

INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

UMI[®]



Modified Algorithms for the Decoding of Hamming Product Codes

BY

ALI AHMAD SALEH AL-SHAIKHI

A Thesis Presented to the
DEANSHIP OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

In

ELECTRICAL ENGINEERING

MAY 2001

UMI Number: 1404198

UMI[®]

UMI Microform 1404198

Copyright 2001 by Bell & Howell Information and Learning Company.

All rights reserved. This microform edition is protected against
unauthorized copying under Title 17, United States Code.

Bell & Howell Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346

KING FAHD UNIVERSITY OF PETROLEUM AND MINERALS

DHAHRAN, SAUDIA ARABIA

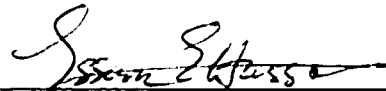
DEANSHIP OF GRADUATE STUDIES

This thesis, written by **Ali Ahmad Saleh Al-Shaikhi** under the direction of his thesis advisor and approved by his thesis committee, has been presented to and accepted by the Dean of Graduate Studies, in partial fulfillment of the requirements for the degree of **MASTER OF SCIENCE IN ELECTRICAL ENGINEERING.**

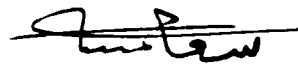
Thesis Committee



Dr. Maan A. Kousa, **Chairman**




Dr. Essam E. Hassan, **Member**



Dr. Saud A. Al-Semari, **Member**



Dr. Samir A. Al-Baiyat
Chairman, Electrical Engineering Department



Prof. Osama A. Jannadi
Dean, Graduate studies

June 2, 2001
Date



Dedication

All praises goes to Allah

*My Lord,
Accept it and make it with the good deeds*

The last prayer is AlHamduLillahi Rabbi Ala'lameen

Acknowledgements

I praise and thank Almighty Allah, the Most Compassionate, and the Most Merciful. He blesses me with his ever-enduring mercies. May his peace and blessing be upon Prophet Muhammad, and his family.

I acknowledge the support and facilities provided by the Electrical Engineering Department at King Fahd University of Petroleum and Minerals.

I would like to express my deep thanks and appreciation to my thesis advisor, Dr. Maan Kousa. He is really an exceptional person in all advising aspects and he is a pioneer in his field. His clever points, ideas, and suggestions led to such a good and precise work. I also remain grateful to the other thesis committee members, Dr. Saud Al-Semari and Dr. Essam Hassan for their patience, helpful suggestions, and kind cooperation.

I also want to thank all my relatives and friends for their prayers, caring, and understanding. Special thank to Mr. Ali Mugaibel for his advice and encouragement. I would like to acknowledge the big support of my father, the prayers of my mother, and the understanding of my brothers and sisters. Last but not least, I would like to thank my wife, Halimah, a loving and considerate partner. Without her support, prayers, and encouragement, this work could not be accomplished. All what they did made this work to appear to light.

Contents

Acknowledgement.....	i
Abstract (English).....	ix
Abstract (Arabic).....	x
1 INTRODUCTION.....	1
1.1 GENERAL OVERVIEW.....	1
1.2 NOISE.....	3
1.2.1 <i>AWGN</i>	3
1.2.2 <i>Fading</i>	3
1.3 CHANNEL.....	6
1.4 MITIGATION TECHNIQUES.....	7
1.5 ERROR CONTROL CODING.....	7
1.5.1 <i>Block Codes</i>	9
1.5.2 <i>Hamming Codes</i>	10
1.5.3 <i>Interleaving</i>	11

2	PRODUCT CODES	12
2.1	GENERAL OVERVIEW	12
2.2	PRODUCT CODES.....	14
2.2.1	<i>Introduction</i>	14
2.2.2	<i>Product Encoder</i>	14
2.2.3	<i>Product Decoder</i>	15
2.3	ERROR CORRECTING CAPABILITY	17
2.3.1	<i>Random Errors</i>	17
2.3.2	<i>Burst Errors</i>	17
2.3.3	<i>Random and Burst Errors</i>	18
2.4	ERROR CORRECTING ALGORITHMS	19
2.4.1	<i>Cascaded Hamming Codes</i>	20
2.4.2	<i>Ordered Decoding</i>	21
2.4.3	<i>Product Codes and Interleaved Block Codes</i>	23
2.4.4	<i>Near Optimum Decoding</i>	23
2.4.5	<i>Decoding of n-Dimensional Product Codes</i>	24
2.4.6	<i>Randomly Interleaved SPC Product Codes</i>	25
2.4.7	<i>A Multidimensional Block Coding Scheme with Iterative Decoding</i>	26
2.4.8	<i>Soft Decision Decoding</i>	27
2.4.9	<i>Erasure Decoding</i>	28
2.4.10	<i>Other Researches</i>	30
2.5	THESIS CONTRIBUTION	30
2.5.1	<i>Thesis Organization</i>	31

3	HARD DECISION DECODING OF PRODUCT CODES.....	33
3.1	GENERAL OVERVIEW	33
3.2	HARD DECISION DECODING OF PRODUCT CODES	34
3.3	MHDD OF PRODUCT CODES.....	37
3.4	EXTENSION TO THREE DIMENSIONS.....	48
3.5	PERFORMANCE CRITERIA.....	57
3.6	PERFORMANCE COMPARISON.....	60
4	SOFT DECISION AND ERASURE DECODINGS OF PRODUCT CODES.....	68
4.1	GENERAL OVERVIEW	68
4.2	SOFT DECISION DECODING	69
4.3	ERASURE DECODING.....	76
4.4	PERFORMANCE AND COMPLEXITY COMPARISON.....	85
5	SUMMARY, CONCLUSIONS, AND SUGGESTIONS FOR FUTURE WORK.....	94
5.1	INTRODUCTION	94
5.2	SUMMARY AND CONCLUSIONS	95
5.3	SUGGESTIONS FOR FUTURE WORK	97
	Nomenclature.....	98
	Bibliography	102

List of Tables

3.1	Comparison Between OHDD and MHDD of (49,16) Product Code	43
4.1	Timing Ratios of the Algorithms	93

List of Figures

1.1	General Communication System	2
2.1	Product Code Array	16
2.2	Diagonal Interleaver of the (49,16) Product Code.....	22
3.1	The OHDD Algorithm.....	35
3.2	Uncorrectable Four-Error Pattern	36
3.3	The Simplified MHDD Algorithm	39
3.4	Uncorrectable Four-Error Patterns.....	42
3.5	Hard Decision Decoder Performance over AWGN Channel.....	45
3.6	Hard Decision Decoder Performance over Fading Channel of 0.1 Fade Rate	46
3.7	Hard Decision Decoder Performance over Fading Channel of 0.001 Fade Rate	47
3.8	The Extended MHDD Algorithm	51
3.9	Hard Decision Decoder Performance of Three Dimensional (343,64) Product Code over AWGN Channel	56
3.10	Ordinary Hard Decision Decoder Performance of (49,16) Product Code over AWGN Channel	58

3.11	Modified Hard Decision Decoder Performance of (49,16) Product Code over AWGN Channel	59
3.12	Performance Comparison of Hard Decision Decoder of (49,16) Product Code over AWGN Channel.....	61
3.13	Performance Comparison of Hard Decision Decoder of (225,121) Product Code over AWGN Channel.....	62
3.14	Performance Comparison of Hard Decision Decoder of (49,16) Product Code over Fading Channel of 0.1 Fade Rate	63
3.15	Performance Comparison of Hard Decision Decoder of (225,121) Product Code over Fading Channel of 0.1 Fade Rate	64
3.16	Performance Comparison of Hard Decision Decoder of (49,16) Product Code over Fading Channel of 0.001 Fade Rate	65
3.17	Performance Comparison of Hard Decision Decoder of (225,121) Product Code over Fading Channel of 0.001 Fade Rate	66
3.18	Different Rounding Decoding of (49,16) Product Code Using Dimension by Dimension Decoding over AWGN Channel.....	67
4.1	The OSDD Algorithm	70
4.2	The MSDD Algorithm.....	71
4.3	Soft Decision Decoder Performance over AWGN Channel	73
4.4	Soft Decision Decoder Performance over Fading Channel of 0.1 Fade Rate.....	74
4.5	Soft Decision Decoder Performance over Fading Channel of 0.001 Fade Rate.....	75

4.6	The OED Algorithm	79
4.7	The MED Algorithm.....	80
4.8	Erasure Decoder Performance over AWGN Channel	82
4.9	Erasure Decoder Performance over Fading Channel of 0.1 Fade Rate.....	83
4.10	Erasure Decoder Performance over Fading Channel of 0.001 Fade Rate.....	84
4.11	Comparison of Different Decoding Algorithms of (49,16) Product Code over AWGN Channel	86
4.12	Comparison of Different Decoding Algorithms of (225,121) Product Code over AWGN Channel	87
4.13	Comparison of Different Decoding Algorithms of (49,16) Product Code over Fading Channel of 0.1 Fade Rate	88
4.14	Comparison of Different Decoding Algorithms of (225,121) Product Code over Fading Channel of 0.1 Fade Rate	89
4.15	Comparison of Different Decoding Algorithms of (49,16) Product Code over Fading Channel of 0.001 Fade Rate	90
4.16	Comparison of Different Decoding Algorithms of (225,121) Product Code over Fading Channel of 0.001 Fade Rate	91

THESIS ABSTRACT

Name: Al-Shaikhi, Ali Ahmad Saleh
Title: Modified Algorithms for the Decoding of Hamming Product Codes
Degree: Master of Science
Major Field: Electrical Engineering
Date of Degree: May, 2001

Product codes attracted many researchers resulting in an explosive amount of literature. This is because product codes deal with the concept of constructing long powerful codes without increasing the complexity of the associated decoding, by employing shorter component codes. Also, product codes are effective in recovering both random and burst errors.

Most proposed decoding algorithms of product code either do not exploit the full capability of the code or achieve high correction capability at the cost of increased complexity and delay. The objective of this thesis is to improve the performance of the product codes without a significant increase in decoding complexity. Specifically, the thesis introduces three modified algorithms. The modified hard decision decoder, MHDD, algorithm develops a decoding algorithm that is capable of reaching the theoretical error correction capability of the product code without a significant increase in the decoding complexity. The extension of the MHDD to three dimensions is presented. The extended MHDD algorithm is also capable of reaching the theoretical error correction capability. The modified soft decision decoder, MSDD, algorithm investigates the performance of soft decision decoding for binary symbols to improve the performance by taking advantage of soft decoding in both dimensions. The modified erasure decoder, MED, algorithm investigates erasure decoding for binary symbols to improve the performance by erasing specific values instead of erasing whole rows or columns.

**KING FAHD UNIVERSITY OF PETROLEUM AND MINERALS,
DHAHRAN, SAUDI ARABIA**

خلاصة الرسالة

الأسم : علي أحمد صالح الشبخي
العنوان : خوارزمات معدلة لتفسير الترميزات الضريبية الخاصة بهامنج
التخصص : الهندسة الكهربائية
التاريخ : مايو ٢٠٠١ م

لقد جذبت الترميزات الضريبية كثيراً من الباحثين ، فنتج عن ذلك كم هائل من الأبحاث ، وذلك لأن الترميزات الضريبية تتعامل مع فكرة إنشاء ترميزات طويلة وقوية بدون زيادة في تعقيد تفسير هذه الترميزات ، وذلك بتوظيف ترميزات أقصر تؤلف الترميزات الضريبية . إن الترميزات الضريبية مفيدة جداً في التعامل مع الأخطاء العشوائية وكذلك الأخطاء المتصلة .

إن معظم الخوارزمات المعروضة الخاصة بالترميزات الضريبية ، إما أنها لا تستغل القدرة الكاملة للترميز أو أنها تنجز قدرة عالية لتصحيح الأخطاء على حساب الزيادة في التعقيد والتأخير لتفسير الترميز الضريبي . الهدف من هذا البحث تحسين أداء الترميزات الضريبية بدون زيادة كبيرة في تعقيد تفسير هذه الترميزات . يعرض هذا البحث تحديداً ثلاثة خوارزمات معدلة لتفسير الترميزات الضريبية . الخوارزم المعدل لمفسر القرار الحاسم (MHDD) يطور خوارزم مفسر قادر للوصول للإمكانية التصحيحية النظرية للأخطاء الواقعة في الترميزات الضريبية بدون زيادة كبيرة في تعقيد تفسير هذه الترميزات . إطالة هذا الخوارزم (MHDD) لثلاثة أبعاد يعرض أيضاً في هذا البحث . الخوارزم المطال لثلاثة أبعاد هو أيضاً قادر للوصول للإمكانية التصحيحية النظرية للأخطاء الواقعة في الترميزات الضريبية . الخوارزم المعدل لمفسر القرار غير الحاسم (MSDD) يحسن أداء المفسر غير الحاسم للرموز الثنائية بإستثمار المفسر غير الحاسم في كلا بعدي الترميز الضريبي . الخوارزم المعدل لمفسر القرار الماسح (MED) يحسن أداء مفسر القرار الماسح للرموز الثنائية بمسح قيم محددة بدلاً من مسح صفوف كاملة أو أعمدة كاملة في الترميز الضريبي .

درجة الماجستير في العلوم

جامعة الملك فهد للبترول والمعادن

الظهران - المملكة العربية السعودية

CHAPTER 1

INTRODUCTION

In this introductory chapter, we review the basic parts of a communication system. It will be observed that a channel code is an essential part of any modern communication system. A channel-encoding scheme, called Product Code, will be presented and some aspects of the code will be introduced.

1.1 General Overview

Consider Figure 1.1 that shows a general digital communication system. Channel coding is an integral part of the system. Channel coding refers to the class of signal transformations designed to improve communication performance by enabling the transmitted signals to better withstand the effects of various channel impairments, such as noise, fading, and jamming [1].

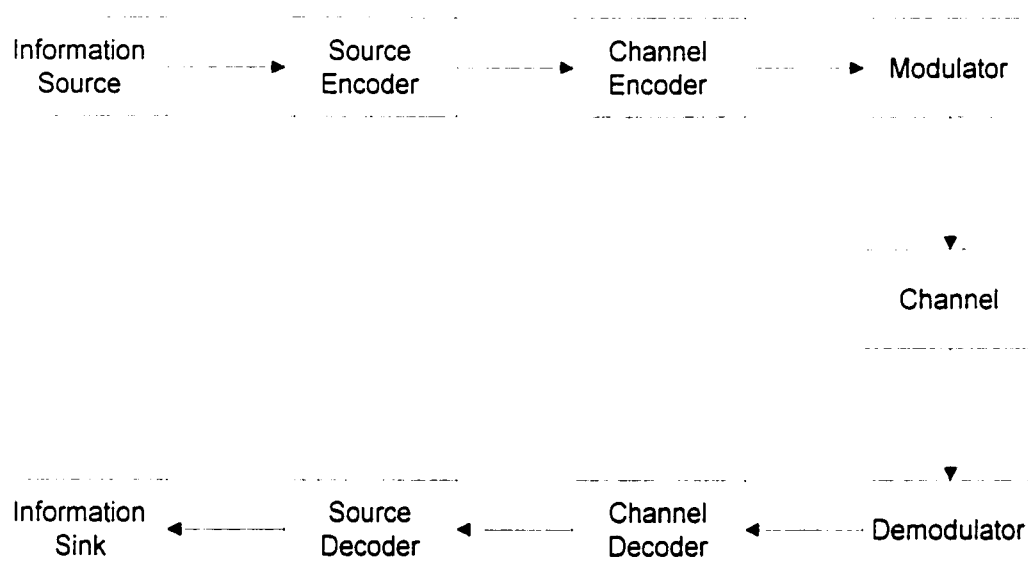


Figure 1.1 : General Communication System

To overcome such noise and interference and, thus, increase the reliability of the signals transmitted through the channel, it is often necessary to introduce some redundant bits. The redundant bits can be then used for the detection and/or correction of errors. On the next sections, different aspects of a communication system will be discussed. Those aspects are related to this thesis work, and are introduced to provide a better understanding of what is done in this work.

1.2 Noise

The signal-to-noise power ratio (*SNR*), defined below, is a good measure of performance at various points of a coding system. The *SNR* can degrade by the decrease of the signal power (signal loss) or the increase of the noise power (interference), however, the net effect on the *SNR* (E_b / N_0) is the same [1].

$$SNR = \frac{\text{signal power}}{\text{noise power}} \quad (1.1)$$

1.2.1 AWGN

The *additive white Gaussian noise* (AWGN) is completely characterized by its variance. The variance of AWGN is infinite, however, the variance of the filtered AWGN is finite and equals $N_0/2$, and where the factor 2 indicates that this is a two-sided power spectral density. So, it will be assumed that the noise of interest is the output noise of a correlator or a matched filter with variance $\sigma^2 = N_0/2$.

1.2.2 Fading

Any microwave radio channel like mobile channel faces the problem that the impulse response of the channel is, in general, time varying [2][3]. This is because of the

constantly changing characteristic of the channel due to the random motion of the ions in the ionospheric layers. So, two received signals transmitted through such channel will be received differently and this difference will be random rather than deterministic. Also, there may be a multiple of paths for a signal with different amplitudes and phases from the transmitter to the receiver. These relatively delayed signal components interfere with each other at the receiver causing fading. At times, the signals from different paths add up constructively and at other times they add up destructively. Thus, the amplitude variations in the received signal are due to the time-variant multi-path characteristics of the channel. These time-variant and multi-path propagation cause frequency dispersion and time dispersion, respectively, of the received signal leading to different types of channels.

Time dispersion

Time dispersion is characterized by the *root mean square (rms) delay spread*, τ_{rms} , and the *coherence bandwidth*, B_c . The *rms delay spread* is the square root of the second central moments of the power delay profile and it is a natural phenomenon caused by reflected and scattered propagation paths in the channel. However, the coherence bandwidth is a defined relation derived from the *rms* delay spread and it is the range of frequencies for which the channel passes all spectral components with equal gain and linear phase.

If the bandwidth, B_c , over which the channel has constant gain and linear phase is greater than the bandwidth of the transmitted signal, then the received signal will undergo a *frequency non-selective* or *flat fading*. This implies that the spectral characteristics of the

signal are preserved at the receiver while the strength of the received signal changes with time.

If the bandwidth, B_c , over which the channel has constant gain and linear phase is less than the bandwidth of the transmitted signal, then the received signal will undergo a *frequency selective fading*. This implies that the received signal includes multiple versions of the transmitted signal, which are attenuated and delayed in time, and hence the received signal is distorted. This delay (dispersion) in time smears symbols into adjacent ones causing *intersymbol interference* (ISI). The delay in time, viewing it in frequency, means that different frequency components of the received signal have different gains and hence the channel is frequency selective.

Frequency dispersion

Frequency dispersion describes the time varying nature of the channel caused by the relative motion between the mobile and the base stations or by the movement of objects in the channel. Frequency dispersion is characterized by *Doppler spread*, f_D , and *coherence time*, T_c .

Doppler spread is a measure of the spectral broadening and it is the range of frequencies over which the received Doppler spectrum is essentially non zero. Coherence time is a statistical measure of the time duration over which the channel impulse response is essentially invariant.

If the coherence time is less than the symbol period of the transmitted signal, the channel is a *fast fading channel*. The impulse response of such channel changes rapidly within the symbol duration leading to signal distortion.

In a *slow fading channel*, the channel impulse response changes at a rate much slower than the transmitted base-band signal. It means that the Doppler spread of the channel is much less than the bandwidth of the base-band signal.

1.3 Channel

Work is tested by assuming the presentation of AWGN and/or fading. To do so, the bit error rate (*BER*) will be calculated against *SNR*. As mentioned earlier, the AWGN will be modeled as a filtered AWGN with variance $\sigma^2 = N_0/2$. For fading channel, it is known that the envelope of the sum of two quadrature Gaussian noise signals obeys a Rayleigh distribution [3]. So, the fade will be modeled as a Rayleigh fading. This model is widely used and it is a realistic one. Rayleigh fading model gives all the characteristics of a narrow band channel leading to the conclusion that the channel under consideration is a flat fading channel. This flat fading channel is simulated for different *fade rate*, where the *fade rate* is nothing but the normalized Doppler spread, $f_D T$ where T is the symbol duration. The *fade rate* will indicate how fast the channel fade is. Unless otherwise mentioned, two values of fade rate will be considered. Those are 0.1 and 0.001. Fading channel simulation is done using Jake's method at a mobile speed of 96 Km/hr and carrier frequency of 900 MHz. This method was tested against theoretical statistics, which are the probability density function and the autocorrelation function. Good agreement was found with theory [4].

1.4 Mitigation Techniques

To militate against such noise and fading introduced by the channel, there are many ways. Those include diversity, equalization, and channel coding. Equalization is applied when the channel introduces ISI, which is not considered here. Channel coding, which is considered in this thesis, incorporates extra information (redundancy) into the transmitted data, which can then be used to detect and/or correct errors. Diversity techniques are based on the concept of supplying the receiver with several replicas of the same information signal transmitted over independently fading channels. So, the probability that all signal components will fade simultaneously is reduced considerably. Diversity can be regarded as brute force use of redundancy, in which each symbol is repeated L times. From a coding vision, diversity involves the use of a simple repetition code of rate $1/L$. Selection of a code will lead to a more efficient system, while maintaining the benefits of the diversity concept. It has been proven that an error correcting code can provide an effective order of diversity equal to its minimum distance [5].

This work concentrates on channel coding with/without interleaver. Channel coding fulfils our need regarding the types of thermal noise and fading channel that were modeled. Next, brief introduction about error control coding and interleaver will be presented.

1.5 Error Control Coding

A major concern in data communication systems is how to control transmission errors caused the channel noise and/or fading so that reliable data can be delivered to the user.

Different data communication systems attempt to make use of coding differently in order to enhance their performance, based on one or more performance criteria. Basically, there are two fundamental techniques for error control, which are automatic repeat request (ARQ) and the forward error control (FEC) [1]. The former employs pure error detection in which an acknowledgement type of protocol is used with requests for repetitions of unaccepted messages and the latter employs pure error correction. FEC is considered.

FEC could be block codes or convolutional codes. Two or more codes can be combined to produce a more powerful error control code called layered code. Layering [6] can be done in parallel or serially. Parallel layering is done for convolutional codes to form what is called turbo codes. However, serial layering is found in block codes.

There are two kinds of serially layered block codes, which are product codes and concatenated codes [7]. The goal is to form long powerful codes, without the complexity of the associated decoding, by employing shorter component codes. Concatenated codes are serially layered codes in which a code followed by another. They use two levels of coding, an inner code and an outer code, to achieve the desired error performance. The inner code is usually to correct most of the channel errors and the outer code, higher rate code, reduces the probability of error to the specified level. Usually, there is an interleaver between the two coding steps to spread any error bursts that may appear at the output of the inner coding operation. Product codes, which will be discussed on the next chapter, are serially layered codes [8] but layering is done of one code on top of the other one. The primary difference in concatenated codes is that the two component codes are defined over different field sizes [6]. Here, block codes are considered.

1.5.1 Block Codes

The input to the encoder is assumed to be a sequence of bits occurring at a rate of R bits/s. In block encoding, blocks of k information bits are encoded into blocks of n bits ($n > k$). Each block of n bits from the encoder constitutes a codeword contained in a set of $M = 2^k$ possible codewords. The code rate, defined as the ratio k/n and denoted by R_c , is a measure of the amount of redundancy introduced by the encoder.

The binary digits from the encoder are fed into a modulator, which maps each bit into an elementary signal waveform. Binary phase shift keying (BPSK) is considered in this work. The channel corrupts the transmitted signal by AWGN and/or Rayleigh fading. The resulting received signal is processed first by the demodulator and then by the decoder.

Remember that, ε , the channel bit error probability is related to E_b/N_0 , bit energy per noise spectral density, for BPSK by the following

$$\varepsilon = Q\left(\sqrt{2 \frac{k}{n} \frac{E_b}{N_0}}\right) \quad (1.2)$$

where $Q(x)$ is the complementary error function [1].

