BIT FLIPPING DECODING FOR BINARY PRODUCT CODES

Sibonginkosi Ntuli

A DISSERTATION SUBMITTED TO THE FACULTY OF ENGINEERING AND THE BUILT ENVIRONMENT, AT THE UNIVERSITY OF THE WITWATERSRAND, IN FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE IN ENGINEERING

September 2013

Declaration

I declare that this dissertation is my own, unaided work, except were otherwise acknowledged. It is submitted for the degree of Master of Science in Engineering at the University of the Witwatersrand, Johannesburg, South Africa. It has not previously been submitted for any degree or examination at any other university.

Candidate Name :…….....

Candidate Signature :….....

Date :(Day)…… (Month)…… (Year)……

Abstract

Error control coding has been used to mitigate the impact of noise on the wireless channel. Today, wireless communication systems have in their design Forward Error Correction (FEC) techniques to help reduce the amount of retransmitted data. When designing a coding scheme, three challenges need to be addressed, the error correcting capability of the code, the decoding complexity of the code and the delay introduced by the coding scheme. While it is easy to design coding schemes with a large error correcting capability, it is a challenge finding decoding algorithms for these coding schemes. Generally increasing the length of a block code increases its error correcting capability and its decoding complexity.

Product codes have been identified as a means to increase the block length of simpler codes, yet keep their decoding complexity low. Bit flipping decoding has been identified as simple to implement decoding algorithm. Research has generally been focused on improving bit flipping decoding for Low Density Parity Check codes. In this study we develop a new decoding algorithm based on syndrome checking and bit flipping to use for binary product codes, to address the major challenge of coding systems, i.e., developing codes with a large error correcting capability yet have a low decoding complexity. Simulated results show that the proposed decoding algorithm outperforms the conventional decoding algorithm proposed by P. Elias in BER and more significantly in WER performance. The algorithm offers comparable complexity to the conventional algorithm in the Rayleigh fading channel.

Dedication

To Lindie and Lwethu, my family, for the support and care you have given me throughout my studies. Thanks for your words of encouragement and support.

Love you lots.

"UJehova ungumalusi wami, angiyikuswela"

Acknowledgements

My heartfelt gratitude to my supervisors, Dr. Jacobus J. Versfeld and Dr. Ling Cheng. It was inspiring to work with you. Thank you for the patience, guidance and enthusiasm throughout the project. I would also wish to thank Prof. Rex van Olst, for allowing me to join such an amazing group of researchers. Thanks to all my colleagues, Familua, Mpendulo thanks for all your assistance, your comments and suggestions have been a guiding hand throughout my research. It was a pleasure working with you guys. I would also like to thank my mum, dad, Thabo, Mgcini and Suku. Thanks for egging me on when there was need and for your patience with me when I was difficult. You always have been pillars in my life, and I hope one day you will afford me the opportunity to support you in your own endeavours. Lastly I would like to thank my sponsors Telkom SA, without whom this would have been impossible.

Table of Contents

List of figures

List of abbreviations

ACK: .... acknowledgement

ARQ: ....Automatic Repeat reQuest

AWGN:..... ..... additive white Gaussian noise

BCH:..... ..... Bose-Chaudhuri-Hocquenghem

BER: .... bit error rate

BF: .... bit flipping

BLER:..... .....block error rate

BPSK:..... .....binary phase-shift keying

FEC:..... .....foward error correction

LDPC:..... .....Low-density parity-check

MAP: ....maximum a posteriori

ML:..... .....maximum likelihood

MLG:..... .....majority logic

NAK: ....negative acknowledgement

SER:..... .....symbol error rate

SNR: .... signal-to-noise ratio

SPC:..... ..... single parity check

VA:..... ..... Viterbi algorithm

WER:..... ..... word error rate

XOR: ....exclusively OR

List of notations

ω	Attenuation factor
B	Bandwidth
n	Binary bit being transmitted in BPSK signal
T_b	Bit duration
R_T	Bit Transmission rate
u	Block of information
s_n	BPSK transmitted signal
$\boldsymbol{M}$	Candidate Error Matrix
C	Channel capacity
f_c	Carrier frequency
C	Code
R	Code rate
$\boldsymbol{Y}$	Column Parity Check Vector
$\mathbb{C}$	Complexity
$\deg(g(x))$	Degree of the polynomial, $g(x)$
τ	Elapsed time variable
E_b	Energy per bit
ε	Erasure
t	Error correcting capability
e	Error vector or sequence
$c(t)$	Filter response
$\boldsymbol{F}$	Flip matrix
δ	Fractional minimum distance
G	Generator matrix
$g(x)$	Generator polynomial
I_k	Identity matrix of size $k \times k$
G_{ri}	i^{th} row of the matrix G

s_i	i^{th} component of the matrix s
$m(x)$	Message polynomial
d_{min}	Minimum distance
v'	New vector or sequence
$n(t)$	Noise signal
k	Number of information bits in codeword
H	Parity check matrix
α	Primitive element of a BCH code
$r(t)$	Received signal
r	Received vector or sequence
$rem(x)$	Remainder from dividing two polynomials
$\boldsymbol{X}$	Row Parity Check Vector
$\frac{E_b}{N_o}$	Signal to noise ratio
s	Syndrome
t	Time
$c(\tau;t)$	Time-variant channel impulse response
n	Total number of codeword bits
$s(t)$	Transmitted signal
v	Unique codeword
$\mathbb{R}$	Vector with the indices of the rows with errors
$A(x)$	Weight enumerator polynomial

This page intentionally left blank

Chapter 1 - Introduction

1.1 Introduction

The wireless channel is one of the harshest communication channels. It poses a challenge to designers of communication systems, as little or no control can be exercised over the external noise sources and interfering signals interacting with the communication signal. Modern digital wireless communication systems use Forward Error Correction (FEC) as an option for mitigating the effect of noise encountered on the channel to make communication more reliable. Generally using long block codes increases the error correcting capability of a FEC system. Unfortunately, increasing the block length generally tends to also increase the decoding complexity of the FEC system. Most of the delay in a FEC system is attributed to the decoding complexity of the algorithm used.

