porter, but so far from the correction capacity from the codes to those that are applied, and someone need restrictive conditions about the codes type that they can used, until the appearance of Ehrhard or Duursma Algorithms.

Nowadays is developing more fast and efficient algorithms (although losing generalization) based in the design of majority decoding from Fewe and Rao, which use lineal recurrent relations (Sakata Algorithm) or Gröbner bases (Saints i Heegard Algorithm).

However the codes AG (Algebraic-Geometrics) aren't more implemented in the practice from engineers because the mathematical depth of the underlying idea, for this reason, the mathematicians are making a description of the codes and the practical treatment using a more elemental approach.

However the parameters from AG codes are better than classic codes in assinthetic sense (that is same, for large codes) for another hand classic codes nowadays used have faster and effective decode.

Finally, although decode methods for Goppa’s geometric codes are effective, their process is so difficult and involve complex algorithms based in computational AG methods. For Example semigroups of Weierstrass and his distance of Feng-Rao, or Hamburger-Noether and Brill-Noether algorithms.
Only say to end, that the study of the theory of codes is intimately tied to a series of topics own of the discrete mathematics such as reticul, closed of spheres, exponential sums, the theory of graphs or the arithmetic geometry, as well as another various origin themes series such as the theory of the information, the cryptography, the computational Algebra or the theory of the sign.

Introduction to Coding Theory

Coding theory is the study of methods for efficient and accurate transfer of information from one place to another. The theory has been developed for such diverse applications as the minimization of noise from compact disc recorders, the transmission of financial information across telephone line, data transfer from one computer another or from memory to the central processor, and information transmission from a distance source such as a weather or communications satellite or the Voyager spacecraft which sent pictures of Jupiter and Saturn to Earth.

The physical medium through which the information is transmitted is called a channel. Undesirable disturbances, called noise, may cause the information received to differ from what was transmitted. Noise may be caused by sunspots, lightning, folds in a magnetic tape, meteor showers, competing telephone messages, random radio disturbance, or many other things. Coding theory deals with the problem of detecting and correcting transmission errors caused by noise on the channel.

In practice, the control we have over this noise is the choice of a good channel to use for transmission and the use of various noise filters to combat certain types of interference which may be encountered.

Once we have settled on the best mechanical system for solving these problems, we can focus our attention on the construction of the encoder and the decoder. Our desire is to construct these in such a way as to effect:
1. fast encoding of information
2. easy transmission of encoded messages
3. fast decoding of received messages
4. correction of errors introduced in the channel, and
5. maximum transfer of information per unit time.

The primary goal is the fourth of these. The problem is that it is not generally compatible with the fifth, and also may not be specially compatible with the other three. So any solution is necessary a trade-off among the five objectives.

Basic Assumptions

The information to be sent is transmitted by a sequence of 0s and 1s. We call a 0 or a 1 a digit. A word is sequence of digits. The length of a word is the number of digits in the word. A word is transmitted by sending its digits, one after the other, across a binary channel. Each digit is transmitted mechanically, electrically, magnetically, or other used by one of two types of easily differentiated pulses.
A block code is a code having all its words of the same length. We also need to make certain basic assumptions about the channel. The first assumption is that a codeword of length \( n \) consisting of 0s and 1s is received as a word of length \( n \) consisting of 0s and 1s although not necessarily the same as the word that was sent. The second is that there is no difficulty identifying the beginning of the first word transmitted. The final assumption is that the noise is scattered randomly as opposed to being in clumps called bursts. That is, the probability of any one digit being affected in transmission is the same as that of any other digit and is not influenced by errors made in neighboring digits.

In a perfect, or noiseless, channel, the digit sent, 0 or 1, is always the digit received. If all channels were perfect, there would be no need for coding theory. But fortunately (or unfortunately, perhaps) no channel is perfect; every channel is noisy. Some channels are less noisy, or more reliable, than others.

**Correcting and Detecting Error Patterns**

Suppose a word is received that is not a codeword. Clearly some errors have occurred during the transmission process, so we have detected that an error (perhaps several errors) has occurred. If however a codeword is received, then perhaps no errors occurred during transmission, so we cannot detect any error. The concept of correcting an error is more involved.

**Weight and Distance**

Let \( v \) be a word of length \( n \). The Hamming weight, or simply the weight, of \( v \) is the number of times the digit 1 occurs in \( v \). We denote the weight of \( v \) by \( \text{wt}(v) \).

Let \( v \) and \( w \) be words of length \( n \). The Hamming distance, or simply distance, between \( v \) and \( w \) is the number of positions in which \( v \) and \( w \) disagree. We denote the distance between \( v \) and \( w \) by \( d(v,w) \).