The demodulator is a matched filter to the signal waveform corresponding to each transmitted signal [1]. The demodulator can be used to make firm decisions onto whether each decoded bit is a 0 or a 1, for two quantization levels. In this case the demodulator makes a hard decision on each bit. The detected bits from the demodulator are fed into the decoder to recover the information sequence. Since the decoder operates on the hard decisions made by the demodulator, the decoding process is termed hard decision decoding. However, the unquantized output, or more than two levels of quantization,

from the demodulator can be fed to the decoder. The decoder now uses the additional information contained in the unquantized samples to recover the information sequence with a higher reliability than that achieved with hard decisions. The resulting decoding process is termed soft decision decoding. This work employs the class of Hamming codes. So, some little details about this class will be discussed.

1.5.2 Hamming Codes

A binary Hamming code is an $[n=2^m - 1, k=2^m - 1 - m, d_{min}=3]$ code for $m > 2$. The columns of the parity check matrix H consists of all non-zero binary vectors of length m (m - tuple). As the minimum distance of these codes is three, they are single error correcting codes. Furthermore, they are perfect codes meaning that the number of correctable error patterns equals the number of distinct syndromes. Weight distribution of Hamming codes is known. Hamming codes are easily encoded and decoded. They can be made cyclic which are even simpler to encode and decode. In this work, the (7, 4, 3) and (15, 11, 3) Hamming codes will be used in a layered manner.

Coding theory has been based on the assumption that each symbol is affected independently by noise so that the probability of an error pattern depends only on the number of errors. This referred to as random errors. Codes are designed to correct any pattern of t errors or less in a block of n symbols. If more than t errors occur, the received codeword will be incorrectly decoded. For a random Binary Symmetric Channel, BSC, of crossover probability ε , where an (n, k, d_{min}) code is used, the probability of a received codeword to be in error is as follows

$$P_{block} = \sum_{i=t+1}^n \binom{n}{i} \varepsilon^i (1 - \varepsilon)^{n-i} \quad (1.3)$$

This is the probability [2] that a decoded codeword is wrong. This does not mean that each bit in this codeword is erroneous. A decoded codeword is wrong whenever at least one bit is wrong. Since some bits of the wrongly decoded codeword may still be correct, a useful quantity to calculate is the BER after decoding, which is used in this work.

1.5.3 Interleaving

Time diversity of channel codes in digital communication system can be improved by incorporating interleaving, without adding any overhead information. However, the system will incur interleaver and deinterleaver delay. Interleaving is an effective way to combat error bursts occurring on fading channels. It helps to randomize error burst and hence good codes that are designed for independent error channels can be utilized for burst error channels. Several types of interleavers have been proposed in the literature [3][9][10]. Convolutional, diagonal, inter block, and block interleavers are in existence. The block interleaver and diagonal interleaver are considered in this work. The interleaver is modeled as a buffer of J rows, referred to as the depth and K columns called the span. The channel symbols are written into the array row-wise and readout into the channel column-wise or the opposite. The size of the interleaver is JK . At the receiver, the deinterleaver performs the reverse operation. It has been shown [4] that near ideal interleaver performance is obtained if J and K are chosen as follows, where Δ is the allowable delay.

$$J \geq \frac{1}{4f_d T} \quad (1.4)$$

$$K = \frac{1}{T} \frac{\Delta}{J} \quad (1.5)$$

CHAPTER 2

PRODUCT CODES

2.1 General Overview

As mentioned in chapter 1, redundant symbols must be introduced for error detection and/or correction. To have error free symbols, the redundant symbols must be long, ideally infinite, meaning that the rate of the code must be very small, ideally zero. However, Shannon showed that if the rate of the source code is less than channel capacity, then it is possible to have codes such that the probability of erroneous decoding is arbitrarily small [2]. This is the channel-coding theorem, which is an existence proof of such codes, but it does not tell us how to construct such good codes.

Coding (for error-correction or detection) has been well accepted as a powerful technique for digital communication systems design over fading and non-fading channels. It is known from information theory that the performance of a code is proportional to its block

length, for block codes, or to its constraint length, for convolutional codes. However, increasing the code length results in an increased complexity at the decoder. Researchers have been trying to introduce long codes but yet possess enough structure to be decoded easily. Today, concatenated coding schemes are considered one of the best solutions for powerful protection against errors [7]. The motivation for using concatenated coding schemes is to achieve the same performance as that of a single and powerful error correcting code with lower decoding complexity by associating two (or more) less powerful error correcting codes for data coding. Practically, the number of codes in a concatenated code is limited to two codes. One of the most successful proposals is the introduction of product codes by Elias [7].

The introduction of product codes attracted many researchers resulting in an explosive amount of literature. This is because product codes deal with the concept of constructing long powerful codes without increasing the complexity of the associated decoding, by employing shorter component codes [11]. Also, product codes are effective in dealing with both random and burst errors. However, the correction capability of the product codes most of the time cannot be reached without complex decoding designs [11].

This chapter starts with introducing product codes. Both the encoder and the decoder are reviewed. An up-to-date literature survey is then presented on decoding algorithms for random and burst errors. The chapter ends with stating the thesis contribution and highlighting the thesis organization.

2.2 Product Codes

2.2.1 Introduction

Product codes, a kind of series concatenated coding schemes [7][8], were introduced as early as 1954 by Elias [12]. They were the first family of codes shown to asymptotically achieve error free performance with a nonzero code rate [12]. Product codes were not given much attention for a long time because they did not yield good performance. The deceiving performance of product codes was mainly due to the use of sub-optimal hard decoders for decoding the rows and the columns of the matrix. The authors in [8] proposed a near optimum iterative algorithm for soft decision decoding of product codes. It has been shown that for BCH product codes, a BER of 10^{-5} can be achieved with a signal to noise ratio of 2.5 ± 0.2 dB of Shannon's theoretical limits [8]. So, series-concatenated (product) and parallel-concatenated (turbo) coding schemes are now comparable in terms of performance. The dominating factor in the choice of one solution among these different candidates is the complexity of the decoders.

In the following sections the structure of both the encoder and the decoder are explained. The error-and-erasure, near optimum performance, and cascaded Hamming codes are briefed. Some other decoding designs are also highlighted.

2.2.2 Product Encoder

Errors occur independently at random and/or in bursts. So, the objective is to design codes capable of correcting those two types of errors. Interleaving is often implemented to handle burst errors. Actually, product codes are nothing but interleaved block codes because codewords are written into arrays row-wise and read out column-wise. Generally

speaking the rows of the array are generated from one code, c_1 , and the columns are generated from another code, c_2 . In effect, a product code combines two codes $c_1 (n_1, k_1, d_1)$ and $c_2 (n_2, k_2, d_2)$ where n_i is the total length of the code, k_i is the length of the information bits and d_i is the minimum distance of the code. The resultant code is the product code $c (n_1 n_2, k_1 k_2, d_1 d_2)$. Figure 2.1 shows the structure of the product code. To encode the product code using two codes $c_1 (n_1, k_1, d_1)$ and $c_2 (n_2, k_2, d_2)$, arrange the information bits in an array of $k_2 \times k_1$. Then, encode each of the k_2 rows using code c_1 . Next, encode each of the n_1 resultant columns using code c_2 . The resultant code is a product code $c (n_1 n_2, k_1 k_2, d_1 d_2)$. Alternatively, the same product code c can be obtained by first encoding all the k_1 columns using code c_2 , and then the resultant n_2 rows are encoded using code c_1 [11][13][14].

2.2.3 Product Decoder

Most of the research on product codes concentrates on decoding. Since product codes are generated from simple codes, they, in general, have low complexity decoding algorithms. However, as researchers are seeking better performance, the complexity of decoding increases.

Decoding of product codes can be done for random errors, burst errors, or both. It can as well be applied to erasures. Other new trends in decoding have been proposed. In the following subsections, we provide a literature survey of decoding techniques and algorithms. Before that we review the error correcting capability of product codes for random errors only, burst errors only, and mixed random & burst errors.

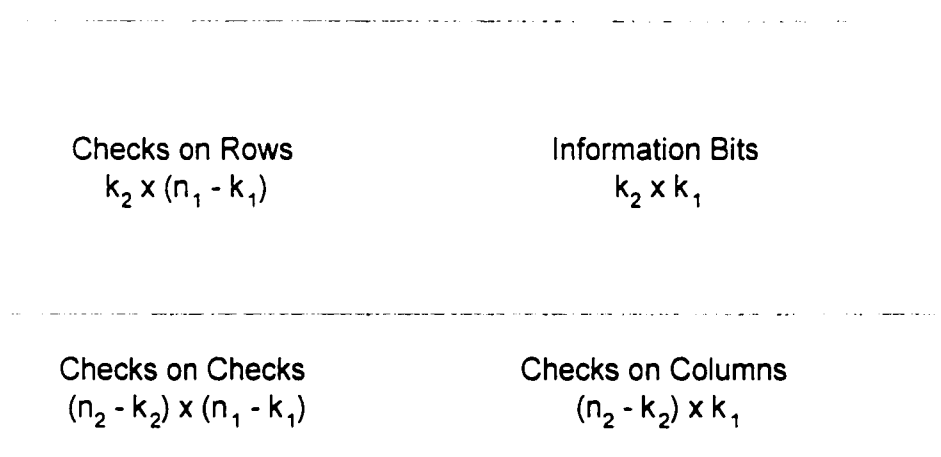


Figure 2.1 : Product Code Array

2.3 Error Correcting Capability

Errors can happen randomly, in bursts, and/or random & burst. This depends on the kind of channel under consideration. The correcting capability of the product codes for such errors are different.

2.3.1 Random Errors

The minimum distance of the product code is $(d_{min}=d_1d_2)$ [11]. This can be seen as follows [14], d_1 is the minimum distance of code c_1 , and d_2 is the minimum distance of code c_2 . Then, a code word in the product code must have at least d_1 nonzero elements in each row and at least d_2 nonzero elements in each nonzero column. Thus, the minimum distance of the product code is exactly d_1d_2 . Therefore, at least one such code word exists. So, the random error correction capability, t , of the product code is $\lfloor (d_1d_2-1)/2 \rfloor$ or fewer random errors.

2.3.2 Burst Errors

Product codes are also effective in fading channels, which result in burst errors. If b_1 and b_2 are the burst error correcting capabilities of code c_1 and code c_2 , respectively, then it can be shown that the burst error correction capability, b , of the product code is at least $\max(n_1b_2, n_2b_1)$ if burst errors are only considered [13]. This can be shown as follows [13][14]. Assume a code array is transmitted row-by-row and then rearranged back at the receiver into an array row-by-row. Any existing error burst of length n_1b_2 or less will affect no more than b_2+1 consecutive rows. So, at the receiver, each column is at most affected by a burst of length b_2 . Now, decoding the received code array on the basis of a column-by column will correct that length of burst errors. Therefore, the burst error

correction capability of the product code is at least n_1b_2 . On the other hand, assume a code array is transmitted on a column-by-column basis and decoded on a row-by-row basis. It can be shown that any error burst of length n_2b_1 or less can be corrected. Therefore, the burst error correction capability of the product code is at least n_2b_1 . Consequently, one may conclude that the burst error correction capability, b , of the product code is at least $\max(n_1b_2, n_2b_1)$ if burst errors are only considered.

Here, it is important to emphasize that the burst error correction capability, b , of the product code is at least $\max(n_1b_2, n_2b_1)$ if the decoding is done on the basis of column-by-column or row-by-row. If the decoding is done on the basis of column-by-row or row-by-column, then the burst error correction capability, b , of the product code is at least $\max(n_1b_2+b_1, n_2b_1+b_2)$, although this capability is only claimed for cyclic product code.

2.3.3 Random and Burst Errors

Product codes could be used for simultaneous random error correction and burst error correction [15][16]. It does not mean that one array can contain a full complement of both burst and random errors, however, it means that there can never be confusion between burst errors and random errors if both fall within the error correction capability of the product code. This can be shown as follows. Assume that d_1 is the minimum distance of code c_1 correcting t_1 random errors and d_2 is the minimum distance of code c_2 correcting t_2 random errors, then we have shown that the random error correction capability, t , of the product code is $\lfloor (d_1d_2-1)/2 \rfloor$ or fewer random errors. Simultaneously, the burst error correction capability, b , of the product code is at least $\max(n_1t_2, n_2t_1)$ or shorter burst errors. Now, if n_1t_2 is greater than or equals n_2t_1 then b will be n_1t_2 . Consider an error burst of $b = n_1t_2$ or less. When this error burst is arranged in an array of n_2 rows

by n_1 columns, then each column contains at most t_2 errors. If this burst and some random error pattern of t or fewer errors are in the same co-set (codewords corrupted with the same error pattern) of the product code then the sum of these two error patterns is a code array in the product code. As a result, each column of the sum array must either have no nonzero components or have at least d_2 nonzero components. Each nonzero column of the sum array of the product code must be composed of at least $(d_2 - t_2)$ errors from the random error pattern and at most t_2 errors from the burst error pattern. Since there are at most t random errors, these errors can be distributed among at most $\lfloor t/(d_2 - t_2) \rfloor$ columns. This leads to conclude that the sum array must contain at most $\lfloor t/(d_2 - t_2) \rfloor * t_2 + t$ nonzero components. However,

$$\lfloor t/(d_2 - t_2) \rfloor * t_2 + t \leq t * \{ \{ t_2 / (d_2 - t_2) \} + 1 \} < 2t < d_1 d_2$$

So, the sum array contains fewer than $d_1 d_2$ nonzero components and hence it is not a code array in the product code. This contradiction implies that those two types of errors can not be in the same co-set of a standard array. They can be both the co-set leaders (different error patterns leading every co-set in the standard array) and hence be corrected. The same arguments can be applied to rows instead of columns of the sum array. So, the burst error correction capability, b , of the product code is at least $\max(n_1 t_2, n_2 t_1)$ and the simultaneous random error correction capability of the product code is t .

2.4 Error Correcting Algorithms

Most of the existing decoding algorithms are built around error correcting. Product codes are effective in correcting random errors as well as burst errors.

However, the existing decoding schemes of product codes cannot achieve the correction capability, t , without complex decoding. This is because error correction of product codes depends on the distribution of the errors in the codeword array [11]. Actually, it is not easy to characterize correctable error patterns because they depend on how the correction is done. The normal dimension by dimension (row-column or vice versa) decoding is not effective.

Researchers were trying to solve this problem of poor performance. Double decoding, i.e. performing row-column decoding twice *may* improve the correction capability but it will increase delay [13]. We review some decoding algorithms to see how researchers tackle the decoding of product codes or how they look at different aspects of product codes.

2.4.1 Cascaded Hamming Codes

Kousa analyzed a multi-dimensional coding scheme based on Hamming codes and introduced the use of the Hamming codes in a cascaded fashion, called cascaded Hamming codes [5]. The two-dimensional scheme is the popular product code. He started with n information bits. These bits are the zeroth level of cascading since they are yet uncoded for error correction. These bits can go to one or more stages of encoding (cascading). For a particular (N, K) Hamming code, the n bits are divided into groups of K bits. Each set of K bits is then mapped into N bits by means of an (N, K) Hamming encoder. The total number of bits at this stage is $(n/K)N$. This stage is referred to as the first level of cascading. The second level of cascading is formed by sorting the $(n/K)N$ available bits into groups of K bits again. These bits should constitute $(n/K^2)N$ such groups. Each group is again encoded into N bits by the same encoder. At the end of this

stage, there should be $(n/K^2) N^2 = n(N/K)^2$ bits. Continuing this process, any level of cascading can be reached. The author formulated a recursive equation that finds the bit error rate of any intended level in terms of the bit error rate of the previous level. For an (N, K) Hamming code it was shown that

$$\varepsilon' = \frac{1}{N} \sum_{i=0}^N [(i+1) \binom{N}{i} - A_i - 2A_i(N-i+1)] \varepsilon' (1-\varepsilon)^{N-i} \quad (2.1)$$

where A_i is the number of code words of weight i , ε' is the bit error rate of level one, and ε is the bit error rate of level zero (channel bit error rate). The bit error probability for higher dimensions can be found recursively in the same manner. The above relation was derived under the assumption of random errors. Therefore, it assumes that the bits are well interleaved before each level of encoding. This is to ensure that the bit errors in each block are statically independent. The author did suitable interleaving for a particular (N, K) Hamming code and up to three levels of cascading have been shown. For two levels of cascading, the resulting code is nothing but a product code as mentioned before.

2.4.2 Ordered Decoding

Chaehag *et al* used a diagonal interleaver, Figure 2.2, to randomize errors and an ordered decoding scheme for decoding [10]. A good performance was claimed at the expense of the huge amount of computation. Specifically, the authors provided a decoding algorithm for product codes over mobile data communication. They used Reed Solomon (RS) codes for both the rows and the columns of the product code. To reduce the effect of burst errors, a diagonal interleaver is used for the product code. The interleaver rearranges the bits of a codeword to reduce the errors in a row codeword and in a column codeword.

1	44	38	32	26	20	14
8	2	45	39	33	27	21
15	9	3	46	40	34	28
22	16	10	4	47	40	35
29	23	17	11	5	48	42
36	30	24	18	12	6	49
43	37	31	25	19	13	7

Figure 2.2 : Diagonal Interleaver of the (49,16) Product Code

The interleaver outputs the bits to the diagonal direction. For decoding, an ordered decoding scheme is used. The ordered decoding calculates probabilities of correct decoding for each row and column. The most reliable row or column codeword is decoded first to reduce the probability of decoding error, and then the second most reliable row or column codeword is decoded, and so on. They showed that their scheme has a 5.5dB gain over ordinary decoding. However, the required computation for their ordered decoding scheme is very large, and it is required once per product codeword. One important factor to mention is that their performance measure was not the normal BER but the bit failure rate.

2.4.3 Product Codes and Interleaved Block Codes

Dongfeng and Lijun compared product codes with interleaved block codes over mobile channel [16]. They applied nine kinds of product codes where the component codes are BCH codes. For every kind of product code, they chose an interleaved BCH code having the same elapsed time (delay). They concluded that product codes can improve the BER of mobile communication channels by one order of magnitude better than interleaved BCH codes in general with the same delay using modulation schemes such as FSK, DPSK, and 8-ary PSK.

2.4.4 Near Optimum Decoding

Pyndiah *etal* applied a new algorithm that is very effective in dealing with random error and burst error [7] [8][17]. The authors showed how powerful a product code could be. It is claimed that a near optimum performance was obtained [17][8]. However, as mentioned earlier, decoding complexity was increased drastically. They used an iterative

decoding algorithm for product codes similar to the concept of convolutional turbo code, called product turbo codes. It is based on soft-input soft-output decoders for decoding the component codes so that near optimum performance is obtained at each iteration. This soft-input soft-output decoder is a Chase decoder, which delivers soft outputs instead of binary decision. Channel state information (CSI) and a posteriori probability for decoded symbols are needed in the algorithm. The concept of ranked decoding is proposed for maximum likelihood sequence estimation (MLSE) in which the decoding process starts with the most reliable symbol and works toward the least reliable one. Error performance is evaluated in a Gaussian channel. For a Rayleigh fading channel, the error performance was evaluated for maximum likelihood demodulation of M -ary orthogonal signal. Then, they tried to reduce the complexity of the turbo product code and offered a very low complexity product turbo codes [7]. The complexity of the new one is about one tenth of that of the near optimum one for a degradation coding gain of only 0.7 dB. So, the reduced complexity of the near optimum turbo product code offers a good compromise between complexity and performance. Also, the complexity does not depend on the number of iterations done because the same decoder can be used for several iterations. This results in reduction in time which make this algorithm advantageous over convolutional turbo codes, which exhibit delay. However, they did not provide any clear theoretical justification for its good behavior.

2.4.5 Decoding of n -Dimensional Product Codes

One good such research is given in [12]. It is stated that decoding product codes on row-column basis is successful but this is at the expense of increased complexity and delay. They applied single parity check codes and Hamming codes for the component codes.

They tested the performance for two and more dimension levels. As the dimension increases, the performance improves. They applied hard decision and soft decision decoding and found that much better performance can be obtained from soft decision decoding. For the soft decision decoding two techniques were compared, maximum a posteriori probability (MAP) decoding and the log-likelihood domain. The former is better than the latter. In the former, more than one cycle (iteration) is applied, but as the number of cycles increases, the probabilities of the received symbols become correlated. So, the performance improves toward a limit with each iteration and it is better for higher dimensional product codes than lower ones because high dimensional product codes have less correlation between the decoding cycles. The latter has an advantage of determining the code-word probabilities from the a priori probabilities of the information bits meaning that the bits of check on check need not to be transmitted resulting in increasing the rate of the code. However, the minimum distance now is less than before. If those bits are kept, the performance of the latter is better. Those two soft decoding techniques are difficult to be used and time consuming.

2.4.6 Randomly Interleaved SPC Product Codes

Rankin and Gulliver proposed a scheme to reduce the number of low weight codewords and hence improve the performance of the product codes [18]. The authors considered single parity check (SPC) product codes, which are randomly, interleaved (RI) between the encoding of each dimension. For encoding, simply, after each parity check equation is encoded in a single dimension, the data are interleaved before the next dimension is encoded. For decoding, the RI SPC product code must be decoded in the reverse order of the encoding process. The component decoders are maximum a priori (MAP) decoders in

the log likelihood domain, hence the bit error probability in the component code is minimized. Simulation was done for two to five dimensional RI SPC product codes. The performance was exceptional especially as the size of the component code increases and/or the number of dimensions increases. For four-dimensional (20, 19) RI SPC, the code is only 0.63 dB away from the channel capacity at BER of 0^{-5} . The only disadvantage is the exponential increase in the block length as the number of dimensions increases and as the size of the component code increases.

2.4.7 A Multidimensional Block Coding Scheme with Iterative Decoding

Sweeny *et al* proposed an iterative decoding algorithm that uses Dorsch algorithm instead of the Chase algorithm that was used by Pyndiah [19]. The Chase algorithm produces alternative solutions to the soft inputs by testing various numbers of patterns. However, because an algebraic decoder has to be invoked for each error pattern, the number of decoding tries is kept relatively small and it is applied only to codes for which an algebraic decoder is available. The main advantage of the Dorsch algorithm over the Chase algorithm is its complexity. The Dorsch algorithm has to sort the received bits once according to their reliability and then only needs to re-encode the most reliable bits. Also, each new re-encoding does yield a new codeword while the Chase algorithm may fail to produce a solution. Another advantage is that the Dorsch algorithm is applicable to any binary code whereas the Chase algorithm can only be applied to codes for which an algebraic decoder is available. The authors also implemented the Dorsch algorithm to non-binary codes. The Dorsch algorithm showed an improved performance over the standard Chase algorithm. Also, the authors compared their scheme with that of Pyndiah in the case of the product code formed using (15, 12, 4) extended Reed-Solomon code

with a total of 4 iterations. They increased the number of decoding tries to 100 and achieved a BER of 10^{-5} at lower SNR of 2.5-2.6 dB with only a marginally increased complexity of their scheme.

2.4.8 Soft Decision Decoding

There is also soft decision decoding. Some of the previously mentioned algorithms used a kind of soft decision decoding.

Let E denote the transmitted signal energy per codeword, E_c denote the transmitted signal energy per bit in the codeword, and E_b denote the transmitted energy per information bit. Since there are n bits in a codeword, $E = n E_c$, and since each codeword conveys k bits of information, the energy per information bit is $E_b = E / k = n E_c / k = E_c / R_c$. The codewords are assumed to be equally likely a priori with prior probability $1/M$, where $M=2^k$.

In soft decision decoding [2], the optimum receiver can be realized as a parallel bank of M filters matched to the M possible transmitted signals. The outputs of the M matched filters which encompasses the transmission of n bits in the codeword are compared and the codeword corresponding to the largest matched filter output is selected. The receiver implementation can be simplified. So, an equivalent optimum receiver can be realized by the use of a single filter matched to the BPSK waveform used to transmit each bit in the codeword followed by a decoder which forms the M decision variables corresponding to the M codewords. To be specific, let $y_j, j = 1, 2, \dots, n$, represents the n sampled outputs of the matched filter for any particular codeword. Since the signaling is BPSK, the output y_j may be expressed either as

$$y_j = -2E_c + v_j \quad (2.2)$$

when the j th bit of a codeword is a 1 or as

$$y_j = 2E_c + v_j \quad (2.3)$$

when the j th bit of a codeword is a 0. The variable v_j represents AWGN with zero mean and $2E_c N_0$ variance. From knowledge of the M possible transmitted codewords and upon reception of the y_j , the optimum decoder forms the M decision variables

$$U_i = \sum_{j=1}^n (2c_{ij} - 1)y_j \quad i = 1, 2, \dots, M \quad (2.4)$$

where c_{ij} denotes the bit in the j th position of the i th codeword. Thus, if $c_{ij} = 1$, the weighting factor $2c_{ij} - 1 = 1$, and if $c_{ij} = 0$, the weighting factor $2c_{ij} - 1 = -1$. In this manner, the weighting $(2c_{ij} - 1)$ aligns the signal components in y_j such that the decision variable corresponding to the actual transmitted codeword will have a mean value $2nE_c$ while the other $M - 1$ decision variables will have smaller mean values.