Product codes have been used to increase the block length, thus increasing the error correcting capability of FEC systems, while keeping the decoding complexity low. The conventional decoding algorithm proposed by Elias [1], in 1954, has been shown to be an effective decoding algorithm despite its shortcomings. In this study we propose a new decoding algorithm for

product codes. The algorithm addresses the permanent error problem identified by Abramson in [2], that cannot be corrected by the conventional decoding algorithm. It is based on the bit flipping decoding algorithm first presented by Gallager [3] for Low-density parity-check (LDPC) codes. The bit flipping decoding algorithm has been identified to be a low complexity decoding algorithm. This ensures that the proposed decoding algorithm has a low complexity, thereby addressing the problem of complexity and delay introduced by decoding algorithms in modern communication systems.

1.2 Background

In recent years there has been an increase in research on wireless communication. People have moved away from using fixed line devices for communication, opting for the convenience of untethered devices like cellular phones, tablet computers and notebooks [4]. The emergence of digital wireless communication has been a major driving factor for the move to tetherless connectivity. Wireless communication has been readily adopted in developing nations where there is limited fixed line infrastructure.

The wireless channel is a harsh communication channel. Two major aspects of wireless communication make it challenging to use. The first of these is the phenomenon of fading. Secondly, unlike in the wired channel where the path between transmitter-receiver pair can be viewed as an isolated point-to-point link, in wireless communication, data is transmitted over the air. This subjects wireless communication to interference.

Fading is the time variation of the received signal power caused by changes in the wireless channel, or the path taken by the signal. In a fixed environment, for example, two fixed microwave towers communicating with line of sight, fading can be affected by changes in the atmospheric conditions, such as rainfall. In an environment where one of the receiving devices is moving relative to the other, the obstacles in the path between the two devices change over time, creating complex transmission effects [5]. This makes fading a challenging problem in mobile wireless network design. Fading can be classified as either fast or slow. Fast fading occurs in the urban environment where the built up environment causes rapid variations in signal strength. Slow fading occurs when the variations in signal strength occur when the mobile user covers

distances in excess of the wavelength. Fading can also be classified as flat fading or selective fading. Flat fading occurs when all the frequency components of the received signal fluctuate in the same proportions. Selective fading affects different spectral components of the radio signal unequally.

Cellular radio systems are subject to co-channel interference because cellular radio systems make use of frequency reuse. Frequency reuse also introduces adjacent channel interference. Adjacent channel interference arises when neighbouring cells use carrier frequencies that are adjacent to each other. Other forms of interference that affect wireless systems are multiple-access interference and narrowband interference (NBI). Multiple-access interference is resultant from multiple users accessing the same wireless network. Ultra-wide band (UWB) systems which operate at very low power spectral densities are affected by NBI [6].

Another factor that affects wireless communication is multipath propagation. Reflection, diffraction and scattering play a role in multipath propagation. Reflection of electromagnetic waves occurs when the transmitted signal encounters an object that is larger than its wavelength. Diffraction occurs at the edge of object that is larger than the wavelength of the signal and the signal cannot pass through the object. The signal is propagated in different directions with the edge acting as a source. If the object in the path of the signal is smaller than the wavelength of the signal, scattering occurs. The incoming signal is scattered by the object into several weaker outgoing signals [5]. For cellular networks, objects such as lamp posts, and traffic signs scatter the signals. Multipath propagation results in multiple copies of the signal arriving at the receiver's antenna. If the copies of the signal are out of phase, this can reduce the power of the signal making it difficult for the receiver to detect it.

Battery life is also factor in the development of mobile devices [7]. Devices connected to a wired network are usually powered by the electrical cables provided in the building. Wireless devices are generally meant to be portable and mobile and are typically powered by a battery [8]. Although processor and memory development have increased in the last couple of years, the development of batteries has been slower. According to [9], battery capacity doubles after 35 years. Though battery capacity has improved, it still lags behind in terms of development when compared to other parts of mobile devices [9]. Over the years the processing power and memory in mobile devices has increased, enabling mobile devices to offer more complex operations than

when they were first conceived. For example the first set of cellular phones offered voice communication only. The development of smart phones has seen the capabilities of cellular phones increased. Smart phones offer the user the ability to send and receive multimedia messages, browse the internet, and connect to various peripheral devices. The information sent by smart phones is mainly of the nature where integrity of the received information must be maintained. Corruption of the information by the channel requires the information to be re-sent thereby consuming more battery power. Users on the other hand require a longer battery life. To enable this, the mobile devices have to resend as little of the received corrupted information as possible.

Error control is used to mitigate the impact of noise on the wireless channel and improve the reliability of the channel [10]. Two error control coding strategies are used in wireless communication namely, error detection coding and error correction coding [11]. Error detection coding is a means where the presence of errors is detected in received vectors. Error detection coding is used in conjunction with schemes such as Automatic Repeat reQuest (ARQ) to control errors. In error detection coding a codeword, with redundant data added, is transmitted to the receiver. The receiver uses cyclic redundancy checks, syndromes, or other means to determine the presence of errors in the received vector. If there are any errors, ARQ or error correction can be used to obtain the correct codeword [12]. Error correction coding is a means by which the receiving device corrects the errors that are introduced by the communication channel. The ability of a code to correct errors is related to its ability to detect errors [13]. The receiver checks the received vector for the presence of errors. If the exact locations and number of errors are established, the errors are corrected and the vector is decoded into the original information that was encoded. If the receiving device cannot establish the exact positions of the errors, the received vector will be decoded incorrectly [14]. ARQ and forward error correction (FEC), two techniques used for controlling transmission errors, are covered in greater detail in Section 2.4.

1.3 Product codes

In the wireless environment, the costly resources of power and bandwidth make it a challenge to choose the appropriate error control scheme. Product codes provide a means to obtain long and powerful codes while keeping the decoding complex low, by using concatenation of simpler

constituent codes. They were first presented by Elias in [1]. A product code can be defined as a multidimensional array of linear block codes. A two dimensional product code can be represented as a set of matrices where each row is a codeword in one constituent code, and each column is a codeword in another constituent code. The decoding complexity of the product code can be kept low by decoding each row and column separately [15], [16], [17].