Note that the distance between \( v \) and \( w \) is the same as the weight of the error pattern \( u = v + w \):

\[
d(v,w) = \text{wt}(v+w)
\]

**Maximum Likelihood Decoding**

There are two quantities over which we have no control. One is the probability \( p \) that our BSC will transmit a digit correctly. The second is the number of possible messages that might be transmitted. The two basic problems of coding then are:

- **Encoding.** We have to determine a code to use for sending our messages. We must make some choices. First we select a positive integer \( k \), the length of each binary word corresponding to a message. Next we decide how many digits we need to add to each word of length \( k \) to ensure that as many errors can be corrected or detected as we require; this is the choice of the codewords.
and the length of the code, n. To transmit a particular message, the transmitter finds the word of length k assigned to that message, then transmits the codeword of length n corresponding to that word of length k.

- Decoding. A word w in k^n is received. We now describe a procedure called maximum likelihood decoding, or MLD, for deciding which word v in C (binary code, or set of words) was sent. There are actually two kinds of MLD.

1) Complete Maximum Likelihood Decoding, or CMLD. If there is one and only one word w in C closer to w than any other word in C, we decode w as v. That is, if d(v,w)<d(v₁,w) for all v₁ in C, v₁≠v, then decode w as v. If there are several words in in C closest to w, i.e., at the same distance from w, then we select arbitrarily one of them and conclude that is was the codeword sent.

2) Incomplete Maximum Likelihood Decoding, or IMLD. If there is a unique word v in C closest to w, then we decode w as v. But if there are several words in C at the same distance from w, then we request a retransmission. In some cases we might even ask for a retransmission if the received word w is too far away from any word in the code.

Error-Detecting Codes

We now make precise the notion of when a code C will detect errors. Recall that if v is in C is sent and w in k^n is received, then u=v+w is the error pattern. Any word u in k^n can occur as an error pattern, and we wish to know which error patterns C will detect.

We say that code C detects the error pattern u if and only if v+u is not a codeword, for every v in C. In other words, u is detected if for any transmitted codeword v, the decoder, upon receiving v+u can recognize that it is not a codeword and hence that some error has occurred.

The table constructed for IMLD can be used to determine which error patterns a code C will detect. The first column lists every word in k^n. Hence the first column can be reinterpreted as all possible error patterns, in which case the error pattern columns in the IMLD table then contain the sums v+u, for all v in C. If in any particular row none of these sums are codewords in C, then C detects the error pattern in the first column of that row.

An alternative and much faster method for finding the error patterns that code C can detect is to first find all error patterns that C does not detect; then all remaining error patterns can be detected by C. Clearly, for any pair of codewords v and w, if e=v+w then e cannot be detected, since v+e=w, which is a codeword. So the set of all error patterns that cannot be detected by C is the set of all words that can be written as the sum of two codewords.

There is also a way of determining some error patterns that code C will detect without any manual checking. Firs we have to introduce another number associated with C.
For a code C containing at least two words the distance of the code C is the smallest of the numbers \(d(v,w)\) as \(v\) and \(w\) range over all pairs of different codewords in C. Note that since \(d(v,w) = wt(v+w)\), the distance of the code is the smallest value of \(wt(v+w)\) as \(v\) and \(w\), \(v \neq w\) range over all possible codewords.

The distance of a code has many of the properties of Euclidean distance; this correspondence may be useful to assist in understanding the concept of the distance of a code.

Now we can state a theorem which helps to identify many of the error patterns a code will detect.

**THEOREM 1.** A code C of distance d will at least detect all non-zero error patterns of weight less than or equal to d-1. Moreover, there is at least one error pattern of weight d which C will not detect.

Remark Notice that C may detect some error patterns of weight d or more, but does not detect all error patterns of weight d.

A code is an t error-detecting code if it detects all error patterns of weight at most t and does not detect at least one error pattern of weight t+1. So, in view of Theorem 1, if a code has distance d then it is a d-1 error-detecting code.

Theorem 1 does not prevent a code C from detecting error patterns of weight d or greater. Indeed, C usually will detect some such error patterns.

**Error-Correcting Codes**

If a word \(v\) in a code C is transmitted over a BSC and if \(w\) is received resulting in the error pattern \(u = v+w\), then IMLD correctly concludes that \(v\) was sent provided \(w\) is closer to \(v\) than to any other codeword. If this occurs every times the error pattern \(u\) occurs, regardless of which codeword is transmitted, then we say that C corrects the error pattern \(u\). That is, a code C corrects the error pattern \(u\) if, for all \(v\) in C, \(v+u\) is closer to \(v\) than to any other word in C. So a code is said to be an t error-correcting code if it corrects all error patterns of weight at most t and does not correct at least one error pattern of weight t+1.