Although the above computation for optimum soft decision decoding is relatively simple, it may still be impractical to compute for all possible codewords when the number of codewords is large, e.g., $M > 2^{10}$. In such cases, sub-optimum soft decision decoding may be used and they are available in the literature. This is out of the scope of this work.

2.4.9 Erasure Decoding

Erasure decoding is a special kind of soft decision decoding. The simplest form of soft decision decoder uses erasure to indicate the reception of a signal whose corresponding symbol value is in doubt. Such a channel has an input alphabet of size Q and an output alphabet of $Q+1$. The extra symbol is called an erasure flag, or simply an erasure. Decoding using error correction decoder for rows and an error-erasure correction decoder

for columns or vice versa is effective in dealing with random errors and helps significantly in fading and bursty channels [14].

The matter now lies on finding the coordinates for erasing. Assuming the order of transmission is along the rows, the decoding method relies on having an error correcting decoder for the rows and an error-and-erasure correcting decoder for the columns.

1. Decode the rows. For any row that cannot be decoded (all rows can be decoded for perfect codes), erase it. For any row i in which corrections are made, record ω_i , the number of corrections.
2. If an odd number of rows have been erased, erase one more row, choosing the one for which ω_i is largest.
3. Calculate the error correction capability of the product code as $[(d_1 (d_2 - e) - 1) / 2]$ where e is the number of rows erased.
4. Decode one column. If decoding succeeds, count $d_1 - \omega_i$ for every position in which the value is corrected and ω_i for every (unerased) position in which no correction occurs. If this count is less than or equal to the error correction capability of the code, then the column is correctly decoded. Otherwise, or if the original decoding failed, erase two more rows (with largest ω_i), recalculate the error correction capability, and repeat.
5. After each successful column decoding, move on to the next column. Rows previously erased remain erased.

There are other erasure techniques for non-binary symbols like Berlekamp-Massey algorithm and Euclid algorithm [15][20]. The algorithms for non-binary symbols are so complicated and Galois Field (GF) arithmetic must be invoked there.

2.4.10 Other Researches

There are also many other techniques in the field of product codes. Henkel and Chung introduced a filling procedure for the array of product codes that will reduce delay [9]. Rajpal and Lin discussed product-coded modulation in which good modulation codes can be constructed for the AWGN and Rayleigh fading channel [21].

2.5 Thesis Contribution

Most proposed decoding algorithms achieve high correction capability at the cost of increased complexity and delay. The objective of this work is to improve the performance of the product codes without a significant increase in decoding complexity. Specifically, the thesis introduces three modified algorithms.

The first modified algorithm is the *modified hard decision decoding* (MHDD) algorithm. The MHDD algorithm develops a decoding algorithm for a product code that is capable of reaching the correction capability of the code without significant increase in the decoding complexity. The algorithm is suitable for component codes that are single error correcting. The MHDD algorithm is extended to three dimensions. The extended MHDD algorithm is also capable of reaching the theoretical error correction capability of the product code in three dimensions.

The second modified algorithm is the *modified soft decision decoding* (MSDD) algorithm. The MSDD algorithm investigates the performance of soft decision decoding for binary symbols. It proposes a simple modification for the theoretical soft decision decoding algorithm for binary symbols to improve the performance.

The third modified algorithm is the *modified erasure decoding* (MED) algorithm. The MED algorithm is to investigate erasure decoding for binary symbols. We will investigate the erasure characterization in [11]. The MED algorithm modifies slightly the erasure characterization in [11]. It selects symbols for erasing instead of whole rows. Conceptually, some improvement in decoding could be achieved without significantly increasing the complexity of decoding.

The above mentioned modified algorithms will be applied to the well-known Hamming codes. The modified algorithms will be evaluated in terms of their correction capability, bit error rate performance, and complexity. They will be tested over AWGN and/or Rayleigh fading channels. They will be compared with relevant algorithms reported in the literature.

2.5.1 Thesis Organization

The next chapter, Chapter 3, introduces the hard decision decoding. The first modified algorithm, the MHDD algorithm, is introduced. The algorithm flowchart is presented and the main aspects are explained. The extension of this algorithm to three dimensions is presented.

In Chapter 4, the soft decision decoding and erasure decoding are explained. The MSDD and the MED algorithms are presented. The algorithms flowchart are presented and explained.

In both chapters, the modified algorithms will be compared with relevant equations, algorithms, and approaches, and they will be tested over AWGN and/or Rayleigh fading channels. A measure of the complexity of the modified algorithms will be provided.

The findings and conclusions of this thesis are summarized in Chapter 5. Also, suggestions for direct extensions of this work that worth further investigations are stated.

CHAPTER 3

HARD DECISION DECODING OF PRODUCT CODES

3.1 General Overview

As mentioned in previous chapters, in hard decision decoding, the n bits from the demodulator corresponding to a received codeword are passed to the decoder. The decoder compares the received codeword with the M possible transmitted codewords and decides for the codeword that is closest in Hamming distance to the received codeword. This minimum distance decoding rule is the optimum decoding in the sense that it results in a minimum probability of a codeword error for the binary symmetric channel. This method is computationally inefficient so syndrome decoding is better to use in the sense that it is optimum and more efficient. Syndrome decoding for Hamming Codes is used.

3.2 Hard Decision Decoding of Product Codes

It had been shown that the minimum distance, d_{min} , of the product code is d_1d_2 . It follows that the random error correction capability, t , of the product code is $\lfloor (d_1d_2-1)/2 \rfloor$ and all-fewer random errors. This capability can not be achieved, except for the simple parity-check component codes, without complex decoding.

So, the straightforward approach of performing row-column decoding, or vice versa, is not effective. We refer to this approach by Ordinary Hard Decision Decoding (OHDD) algorithm, as depicted in Figure 3.1. This approach does not recover all error patterns promised by the minimum distance of the product code although it can recover some higher error patterns [13]. This can be demonstrated as follows. Take the product of two single-error-correcting codes such as the (7, 4) Hamming code. The resultant product code has a minimum distance of 9 and hence should correct all patterns of four and less random errors. If those four errors are arranged in a rectangular shape, as in Figure 3.2 then the simple strategy of performing row-column decoding or vice versa will make things worse. The row correction will add an extra error into each of the infected rows and the column correction will then do the same into each of the erroneous columns. So, the product code will thus be wrongly decoded.

The smallest number of errors that prohibits correct decoding in row-column decoding can be determined [6]. Let t_1 and t_2 be the guaranteed error correcting capability of the row and column codes, respectively. For it to be possible for an array to fail, a certain number of row failures should occur. Row failures may happen when t_1+1 errors occur in any row. For the column decoding to fail, it must be true that at least t_2+1 row failures

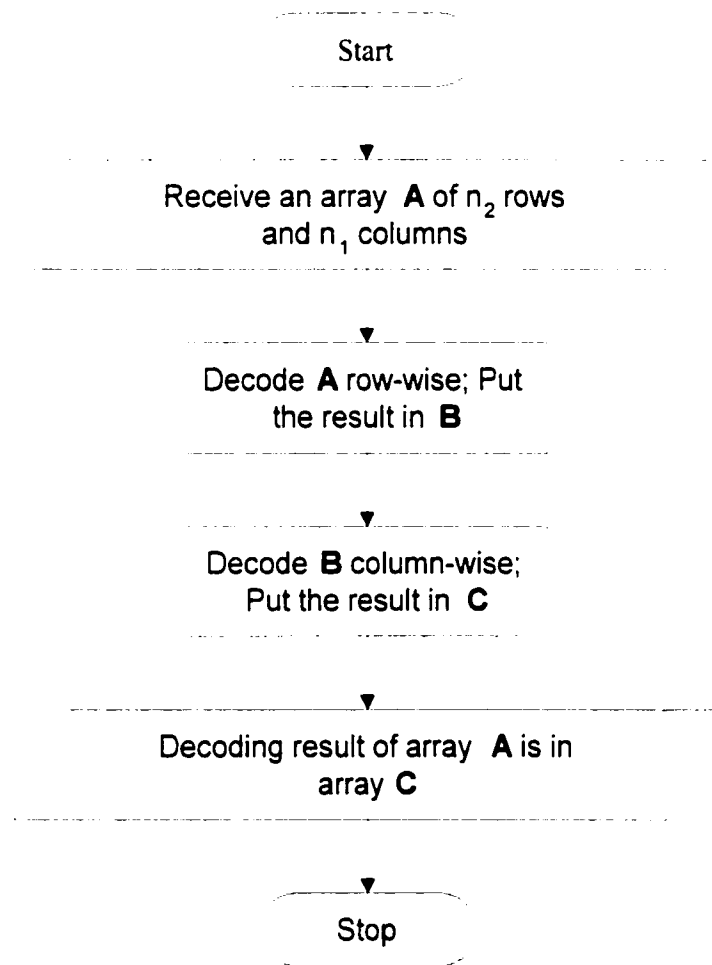


Figure 3.1 : The OHDD Algorithm

	X		X			
	X		X			

Figure 3.2: Uncorrectable Four-Error Pattern

occur and, moreover, these errors are arranged in a particular shape. Thus, any error pattern with the following number of errors, t_g , or less is correctable.

$$t_g = (t_1+1)(t_2+1)-1 \quad (3.1)$$

This is of course less than the capability of the product code, roughly half as large for large distance codes. Take $d_1=d_2=21$ meaning $d_{min}=441$. This product code should correct up to 220 errors but Equation (3.1) yields that only 120 errors are guaranteed to be corrected. Nevertheless, it should be observed that many higher error patterns are in fact correctable.

Double decoding, as mentioned in Chapter 2 *may* improve the correction capability but it will increase delay [13].

3.3 MHDD of Product Codes

Now, we are in a position to present the first modified algorithm which is the MHDD algorithm. This algorithm makes the product code capable of reaching its theoretical capability of correction for product codes generated from single error correcting component codes. Theoretically, component codes having a minimum distance of three will make the generated product code of minimum distance nine. So, such product code should be able to correct all four or less random error patterns. The OHDD algorithm, as discussed earlier, could not do the job. The MHDD algorithm uses Hamming codes as component codes and assumes hard decision symbols are coming from the demodulator. The motivation of the MHDD algorithm is to reach the theoretical capability of the product code. The algorithm basically modifies the basic row-column decoding and tries

to account for the most dominant errors by gathering some information. The flowchart of the MHDD algorithm is shown in Figure 3.3.

Briefly, at the beginning, a codeword \mathbf{A} of hard symbols is received after demodulation. Then, the following steps are made in sequence.

- 1) The received codeword, \mathbf{A} , is decoded row-wise giving \mathbf{A}_R . For any row correction, the row number is indicated, $\mathbf{A}_{R,i}$, the column number is indicated, $\mathbf{A}_{R,i(j)}$, and the number of correction is counted, which is one for Hamming codes. After finishing all rows, we will have information about the rows in which correction was made, $\mathbf{A}_{R,i}$, the place of correction, $\mathbf{A}_{R,i(j)}$, and the total number of correction made, $\{\mathbf{A}_{R,i}\}$. If the total number of corrections made for rows, $\{\mathbf{A}_{R,i}\}$, is not two or three, go to step 3.
- 2) Decode the original received codeword, \mathbf{A} , columns-wise giving \mathbf{A}_C . For any column correction, we will have information about the columns in which correction was made, $\mathbf{A}_{C,j}$, the location of the corrected bit, $\mathbf{A}_{C,j(i)}$, and total number of corrections made for columns, $\{\mathbf{A}_{C,j}\}$. If the location of the corrected bit corresponds to a row in which a correction was done, $\mathbf{A}_{C,i,r}$ such scenario is counted $\{\mathbf{A}_{C,i,r}\}$.
- 3) Decode, \mathbf{A}_R , column-wise giving \mathbf{A}_{RC} . If column correction is made, the location of the correction is indicated, $\mathbf{A}_{RC,j(i)}$. From $\mathbf{A}_{RC,j(i)}$, the number of corrections happened in each row, \forall_i , can be found. If the total number of corrections made for rows, $\{\mathbf{A}_{R,i}\}$, is not two or three, go to step 5.

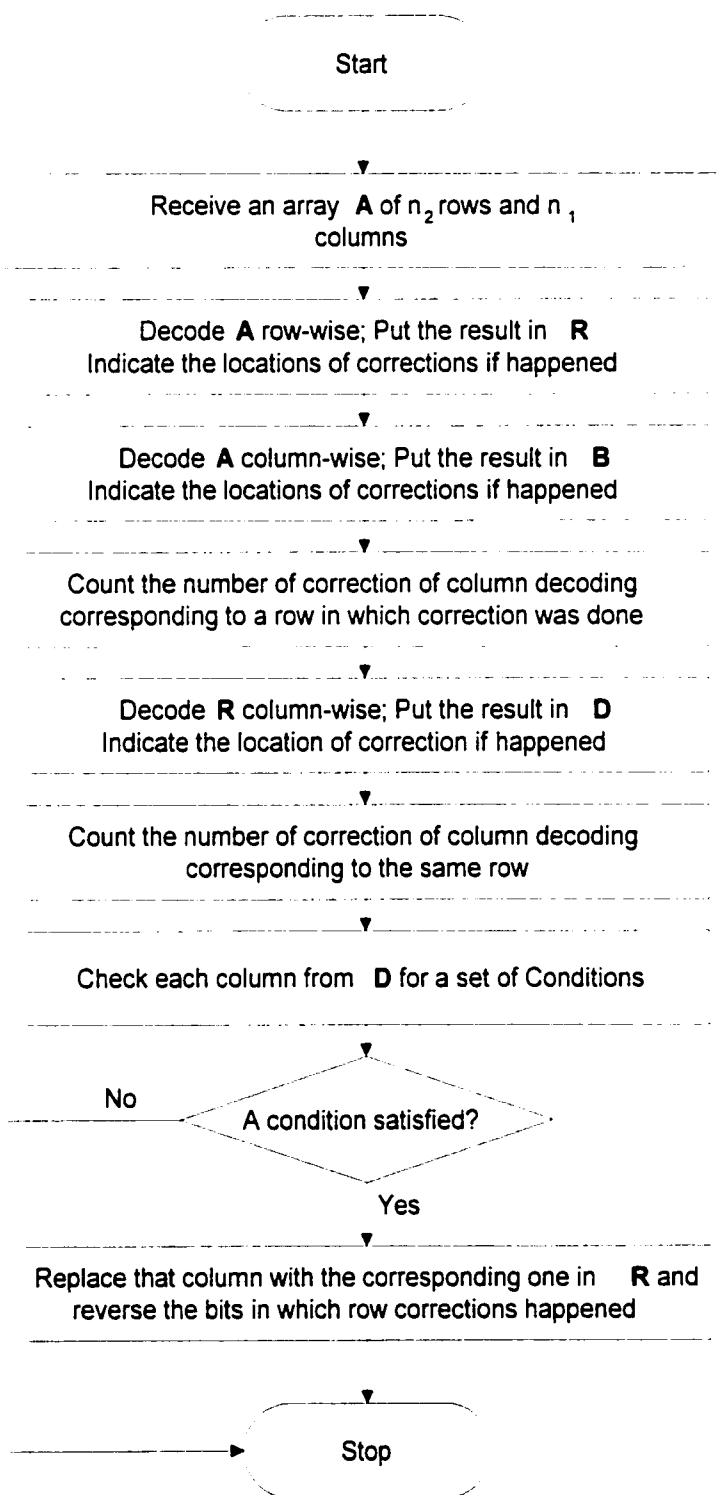


Figure 3.3 : The Simplified MHDD Algorithm

- 4) From all the above gathered information, certain conditions should be checked for each corrected column in A_{RC} . If one of the conditions is satisfied, the corrected column is replaced by the column generated from the codeword after row decoding, A_R , and the bits corresponding to row corrections, $A_{R,i}$ are reversed. Otherwise go to step 5.
- 5) Decoding of codeword A is thus done.

The conditions are set by doing exhaustive search. They try to give an idea about the dominant error patterns that took place for that codeword. Actually, we analyzed the set of actions (the above gathered data) of the decoder after row decoding and after column decoding to find symptoms to map a certain set of actions to a certain error pattern.

To explain how did we reach to the conditions, we start by the four-error pattern that was depicted in Figure 3.2. This error pattern and all other four error patterns can be corrected if the following condition is satisfied. Remember that any condition is checked for every column in A_{RC} if a correction was done for that column.

- $(\{A_{R,i}\} = 2) \& (A_{RC,i} \neq A_{R,i})$.

This condition means that if there were two corrections when the codeword was decoded row-wise and a column correction made a correction outside the rows in which corrections were done. So, reverse the two bits from the corrected column corresponding to where row corrections were done. Although the above condition ensures that the product code will reach its theoretical error correction capability, the BER of the product code may not improve because the condition will affect other higher error patterns that are correctable by the OHDD algorithm. So, let's be more specific in choosing the

uncorrectable four-error patterns. The four-error patterns having rectangular shapes will be corrected by the following condition:

- $(\{A_{R,i}\} = 2) \& (A_{RC,i} \neq A_{R,i}) \& (\forall_i = 3) \& (\{A_{C,j}\} = 2)$.

We still have other four-error patterns that can not be corrected. Figure 3.4 shows those four-error patterns. The four-error pattern in Figure 3.4 (a) will be corrected by the following condition:

- $(\{A_{R,i}\} = 2) \& (A_{RC,i} \neq A_{R,i}) \& (\forall_i = 3) \& (\{A_{C,j}\} = 3) \& (\{A_{C,i,r}\} = 2)$.

The four error-patterns in Figure 3.4 (b) and (c) will be corrected by the following condition:

- $(\{A_{R,i}\} = 2) \& (A_{RC,i} \neq A_{R,i}) \& (1 \leq \forall_i \leq 2) \& (\{A_{C,j}\} \leq 4) \& (\{A_{C,i,r}\} \geq 1)$.

A six-error pattern distributing in three rows each row having two errors while one of the column has three errors, can be corrected by the following condition if columns decoding attempts to add errors:

- $(\{A_{R,i}\} = 3) \& (A_{RC,i} \neq A_{R,i}) \& (\forall_i = 1) \& (\{A_{C,j}\} = 4) \& (\{A_{C,i,r}\} = 3)$.

We can not proceed more since as you try to correct some error patterns others will be affected.

To test the MHDD algorithm against the OHDD algorithm, all error patterns from 1 to 8 are generated and the two algorithms are tested. The result is in Table 3.1. We stopped at 8 error patterns since more than this will have a very small probability to occur and the time needed to be generated is huge, about 2 days for all the nine error patterns. It is seen from the table that the correction capability of the code, which is four, is reached by the MHDD algorithm while it is only three using the OHDD algorithm as Equation 3.1 indicates. Furthermore, it can be observed that the number of errors remaining after

(a)

	X		X		O	
	O		X		X	

(b)

	X	X	O			
		O	X	X		

(c)

	X	X	O			
			X	X	O	

* O: represents an error introduced after row decoding

Figure 3.4: Uncorrectable Four-Error Patterns

Number of Introduced Errors		4	5	6	7	8
Total Number of Error Patterns		211876	1906884	13983816	85900584	450978066
Total Number of Bits		10381924	93437316	685206984	4209128616	22097925234
OHDD Algorithm	Average # of bits in error after decoding per codeword	3.857	3.986	4.408	5.223	6.526
	% of uncorrected patterns	4.371	18.432	42.003	67.322	85.532
MHDD Algorithm	Average # of bits in error after decoding per codeword	0	2.826	4.421	5.195	6.526
	% of uncorrected patterns	0	17.576	41.826	67.318	85.515
% of Corrected Patterns by OHDD and MHDD		95.629	81.499	57.998	32.677	14.468
% of Corrected Patterns by MHDD not OHDD		4.371	0.925	0.1766	0.004107	0.0172
% of Corrected Patterns by OHDD not MHDD		0	0.0694	0	0	0
% of Not Corrected Patterns by Both		0	17.507	41.826	67.318	85.515

Table 3.1: Comparison Between OHDD and MHDD of (49, 16) Product Code

decoding is in average equals the number of errors before decoding and sometimes they are less. Also, other information can be extracted from the table regarding the number of correct arrays (blocks), number of errors, and the number of error patterns corrected by one algorithm and the other could not. The information from the table indicates that the MHDD algorithm outperforms the ordinary algorithm.

To further test the algorithms, they are tested over AWGN and fading channels. Figure 3.5 shows the performance of the two algorithms over AWGN channel. It is noticed that the MHDD algorithm has about one-half dB gain over the OHDD algorithm.

For fading channel, Figure 3.6 and Figure 3.7 show that the two algorithms are comparable for the same fade rate. However, low fade rate gives bad performance. This is because if the magnitude of the fade rate is small, it means slowly fading channel. This means that the channel represents high correlation among the generated fade amplitudes. So, the system with less correlation (fast fading or bigger fade rate) performs better than the one with higher correlation. Also, the MHDD algorithm is better than the OHDD algorithm as the fade rate increases. It can be seen from Figure 3.6 that for 0.1 fade rate, the gain is about 0.5 dB. The difference between the two algorithms is not notable as the fade rate decreases. Also, as the dimension of the product code increases, the performance improves especially at lower fade rate. This is because at low fade rate the number of burst errors is high. So, the errors are more randomized in a higher dimensional product code.