The inherent interleaving in product codes makes them attractive codes for wireless communication channels. Wireless communication channels suffer from noise and fading due to multipath propagation of the signal. Fading severely degrades the performance of data transmission in wireless communication systems [18]. It results in bursty errors in the received vector. Interleaving is used to spread the error bursts into random errors which can be corrected by forward error control codes [19], [20]. A block interleaver arranges the codewords to be transmitted into a number of rows. The bits are then transmitted column-wise. At the receiver the bits are re-blocked and decoded row-wise. Interleaving adds further delay to the decoding process in the communication system. Product codes have the proper structure to deal with burst errors without the need for extra interleaving. They have been used in digital versatile discs where fingerprints and scratches can cause bursty errors [21], [22], [23]. All error patterns where the burst errors are restricted to a number of rows less than half the minimum distance of the column code are correctable when decoding column-wise or burst errors that are restricted to a number of columns less than half the minimum distance of the row code are correctable row-wise [24].

The reasons why we decided to work on binary product codes can be summarized as follows:

- Fading in wireless communication systems introduces burst errors. Products codes offer good error correcting capability in wireless communication systems because of their ability to spread burst errors amongst the rows or columns when decoding. This is because they include interleaving as an in-built feature in their design.
- Most decoding algorithms used for product codes group the received vector into an array of row and column vectors. The decoding algorithms are still based on the conventional decoding algorithm of initially decoding the rows then using the information obtained from decoding the rows to decode the columns. Maximum likelihood decoding algorithms that attempt to decode the entire codeword at once are avoided because of the size of the block. Our aim is to develop a

new decoding algorithm that is capable of correcting all the errors in the received vector at once, error pattern permitting, while ensuring that the BER performance of our algorithm is comparable to the algorithm proposed by Elias in [1].

- Product codes have been generalized into other types of multilevel codes like concatenated codes, array codes and augmented codes. Solutions for decoding product codes are sometimes extended to concatenated codes and codes constructed on the basis of product codes and concatenated codes like those in [25]. We hope that the new decoding algorithm can also be modified and used to decode similar multilevel binary codes.
- Product codes can be used to improve the reliability of the wireless channel. In a noisy wireless channel, devices have to either increase their transmitting power or resend any of the corrupted data. Both methods impact negatively on the battery life of mobile devices [26]. FEC techniques suitable for wireless communication, such as product codes, can be used to improve the reliability of the channel, thereby extending the battery life of the mobile devices [27][28].

Generally, increasing the length of a code improves its error correcting capability. As a general rule, the more powerful a code is, the more difficult it is to decode [29]. To increase the block length of codes while keeping the decoding complexity low, Elias proposed using product codes. Elias proposed using a decoding algorithm that would decode the rows then the columns of the product code. The complexity of decoding would be a linear summation of the decoding complexities of the rows and columns. Most of the decoding algorithms that have been proposed for product codes have followed on the algorithm Elias proposed, only differing in the method used to decode the constituent codes. Few decoding algorithms have been proposed that perform one step decoding of product codes [29].

In 1962 Robert Gallager proposed Low-density parity-check codes. Gallager presented a simple hard decision decoding scheme for LDPC codes [2][29]. By checking the parities of each bit, he could determine which bit was most likely to be erroneous. If the number of parity fails of a bit exceeded a given threshold, the bit was deemed incorrect and flipped. This algorithm is referred to as the bit flipping decoding algorithm. Since Gallager presented the bit flipping algorithm, a lot of research has been done on it [31][32][33][34]. Researchers agree that the bit flipping algorithm is a simple decoding algorithm [35][36]. Bit flipping decoding has generally been confined to LDPC codes.

In binary product codes, each bit affects more than one syndrome, depending on the number of dimensions of the product code. Based on syndrome checking, it is possible to determine the possible error locations in the received vector, and use bit flipping for decoding binary product codes. Little research has been done on developing a bit flipping decoding algorithm for binary product codes other than single parity check product codes [37].

1.4 Problem statement

In this study we investigate how different error patterns affect the parities of two-dimensional binary product codes. After analysing the error patterns, *we aim to develop a decoding algorithm for binary product codes that is based on syndrome checking and bit flipping.*

Based on the structure of product codes, it should be possible for the bit flipping algorithm, in some cases, to correct all the errors in the received vector in one step. Bit flipping decoding is simple to implement. In our study we use the simplicity of bit flipping to develop our decoding algorithm. Turbo decoding used for product codes can achieve a low error rate close to Shannon's limit, but has a high decoding complexity [38].

Our study is restricted to two-dimensional product codes, as they are easier to comprehend. The results obtained from the two-dimensional product codes are scalable to multi-dimensional codes. The developed decoding algorithm is only implementable on binary product codes.

1.5 Scope and objectives of the study

This project aims to study the suitability of product codes for communication in the wireless communication channel by developing a decoding algorithm for binary product codes that is based on syndrome checking and bit flipping. The algorithm is specifically for binary product codes as they only have two symbols. The algorithm will be compared to the conventional decoding algorithm in the additive white Gaussian noise (AWGN) and Rayleigh fading channels. The AWGN channel is often used when developing coding systems. It offers a simple channel model on which to develop coding systems. Radio communication systems operating over a multipath environment, such as indoor wireless communication or mobile telecommunications,

are often modelled as a Rayleigh fading channel and Rician fading channel [39]. In this study, we restrict ourselves to the AWGN and Rayleigh fading channel. We also restrict ourselves to two dimensional product codes. Henceforth, any reference to product codes refers to two dimensional product codes unless otherwise stated. The objectives of the study can be summarized as follows:

- Study the pattern of the errors that are introduced by the AWGN and Rayleigh fading channel in a two-dimensional binary product code.
- Develop a decoding algorithm for binary product codes to be used in the wireless communication channel. The decoding algorithm should offer comparable error correcting capability to the conventional decoding algorithm by decoding errors patterns, like the permanent error pattern, that the conventional decoding algorithm fails to decode.
- Compare the developed algorithm with conventional decoding algorithm in the AWG channel to prove that the developed algorithm is comparable to the conventional decoding algorithm
- Compare the decoding complexity of the developed algorithm to that of the conventional decoding algorithm. Research on LDPC codes has shown that the bit flipping algorithm offers an effective trade-off between error performance and decoding complexity [35]. To reduce the complexity of the decoding, the decoding algorithm will be based on syndrome checking and bit flipping.
- Compare the performance of the developed algorithm to that of the conventional decoding algorithm in the Rayleigh fading channel. Since the developed algorithm will be used for wireless communication, a performance comparison of the bit flipping decoding algorithm and the conventional decoding algorithm will be done using a Rayleigh fading channel. This will give us an opportunity to see how the algorithm performs in a wireless channel and handles burst errors in comparison with the conventional algorithm.

