**Buletin Teknik Elektro dan Informatika**
(Bulletin of Electrical Engineering and Informatics)
Vol.1, No.1, March 2012, pp. 29~36
ISSN: 2089-3191


# Design and Implementation of Error Correcting Codes for Transmission in Binary Symmetric Channel

**Victor N. Papilaya**
Electronic and Computer Engineering Faculty, Satya Wacana Christian University
Jl. Diponegoro 52-60, Salatiga 50711 Jawa Tengah, Indonesia
e-mail: victor.papilaya@staff.uksw.edu

***Abstrak***

*Error-correcting codes (ECC) biasanya digunakan untuk melindungi informasi dari gangguan selama proses pengiriman di dalam media transmisi. Bit-bit Informasi dikodekan (encoding process) ke bentuk ECC, kemudian dikirimkan ke dalam media transmisi. Pada sisi penerima, kode yang diterima dikodekan kembali (decoding process) ke dalam bentuk bit-bit informasi. Paper ini memperkenalkan teknik membuat ECC berbentuk biner yang memenuhi Gilbert-bound dan juga mekanisme encoding dan decoding. Untuk menunjukkan unjuk kerja dari ECC, Binary Symmetric Channel (BSC) digunakan sebagai media transmisi.*

***Kata Kunci:*** *binary error correcting codes, binnary symmetric channel, hard decoding, maximum likelihood decoding, maximum aposteriori decoding.*

***Abstract***

*Error Correcting codes are normally used for protecting transmitted information bits in a noisy channel. The information bits are encoded into error correcting codes which will be transmitted into the channel and on the receiver side, the received codes will be decoded back into the transmitted information bits. In this paper, a technique of generating binary error correcting codes that meet the Gilbert-bound and a simple encoding-decoding mechanism will be presented. To show the performance of the error correcting codes, a Binary Symmetric Channel is considered for transmission.*

***Keywords****: binary error correcting codes, binary symmetric channel, hard decoding, maximum likelihood decoding, maximum aposteriori decoding.*

## 1. Introduction

Error correcting codes (ECC) used for protecting information in a noisy channel. It has been introduced a long time ago by Hamming in 1950 [1]. In his work, Hamming showed that ECC can be used for correcting single error. After his work, there are many researches in the field of error correction, and in general the classical research problems [2] can be divided into three types: codebook's design, channel characterization, and decoder's design.

A codebook, or codes, contains a list of error correcting codes, called code words, has being used in transmissions. The purpose in the design process is to have the maximum cardinality |C| and to maximize the minimum distance $d_{min}$ between code words.

Since the codes will be sent in a channel then the channel characterization becomes one of the important research problems, two general problems to solve, in this research, are find the type of errors and calculate the capacity. The answer of these problems will help in designing codebooks and also decoders.

A good decoder is a decoder that brings minimum error rates in the decoding process and also has minimum decoding complexity. To achieve the minimum error rates, designers must create good decisions mechanism which are based on the behavior of the noises.

This paper will focus on two problems, codebook's design and decoder's design. In the codebook's design, the Gilbert-bound [3] on the cardinality will be considered and the technique used for generating ECC is based on [2]. There are other bounds for binary ECC (see [4][5])

*Received September 17, 2011; Revised January 23, 2011; Accepted February 26, 2012*

but Gilbert-bound is considered because the steps used for generating codes that meet this bound is simple. The decoder's design is based on the hard decision mechanism using maximum likelihood (ML) probability and the maximum aposteriori (MAP) probability. To the best of the author's knowledge, there is only one publication [7] related to comparison between ML and MAP in relation to ECC. In this thesis, Yeong considered convolution codes, instead of block codes as considered in this paper. Furthermore, Yeong just considered one type of convolution codes (CC), CC of rate ½ with the constraint length 3, and used additive Gaussian white noise (AWGN) as the transmission channel. In this paper, the Binary Symmetric Channel (BSC) [6] will be used for the transmission channel. Table 1 shows the differences between Yeong's Thesis and this paper:

Table 1 Comparison between Yeong's thesis and this paper.