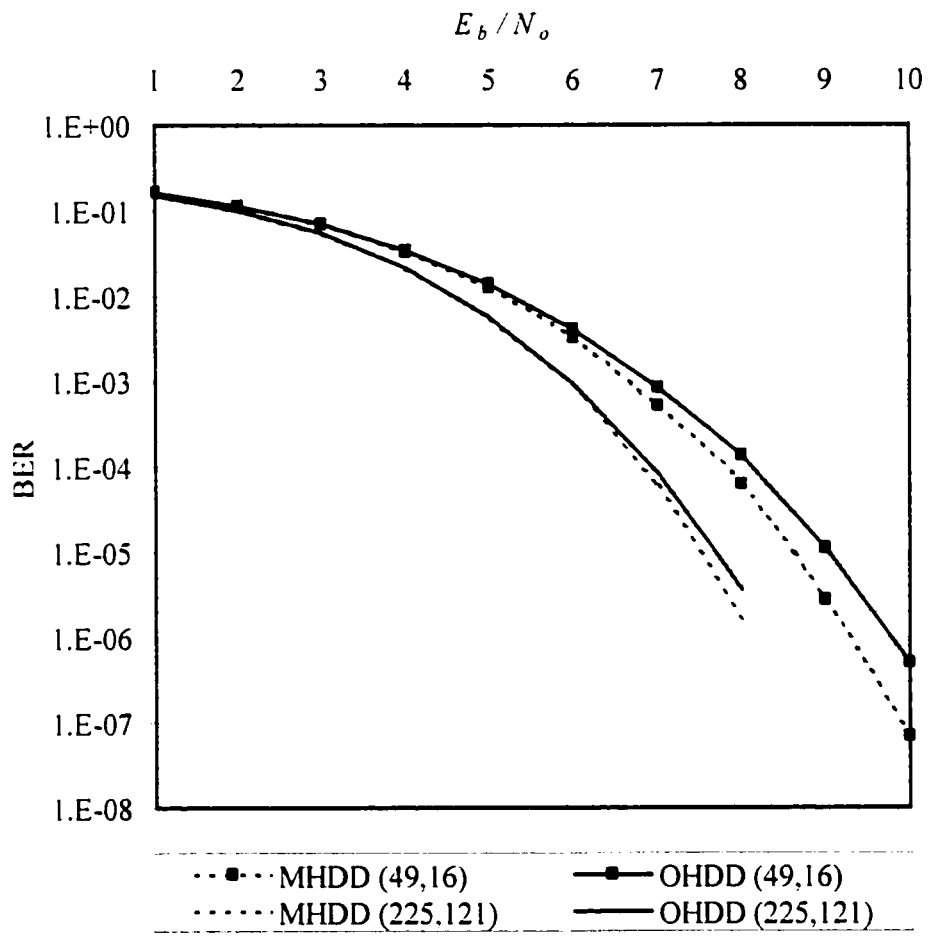


Figure 3.5 : Hard Decision Decoder Performance over AWGN Channel

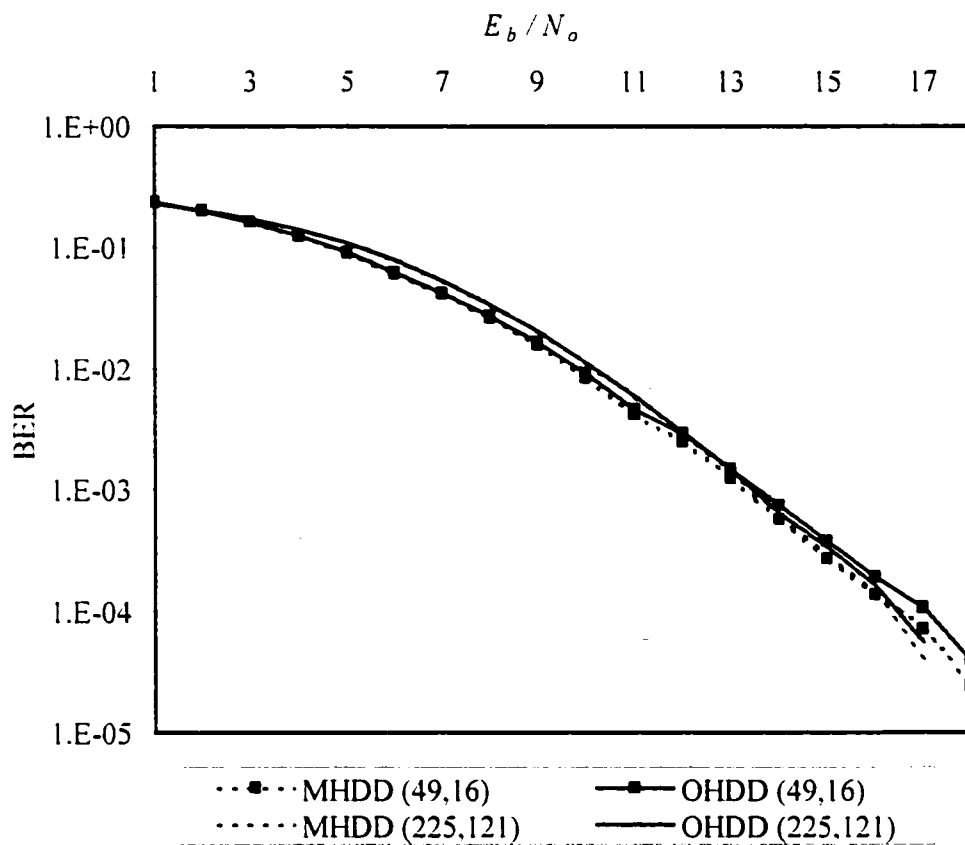


Figure 3.6 : Hard Decision Decoder Performance over Fading Channel of 0.1 Fade Rate

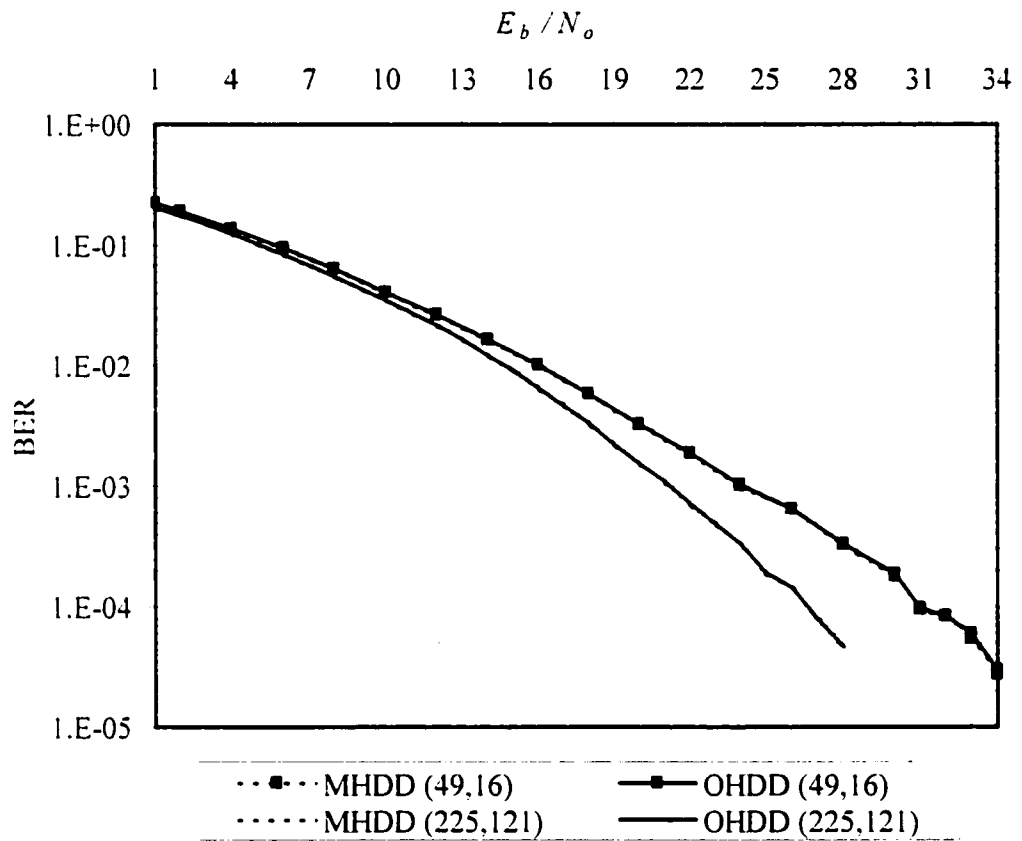


Figure 3.7 : Hard Decision Decoder Performance over Fading Channel of 0.001 Fade Rate

3.4 Extension to Three Dimensions

Now, let's extend the MHDD algorithm to a three-dimensional product code. We saw earlier that the MHDD algorithm does reach the theoretical capability of error correction, which is four for the Hamming product code. By the same concept, the extension to three dimensions will be done by investigating the most dominant error patterns.

First, we would like to characterize the errors from zero to three by noting what will happen to these errors after row decoding and then column decoding to the row-decoded codeword. We will have the following cases:

- I. Case 1: No errors; In this case no row or column corrections will take place.
- II. Case 2: One error; In this case a single row correction takes place.
- III. Case 3: Two errors; The errors in this case could be in two different rows or in the same row. For the former, we will have two row corrections and no column corrections. For the latter, we will have one row correction and three column corrections where all column corrections are in the same row in which correction was done.
- IV. Case 4: Three errors; we will have one of the following three cases.
 - a) If the errors are all in different rows, we will have three row corrections and no column corrections.
 - b) If the errors are all in one row, we will have two scenarios. If the three errors do not correspond to a valid codeword, we will have one row correction and four column corrections where all column corrections are in the same row in

which correction was done. If the three errors correspond to a valid codeword, we will have no row corrections and three column corrections.

- c) If the errors are distributed into two rows, we will have two row corrections and three column corrections where all column corrections are in one of the two corrected rows.

All errors of four and five will have different cases than those mentioned above. So, a plane containing four or five error-pattern will be recognized and distinguished from other planes having less error patterns. More than five errors may or may not produce one of the cases above.

Now, the extension to three dimensions is as follows. A codeword \mathcal{A} of hard symbols is received after demodulation.

- 1) Decode a plane consisting of rows and columns using the MHDD algorithm.
- 2) If one of the above cases happened, the case number is indicated for this plane. Otherwise, a flag is set to one for this plane.
- 3) Repeat steps 1 and 2 for all planes. After finishing step three, every plane is either corresponding to one of the cases or it has a flag set to one.
- 4) Find the two planes, if any, in which the bits may be reversed with the following conditions.
 - a) Condition 1: If the total number of flags that are set to one is two, the bits to be reversed are corresponding to the planes where the flags are set to one.
 - b) Condition 2: If the total number of flags that are set to one is one and only one highest case can be decided, the bits to be reversed are corresponding to where the flag is set to one and where the highest case is in.

- c) Condition 3: If no flags are set to one and we have only two cases, those two cases should be either two cases of 4 or case 4 and case 3. The bits to be reversed are corresponding to where the two cases are in.
 - d) Condition 4: If no flags are set to one and we have three cases, those three cases should be two of case 4 and one of case 1. The bits to be reversed are corresponding to where the two highest cases are in.
- 5) Decode one vector from the third dimension.
- a) If no correction takes place, the decoding of that vector is finished.
 - b) If a correction took place, check the above conditions.
 - If none of the conditions are satisfied, decoding that vector is finished.
 - If a condition is satisfied, check the location of the error. If the error location corresponds to one of the chosen planes in which the bits will be reversed, accept the decoding of that vector. Otherwise, the error location does not correspond to any of the chosen planes in which the bits will be reversed, do not accept the decoding of that vector. Instead, reverse the bits in the chosen planes.
- 6) Repeat step5 for all vectors in the third dimension.

The above algorithm will be called the extended MHDD algorithm and its flowchart is shown in Figure 3.8. It is guaranteed to reach the theoretical error correcting capability of the three dimensional product codes of 13 while the extended OHDD algorithm can correct up to 7 errors only.

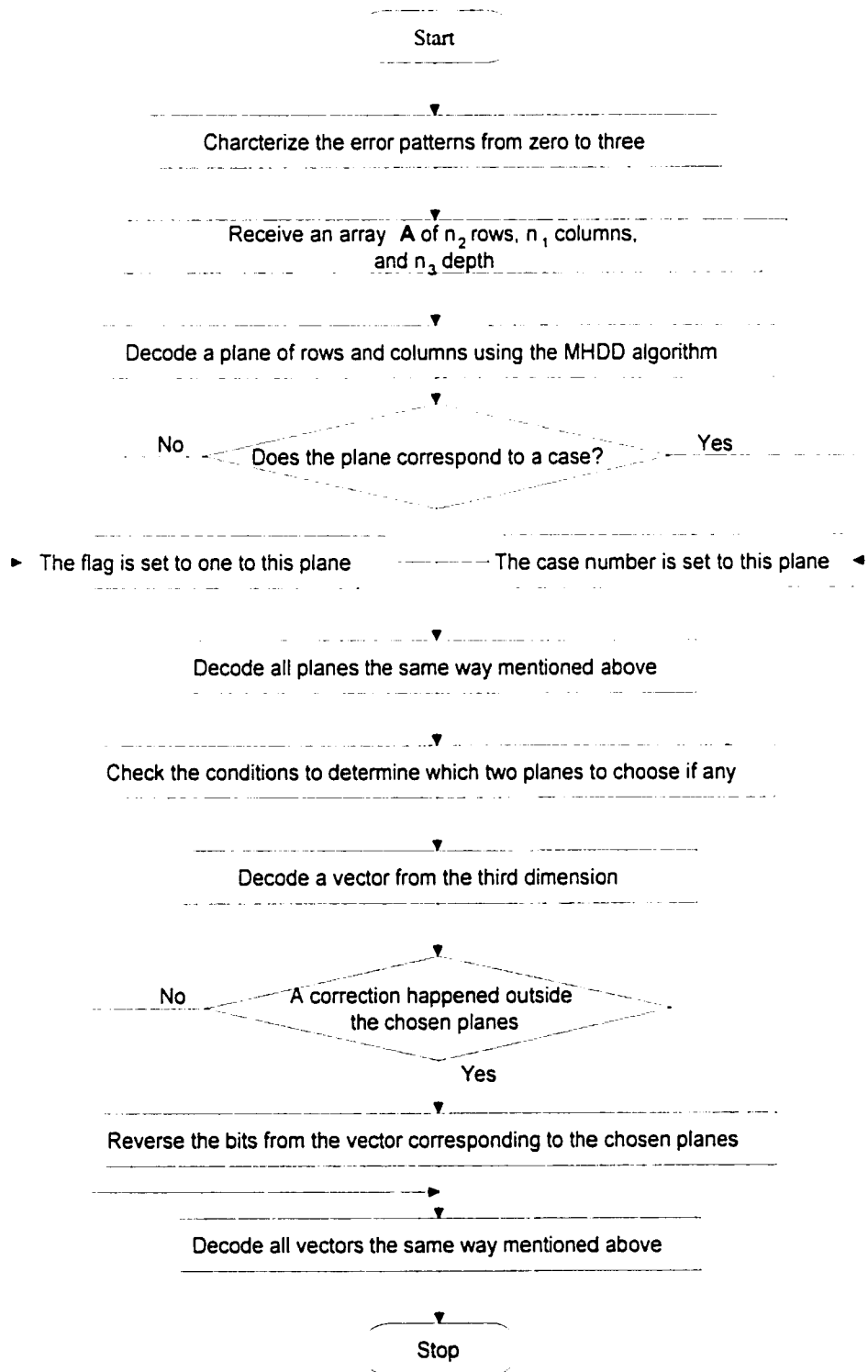


Figure 3.8 : The Extended MHDD Algorithm

To prove that the extended MHDD algorithm reaches the theoretical limit of 13, we will examine 13-error patterns. Since the MHDD algorithm can correct up to 4 errors, it is guaranteed that the extended MHDD algorithm can correct all error patterns from 1 to 9. This can be proven by simply invoking Equation 3.1, where t_1 equals 4 and t_2 equals 1. We are left with the errors from 10 to 13. Remember that the correction of a plane may fail if a plane has five or more errors. We will consider the distribution of the error patterns into two to five planes only. If all the errors are in one plane, the third dimension will correct any errors left after planes corrections.

1. Ten errors in two planes:

- a) Five and five: We will have two flags set to one, condition 1. If the decoding of a vector from the third dimension attempts to correct an error outside the two planes, do not accept the vector decoding but reverse the bits corresponding to the two planes. However, if the decoding of the vector from the third dimension attempts to correct an error inside one of the two planes, accept the vector decoding.
- b) Four and six: The plane containing four will indicate a flag and the one with six may indicate a flag, making a total of two flags, condition 1. If the plane containing the six errors does not indicate a flag, still it will be chosen as a case, condition 2. The decoding of a vector from the third dimension always attempts to correct an error inside the plane containing the six errors.
- c) Three and seven, two and eight, one and nine: The plane containing the lower number of errors will indicate a case definitely. Irrespective of the plane with higher number of errors, the decoding of a vector from the third dimension always

attempts to correct an error inside the plane containing the higher number of errors.

2. Eleven errors in two planes:

- a) Five and six, four and seven: They are like ten errors in two planes, part (b).
- b) Three and eight, two and nine, one and ten: They are like ten errors in two planes, part (c).

3. Twelve errors in two plane:

- a) Six and six: If the two planes containing the errors indicate flag, it is like ten errors in two planes, part (a). If one of them indicates a flag, it is like ten errors in two planes, part (b). If the two planes do not indicate a flag, they will be chosen since they are two cases of 4, condition 3. So, the decoding of a vector from the third dimension attempts to correct an error outside the two planes, do not accept the vector decoding but reverse the bits corresponding to the two planes.
- b) Five and seven, four and eight: They are like ten errors in two planes, part (b).
- c) Three and nine, two and ten, one and eleven: They are like ten errors in two planes, part (c).

4. Thirteen errors in two planes:

- a) Six and seven: If the two planes containing the errors indicate flag, it is, like, ten errors in two planes, part (a). If one of them indicate a flag, it is like ten errors in two planes, part (b). If the two planes do not indicate a flag, they will be chosen since they are two cases one of case 4 and the other is case 3, condition 3. So, the decoding of a vector from the third dimension attempts to correct an error outside

the two planes, do not accept the vector decoding but reverse the bits corresponding to the two planes.

- b) Five and eight, four and nine: They are like ten errors in two planes, (b).
- c) Three and ten, two and eleven, one and twelve: They are like ten errors in two planes, part (c).

From the above discussion for the errors that were distributed in two planes, the distribution of errors in three, four, and five planes will cause a problem for the third dimension if two planes are left in error after plane corrections. This will limit our discussion to the following distributions:

1. Eleven errors in three planes, we will have only one situation that will produce two planes in error after planes corrections.
 - a) One, five and five: It is like ten errors in two planes, part (a).
2. Twelve errors in three planes:
 - a) One, five and six: It is like ten errors in two planes, part (b).
 - b) Two, five and five: It is like ten errors in two planes, part (a).
3. Thirteen errors in three planes:
 - a) One, five and seven: It is like ten errors in two planes, part (b).
 - b) One, six and six: It is like twelve errors in two planes, part (a). However, if the three planes do not indicate a flag, they are all cases. Those cases are exactly the one set by condition 4. So, the highest two cases will be chosen. So, the decoding of a vector from the third dimension attempts to correct an error outside the two chosen planes, do not accept the vector decoding but reverse the bits corresponding to the two chosen planes.

- c) Two, five and six: It is like ten errors in two planes, part (b).
- 4. Twelve errors in four planes, we will have only one distribution that will produce two planes in error after planes corrections.
 - a) One, one, five and five: It is like ten errors in two planes, part (a).
- 5. Thirteen errors in four planes:
 - a) One, one, five and six: It is like ten errors in two planes, part (b).
 - b) One, two, five and five: It is like ten errors in two planes, part (a).
- 6. Thirteen errors in five planes, we will have only one distribution that will produce two planes in error after planes corrections.
 - a) One, one, one, five and five: It is like ten errors in two planes, part (a).

If the errors are distributed into more than five planes. they will be corrected.

To test the extension to three dimensions, the extended MHDD algorithm is compared with the extended OHDD algorithm. Figure 3.9 shows the performance of these two extended algorithms over AWGN channel. Figure 3.9 shows that the extended MHDD algorithm outperforms the extended OHDD algorithm. It has about 0.5-dB gain over the OHDD algorithm.

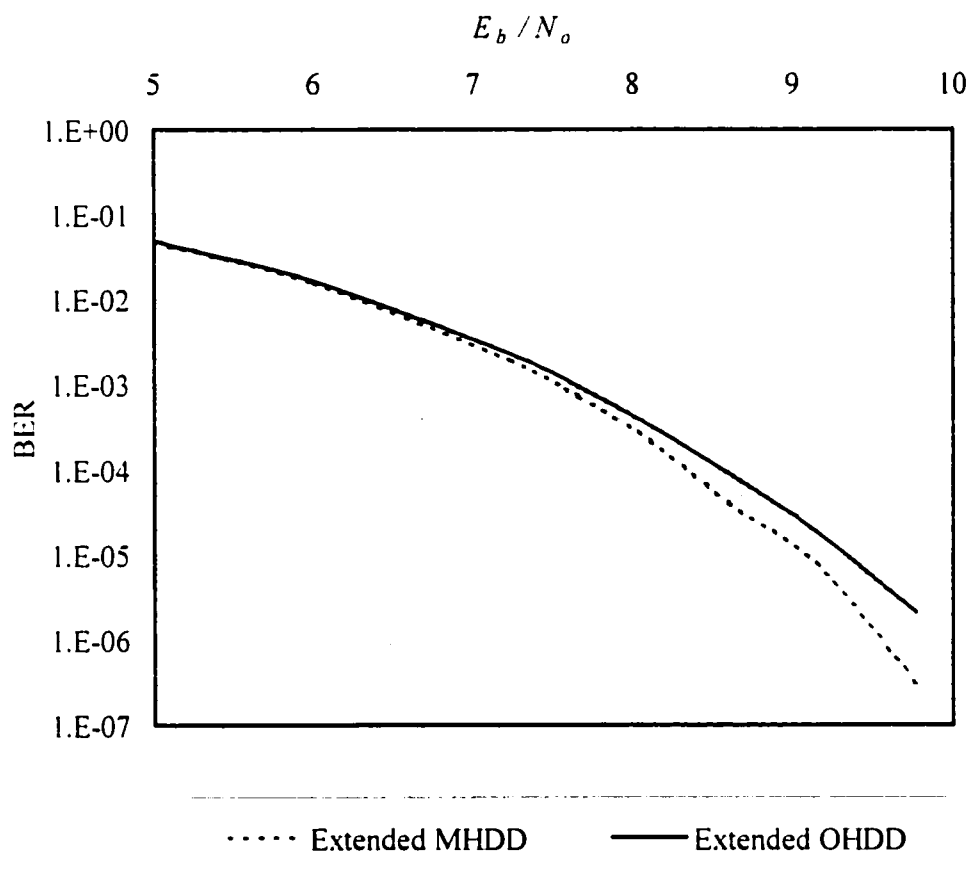


Figure 3.9 : Hard Decision Decoder Performance of Three Dimensional (343,64) Product Code over AWGN Channel

3.5 Performance Criteria

The block error rate was given in Equation 1.3. That formula is applicable for t -error correcting code in which all error patterns up to t are correctable and all error patterns more than t are not correctable. Unfortunately, that equation is very underestimating when applied to product codes. This is because any product code which can correct all error patterns up to t errors, can correct many higher error patterns. To accommodate this situation, Equation 1.3 can be modified as follows

$$P_{block} = \sum_{i=t+1}^n B_i \varepsilon^i (1 - \varepsilon)^{n-i} \quad (3.2)$$

where B_i is the number of uncorrectable patterns of i errors and ε is the channel bit error probability. The values for B_i are taken from Table 3.1 for up to eight error patterns. Higher error patterns are assumed to be uncorrectable. Consequently, the bit error probability, Equation 3.2, can be modified to the form:

$$P_{bit} \approx \frac{1}{n} \sum_{i=t+1}^n i B_i \varepsilon^i (1 - \varepsilon)^{n-i} \quad (3.3)$$

where i means that the number of errors after decoding is, in average, equal to the number of errors before decoding.

Figure 3.10 and Figure 3.11 compare the OHDD algorithm and the MHDD algorithm with Equations 2.1 and 3.3 over AWGN channel. Figures 3.10 shows a perfect matching between the simulation of the OHDD algorithm and equation 2.1. Also, Figure 3.10 and Figure 3.11 show that Equation 3.3 is a very good approximation for the bit error rate of the OHDD algorithm and the MHDD algorithm.