1.6 Organization of thesis

Chapter 1 of the thesis introduces the problem investigated in this research. Section 1.2 gives background information on wireless communication, stating the problems encountered on the wireless communication, namely, interference and fading. Interference can either be inter-

channel or co-channel. The section also gives a brief introduction of error control techniques that are available for improving wireless communication. Section 1.3 briefly introduces product codes. Product codes were first presented by P. Elias. They allow building long block codes, while keeping their decoding complexity relatively low. The section also gives the reason why product codes were chosen for this study. Section 1.4 defines the problem statement. Product codes are decoded by iteratively decoding the component codes. By studying the error patterns, we devise a new decoding algorithm for product codes based on bit flipping. Section 1.5 states the scope and objectives of the project. The main objective of the study is to *develop a decoding algorithm for binary product codes that is based on syndrome checking and bit flipping*. The last section of the chapter describes the scope of the report.

Chapter 2 of the thesis gives background information on digital communication systems. It presents mathematical channel models that are used when studying digital communication. The first of these is the AWGN channel. The AWGN channel is used as a tractable channel to develop digital communication channels. Other channels are the Linear Filter channel and Rayleigh fading channel. The chapter presents two error control techniques that are used in wireless communication namely ARQ and FEC. Three different ARQ methods are presented. Section 2.5 defines block codes as a FEC technique. This section defines three types of block codes, namely, the Repetition codes, Hamming codes and Bose-Chaudhuri-Hocquenghem (BCH) codes and their decoding algorithms. The chapter also defines product codes, presenting the different decoding algorithms that are used for them. Section 2.7 presents the bit flipping decoding algorithm introduced by Gallager for decoding LDPC codes.

Chapter 3 of this thesis gives a survey of the literature on product codes and bit flipping. It starts by giving a brief history on the development of product codes. When product codes were first proposed by Elias, he proposed them with Hamming codes as constituent codes. He stated that other systematic codes like Golay codes could be used as constituent codes. Sections 3.3 and 3.4 provide a literature review on cyclic product codes, that were first presented by Burton and Weldon, and single parity check product codes, that were introduced by Bahl and Chien in 1971 respectively. Other codes in Chapter 3 are array codes and augmented codes. Chapter 3 also presents literature on the bit flipping decoding algorithm. Little work has been done on

improving the bit flipping decoding algorithm, because of its simplicity when it was first presented.

Chapter 4 presents the methodology section of the thesis. Section 4.2.2 introduces the different error patterns that influenced the development of our bit flipping decoding algorithm. Section 4.2.3 then presents the definitions of the Row Parity Check Matrix and the Column Parity Check Matrix, explaining how they are used to obtain the Candidate Error Matrix. Section 4.2.3 gives a detailed description of the new bit flipping algorithm for binary product codes. It shows how the algorithm uses two types of error patterns to converge the area of bits that are probably in error (the shadow area).

In Chapter 5 we present our results. The chapter gives a detailed description of the different error patterns stating under which channel conditions they are most likely to occur. Section 5.3 describes the different ways for measuring the performance of a coded communication system, namely the bit error rate (BER), block error rate (BLER) or Word error rate (WER), and the symbol error rate (SER). Chapter 5 also shows the BER and WER curves of the conventional and the bit flipping decoding algorithm. Lastly Chapter 5 gives a detailed description of partial syndrome decoding, and how it was used in conjunction with lookup tables to lower the decoding complexity of the bit flipping decoding algorithm.

Chapter 6 concludes the report, discussing how the objectives of the project were met. It also presents further work that can be done to improve the study and the developed algorithm.

Chapter 2 - Background Information

2.1 Introduction

In most communication, errors will occur. In some communication, the message has some inherent redundancy so the communication system can tolerate some of the errors that will occur [40]. For example in a voice communication system, the context of the conversation introduces the redundancy. Digital communication systems are designed with a method to recover from the errors that will occur from time to time during transmission.

Errors in data communication or storage systems come from different sources. The errors could be from random noise, impulse noise, channel fading or physical defects of the media. The communication system must be designed in a manner that curbs these errors. In this chapter we present the digital communication system by discussing the two ways that are used to mitigate the impact of noise. The two ways are used separately or can be combined as a hybrid system. The two methods, error detection coding and error correction coding are collectively known as error control coding. Both offer different advantages, hence in some cases a hybrid of the two is used. In error correction coding, the errors which may be introduced into the data by the

transmission through the communication channel can corrected based on the redundancy in the received data. In error detection coding, the errors introduced are detected [30].

2.2 Overview of a digital communication system

The objective of error control coding in digital communication systems is to detect the presence of errors in the received data, and in some cases correct the errors. Error control coding adds complexity to a communication system. It involves adding redundancy to the message to be transmitted. The redundancy is added in a manner such that the validity of the received message can be checked thereby detecting the presence of the errors [41].

A typical communication system has only three elements, the source device, the channel, and the destination device. The source device could either be a storage device or a device sending information across a network. The channel is the medium through which the information will be transmitted to the destination device.

Figure 2.1 shows the elements of a typical digital communication system. Error control coding is applied after the data is converted into a digital format by the source encoder. Though represented as separate steps in Figure 2.1, it is worth noting that usually the digital modulation and channel coding are designed together [41]. After modulation the data is transmitted on the channel and the steps are carried out in reverse order at the receiving side.