| Paper | Channel | ECC | Investigated Problems |
|---|---|---|---|
| Yeong's thesis | AWGNC | Convolution codes | Decoder's design: the performance between MAP and ML decoding |
| This paper | BSC | Binary block codes | 1. Codebook's design:<br>• The steps needed for generating codes with certain minimum distance and obey Gilbert-bound.<br>• The performance of codes when minimum distance is changed.<br>The performance of codes when rate is changed.<br>2. Decoder's design: The performers between MAP and ML decoding |

## 2. Research Method

This paper is presented in purpose that it can give a clear introduction to ECC and also to prove the theoretical information that MAP is better than ML, because it considers the probability of codewords being sent. For this reason, in this section we will discuss some background informations. And after that we will design and do the simulation scenarios for showing the influence of important parameters of ECC, and the performance of both decoding mechanisms, ML and MAP.

### 2.1. The Parameters of a Codebook

There are three important parameters of a codebook. They are cardinality |C|, minimum distance $d_{min}$ and rate R [2]. The number of code words in a codebook is called cardinality and the smallest distance. Hamming distance, between code words is called minimum distance. Knowing the |C|, one could determine the length of information bits $k$ which is equal to $\log_2(|C|)$. The error detection and correction capability of codes is determined by $d_{min}$ where the detectable errors = $d_{min}$-1 and the correctable errors = ($d_{min}$-1) / 2. The rate R is equal to k/n where $n$ is the length of a code word and n ≥ k. The basic idea behind ECC is to add redundancy bits into the information bits and indicated by n-k.

As an example, given a Codebook C = [0000, 0011, 1100, 1111]. The codebook contains four code words of length 4 which are different in at least two positions. This means |C|=4, $d_{min}$= 2, k=2, n=4, R=1/2, and redundancy bits = 2.

### 2.2. The Gilbert-bound

The Gilbert-bound determines the lower bound M on the cardinality. This bound takes into the consideration the length n of a code word and the minimum distance $d_{min}$ of a codebook.

$$M \geq \left\lceil \frac{2^{n}}{\sum_{i=0}^{d_{min}-1} \binom{n}{i}} \right\rceil \quad (1)$$

From the equation (1) above, if n =4 and $d_{min}$ = 2 then M ≥ 4.

### 2.3. Generating Binary Error Correcting Codes

The steps [2] of generating binary ECC that meet the Gilbert-bound are:

1. Start: Select code word from $2^n$ possible words.
2. Remove all words at distance < $d_{min}$ from selected code word
3. Select one of the remaining as next code word
4. Go to 2, unless no possibilities left.

As an example, suppose we are going to generate a codebook with $d_{min}$ = 2 and n = 4. This configuration gives us $2^4$ possible words as follows:

| | | | |
|---|---|---|---|
| 0000 | 0100 | 1000 | 1100 |
| 0001 | 0101 | 1001 | 1101 |
| 0010 | 0110 | 1010 | 1110 |
| 0011 | 0111 | 1011 | 1111 |

Suppose we choose zeros code word 0000 as an initial code word then we have to delete other words which differ at one position from zeros code word,

| | | | |
|---|---|---|---|
| **0000** | ~~0100~~ | ~~1000~~ | 1100 |
| ~~0001~~ | 0101 | 1001 | 1101 |
| ~~0010~~ | 0110 | 1010 | 1110 |
| 0011 | 0111 | 1011 | 1111 |

Since the number of remaining words is still larger than the lower bound and the minimum distance is still 1 (because the word 1110 and 1111 are differ in one position), we have to choose the next code word to be considered. Assume we select the word 1111 then the other words that must be deleted are the words which are differ in 1 position from the selected word.

| | | | |
|---|---|---|---|
| **0000** | ~~0100~~ | ~~1000~~ | 1100 |
| ~~0001~~ | 0101 | 1001 | ~~1101~~ |
| ~~0010~~ | 0110 | 1010 | ~~1110~~ |
| 0011 | ~~0111~~ | ~~1011~~ | **1111** |

Until this step, the remaining words have already minimum distance of 2. This means there is no need to continue the steps. Our final Codebook C is [0000, 0011, 0101, 0110, 1001, 1010, 1100, 1111].