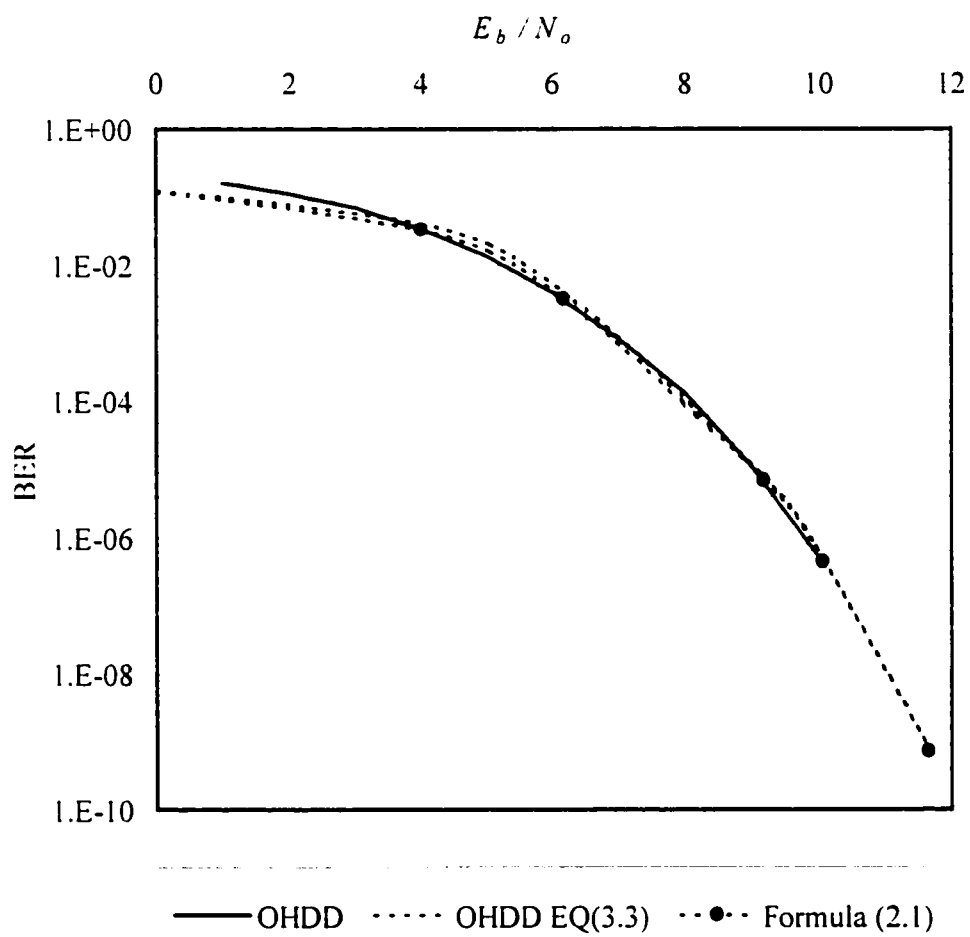


Figure 3.10 : Ordinary Hard Decision Decoder Performance of (49,16) Product Code over AWGN Channel

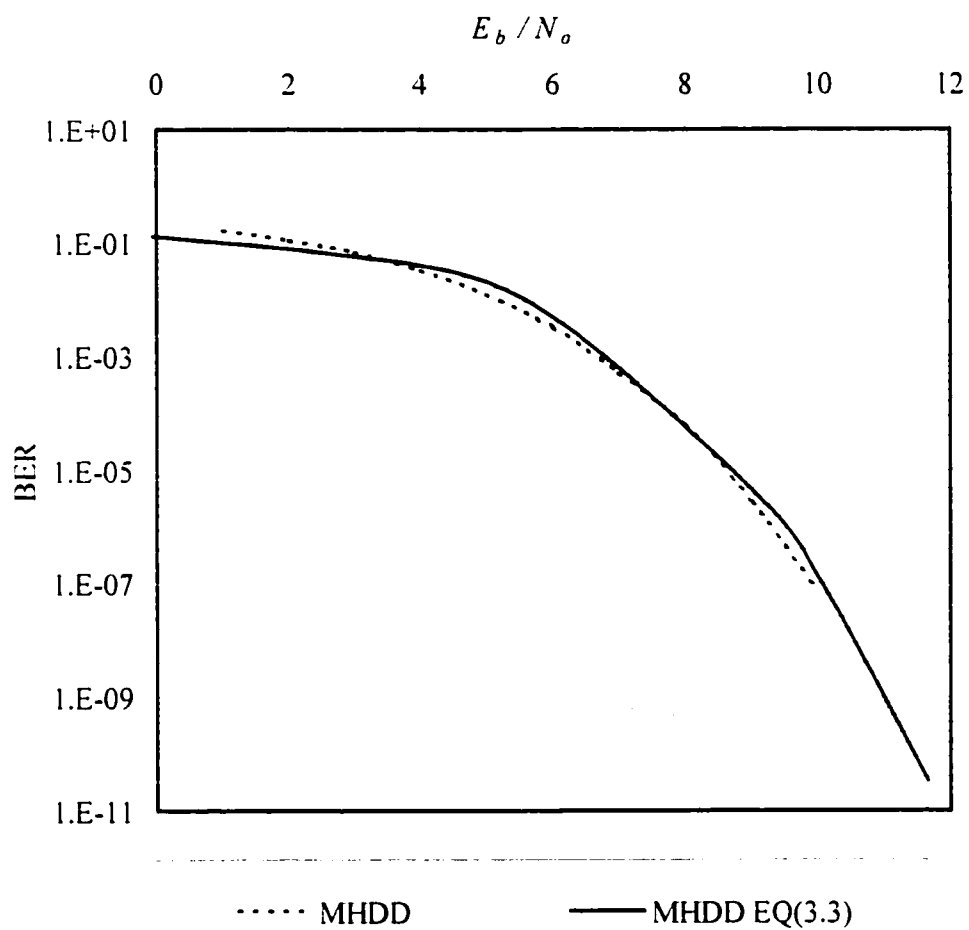


Figure 3.11 : Modified Hard Decision Decoder Performance of (49,16) Product Code over AWGN Channel

3.6 Performance Comparison

Now, the MHDD algorithm will be compared to one more algorithm, which is the ordered decoding algorithm introduced in Chapter 2. The effect of diagonal interleaver, which has been introduced in Chapter 1, on these algorithms will be tested.

Figure 3.12 and Figure 3.13 compare the OHDD and MHDD algorithms incorporating diagonal interleaver with the ordered decoding method over AWGN channel. The figures show that the diagonal interleaver has no effect over AWGN channel. Also, the ordered decoding method performs worse than the other two algorithms over AWGN channel.

Figure 3.14 and 3.15 compare the same algorithms over fading channel with fade rate of 0.1 while Figure 3.16 and 3.17 compare them for a fade rate of 0.001. The four Figures show that as the fade rate increases, the diagonal interleaver provides some significance. The ordered decoding method in general works worse than other algorithms. In general, the ordered decoding method and the diagonal interleaver are affected mutually by the length of the product code and the fade rate of the channel. However, as the length of the product code increases or as the fade rate increases, the performance, generally, improves.

Next we test the effect of increasing the number of decoding iterations. Figures 3.18 shows the performance of the OHDD algorithm over AWGN channel, under one iteration (row only) to six iterations (row then column trials). It can be observed that row-column decoding gives a big advantage over the row only decoding. A third decoding iteration still provides some significance, but more iteration is of no practical value and is merely a waste of time.

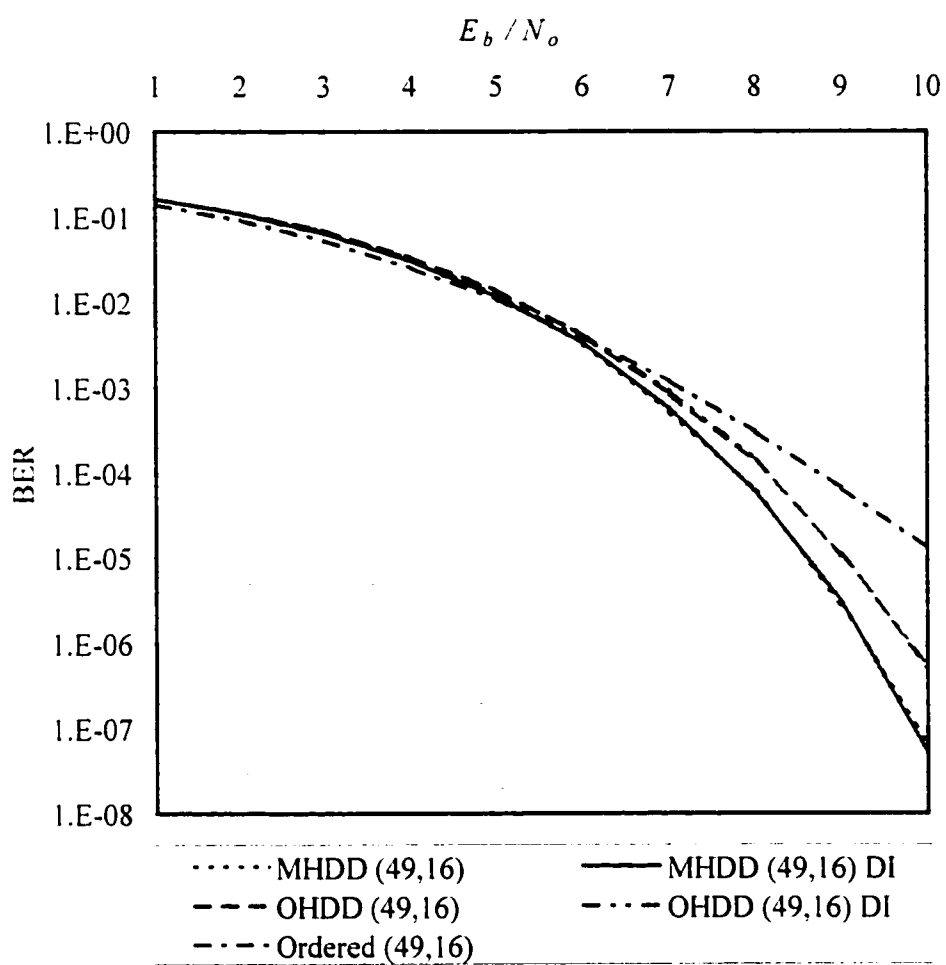


Figure 3.12 : Performance Comparison of Hard Decision Decoder of (49,16) Product Code over AWGN Channel

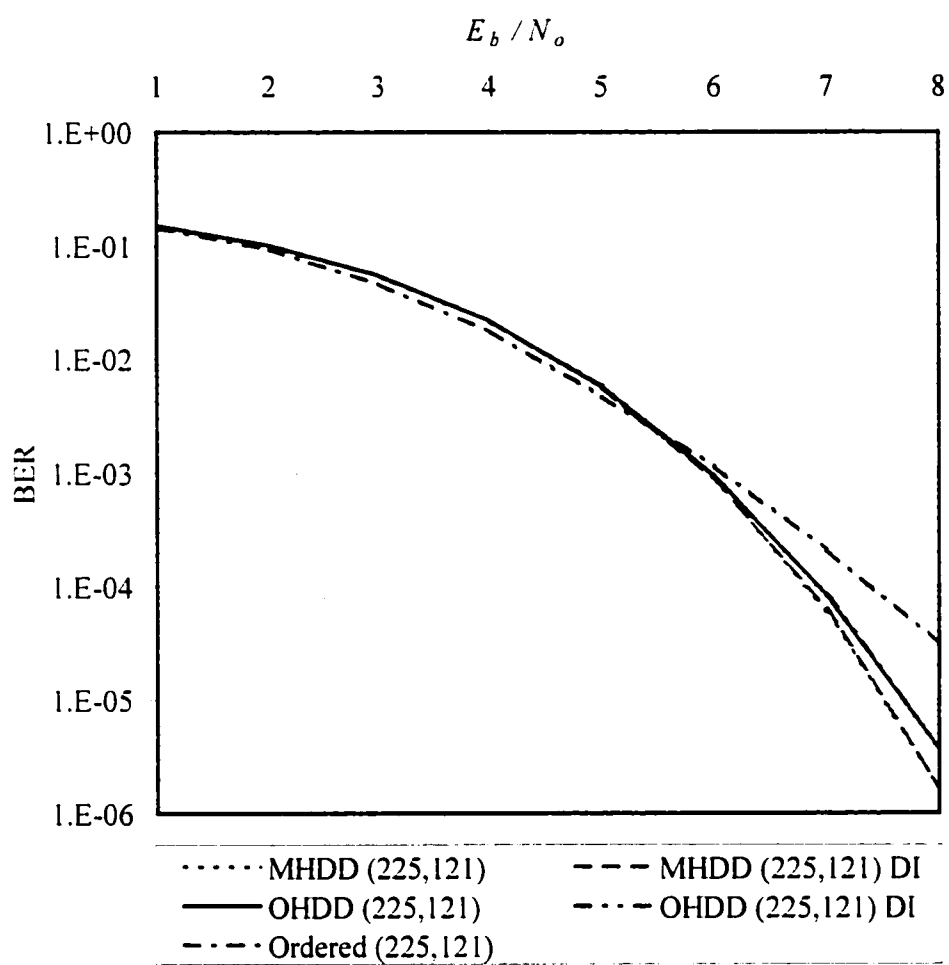


Figure 3.13 : Performance Comparison of Hard Decision Decoder of (225,121) Product Code over AWGN Channel

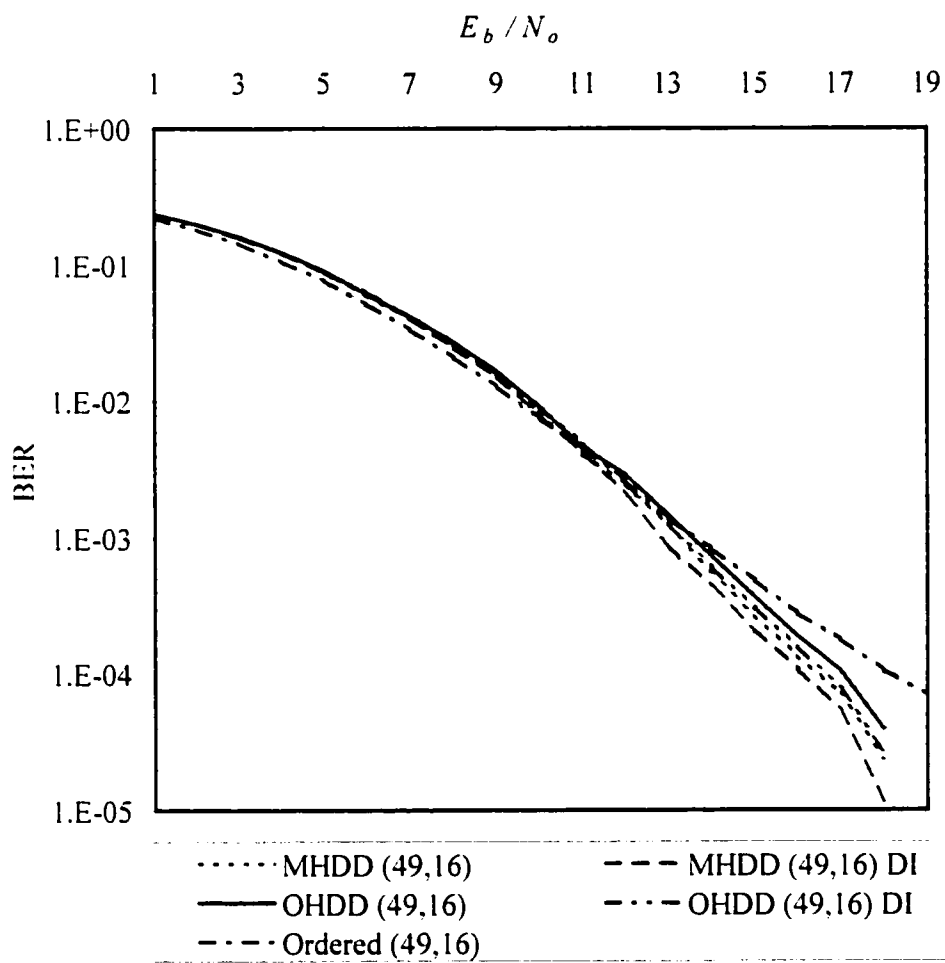


Figure 3.14 : Performance Comparison of Hard Decision Decoder of (49,16) Product Code over Fading Channel of 0.1 Fade Rate

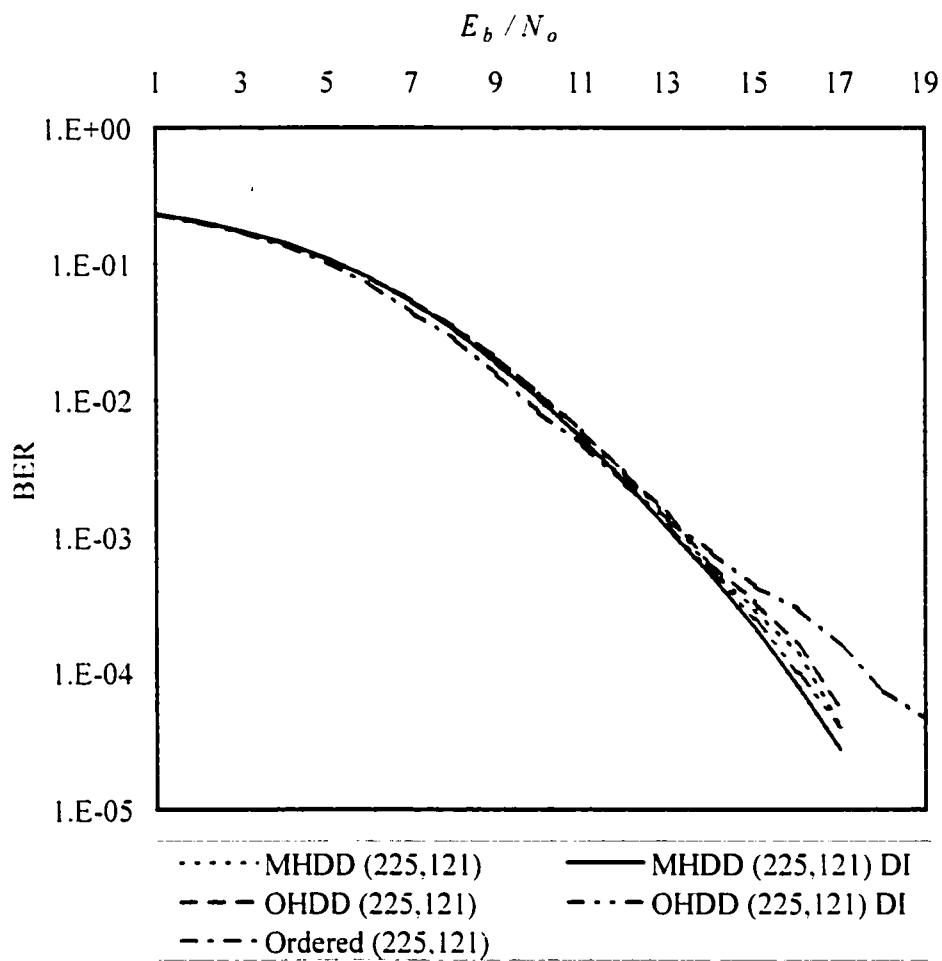


Figure 3.15 : 3.15 : Performance Comparison of Hard Decision Decoder of (225,121) Product Code over Fading Channel of 0.1 Fade Rate

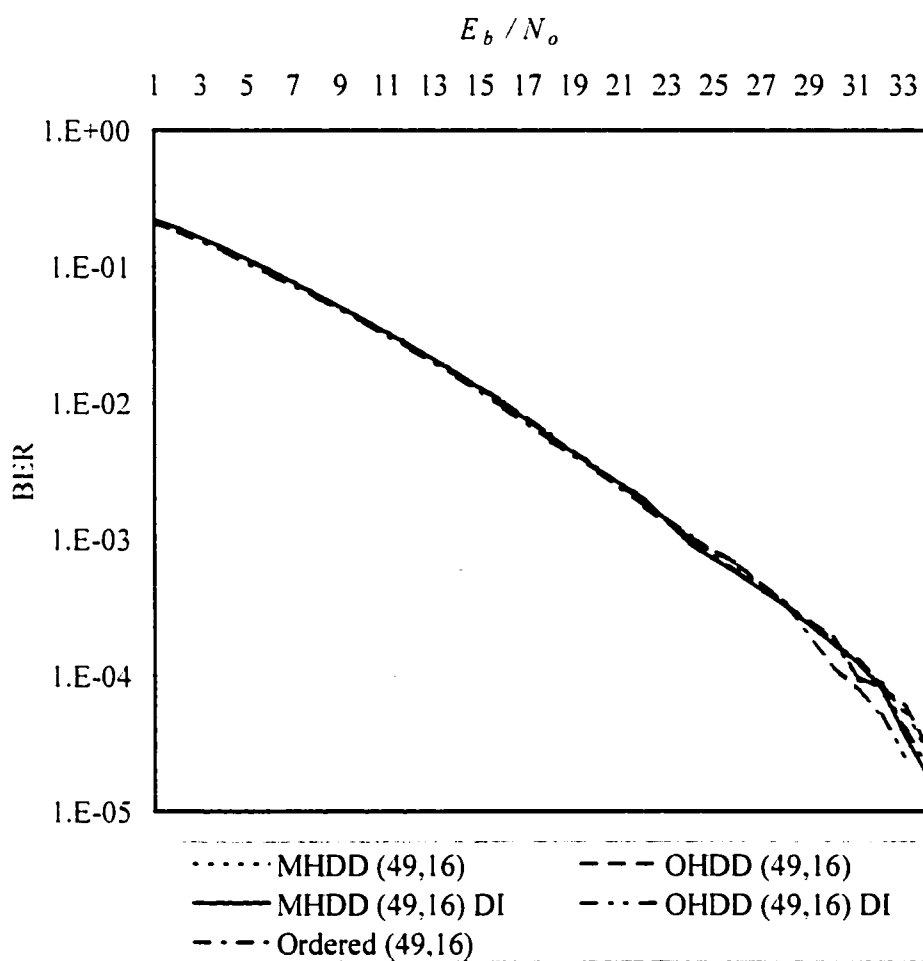


Figure 3.16 : Performance Comparison of Hard Decision Decoder of (49,16) Product Code over Fading Channel of 0.001 Fade Rate

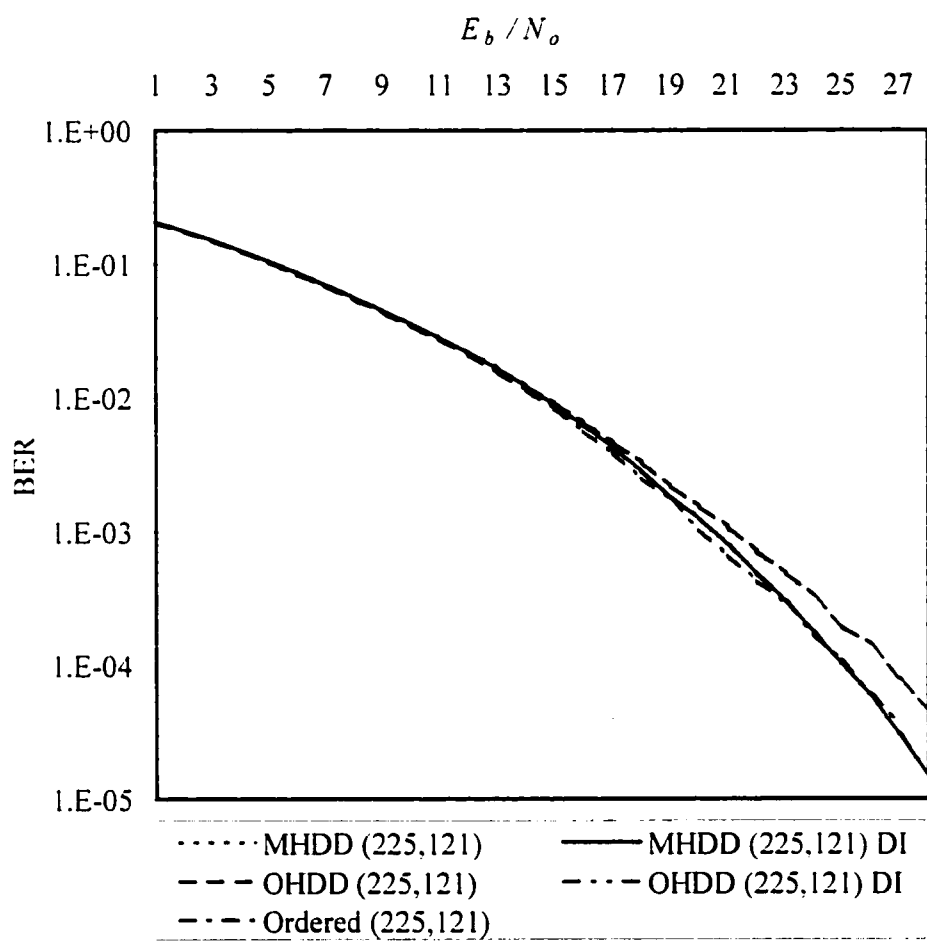


Figure 3.17 : Performance Comparison of Hard Decision Decoder of (225,121) Product Code over Fading Channel of 0.001 Fade Rate

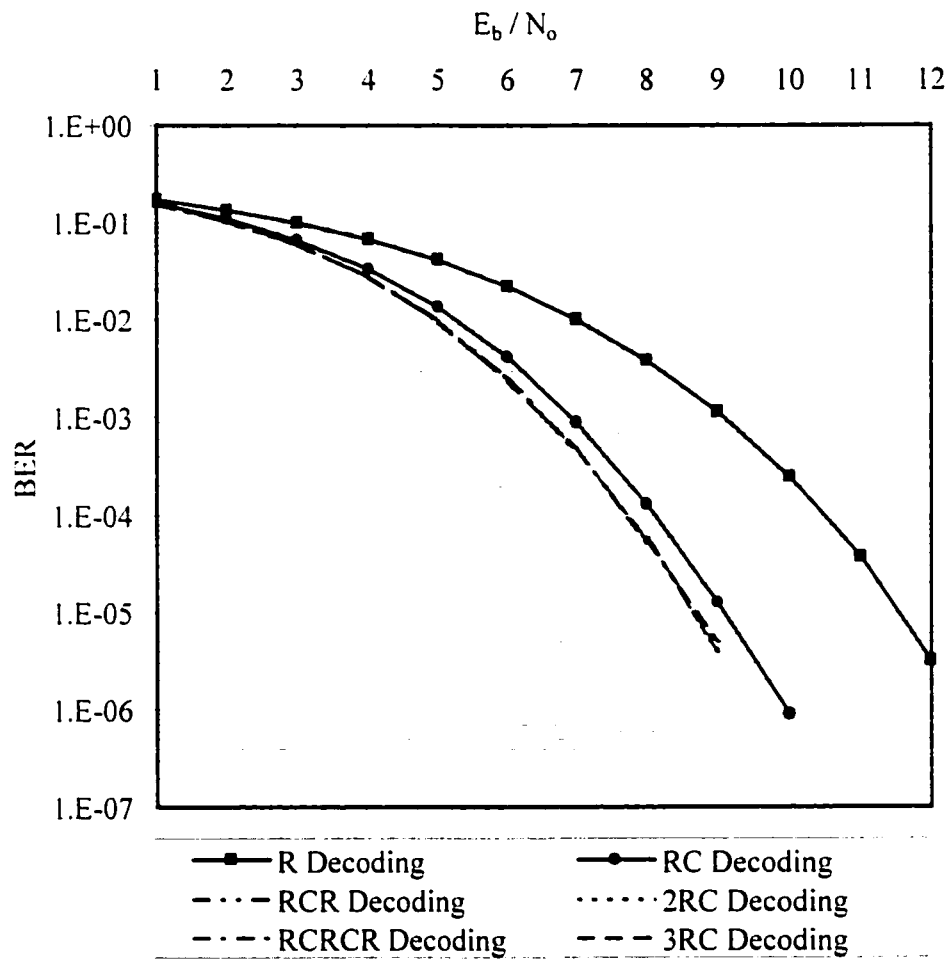


Figure 3.18 : Different Rounding Decoding of (49,16) Product Code of Hard Decision Decoding Over AWGN Channel

CHAPTER 4

SOFT DECISION AND ERASURE

DECODINGS OF PRODUCT

CODES

4.1 General Overview

In a hard decision receiver, the detector limits its range of choices to the same group of choices available at the transmitter. If the transmitter makes binary transmission, then the receiver will do binary decision. Unfortunately, hard decisions destroy information that can improve the overall performance of the communication system. In some cases, the received signal may not offer a clear choice of what symbol has been transmitted. Rather than force a decision that is likely to be incorrect, the soft decision receiver uses an

expanded selection of choices to communicate the quality of the received symbol to the error control decoder.