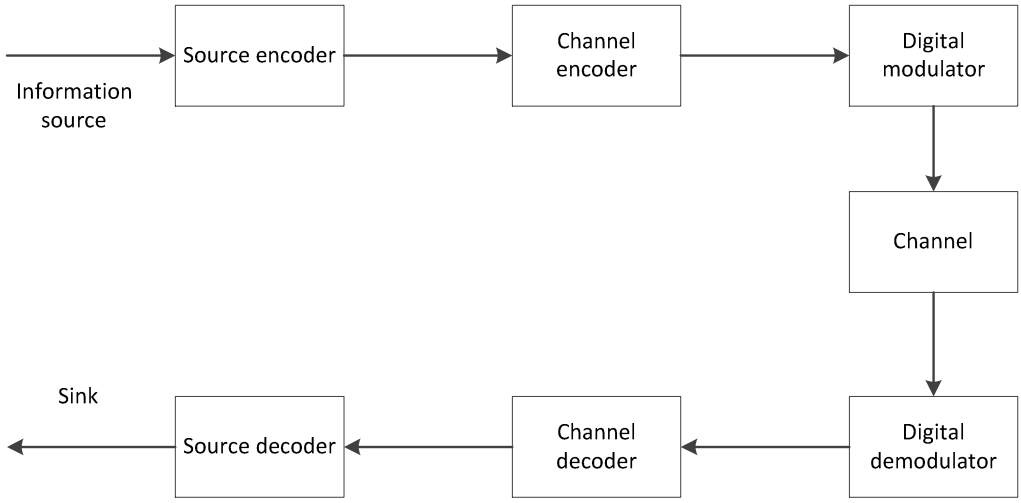

Figure 2.1: Elements of a digital communication

Source encoding – In source encoding, the information is converted into a digital message. Redundancy is removed from the digital message using compression techniques. In the case of speech, video, or images, lossy compression techniques can be used. Lossy compression techniques do not reproduce the original information on decompression. Rather the reconstituted message will differ from the original in a way that the final user does not lose the relative message that was transmitted initially. In the case of text, executable files or documents, lossless compression techniques can be used. The reconstituted message matches exactly the original message that was compressed. Lossy compression techniques are more effective when compressing information [41].

Channel coding – Channel coding can be referred to as error control coding. The objective of error control coding is to average the effects of channel noise of several transmitted signals. Redundancy is added into the transmitted sequences resulting in more bits being transmitted than those needed to represent the actual information. The redundancy allows the detection of the presence of errors in the received information [42].

Modulation – Modulation can be considered as a way of mapping an information signal on to a carrier signal. Initially the digital symbols are converted into a baseband information signal. The rate of the baseband signal changes at a rate comparable with the digital symbols. Examples of baseband signals include Non Return to Zero-Level (NRZ-L), Manchester and Differential Manchester format. Baseband signals are low frequency signals. They can be either digital or analogue. Figure 2.2 shows a representation of binary bits using three digital baseband signals [43].

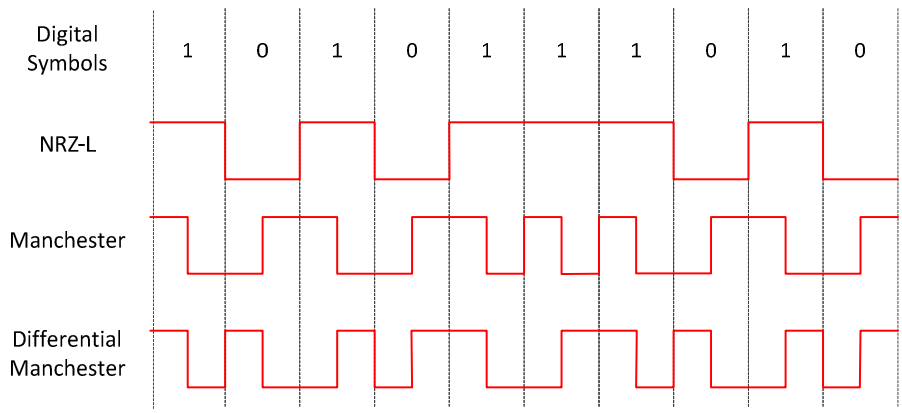

Figure 2.2: Digital signal encoding formats

It is possible to transmit the information signal as it is, but usually it is modulated on to a carrier signal with a higher frequency. This allows engineers to design communication systems that use specific frequencies for the transmitted signals thereby enabling spectrum management. Higher frequency signals have a smaller wavelength, allowing for smaller antennas on the communicating devices and allowing portability in some devices. When wireless communication was first introduced, it could only be used for ships because they were large enough to carry the antennas required [44]. The introduction of smaller antennas has led to mobile communication via smaller devices like modern cellular phones. Using carrier signals also allows the transmission of data across some media, like optical fibres that do not carry signals at some frequencies [43].

In modulation techniques, the data varies either the frequency or the amplitude or the phase of the carrier signal. The three basic classes of digital modulation are

- Amplitude-shift keying (ASK),
- Frequency-shift keying (FSK), and
- Phase-shift keying (PSK).

In ASK the amplitude of the carrier is varied in response to information. The other parameters of the carrier signal are kept fixed. In FSK, the frequency of the carrier is changed. A particular frequency is used for a 1 and another frequency is used for a 0 if we are producing a binary FSK signal. In PSK, we change the phase of the carrier to indicate information. The change in phase denotes a change in the bit being transmitted.

An example would be to multiply the NRZ baseband signal by a sinusoidal carrier whose frequency is a multiple of the transmitted bit rate to ensure that a whole number of carrier cycles are contained in a single bit interval. The transmitted signal over a single-bit interval is either the sinusoidal carrier or a phase shift of 180°, the inverse of the carrier. This scheme is known as Binary Phase Shift Keying (BPSK). BPSK is variation of PSK and is represented by the equation

$$s_n = \sqrt{\frac{2E_b}{T_b}}\cos(2\pi f_c t + \pi(1-n)), \text{ where } n = 0 \text{ or } 1, \tag{2.1}$$

where E_b is the energy per bit, T_b is the bit duration, f_c is the frequency of the carrier signal t is time and n is the binary bit being transmitted. Figure 2.3 shows the resultant wave for BPSK modulation.