### 2.4. Maximum Likelihood Decoding and Maximum Aposteriori Decoding

Given a codebook which contains code words $\mathbf{C_i}$ where 1 ≤ i ≤ |C|. These code words are sent into a noisy channel. And on the receiver side, we receive the code words that contain errors. Assume we received a word **y** then the question is what the most probable code word $\mathbf{C_i}$ that has been sent is. This is a basic question on the decoding process and can be solved using the join probability [8] between $\mathbf{C_i}$ and **y**. This probability can be written as follows:

$$P(\mathbf{C_i},\mathbf{y}) = P(\mathbf{y}) * P(\mathbf{C_i}|\mathbf{y}) \quad (2)$$

$$P(\mathbf{C_i},\mathbf{y}) = P(\mathbf{C_i}) * P(\mathbf{y}|\mathbf{C_i}) \quad (3)$$

The idea behind the Maximum Likelihood (ML) decoding is based on the equation (2) and the Maximum Aposteriori (MAP) decoding is based on the equation (3). As can be seen, ML does not take into account the probability of a code word but MAP does. Since the decoding process can only be done after receiving the word **y** then P(**y**) = 1. If we assume that all code words have the same probability to be sent then P($\mathbf{C_i}$) can be just considered as a constant. This means we can omit the P($\mathbf{C_i}$) in the decoding process and this also means that

the MAP decoding will be the same as the ML one. In case of code words have different probability and then MAP uses this information in the decoding process.

### 2.5. Binary Symmetric Channel

The following Figure 1 shows the construction of a Binary Symmetric Channel (BSC).

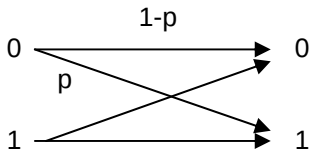


Figure 1. Binary Symmetric Channel

The input and output of the BSC are bits. The type of noise introduced by this channel is additives noise which means the noise will be added into our transmitted bits. This noise can also be seen as a "flip" noise, because if the input is 0 and the error probability p takes place then the output will be 1, or vice versa.

The following Figure 2 shows the clear situation about how the noises are added into our transmitted bits. **E** is the binary error sequence such that P(1)=1-P(0) = p. **X** is the binary information sequence and **Y** is the binary output sequence. As an example, given the transmitted bits **X** = [0101 0101 0101] and the noise bits **E** = [1111 0000 1111] then the received bits **Y** can be calculated as follows:

X = 0 1 0 1 0 1 0 1 0 1 0 1
E = 1 1 1 1 0 0 0 0 1 1 1 1
XOR
Y = 1 0 1 0 0 1 0 1 1 0 1 0.

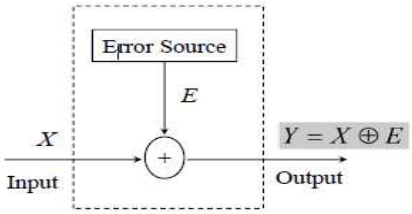


Figure 2. Communication model of additive noise channel.

### 2.6 Encoding and Decoding Mechanism

An encoding process is a process of mapping our information bits into code words. As an example, given a codebook *C* of distance 4 which contains two code words of length 4, C = [0000, 1111]. In this case, we have information bits of length k = 1 which simply means that we could send just 0 or 1. Since the |C| and the number of information bits are equal then we can do one-to-one mapping. One possible mapping will be 0 → 0000 and 1 → 1111. Table 2 shows the example of encoding process.

Table 2. Encoding process.

| Information bits | Encoding's output |
|---|---|
| 1010 | 1111000011110000 |

A decoding process is the reverse process of the encoding one. The idea behind the ML decoding is to choose the code word which has less number differences from the received one.

Table 3 ML decoding process.

| Sent | Received | Number of difference | | Decoding's output |
|---|---|---|---|---|
| | | $C_1$ = 0000 | $C_2$ = 1111 | |
| 0000 | 0000 | 0 | 4 | 0000 |
| 0000 | 0001 | 1 | 3 | 0000 |
| 1111 | 1111 | 4 | 0 | 1111 |
| 1111 | 0111 | 3 | 1 | 1111 |
| 1111 | 0011 | 2 | 2 | 0000/1111* |

* confusion

As can be seen above, if there are more than one code words that give the same least number of difference then the confusion takes place. In this situation, the decoder's output is unknown and can be chosen arbitrarily.