4.2 Soft Decision Decoding

In Chapter 2, we introduced the soft decision decoding. A product code can be decoded softly by combining the demodulator with the decoder. Every codeword should be correlated to all possible codewords, which are 2^{16} for the (49, 16) product code and then choose the one with highest correlation. This is not practical. A more realistic approach is proposed [12] and illustrated in Figure 4.1. Soft decoding will be performed for rows or columns only. The results (symbols) from the first (soft) decoding are hard output. The other dimension will be decoded hardly. We refer to the above algorithm as the Ordinary Soft Decision Decoding (OSDD) algorithm. By doing so, the advantage of soft decoding is exploited in one dimension only, rows or columns. This algorithm can be modified to take advantage of soft decoding in both dimensions, as illustrated in Figure 4.2.

Considering a received codeword \mathbf{A} of soft symbols, the following steps are made in sequence.

- 1) The received codeword, \mathbf{A} , is decoded row-wise yielding \mathbf{A}_R . Decoding is done on the basis of Equation 2.3. For each row decoding, we will end up with the decoded row, $\mathbf{A}_{R,i}$, and an associated weight of the row, $\mathbf{W}_{R,i}$.
- 2) The received codeword, \mathbf{A} , is decoded column-wise yielding \mathbf{A}_C . Decoding is done on the basis of Equation 2.3. For each column decoding, we will end up with the decoded column, $\mathbf{A}_{C,j}$, and an associated weight of the column, $\mathbf{W}_{C,j}$.

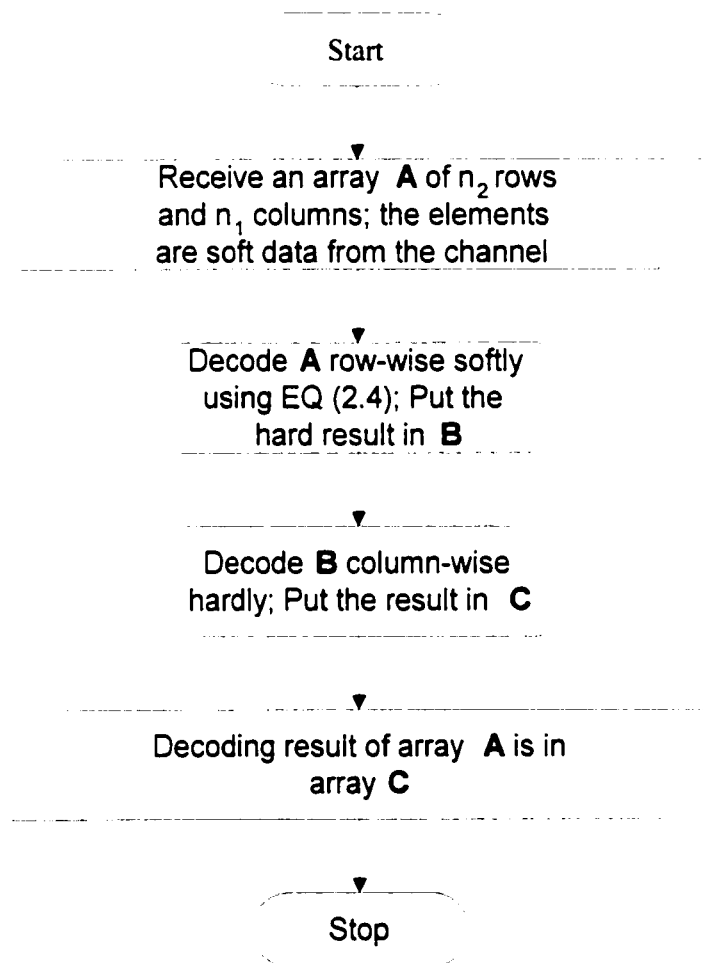


Figure 4.1 : The OSDD Algorithm

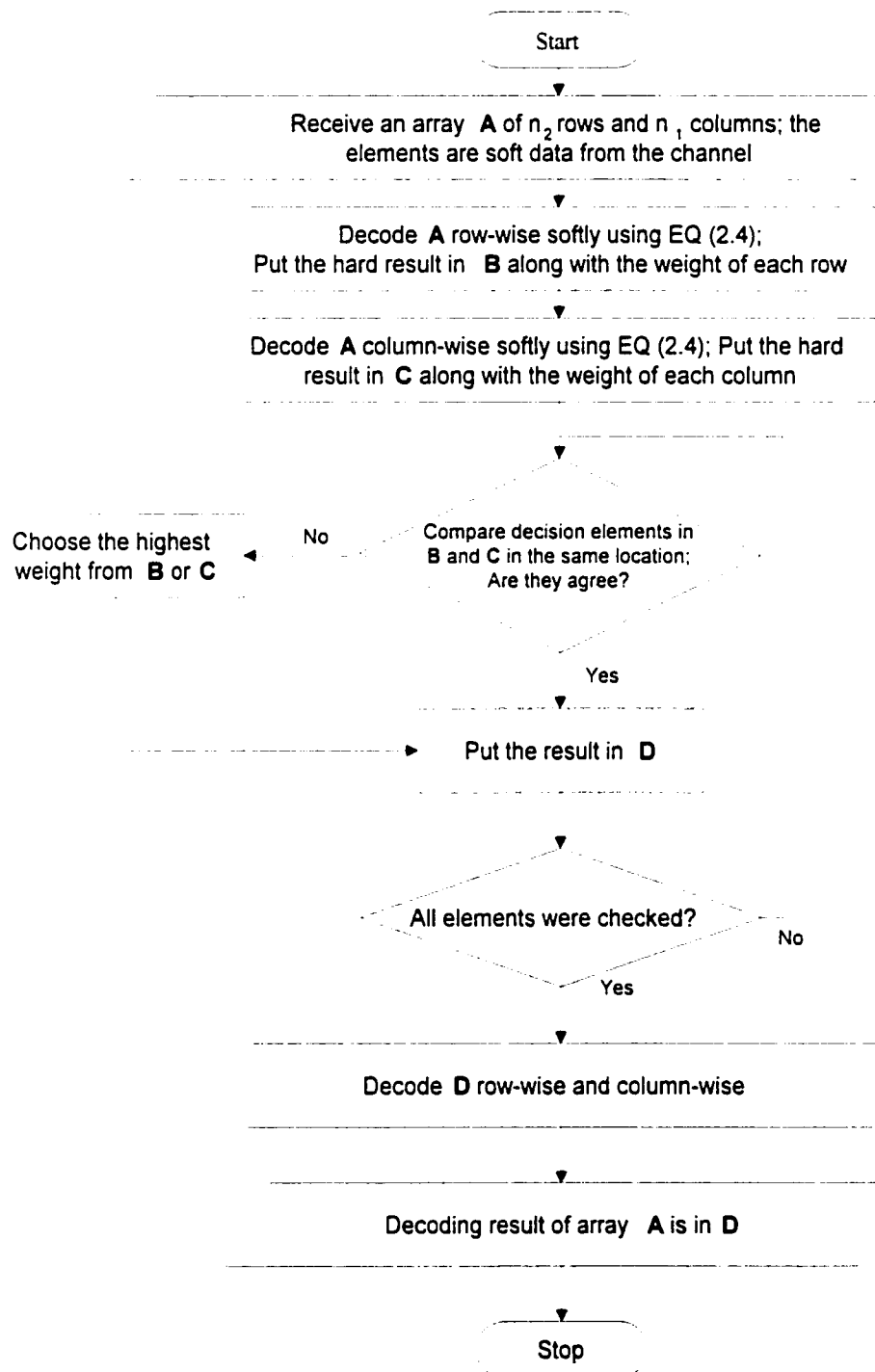


Figure 4.2 : The MSDD Algorithm

- 3) The symbol at the intersection of $A_{R,i}$ and $A_{C,j}$ is decoded by comparing its value in $A_{R,i}$ and $A_{C,j}$; If they agree, it is decoded as such. Otherwise, it is decoded based on the maximum of the two weights $W_{R,i}$ and $W_{C,j}$.
- 4) Re-decode the resultant symbols hardly row-wise and then column-wise because of the existence of conflict from the comparison in step 3. Decoding codeword A finished.

We refer to the modified algorithm as the Modified Soft Decision Decoding (MSDD) algorithm.

The two algorithms were tested over AWGN and fading channels. The results are similar to those observed in hard decision decoding. For AWGN channel, Figure 4.3 shows that a MSDD has more than 0.6 dB gain over OSDD at a BER of 10^{-5} . Also, the OSDD algorithm was tested by start decoding the columns instead of rows and by decoding hardly one more iteration, but essentially no difference was observed in both cases.

Figure 4.4 and Figure 4.5 show the performance of the two algorithms over fading channels of 0.1 and 0.001 fade rate respectively. It can be observed that as the fade rate increases, the performance of both algorithms improves. However, the difference between the two algorithms is not notable especially as the fade rate decreases. Also, as the length of the product code increases, the performance improves especially at lower fade rate.

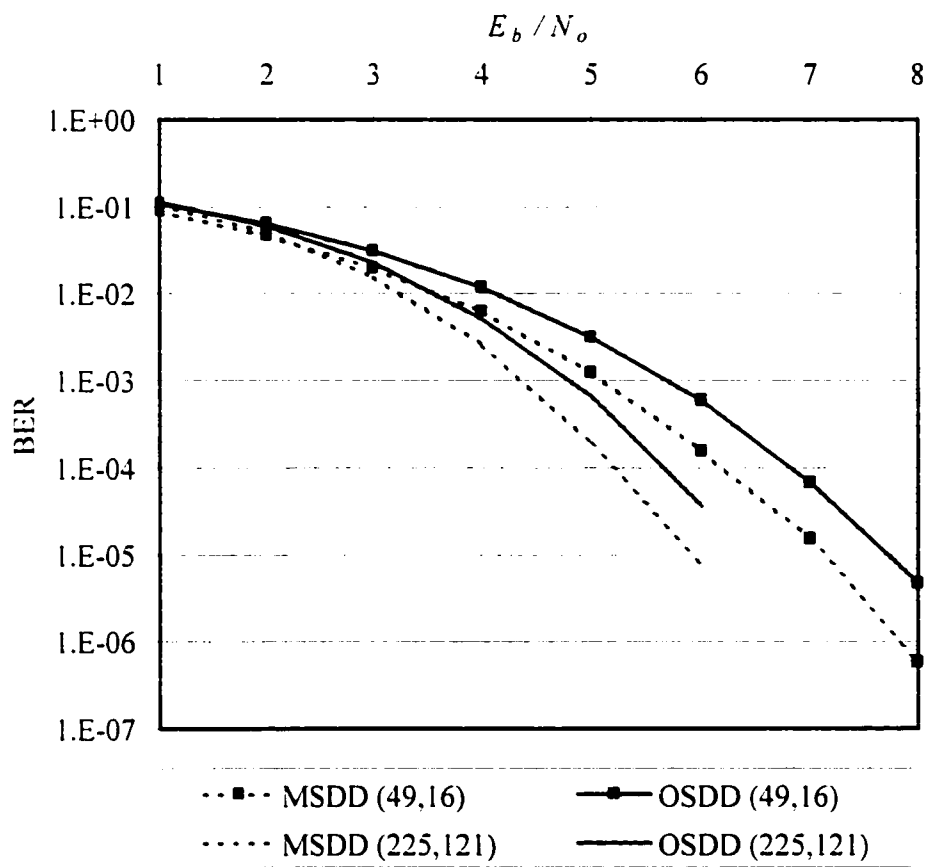


Figure 4.3 : Soft Decision Decoder Performance over AWGN Channel

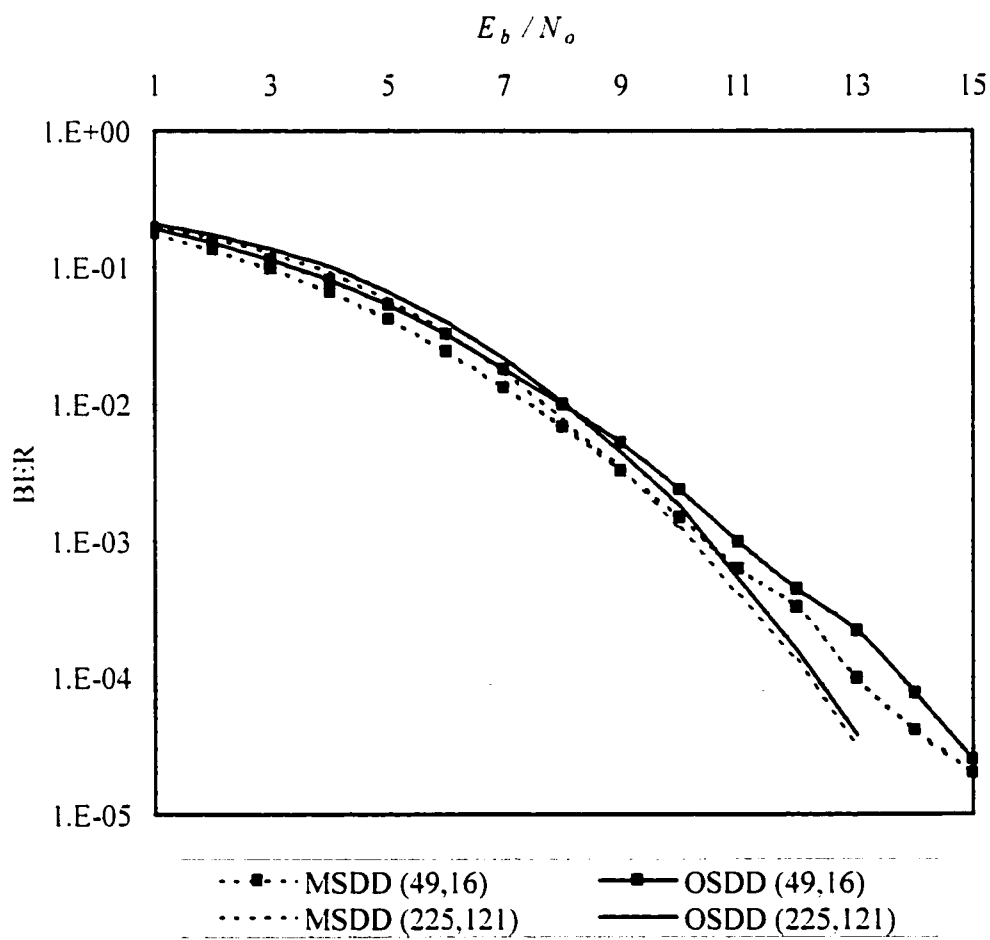


Figure 4.4 : Soft Decision Decoder Performance over Fading Channel of 0.1 Fade Rate

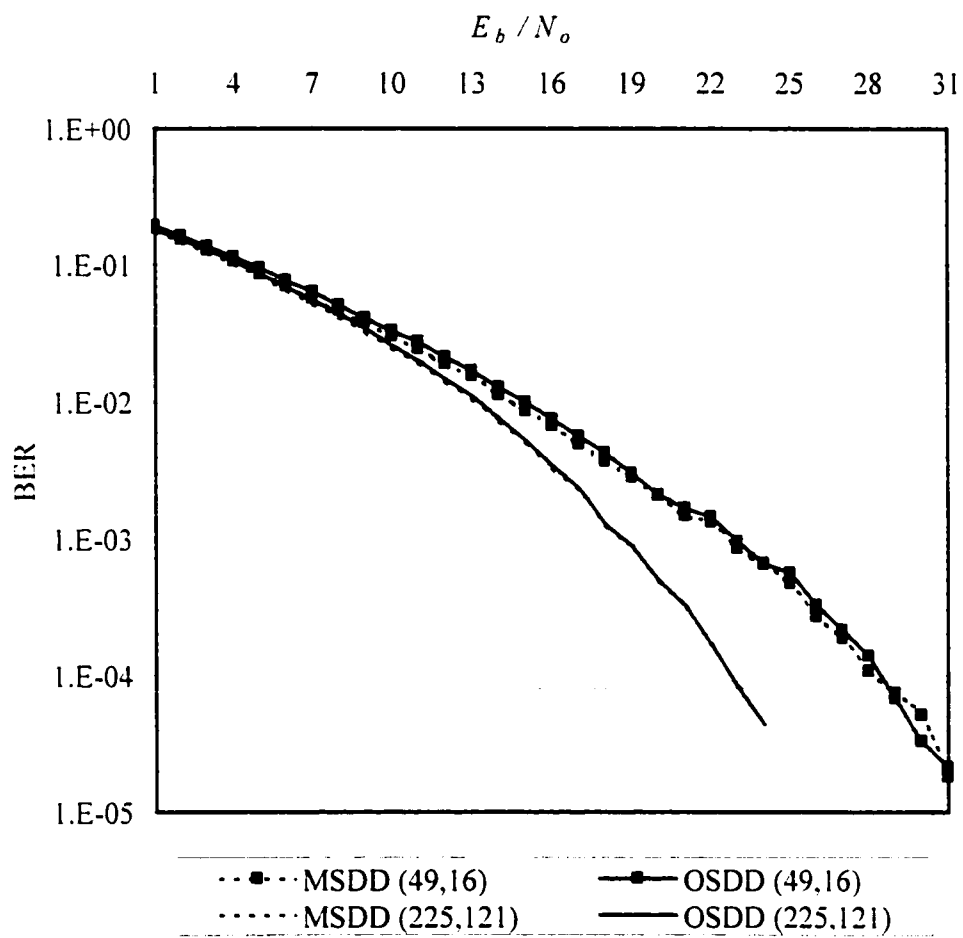


Figure 4.5 : Soft Decision Decoder Performance over Fading Channel of 0.001 Fade Rate

4.3 Erasure Decoding

The simplest form of soft decision decoder discussed earlier, uses erasure to indicate the reception of a signal whose corresponding symbol value is in doubt. Generally, if a demodulator makes a symbol error, two parameters are needed to correct that error, the location and the correct symbol value. In the case of binary symbols, only the error location is needed for correction. However, in the erasure binary correction, all what is needed is the symbol value because symbol location is known. So, decoding of erased codewords can be simpler than error correcting.

Error control codes can be used to correct erasures or to correct errors and erasures simultaneously. For a code having a minimum distance d_{min} , any pattern of ρ or fewer erasures can be corrected if $d_{min} \geq \rho + 1$. Moreover, any pattern of α errors and γ erasures can be corrected simultaneously if $d_{min} \geq 2\alpha + \gamma + 1$. Simultaneous erasure correction and error correction can be decoded in the following way [20].

1. Given a received word r , place zeros in all erased positions and decode normally. Label the resulting code word c_0 .
2. Now place ones in all erased positions and decode normally. Label the resulting code word c_1 .
3. Compare c_0 and c_1 to r after placing zeros and ones, respectively, selecting as the final decoded output the code word that is closest in hamming distance to r , after placing zeros and ones, respectively.

It is simple to show that this mechanism works. We first assume that the number of errors and erasures caused by the channel satisfies the constraint $d_{min} > 2\alpha + \gamma$ for a pattern of α errors and γ erasures simultaneously. If we assign zeros to the γ -erased positions, we generate α_0 errors. Making the total number of errors at the input to the decoder equal to $(\alpha + \alpha_0)$. When we assign ones to the erased positions, we end up with $(\alpha + \alpha_1) = (\alpha + \gamma - \alpha_0)$ errors. Either α_0 or $(\gamma - \alpha_0)$ must be less than or equal to $\gamma/2$. It follows that, in at least one of the decoding operations, the total number of errors α_t will satisfy $2\alpha_t \leq 2(\alpha + \gamma/2) < d_{min}$. So, at least one of the decoding operations yields the correct code word.

It should be noted that the speed of the errors-and-erasures decoder is half that of the errors-only decoder (or it has twice the hardware). This is because now two standard decoding operations must be performed for each received word containing erasures.

It is interesting to note that one can correct twice as many erasures as one can correct errors. This can be explained intuitively by noting that we have more information about the erasures to begin with. We know exactly where the erasures are, but we have no idea as to the location of the errors.

Previously in Chapter 2, an algorithm was introduced that was capable of reaching the theoretical minimum distance of the product code by determining which coordinates to erase. That algorithm is depicted in Figure 4.6. We refer to that algorithm by the Ordinary Erasure Decoding (OED) algorithm. To see how this algorithm works, consider once more the four-error pattern that was introduced in Figure 3.2. Assume single error correcting component codes. After decoding all rows, two more errors will be introduced in the two rows having two errors. For column decoding, if a correction happened, an

error is introduced in a row that was decoded without correction. So, it means that there were three errors in that row in addition to the two errors presumably corrected in the other two rows. This makes a total of 5 errors which exceeds the error correcting capability, 4, of the product code. So, column decoding is not accepted and the two rows corresponding to where a decoding was done are erased. Now, the column decoder will successfully fill the erasures.

The OED algorithm, Figure 4.6, for binary symbols can be improved. The OED algorithm erases whole rows or columns, which means that some good information is erased. We are presenting a simple modified technique that erase particular symbols instead of complete rows or columns. The algorithm is explained in Figure 4.7.