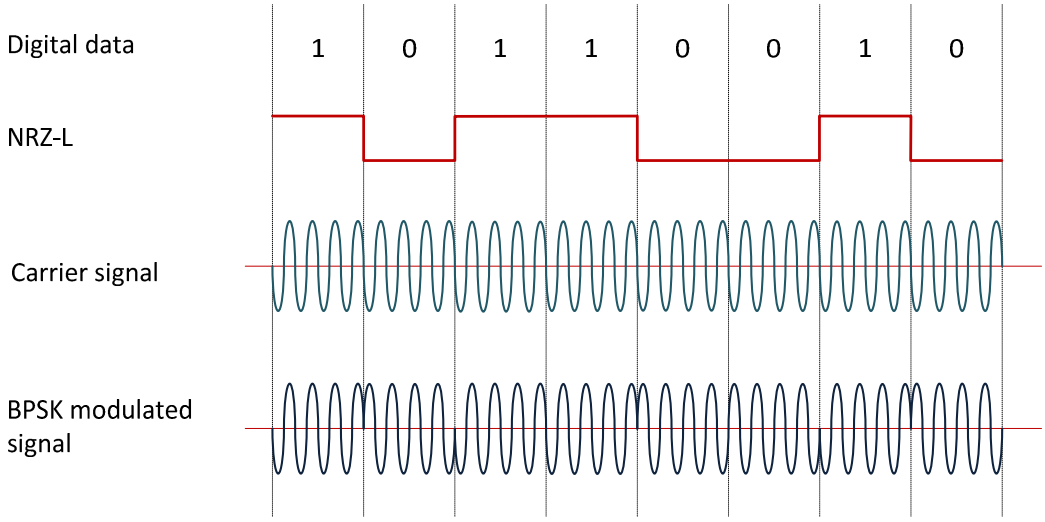

Figure 2.3: BPSK modulation

The modulation technique used in a communication system is determined to some extent by the channel's anticipated noise, interference, and distortion characteristics. In bandwidth limited channels, multilevel modulation schemes such as M-ary Phase Shift Keying (MPSK) and Quadrature Amplitude Modulation (QAM) may be used [45][41].

The channel – There are a various number of channels. They include twisted copper pairs, high frequency radio links, microwave links, electricity lines used in power-line communication, satellite links and optical fibres. It is worth noting, in information theory storage devices can be considered to be channels. Each channel presents its own challenges to the transmission of information. In twisted copper pairs, the transmitted signal can be distorted by crosstalk from other communicating lines, thermal noise, poor termination of the cable and other factors [41]. The way in which the channel impacts on the transmitted signal is described using a number of mathematical models like:

- Symmetric channel,
- Additive White Gaussian Noise (AWGN) channel,
- Bursty channel,

- and Fading channel [14].

Demodulation – Demodulation is performed by the receiving device. It is the opposite of modulation. The receiving device attempts to reconstruct the digital codewords that were transmitted by the sending device before modulation, so that they can be passed to the decoder. The receiving device compares the received signal to the carrier signal to enable it to reconstruct the baseband signal [41].

Channel decoding – In the decoding process, the redundancy that was added by the transmitting device is used to determine whether the received vector has errors or none. If the encoding technique is an error correcting technique, the codeword is corrected for errors, provided that the errors are within the error correcting capability of the code. The decoder is able to determine if the codeword has errors because there is a limited number of codewords within the space of possible codewords. If the received sequence is not a valid codeword, then errors have occurred during the demodulation stage of the communication process. The simplest error correcting decoding method is known as maximum likelihood decoding. The received erroneous sequence of symbols is compared with all the possible valid codewords. The codeword with the closest match to the received sequence is deemed to be the original codeword that was transmitted [41][14].

Source decoder – Finally the information from the channel decoder is forwarded to the source decoder. The source decoder uses this information to try and reconstruct the original message. [41].

2.3 Mathematical models for communication channels

The communication channel provides the link between the transmitting and the receiving devices. It is the physical medium that is used to send the signal. Examples of communication channels include telephone lines, optical fibres, microwave radio channels, cellular phone channels, satellite communication channels, etc. Whichever the physical medium used to transmit the signal, there is a probability that the signal will be corrupted in a random manner [45]. The channel introduces a number of effects such as attenuation, distortion, interference, and

noise. These effects of the channel manifest as errors in the demodulated data stream [41]. Mathematical models are used to represent the communication channel in the design of communication systems. The models are used to represent the most important characteristics of the channel. In this section we represent some communication channel models.

2.3.1 The additive noise channel

One of the most used mathematical models for a communication channel is the additive noise channel illustrated in Figure 2.4. The transmitted signal $s(t)$ is corrupted by an additive random noise $n(t)$.

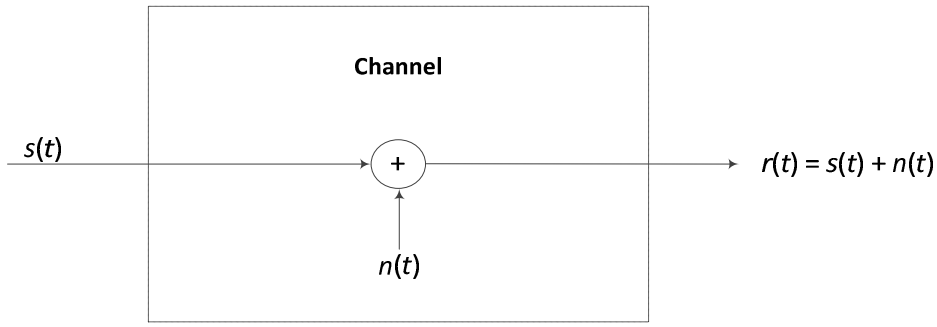

Figure 2.4 : The additive noise channel [45]

The noise may come from electronic components and amplifiers at the receiver of the communication system or from interference encountered in transmission. Equation (2.2) describes the additive noise channel as:

$$r(t) = s(t) + n(t) \tag{2.2}$$

Channel attenuation can be incorporated into the mathematical model. If the signal undergoes attenuation by an attenuation factor ω, the received signal can be described using the following equation:

$$r(t) = \omega \cdot s(t) + n(t) \tag{2.3}$$

The noise signal, $n(t)$, is random in nature. It is commonly described using its mean and variance. To find the mean and variance of a random signal, we need to know its probability distribution. The primary source of noise in the channel is thermal noise generated by the electronic components at the receiver. This type of noise is characterized by a Gaussian distribution. The model also assumes that in the frequency domain, the power spectrum of the random noise signal is uniform over the whole frequency range where communication takes place. The resulting mathematical model of the channel is usually called the additive white Gaussian noise (AWGN) channel. This channel model is applied to a broad class of physical communication channels and because of its mathematical tractability, this is the most used channel model in the analysis and design of communication systems [45][46]. The AWGN channel offers a platform to develop other complex channels. It also offers a starting point when studying the basic performances of a coding system.