In the MAP decoding, the number of confusions can be reduced because the probability of every code word is taken into account. Consider the following example, given P(0000) = 1/3, P(1111)=2/3 and the error probability p = 0.1 then the Table 4 shows the decoding's result.

Table 4. MAP decoding process.

| Sent word, x | Received word, y | Join Probability (Ci, y) | | Decoding's output |
|---|---|---|---|---|
| | | C1 = 0000 | C2 = 1111 | |
| 0000 | 0000 | 0.2187 | ≈ 0 | 0000 |
| 0000 | 0001 | 0.0243 | 0.0006 | 0000 |
| 1111 | 1111 | ≈ 0 | 0.4374 | 1111 |
| 1111 | 0111 | 0.0003 | 0.0486 | 1111 |
| 1111 | 0011 | 0.0027 | 0.0054 | 1111 |

It is interesting to see how the decoder can make decision in the confusing situation using the join probability as explained above. We will take the data from the last row of the Table 4 as an example. The MAP is written as follows: $P(\mathbf{C_i},\mathbf{y}) = P(\mathbf{C_i}) * P(\mathbf{y}|\mathbf{C_i}) = P(\mathbf{C_i}) * p^e * (1-p)^{n-e}$, where *e* is the number of difference between received word **y** and $P(\mathbf{C_i})$, and *n* is the length of a code word. The join probability between the first code word and the received word is: $P(0000, 0011) = P(0000) * P(0011|0000) = 1/3 * 0.1^2 * (1-0.1)^2 = 0.0027$, and the join probability between the second code word and the received word is: $P(1111, 0011) = P(1111) * P(0011|1111) = 2/3 * 0.1^2 * (1-0.1)^2 = 0.0054$. From this calculation, one can see that the join probability between the received code and the second code word give the highest value. For this reason, the second code word is chosen as the decoding's output.

**2.7 Simulations**

In the simulation, we will show the performance of ECC in terms of the bit error rate (BER) as a function of the bit error probability *p*, the transition probability of a BSC. The BER is calculated as follows:

$$\text{BER} = \frac{\text{number of incorrect bits}}{\text{number of bits being sent}} \quad (4)$$

**2.7.1 Simulation's Steps**

1. Generate information bits → **U**
2. Encode information bits → **X** = encode(**U**). To simulate the performance of the ML decoding, the probability of every code word $P(C_i)$ is set to be equal, 1/|C|, and to simulate the MAP decoding, all $P(C_i)$ are different.

   $P(C_1) \neq P(C_2) \neq \ldots \neq P(C_{|c|})$, where $P(C_1) + P(C_2) + \ldots + P(C_{|c|}) = 1$

This step is not done for the uncoded transmission.

3. Send **X** into the channel to produce received bits **R** = **X** + **E**, where **E** is the vector of noise bits. The bits in **E** are generated with following probability: P(1) = 1-P(0) = p. **X** = **U** in the uncoded transmission.
4. Decode received bits → **U' =** decode(**R**). Since this step is not done for the uncoded transmission then in this situation **U'** = **U** + **E**.
5. Calculate the bit error rate → BER = |**U'** \ **U**| / |**U**|

### 2.7.2 Codes' Configuration

The following Table 5 shows the codebooks used in the simulation.

Table 5. Codebooks used in the simulation

| Codebook's ID | Codebook | Codebook's Parameters |
|---|---|---|
| 1 | [0000], [0011], [1100], [1111] | $d_{min}$ = 2, k=2, n=4, R=1/2 |
| 2 | [0000], [0011], [0101], [0110], [1001], [1010], [1100], [1111] | $d_{min}$ = 2, k=3, n=4, R=3/4 |
| 3 | [00000], [00111], [11001],[11110] | $d_{min}$ = 3, k=2, n=5, R=2/5 |
| 4 | [00000], [11111] | $d_{min}$ = 5, k=1, n=4, R=1/5 |

## 3. Results and Analysis

The simulations have been divided into three important points. The first one is to see the influence of the inputs' probabilities on the decoding's performance. The codebook 1 was used. The result shows (see Figure 3.) that the performance of the ML decoding when inputs' probabilities are equal is better than it when inputs are not equiprobable. In the equiprobable inputs, ML's performance is equal to the MAP one. These results show the agreement with the background theory about MAP and ML (see section 2.4).