Considering a received codeword A of soft symbols with the weight of each symbol, W_{ij} , before demodulation. Then, the following steps are made in sequence.

- 1) For each row, erase the worst two symbols depending on their weights and demodulate the other symbols yielding A_E .
- 2) Erasure decoding for A_E is done for each row yielding A_{ER} .
- 3) Decode A_{ER} column-wise hardly.
- 4) Decoding codeword A finished.

We refer to the modified algorithm as the Modified Erasure Decoding (MED) algorithm.

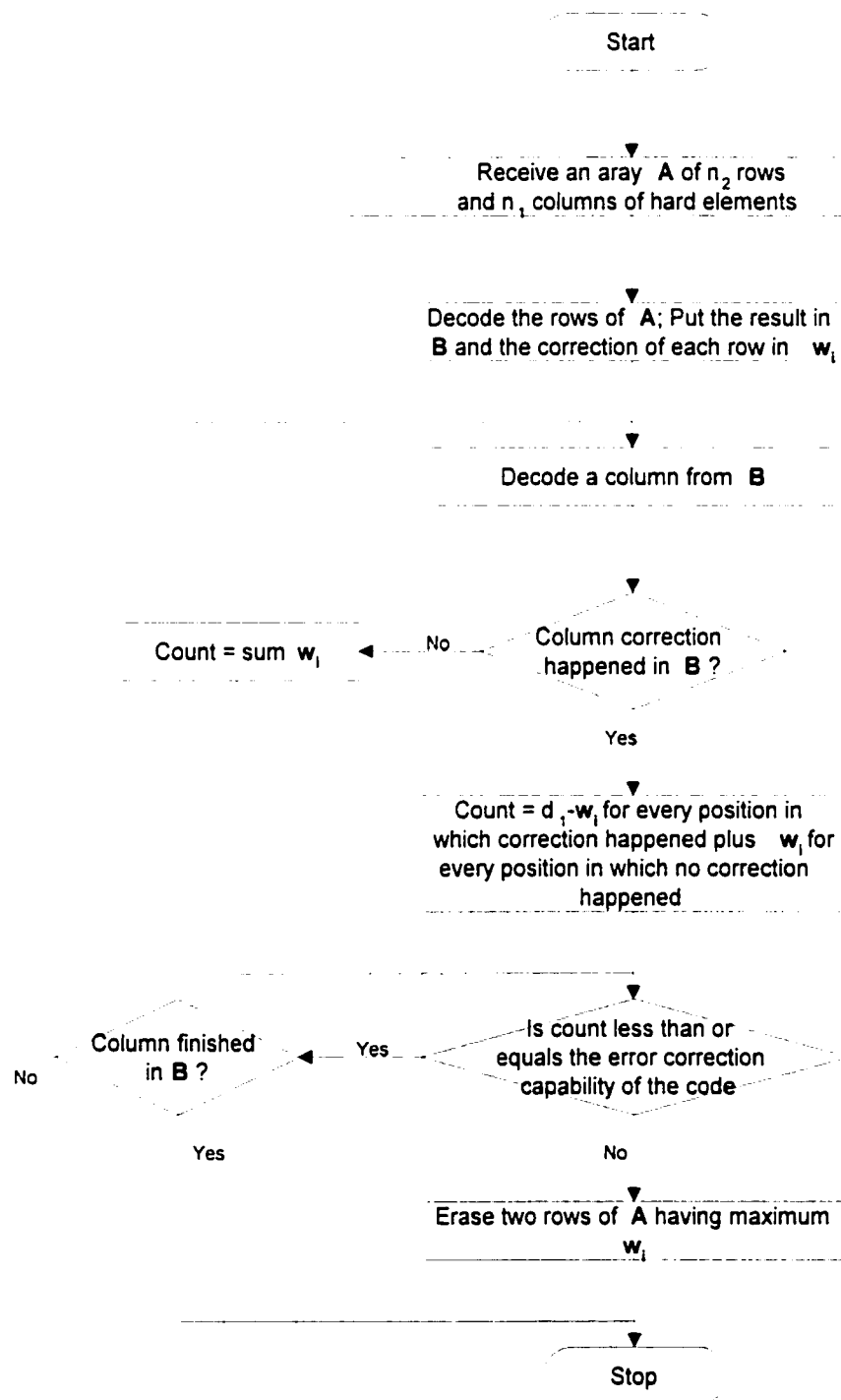


Figure 4.6 : The OED Algorithm

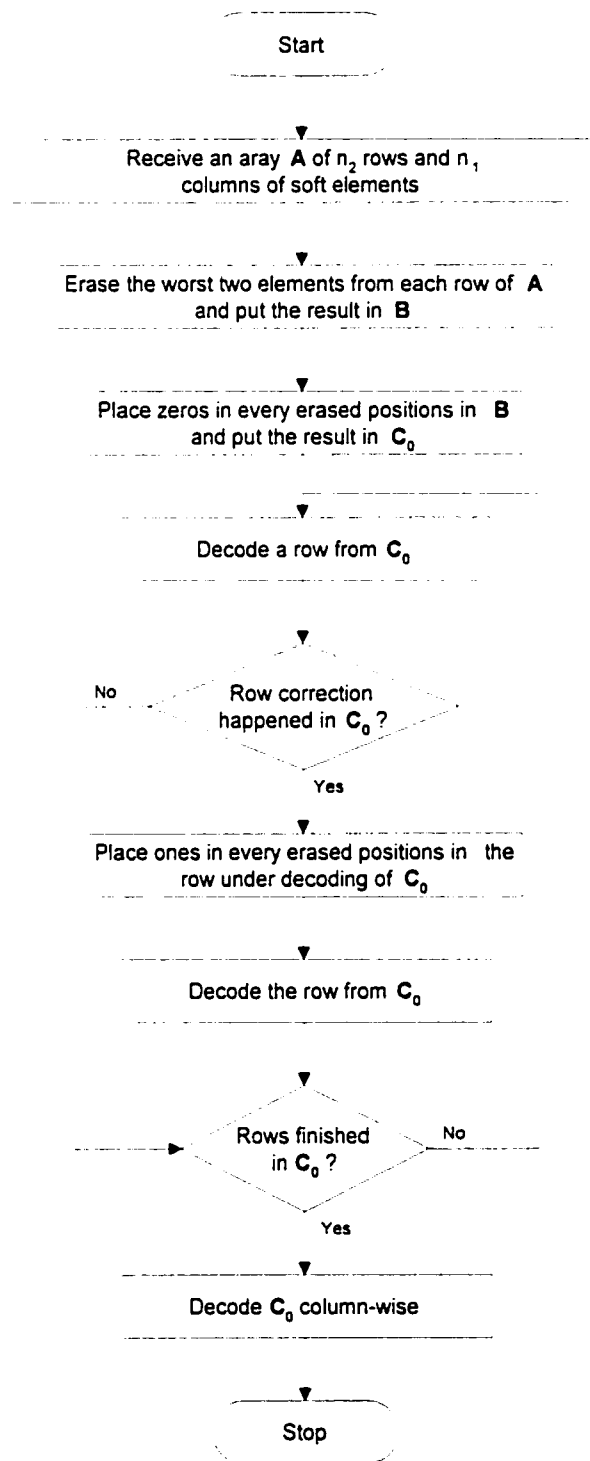


Figure 4.7 : The MED Algorithm

The two algorithms were tested over AWGN and fading channels. Figure 4.8 shows that the MED algorithm has more than one-half dB gain over the OED algorithm at a BER of 10^{-5} .

Figure 4.9 and Figure 4.10 show the performance of the two algorithms over fading channels of 0.1 and 0.001 fade rate respectively. It can be observed that as the fade rate increases, the performance of the two algorithms improves. The difference between the two algorithms is slightly big as the fade rate decreases. Also, as the length of the product code increases, the performance improves especially at lower fade rate.

It can be seen from Figure 4.9 that for big fade rate, less correlation, it is not good to delete whole rows. So, the performance of the OED is bad and it is worse for the product code of higher length. However, as the fade rate decreases, the product code of higher length is better since it randomizes the errors more.

Regarding the length of the code, the curves indicate that the strength of the code is seen only after an E_b/N_0 threshold has been exceeded. For values of E_b/N_0 less than the threshold, the coding manifests itself only as overhead bits resulting in reduced energy per bit. Before the threshold is exceeded, the redundant bits are simply excess load without the ability to improve performance. Once the threshold is exceeded, the performance improvement of the code is more than the compensation for the reduction in energy per coded bit.

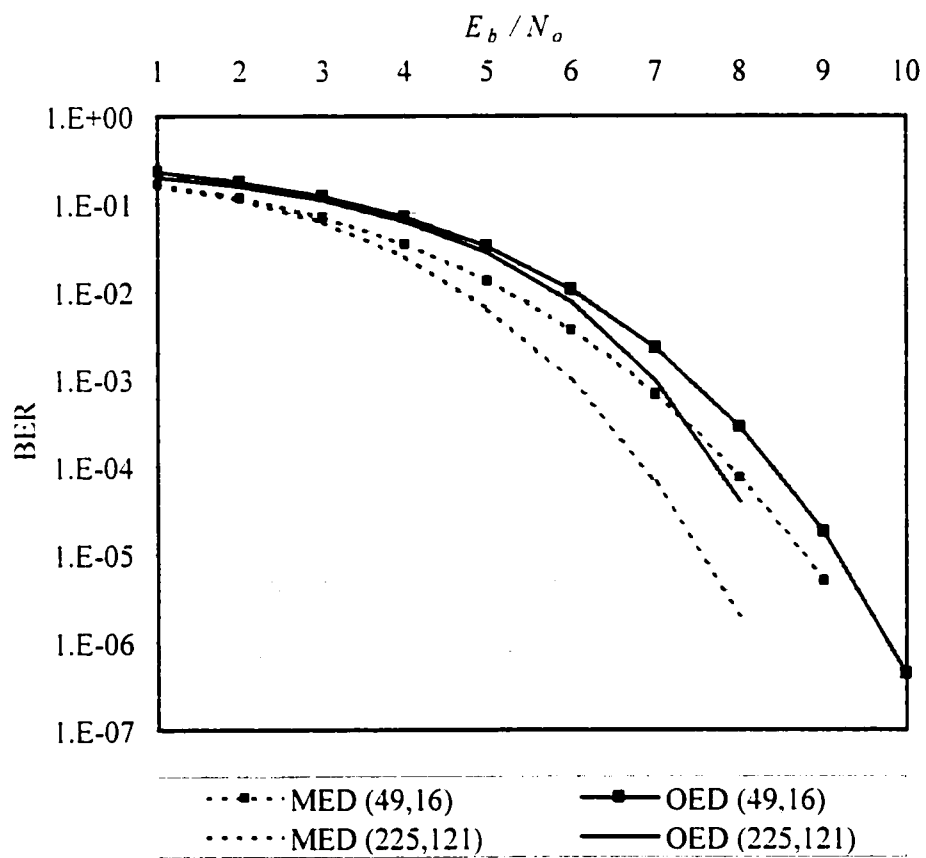


Figure 4.8 : Erasure Decoder Performance over AWGN Channel

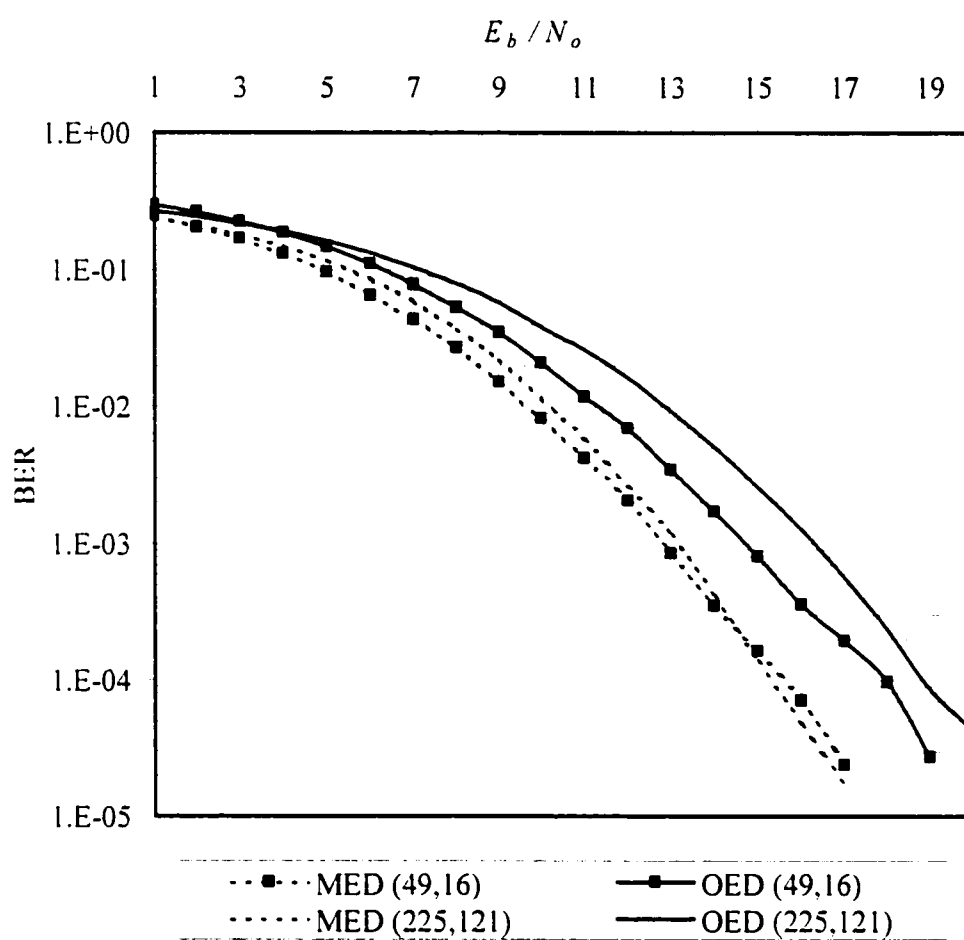


Figure 4.9 : Erasure Decoder Performance over Fading Channel of 0.1 Fade Rate

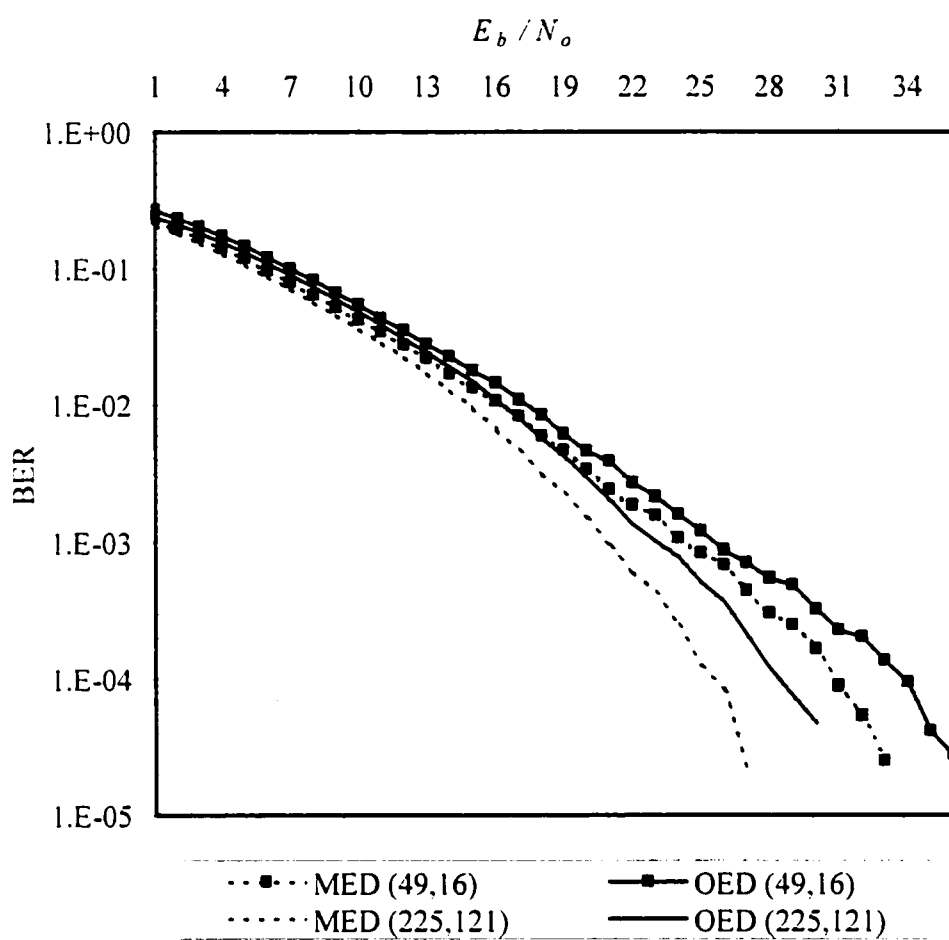


Figure 4.10 : Erasure Decoder Performance over Fading Channel of 0.001 Fade Rate

4.4 Performance and Complexity Comparison

Previously, every presented modified algorithm was compared to the corresponding ordinary one. In this chapter, the modified algorithms will be compared to each other. Also, the complexity of the modified algorithms will be evaluated to have a measure of the cost of the modifications giving the better performances.

Figure 4.11 and Figure 4.12 show the comparison among the six algorithms over AWGN channel. It is important to notice that the MHDD algorithm is better than the OED algorithm by 0.6 dB at BER of 10^{-5} . Also, It is better by about 1 dB as the length of the product code increases. Our particular interest in comparing these two algorithms comes from the fact that both algorithms achieve the theoretical error correction capability of the product code.

Figure 4.13 and Figure 4.14 show the comparison among the six algorithms over fading channel of 0.1 fade rate. Again, it can be seen that MHDD algorithm is better than that of the OED algorithm by about 1.8 dB but worse than that of the MED algorithm by 0.8 dB at a BER of 10^{-4} . As the length of the product code increases, the difference between MHDD algorithm and the OED algorithm becomes more significant while the MHDD algorithm is very comparable to the MED algorithm.

Figure 4.15 and Figure 4.16 show the comparison among the six algorithms over fading channel of 0.001 fade rate. Again, it can be seen that the MHDD algorithm is better than the OED algorithm by about 3 dB at a BER of 10^{-4} and it is comparable to the MED algorithm. As the length of the product code increases, the difference between the

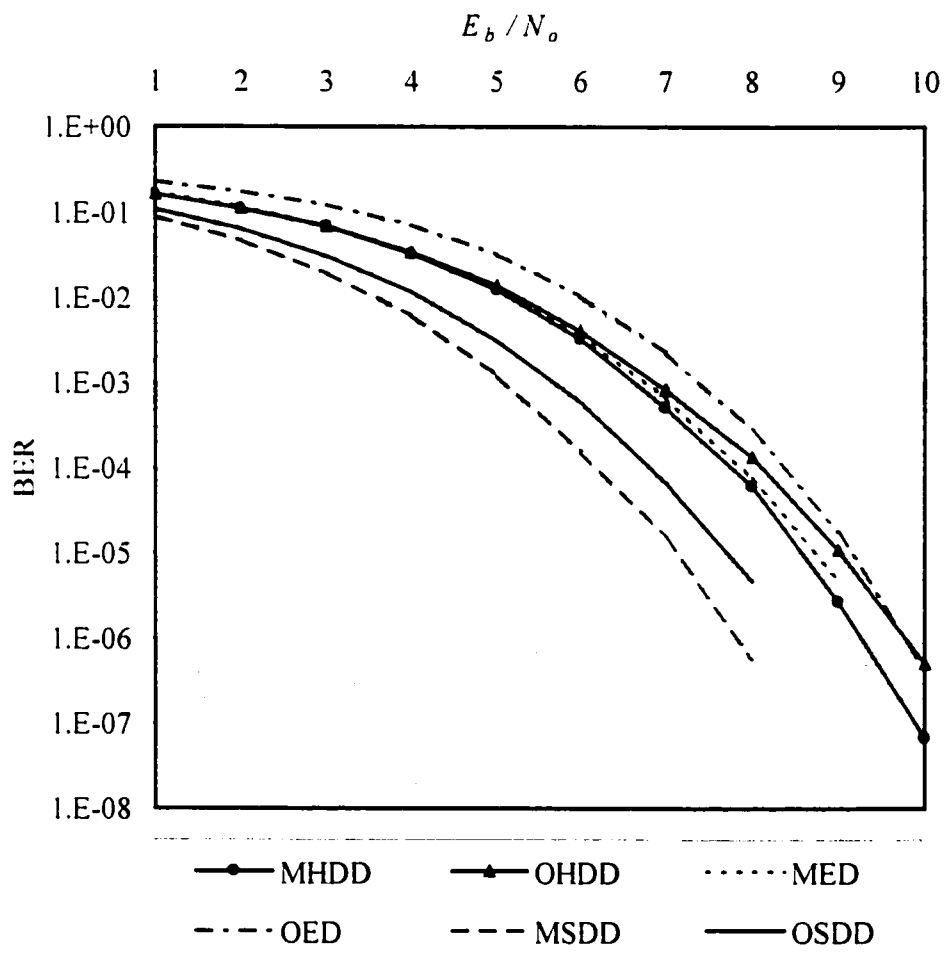


Figure 4.11 : Comparison of Different Decoding Algorithms of (49,16) Product Code Over AWGN Channel

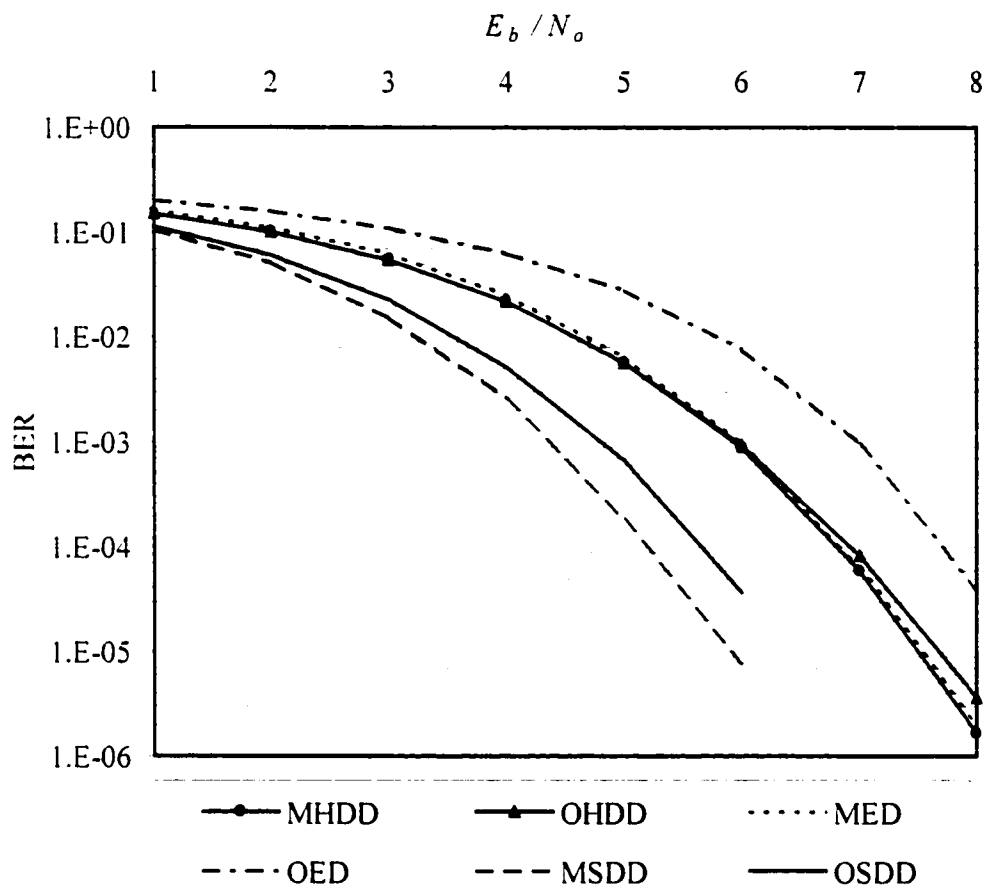


Figure 4.12 : Comparison of Different Decoding Algorithms of (225,121) Product Code Over AWGN