2.3.2 The linear filter channel

In some physical channels, filters are used to limit the bandwidth of the signals to prevent interference among signals. The filters are used to ensure that transmitted signals are confined to specific bandwidth limitations and do not interfere with other signals at different frequencies. In such cases, the channels are generally characterised by a linear filter channel model as shown in Figure 2.5:

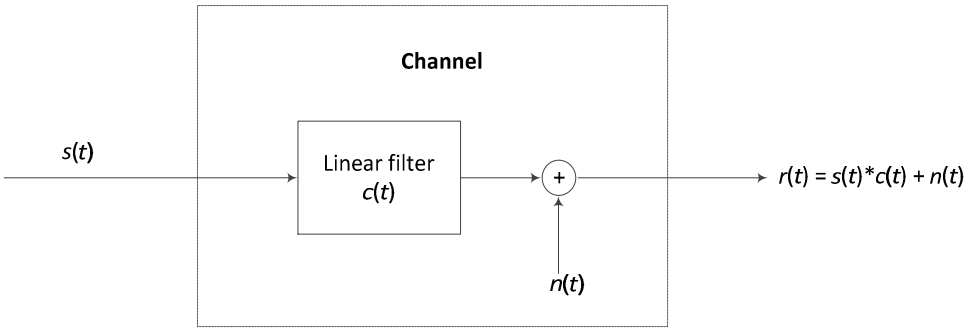

Figure 2.5: Linear filter channel with additive noise [45]

The channel output $r(t)$ for a channel input $s(t)$ and a filter response $c(t)$ is given by

$$r(t) = s(t) * c(t) + n(t),$$

(2.4)

where $*$ represents convolution. In this model the characteristics of the filter representing the channel does not change with time. This means, the filter impulse response does not depend on the time elapsed between observation and application of the input [45][46].

2.3.3 Linear time-variant filter channel

Channels such as mobile cellular radio channels are modelled as linear variant filter channels. The signal in such channels travels through different paths and arrives at the receiving device at different times. The impulse response of the channel varies with the elapsed time hence the various signals are affected by different channel characteristics. The linear filters are characterised by a time-variant channel impulse response $c(\tau;t)$, where $c(\tau;t)$ is the response of the channel at a time t due to an impulse applied at time $t-\tau$ and τ represents the elapsed time variable. The linear time-variant channel with additive noise is illustrated in Figure 2.6.

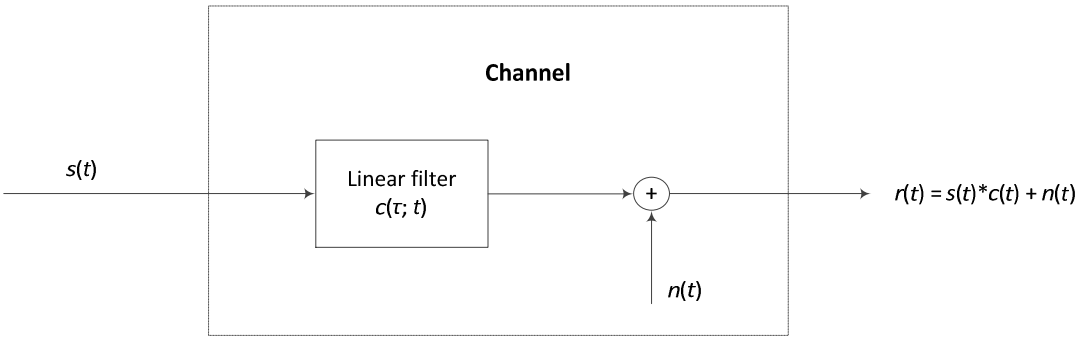

Figure 2.6 :Linear time-variant filter channel with additive noise [46]

The model is described mathematically as

$$r(t) = s(t) * c(\tau;t) + n(t).$$

(2.5)

This equation can be written as

$$r(t) = \int_{-\infty}^{\infty} c(\tau;t)s(t-\tau)d\tau + n(t) .$$

(2.6)

A good model for multipath signal propagation through physical channels, such as mobile cellular radio channels is a special case of (2.6) in which the time-variant impulse response has the form

$$c(\tau;t) = \sum_{k=1}^{L} \omega_k(t)\, \delta(\tau - \tau_k(t)) ,$$

(2.7)

where $\{\omega_k(t)\}$ represents the time-variant attenuation factor for the kth propagation path among L multiple paths and $\{\tau_k\}$ are the corresponding time delays. If (2.7) is substituted into (2.6) the received signal has the form

$$r(t) = \sum_{k=1}^{L} \omega_k(t) s(t-\tau_k) + n(t) .$$

(2.8)

The received signal consists of L multipath components, where the kth component is attenuated $\{\omega_k(t)\}$ by and delayed by $\{\tau_k\}$ [45][46][47]

2.3.4 Rayleigh fading channel

In wireless communication, the signal can travel from the transmitter to the receiver over multiple reflective paths. This is known as multipath propagation. Multipath propagation can result in multipath fading. In designing a wireless communication system, the Rayleigh fading channel can be used for estimating the effects of multipath fading and noise on the wireless communication channel. The Rayleigh channel is used to model wireless communication in channels where there is no distinct dominant path, like wireless communication with no line of sight. For wireless communication with a direct line of sight the Rician fading channel model can be used [48].