The IMLD table can be used to determine which error patterns a code C will correct. In each error pattern column of the table, all possible error patterns (which means each word in \(k^n\)) occurs once and only once.

Also, an asterisk is placed beside the error pattern \(u\) in the column corresponding to a codeword \(v\) in the IMLD table precisely when \(v+u\) is closer to \(v\) than it is to any other codeword. Therefore an error pattern \(u\) is corrected if an asterisk is placed beside \(u\) in every column of the IMLD table.

The distance of a code can be used to devise a test for error-correcting which avoids at least some of the manual checking from the MLD table. The next theorem gives the test. Recall that the symbol \(\lfloor x \rfloor\) denotes the greatest integer less than or equal to the real number x.
THEOREM 2. A code of distance $d$ will correct all error patterns of weight less than or equal to $\lfloor (d-1)/2 \rfloor$. Moreover, there is at least one error pattern of weight $1 + \lfloor (d-1)/2 \rfloor$ which $C$ will not correct.

In view of this theorem it is clear that any code of distance $d$ is a $\lfloor (d-1)/2 \rfloor$-error-correcting code. Theorem 2 does not prevent a code $C$ of distance $d$ from correcting error patterns of weight greater than $\lfloor (d-1)/2 \rfloor$. 
**Block & Linear Block Codes**

**Introduction**

A block code is a rule for converting a sequence of source bits $s$, of length $k$, into a transmitted sequence $t$ of length $n$ bits, where, in order to add redundancy, $n$ will be greater than $k$.

Block codes are not necessarily linear, but in general all block codes used in practice are linear. Linear codes are divided into two categories: block and convolutional codes. In a block code, the transmitted sequence is always a multiple of $n$ long. This is because information is transmitted in blocks of data that are $n$ bits long (hence the name). For an $(n,k)$ code, $C$, the blocks have a length of $n$, $k$ information bits encoded in them, and $n-k$ parity bits. Taking the code $(3,1)$ we have:

- $n = 3$ bits in a block
- $k = 1$ message bit (I)
- $n - k = 2$ parity bits (P)

![Figure 1. This figure shows the information bit and the parity bits.](image)

If each block is $n$ bits long, there are $2^n$ possible combinations. However, linear block codes consist of $2^k$ possible codewords. These $2^k$ possibilities form a subset of the $2^n$ set. These are called valid codewords.

An important feature of linear codes, is that the zero codeword is always a member of the valid codewords.

The minimum distance of a linear code is equal to the lowest weight of a nonzero codeword (the minimum number of bits that must be changed to end up with another codeword).

$t$ is the number of errors that can be corrected. $s$ is the number of errors that can be detected.

Because a decoder must be able to detect an error before it corrects an error, $s$ is always more than or equal to $t$.

- $s \geq t$
- $d_{\text{min}} \geq s + t + 1$
- $d_{\text{min}} \geq 2t + 1$ (for error correction only code)

A linear code maps $k$-bit messages ($2^k$ possible messages) to $n$-bit codewords. Only $2^k$ of the $2^n$ possibilities are used, leaving $2^n - 2^k$ illegal bit sequences. The next step in error correction is assigning each of these illegal bit sequences to their nearest codeword.
Generator and Parity Check Matrix.

The generator matrix for a linear block code is one of the basis of the vector space of valid codewords. The generator matrix defines the length of each codeword $n$, the number of information bits $k$ in each codeword, and the type of redundancy that is added; the code is completely defined by its generator matrix. The generator matrix is a $k \times n$ matrix.

Multiplying the $k$-bit information sequence $u(x)$ by the $(n,k)$ generator matrix gives the $n$-bit codeword.

<table>
<thead>
<tr>
<th>Redundancy (Parity Checks)</th>
<th>Original Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\langle 1 \ 0 \ 0 \rangle$</td>
<td>$\langle 1 \ 0 \ 1 \ 1 \rangle$</td>
</tr>
</tbody>
</table>

$n-k$ bits  \hspace{2cm} k bits

*Figure 2. Systematic Code*

We must remember that the generator matrix will always produce a codeword with the last $k$ bits being equal to the information sequence. When the information bits are passed into the codeword unchanged and parity bits are added this is called a systematic code. If the information bits are not directly represented then the code in called a nonsystematic code. Typically, systematic codes have every advantage of their nonsystematic counterparts plus the encoder only needs to produce $(n-k)$ bits instead of the entire codeword.