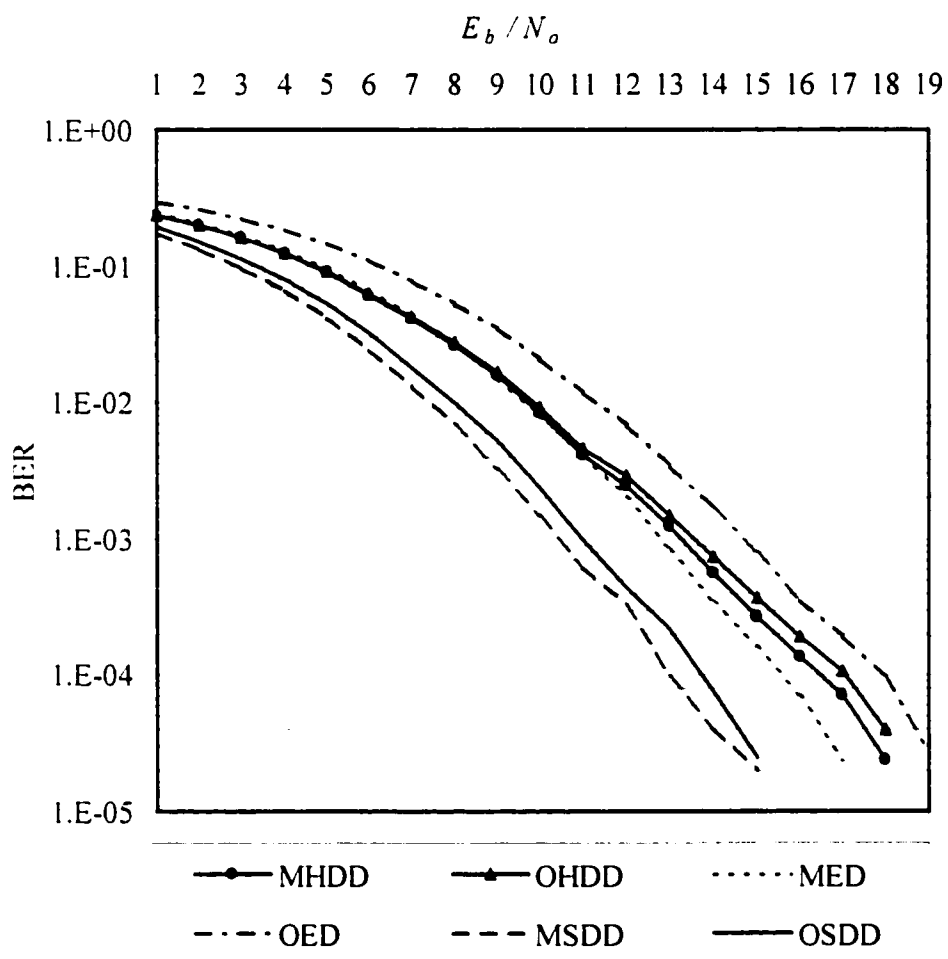


Figure 4.13 : Comparison of Different Decoding Algorithms of (49,16) Product Code Over Fading Channel of 0.1 Fade Rate

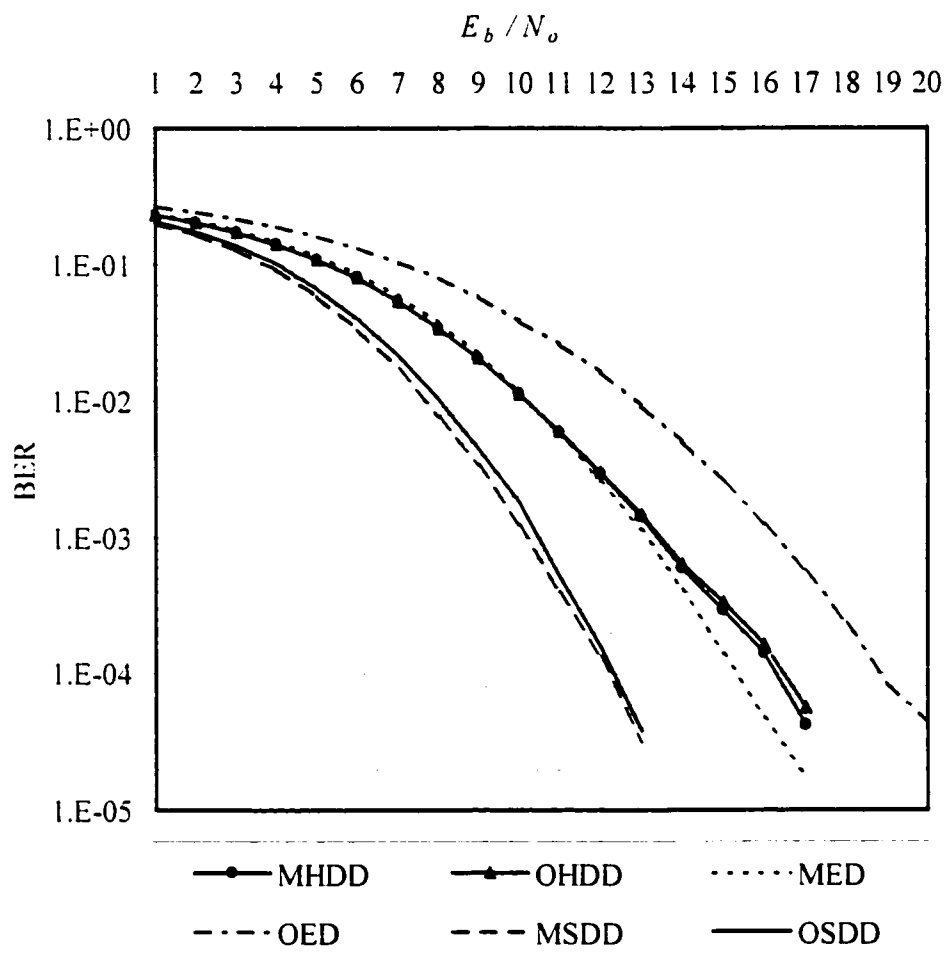


Figure 4.14 : Comparison of Different Decoding Algorithms of (225,121) Product Code Over Fading Channel of 0.1 Fade Rate

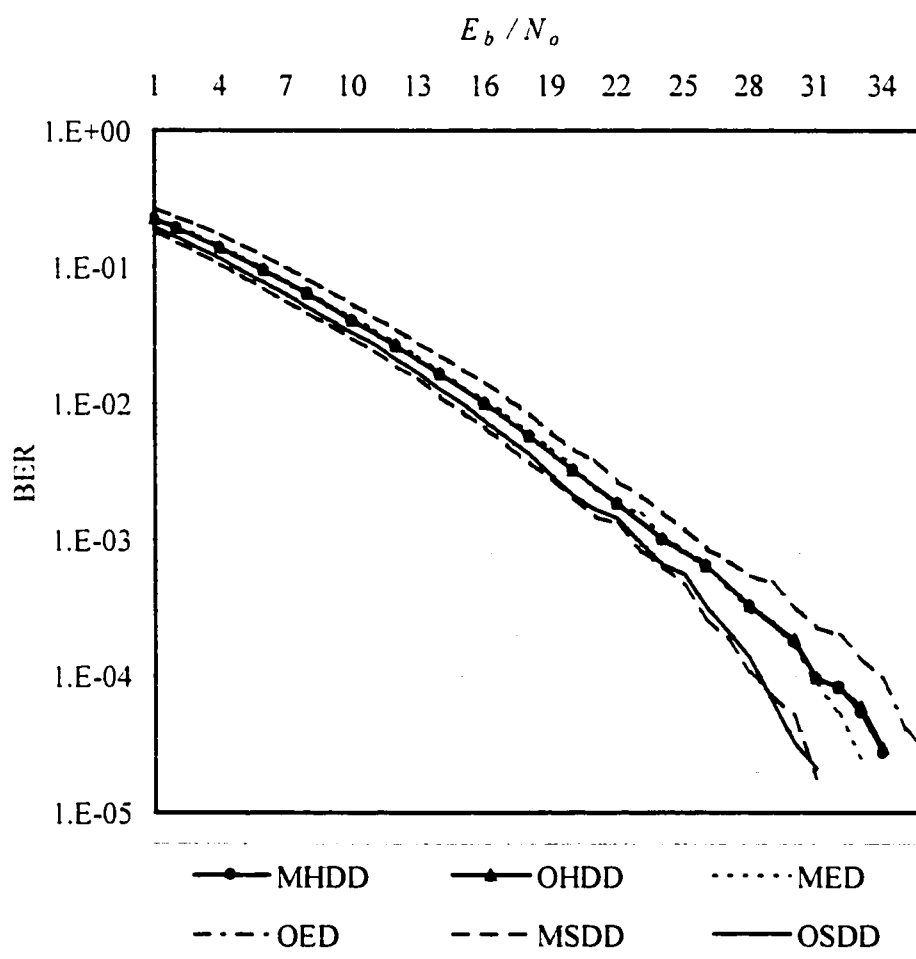


Figure 4.15 : Comparison of Different Decoding Algorithms of (49,16) Product Code Over Fading Channel of 0.001 Fade Rate

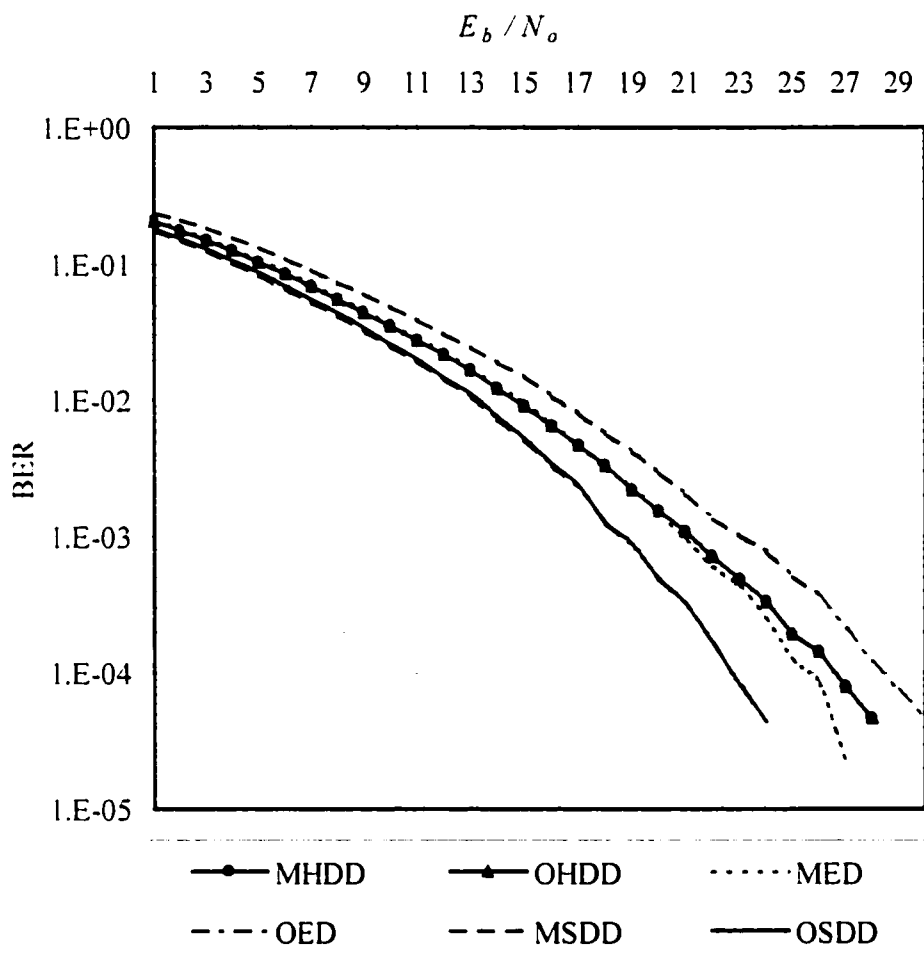


Figure 4.16 : Comparison of Different Decoding Algorithms of (225,121) Product Code Over Fading Channel of 0.001 Fade Rate

MHDD algorithm and the OED algorithm becomes 2 dB at a BER of 10^{-4} and the MHDD algorithm is comparable to the MED algorithm.

Now, the complexity analysis in terms of the time needed for each algorithm to be executed will be compared. The time is measured and compared for the (49, 16) product code for each algorithm. Ten millions arrays (blocks) are processed in a 200 MHz computer to find the time needed for each algorithm. Table 4.1 shows the time ratios of each algorithm to all the others.

The ratios of the modified algorithms to their counter parts of the ordinary ones are seen in Table 4.1 in bold face. The ratio of the MHDD algorithm to the OHDD algorithm is about five, which may appear a big ratio. Although no real complexity analysis is made in the literature, one can observe and deduce from the wording in the literature that this is a reasonable ratio. This claim can be further enhanced by noting that the ratio of the OED algorithm to the OHDD algorithm is about 8.6 although the OED algorithm is only capable of reaching the guaranteed error correction capability. Note that the ratio of the OED algorithm to MHDD algorithm is about 1.7. If the MHDD algorithm is processed just to reach the guaranteed error correction capability, the ratio of the MHDD algorithm to the OHDD algorithm is about 2.2 and the ratio of the OED algorithm to the MHDD algorithm is about 3.9. Finally, it can be seen from Table 4.1 also that the ratio of the MED algorithm to the OED algorithm is about 2.75 and the ratio of the MSDD algorithm to the OSDD algorithm is about 2.27. Those two ratios, I think, are very reasonable.

As a last note, we may add that with today's advancement in technology and process speed, the increased time can be easily absorbed.


	OHDD	MHDD	OED	MED	OSDD	MSDD
OHDD	—	0.198	0.116	0.042	0.02	0.009
MHDD	5.043	—	0.585	0.2125	0.1027	0.0453
OED	8.619	1.71	—	0.363	0.1755	0.0774
MED	23.73	4.705	2.753	—	0.483	0.2132
OSDD	49.1	9.735	5.697	2.069	—	0.441
MSDD	111.3	22.07	12.915	4.69	2.267	—

Table 4.1 : Timing Ratios of the Algorithms

CHAPTER 5

SUMMARY, CONCLUSIONS, AND SUGGESTIONS FOR FUTURE WORK

5.1 Introduction

This thesis adds a good contribution towards finding algorithms that will improve the performance of the product code without a significant increase in decoding complexity. The performance of the product code was studied and analyzed over AWGN and fading channels. In the following section a summary and conclusions of our findings are presented. Suggestions for direct extensions of this work are listed in the last section.

5.2 Summary and Conclusions

Three algorithms were presented. Those algorithms were tested over AWGN and fading channels. The first algorithm, the MHDD algorithm, modifies the basic OHDD algorithm of product codes. The MHDD algorithm reaches the actual theoretical performance guaranteed by the minimum distance of the product codes using Hamming codes as component codes. We showed that the *SNR* of the MHDD algorithm is better than that of the basic OHDD algorithm and of that of the basic OED algorithm by about 0.6 dB at BER of 10^{-5} .

The MSDD algorithm modifies the OSDD algorithm. It performs better than the OSDD algorithm by 0.6 dB at BER of 10^{-5} . The MED algorithm performs better than that of the basic OED algorithm by about 0.6 dB at BER of 10^{-5} , as well. Although the OED algorithm reaches the actual capability of the product code given by its minimum distance, the performance measured by the BER is in general worse than that of the OHDD algorithm. The MHDD algorithm is better than the MED algorithm in terms of BER performance but the difference between them is not very significant.

We saw that the complexity of the modified algorithms is in general larger than that of the ordinary algorithms but the increase is not too big. Actually, the complexity of the MHDD algorithm is less than that of the OED algorithm and it does perform better. Both algorithms reach the actual capability of the product codes. The MSDD algorithm is the most complex in terms of processing timing

We saw that generally as the fade rate increases, the performance improves because for fast fading the errors become more randomized. Diagonal interleaver has no effect over

AWGN channel. It helps over fading channel specially as the fade rate decreases. The ordered decoding method that was suggested in [10] helps over fading channel specially as the fade rate decreases but it has no effect over AWGN channel and over fading channel of high fade rate.

It is known that the performance of a code improves if there is more randomness in the structure. This randomness comes from interleaver which is inherent in the product codes. Diagonal interleaver gives more randomness than what ordinary interleaver does because it randomizes burst errors in both dimensions, rows and columns.

Our simulation results showed that Equation 3.3 is a very good approximation of the BER.

Generally, as the length of the product codes increases, the performance should improve since the performance improvement of the code is more than the compensation for the reduction in energy per coded bit. For the ordinary erasure decoder over fading channel, the performance degrades as the length of the product code increases. This is because as the length increases, a lot of good information is lost when whole rows are erased.

By extending the MHDD to three dimensions, the extended MHDD algorithm also outperforms the extended OHDD algorithm. The extended MHDD algorithm reaches the theoretical error correcting capability of 13 while the extended OHDD algorithm reaches an error correction capability of 7 only.

5.3 Suggestions for Future Work

Future work may consider the following extensions:

- Improving the performance of three dimensional product codes.
- Applying different and more powerful erasure techniques based on the Galois Field arithmetic.
- Applying codes other than Hamming codes, as well as choosing different codes for each dimension.
- Trying finding algorithms for other codes different from single error correcting codes, investigating their performance, and comparing the performance with the actual capability of such codes.
- Trying other soft decision decoding in which more than one iteration can be done. One can then attempt answering some important questions like, is there a limit on the number of iteration and is there a relation between the dimension of product codes and the number of iteration?

Nomenclature

Abbreviations

AWGN	Additive White Gaussian Noise
BER	Bit Error Rate
dB	deciBel; Decibel
MAP	Maximum A posteriori Probability
SNR	Signal to Noise Ratio
E_b/N_0	Signal to Noise Ratio
TC	Turbo Code
PC	Product Code
<i>rms</i>	<i>root mean square</i>
ISI	Intersymbol Interference
ARQ	Automatic Repeat Request
FEC	Forward Error Control
BPSK	Binary Phase Shift Keying
BSC	Binary Symmetric Channel
CSI	Channel State Information
MLSE	Maximum Likelihood Sequence Estimation
MHDD	Modified Hard Decision Decoding
MSDD	Modified Soft Decision Decoding
MED	Modified Erasure Decoding
GF	Galois Field

Symbols

n	Codeword length
t	Number of errors; Theoretical error correction capability
t_g	Guaranteed error correction capability
N_0	Noise spectral density
σ^2	Variance
τ_{rms}	Root mean square delay spread
B_c	Coherence bandwidth
T_c	Coherence time
f_D	Doppler spread
$f_D T$	Fade rate
T	Symbol duration
L	Repetition number of times
k	Information bits
R_c	Code rate
M	All possible codewords
d_{min}	Hamming minimum distance
ϵ	Crossover probability of a BSC
J	Number of rows in a matrix interleaver (span)
K	Number of columns in a matrix interleaver (depth)
K	Information bits for cascaded Hamming codes
N	Codeword length for cascaded Hamming codes

Δ	Allowable delay
b	Burst error correction capability
Q	Alphabet size
E	Energy per codeword
E_c	Energy per one bit in the codeword
E_b	Energy per information bits
y_j	The j th sampled output of a matched filter
v_j	The j th AWGN output of a matched filter of zero mean
c_{ij}	The bit in the j th position of the i th codeword
α	Number of errors for erasure decoding
γ	Number of erasures
w_i	Number of corrections happened per row
e	Number of rows erased
P_{block}	Probability of block error rate
P_{bit}	Probability of bit error rate
A_i	Number of codewords of weight i
B_i	Number of uncorrectable error patterns of i errors
$Q(x)$	Complementary error function
A	Received codeword
A_R	Decoded codeword row-wise
A_C	Decoded codeword column-wise
A_{RC}	Decoded codeword row-column-wise
$A_{R,i}$	Rows in which corrections were done; Decoded row # i
$A_{R,(j)}$	Corrections locations for rows

$\{A_{R,i}\}$	Total number of corrections for rows
$A_{C,j}$	Columns in which corrections were done; Decoded column # j
$A_{C,j(i)}$	Corrections locations for columns
$\{A_{C,j}\}$	Total number of corrections for columns
$A_{C,i,r}$	Corrections locations in columns corresponding to rows corrections
$\{A_{C,i,r}\}$	Total correction locations in columns corresponding to rows corrections
$A_{RC,j(i)}$	Corrections locations for columns after row decoding
\forall_i	Number of corrections, from columns decoding, happened at each row
$W_{R,i}$	Weight of decoded row # i
$W_{C,j}$	Weight of decoded column # j
$W_{i,j}$	Weight of symbol located at row i and column j
A_E	Codeword with erased positions
A_{ER}	Erasur decoding row-wise

Bibliography

- [1] Sklar B., **Digital Communications: Fundamentals and Applications**, New Jersey: *Prentice Hall, Inc.* 1988.
- [2] Proakis J., **Digital Communications**, 2nd Ed *McGraw-Hill Inc.* 1989.
- [3] Rappaport, Theodore, **Wireless Communications Principles & Practice**, New Jersey: *Prentice Hall, Inc.* 1996.
- [4] Adinoyi, Abdulkareem B., **Performance Evaluation of I-Q Trellis Codes Over Frequency-Selective Fading Channels**, Thesis, October 1998.
- [5] Kousa, Maan A., **Adaptive Error-Control Systems for Hybrid ARQ Schemes**, Thesis, June 1988.
- [6] Wilson, Stephen, **Digital Modulation and Coding**, New Jersey: *Prentice Hall*. 1999.
- [7] Pyndiah, Ramesh, Pierre Combettes, and Patrick Adde, "A Very Low Complexity Block Turbo Decoder for Product Codes," *IEEE, communications*, 1996, pp. 101-105.
- [8] Pyndiah, Ramesh Mahendra, "Near-Optimum Decoding of Product Codes: Block Turbo Codes," *IEEE, communications*, 1998, pp. 1003-1010.

- [9] Henkel, W. and Chung, H. Y., "New Filling Procedure to Reduce Delay of Burst-Error Correcting Array Codes," *Electronic Letters*, vol. 30, no. 6, pp. 465-466, Mar. 1994.
- [10] Yi. Chaehag and Lee, Jae Hong, "Interleaving and Decoding Scheme for a Product Code for a Mobile Data Communication," *IEEE trans. on comm.*, vol. 45, no. 2, pp. 144-147, Feb. 1997.
- [11] Sweeney, P., **Error Control Coding: an Introduction**, UK: *Prentice Hall International Ltd.* 1991.
- [12] Rankin, David and Gulliver, T. Aaron, "Asymptotic Performance of Product Codes," *IEEE, communications*, 1999, pp. 431-435.
- [13] Lin. Shu and Costello, Daniel, **Error Control Coding: Fundamentals and Applications**, New Jersey: *Prentice Hall, Inc.* 1970.
- [14] Rhee, Man Young, **Error-Correcting Coding Theory**, USA, *McGraw-Hill Inc.* 1989.
- [15] Shahri, H. and K. K. Tzeng, "On Error-and-Erasures Decoding of Cyclic Codes," *IEEE trans. on Info. Theo.*, vol. 38, no. 2, pp. 489-496, Mar. 1992.
- [16] Yuan, Dongfeng and Zhang, Lijun, "On Performance of Product Codes Employed in Some Typical Rayleigh Fading Channels," *IEEE, telecommunications*, 1998, pp. 28-30.
- [17] Pyndiah, R. et al, "Near-Optimum Decoding of Product Codes: Block Turbo Codes," *IEEE, communications*, 1994, pp. 339-343.

- [18] Rankin, David and Gulliver, T. Aaron, "Randomly Interleaved SPC Product Codes," *IEEE, communications*, 2000.
- [19] Sweeney, P., Wesemeyer, S., and Burgess, D. "A Multidimensional Block Coding Scheme with Iterative Decoding," *IEE, UK: Savoy Place*. 1999.
- [20] Wicker, Stephen B., **Error Control Systems for Digital Communication and Storage**, New Jersey: *Prentice Hall*. 1994.
- [21] Rajpal, Sandeep and Shu Lin, "Product Coded Modulation," *IEEE, communications*, 1993, pp. 7-11.