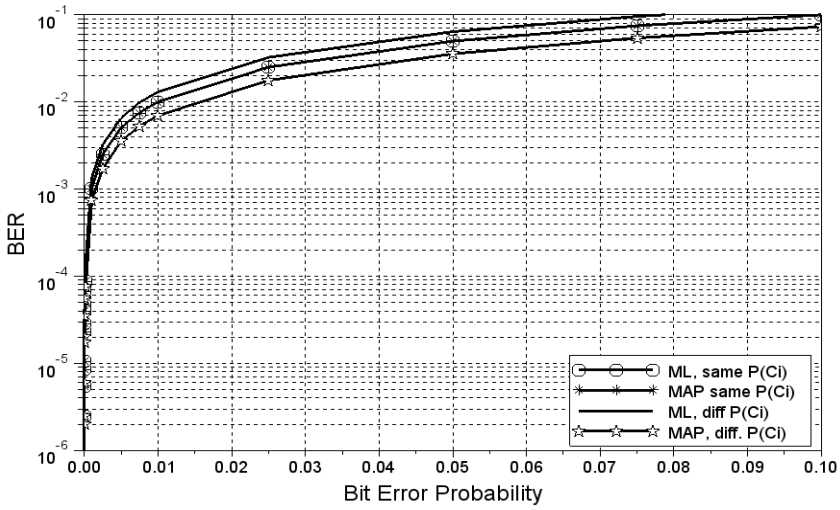


Figure 3. Comparision between MAP and ML decoding.

The second point is to see the influence of the rate R or redundancy. In this simulation, the codebook 1 and 2 were used. As can be seen on the Figure 4, when R gets higher then the performance of the decoding is worst. This is due to the fact that if error takes places then there are many conflicted code words in the codebook 2 compared to the codebook 1. As an example, the word [1111] is sent and the word [0111] is received then there two conflicts code words in the codebook 1: [1111] and [0011], because these two code words are differ in one position from the received one. This means the probability of doing correct decoding is 1/2. On the other hand, in the codebook 2 there are four conflicted code words: [1111], [0011], [0110], and [0101]. In this case, the probability of doing correct decoding gets lower into 1/4.

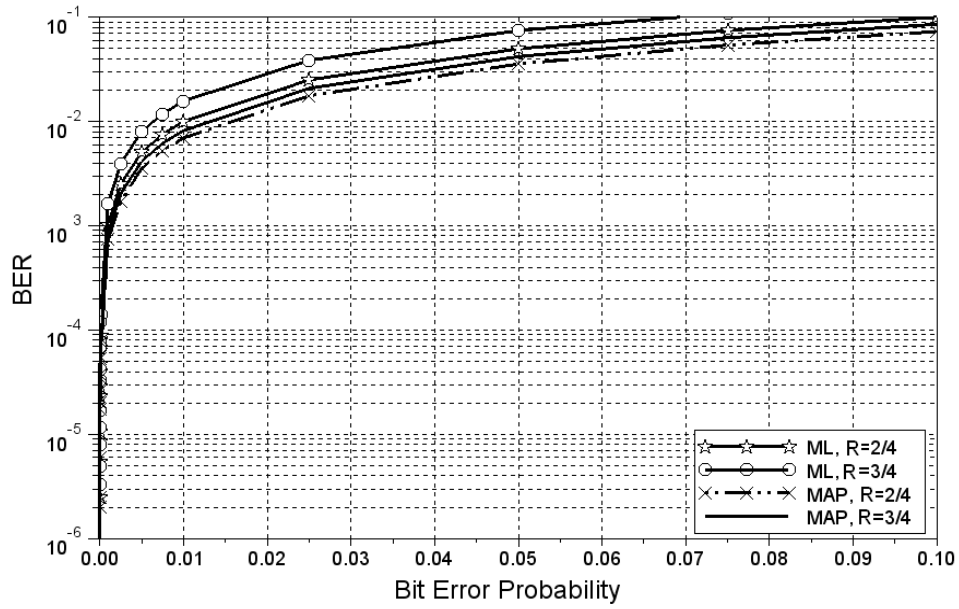


Figure 4. The performance of codes when R is changed.