Syndromes

The **syndrome** is the receive sequence multiplied by the transposed parity check matrix $H$.

$$s(x) = r(x) \ast H^T$$

The syndrome is a $(n-k)$-tuple that has a one to one correspondence with the correctable error patterns. **The syndrome depends only on the error pattern and is independent of the transmitted codeword.**

Most codes do not use all of the redundancy that has been added for error correction. The only two codes known to do this are Hamming $(2^m - 1, 2^m - m - 1)$ and Golay $(23, 12)$ codes. These codes are called **perfect codes.**
The Hamming Code

A neat example of a block code is the (7,4) Hamming code. This is an error-detecting and error-correcting binary code, which transmits N=7 bits for every K=4 source bits. This kind of codes can detect and correct single bit errors or detects double bit errors.

Now we will see how to encode and decode a source sequence using in the transmission a symmetric channel.

The structure of a transmission with a Hamming Coding is:

![Diagram of transmission structure](image)

**Figure 3. Control coding path**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>u</td>
<td>Original message</td>
</tr>
<tr>
<td>v</td>
<td>Codeword</td>
</tr>
<tr>
<td>S(x)</td>
<td>Transmitted signal</td>
</tr>
<tr>
<td>N(x)</td>
<td>Noise signal</td>
</tr>
<tr>
<td>R(x)</td>
<td>Received signal</td>
</tr>
<tr>
<td>r</td>
<td>Received sequence</td>
</tr>
<tr>
<td>u'</td>
<td>decoded received sequence</td>
</tr>
</tbody>
</table>
Encoding a (7,4) Hamming Code.

\[ v = G^t u \]

- \( v \) is the transmitted codeword.
- \( u \) is the source sequence.
- \( G \) is the ‘generator matrix’ of the code. Check bits occupy positions that are powers of 2 or de latter positions. The number of parity/check bits needed is given bye the Hamming rule, a function of the number of bits of information transmitted. The matrix equation is \( G^t = [P | I_4] \) where \( I_4 \) is the 4*4 identity matrix, and \( P \) is the parity matrix.

The rules for obtaining the bits that compose the parity matrix are:
- \( P_0 = I_0+I_2+I_3 \) (which are the first three source bits).
- \( P_1 = I_0+I_1+I_2 \) (which are the last three source bits)
- \( P_2 = I_1+I_2+I_3 \) (which are first, third and fourth source bits)

\( v \) is the transmitted codeword

Decoding a (7,4) Hamming Code

In this special case of linear code, with binary symmetric channel, the decoding task can be re-expressed as syndrome decoding.

\[ s = H r \]

- \( s \) is syndrome.
- \( H \) is the parity checks matrix. \( H = [-P | I_3] \)
- \( r \) is the received vector.

We can find the following two possibilities:

- If the syndrome is zero, that is, all three parity checks agree with the corresponding received bits, then the received vector is a codeword, and the most probable decoding is given by reading out its first four bits. Then \( u' \) is supposed to be the same than \( u \).
  
  One can give the situation in that the errors are not detectable. This happens when the error vector is identical to a non null word code. In this case \( r \) it is the sum of two words code and therefore the syndrome is similar to zero. These errors are non-detectable errors. As there are \( 2k-1 \) non-null words code, there are \( 2k-1 \) non-detectable errors.

- If the syndrome is non-zero, then we are certain that the noise sequence for the present block was non-zero(we have noise in our transmissions). Since the received vector \( v \) is given by \( v = G^t u + n \) and \( H G^T = 0 \) the syndrome decoding problem is then to find the most probable noise vector \( n \) satisfying the equation \( Hn = e \). Once we have find the error vector, we can know which is the original source sequence.

Example of coding and decoding.

We suppose we have the ‘generator matrix’ \( G \) which have the following form:
a) Which is the minimum Hamming distance and its detecting and correcting capacity.

Hamming distance: \( d_H = \text{number of independent columns} + 1 = 3 + 1 = 4 \)

The detecting capacity is: \( d_H = D + 1. \) \( D = 3 \)

The correcting capacity is: \( d_H = 2C + 1. \) \( C = 1 \)

b) Which is the resultant code if the sequence source is 0100 or 1001

If the original message is \( u = 0100 \), to obtain the codeword \( v_1 \) we know we have to apply \( v_1 = u \times G^T \) and the result is \( v_1 = 0100110 \)

If the original message is \( u = 1001 \), to obtain the codeword \( v_2 \) we know we have to apply \( v_2 = u \times G^T \) and the result is \( v_2 = 1001110 \)

c) Which is the original message corresponding to the 11000011 and 11101100 codeword.

\[
H = \begin{pmatrix}
1 & 1 & 1 & 0 & 1 & 0 & 0 \\
0 & 1 & 1 & 1 & 0 & 1 & 0 \\
1 & 0 & 1 & 1 & 0 & 0 & 1
\end{pmatrix}
\]

The first condition to study on decoding the received sequence, is the following equation:

\( s = H \times r = 0 \)

It will give us the result if the syndrome is zero or non-zero.

- \( r_1 = 1100011 \)  
  \( r_1 \times H = s_1 = 000 \)  
  That means that the received sequence is a codeword and then \( u = u' \).

- \( r_2 = 1110110 \)  
  \( r_2 \times H = s_2 = 010 \). That means that we have an error on the vector received. For solving it, we have to make the syndrome table.
We have find which is the error in the transmission so the right received sequence must be:
\[ r_2 = r_2' - e_2 = 1110110 - 0000010 = 1110100 \]
Once that we have find the correct received sequence we can know the decoded received sequence.
\[ u_1' = 1100 \]
\[ u_2' = 1110 \]

**Example for a Hamming Code Real Application**

**Hamming Code (8:4)**
The Hamming code used in teletext is used to reduce the possibility of errors for the sending address and control information.

It is possible to correct a single error in the received byte and detect multiple errors, when there are 2, 4 or 6 error bits. When there are 3, 5, 7 or 8 errors in a byte, this results in a false message being decoded. The hamming code used in teletext is a (8,4) (n,k)

Code Rate Efficiency = k/n = 4/8 = 50%
Where D4 D3 D2 D1 are the data bits and C3 C2 C1 and C4 is the parity bit.

Where \( \oplus \) represents an Exclusive-OR operation (addition without carry)
Message Bits
Hamming Code Protection Bits

<table>
<thead>
<tr>
<th></th>
<th>b8</th>
<th>b7</th>
<th>b6</th>
<th>b5</th>
<th>B4</th>
<th>b3</th>
<th>b2</th>
<th>b1</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>O</td>
<td>0</td>
<td>O</td>
<td>O</td>
<td>0</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>0</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>0</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>D</td>
<td>0</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
</tbody>
</table>

Results of Parity Tests Inference Action

<table>
<thead>
<tr>
<th>A,B,C</th>
<th>Parity Tests</th>
<th>Inference</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>All Correct</td>
<td>Correct</td>
<td>no errors</td>
<td>Accept message bits</td>
</tr>
<tr>
<td>All Correct</td>
<td>Not Correct</td>
<td>error in b7</td>
<td>Accept message bits</td>
</tr>
<tr>
<td>Not all Correct</td>
<td>Correct</td>
<td>multiple errors</td>
<td>Reject message bits</td>
</tr>
<tr>
<td>Not all Correct</td>
<td>Not Correct</td>
<td>single errors</td>
<td>Refer to table above to identify error. Correct error if in message bit</td>
</tr>
</tbody>
</table>
Technical Applications

COMUNICATION SYSTEMS

Teletext systems. [View page 14]
Communication through satellite.
Broadcasting (radio and digital TV).
Telecommunications (digital phones).

INFORMATION SYSTEMS

Logical circuits.
Semiconductor memories.
Magnetic disks (HD).
Optic reading disks (CD-ROM).

AUDIO AND VIDEO SYSTEMS

Digital sound (CD).
Digital video (DVD).
Bibliography &Urls

Bibliography

• A First course in coding theory, Raymond Hill.
• Introduction to coding theory, Van Lint.
• Coding theory, Hoffman, Leonard, Lindrev.
• Error control coding: fundamentals and applications, Shu Lin, D.J. Costello, Jr

Urls

• http://www.ieee.org/about/awards/sums/hamming.htm
• http://www.ee.unb.ca/tervo/ee4253/Hamming.htm
• http://www.home.sch.bme.hu/~sanyo/coding/7cod.html
• http://www331.jpl.nasa.gov/public/imperfectness.html
• http://www-theory.dcs.st-and.ac.uk/~sal/school/cs3010/lectures/forhtml/node3.html
• http://www.en.polyu.edu.hk/~enmzwang/adc/l-notes/node3.html
• http://www.cs.ucl.ac.uk/staff/s.bhatti/d51-notes/node30.html
• http://wol.ra.phi.cam.ac.uk/mackay/itprnn/1997/11/node7.html