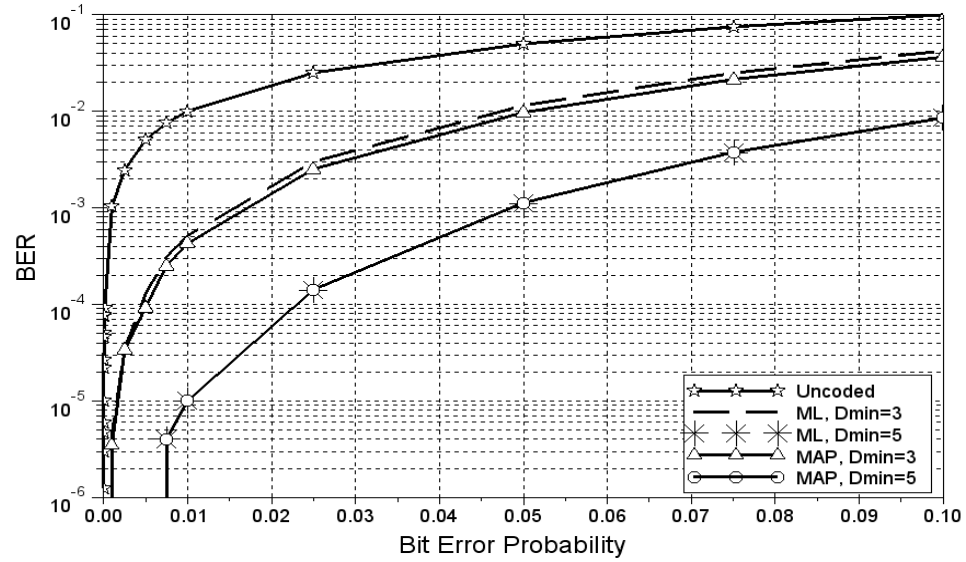


Figure 5. The performance of codes when $d_{min}$ is changed.

The last Figure 5 shows the result of the third important point in the simulation which is about the minimum distance $d_{min}$ and the comparison between uncoded and coded transmission. The codebook 3 and 4 were used for this simulation. From the result, one can see that coded transmission is better than uncoded one. This is because codes can correct errors. Codes with $d_{min}$=3 can correct 1 error and codes with $d_{min}$=5 can correct 2 errors. It can be seen clearly that the performance of codes with $d_{min}$=5 is far better than one with $d_{min}$=3. It is also interesting to see that for the higher distance, $d_{min}$=3, the ML and MAP decoding give slightly different, and even no difference for $d_{min}$=5. This is because in average the maximum number of error in a code of length n=5 is 1 when the maximum bit error probability is 0.1 and it does not exceed the number of error that can be corrected.

## 4. Conclusion

This paper has shown that the binary error correcting codes can protect the information bits during the transmission in a noisy channel. It has also shown the steps for generating binary ECC with certain minimum distance $d_{min}$ from binary sequences. Based on the simulation's result, one could see that the larger minimum distance is the better performance of the codes. Besides, the $d_{min}$, the performance of the codes also depends on the redundancy. Furthermore, it can be seen that emperical data, such as the probability of a code word being sent, can be used for improving decoding's result. MAP decoding uses this kind of information, as a result it is better than ML one.

## References

[1] Hamming R W. Error Detecting and Error Correcting Codes. *Bell Sys. Tech. J*. 1950; 29(2).
[2] Vinck H. A. J. *Error Correction.* The lecture notes on channel coding. Duisburg-Essen University. 2011.
[3] Gilbert E N. A comparison of signaling Alphabets. *Bell Sys. Tech. J*. 1952; 504-522.
[4] Varshamov R R. Estimate of the number of signals in error-correcting codes. Dokl. Akad. Nauk SSSR. 1957; 739 – 741.
[5] Johnson S M. A New Upper Bound for Error-Correcting Codes. IRE Transactions on Information Theory. 1962; pp 203-207.
[6] Cover T M, Thomas J A. Elements of Information Theory. New York: John Wiley & Sons. 1991: Ch.8.
[7] Yeong C N. The Comparison of Maximum-Likelihood and MAP Decoding Techniques. Bachelor Thesis. The University of Queensland; 2002.
[8] Chen Y. *Introduction to Probability Theory*. The lecture notes on information theory. Duisburg-Essen University.