Time-variant multipath channels can be represented statistically using the Rayleigh fading channel. Let the transmitted bandpass signal be

$$s(t) = \mathrm{Re}\left[s_b(t) e^{j2\pi f_c t} \right] ,$$

(2.9)

where $s_b(t)$ is the baseband signal, f_c is the carrier frequency and t is the time. We assume the transmitted signal reaches the receiver through multiple paths. The received signal for the nth path with a propagation delay, $\tau_n(t)$, and an attenuation factor, $\omega_n(t)$, and is given by:

$$r(t) = \sum_n \omega_n(t) s\left[t - \tau_n(t)\right].$$

(2.10)

Substituting (2.9) into (2.10), we get

$$r(t) = \mathrm{Re}\left(\sum_n \omega_n(t) s_b\left[t - \tau_n(t)\right] e^{j2\pi f_c [t - \tau_n(t)]} \right).$$

(2.11)

The baseband equivalent of the received signal is

$$r_b(t) = \sum_n \omega_n(t) e^{-j\theta_n(t)} s_b\left[t - \tau_n(t)\right],$$

(2.12)

where $\theta_n(t) = 2\pi f_c \tau_n(t)$ is the phase of the nth path. The impulse response is

$$c_n(\tau_n; t) = \sum_n \omega_n(t) e^{-j\theta_n(t)} .$$

(2.13)

Large dynamic changes in the medium are required for $\omega_n(t)$ to change sufficiently to cause a significant change in the received signal [45]. It should be noted that $\theta_n(t)$ can change by 2π radian when the delay $\tau_n(t)$ changes by $\frac{1}{f_c}$. Since $\frac{1}{f_c}$ is a small number, $\theta_n(t)$ can change by 2π radians with relatively small motions of the medium. When there are a large number of paths, the central limit theorem can be applied. Each path can be modelled as circularly

symmetric complex Gaussian random variable with time as the variable. This model is called the Rayleigh fading channel model [45][49].

For the purposes of our research, we will initially use the AWGN channel. This channel has a linear addition of white noise with a constant spectral density and a Gaussian distribution of amplitude. The channel does not account for attenuation of the signal, nor interference and dispersion. It provides a simple and tractable mathematical model for studying the proposed decoding algorithm [41]. Its simplicity enables us to look into the performance of the decoding algorithm in comparison with the decoding algorithm proposed by Elias.

2.4 Error control techniques

In the preceding section, the elements of a digital communication system were presented. In any communication system there will be errors introduced by noise in the channel.

In this section we present two methods that have been used to mitigate the impact of errors introduced by the channel on communication. The first is automatic repeat request (ARQ) and the second is forward error correction (FEC).

2.4.1 Automatic repeat request (ARQ)

In ARQ the receiving device uses an error detection technique to determine if the received data has any errors. If errors are detected, the receiving device sends a request to the transmitting device to resend the data. For ARQ to work, the receiving device has to have a means of informing the sending device that the data was received with errors. ARQ only works in half duplex or full duplex communication systems that allow communication in both directions. Like FEC, ARQ works by adding redundancy to the transmitted data. The redundancy is then used to detect the presence of errors at the receiving device [3].

An example of an error detecting communication system would be a single parity check code. A single parity bit is appended to the end of the message bits being transmitted. Assuming the message is a single bit **0** or **1**. A single bit is appended to the message to create a codeword

which has a modulo sum of $\mathbf{0}$. If the message is $\mathbf{1}$, $\mathbf{1}$ is appended resulting in a codeword $\mathbf{11}$, if the message is $\mathbf{0}$, another $\mathbf{0}$ is appended and the transmitted codeword is $\mathbf{00}$. If the received codeword is either $\mathbf{10}$ or $\mathbf{01}$, the receiving device detects that codeword has an error, and requests the sending device to resend the codeword. If the received codeword has no errors, $\mathbf{00}$ or $\mathbf{11}$ is received, the receiving device sends an acknowledgement back to the sending device.

All ARQ systems work in the manner defined by the example, they only differ in the manner in which they recover the codewords with errors. Three types of ARQ exist:

- Stop-and-wait ARQ,
- Go-Back-N ARQ,
- and Selective repeat ARQ.

Stop-and-wait ARQ

In stop-and-wait ARQ, the transmitting device sends a single codeword, and waits for an acknowledgement (ACK) from the receiving device. Two situations could occur, the first being the codeword is received with errors. On receiving a codeword with errors, the receiving device sends a negative acknowledgement (NAK) back to the transmitting device. On receiving the NAK, the transmitting device resends the codeword. It will continue to resend the codeword, until it receives an ACK from the receiving device. The second situation that could occur is that the receiving device receives the codeword without errors and sends an ACK back to the transmitting device. The ACK is damaged by noise in the channel, making it unrecognizable by the transmitting device. The transmitting device uses a timer to handle this situation. On sending the codeword, the transmitting device starts a countdown timer. If the timer expires without the transmitting device receiving either a NAK or ACK, the transmitting device assumes the codeword was never received by the receiving device, and resends the codeword [50][14].

Go-back-N ARQ

This type of ARQ uses sliding windows in its operation. The transmitter and receiver establish a window size at the beginning of transmission. The window size (N) is the number of codewords the transmitting device sends before the receiving device sends an ACK. Assume the window size between two communicating devices A and B is three codewords, with Device A transmitting and B receiving. If A sends the first three codewords, B only sends the ACK

denoting that it has received all three codewords without errors. After A receives the ACK, it sends the next three codewords. If a codeword within the window is detected to have errors, the receiving device sends a NAK to the transmitting device, and discards all codewords that follow the codeword with errors. The transmitter on receiving the NAK responds by sending a new set of codewords equal to the window size, starting with the codeword that was rejected by the receiving device.

Go-back-N ARQ has better efficiency when compared to stop-and-wait ARQ. Less time is wasted waiting for acknowledgements. Using a sliding window allows the communicating devices to negotiate the window size during communication. The window size can be increased when the channel is not noisy. This increases the throughput of the communication system. As the channel becomes noisier, the devices can negotiate on a smaller window size [50][14].

Selective-repeat ARQ

Selective-repeat ARQ is sometimes referred to as selective-reject ARQ. In this type of ARQ, the sending device only resends the codewords that are received with errors. Each incoming codeword is acknowledged. The transmitting device keeps track of the codewords that have been sent by keeping a copy of them in a buffer. Each codeword has a countdown timer. If a codeword receives a positive acknowledgement, it is removed from the buffer. If either the countdown timer runs down or the codeword is negatively acknowledged, the transmitting device resends a copy. Selective-repeat ARQ is more efficient than the other two, but it is very complex to implement. It is expensive on memory as both communicating devices have to store a large number of codewords. The receiving device has to store all the codewords after the codeword with errors so that it can assemble them in the right sequence when a copy with no errors is received. Selective-repeat is used in satellite communication where there is a long propagation delay [50][14][43].

2.4.2 Forward error correction (FEC)

The second method used to control the presence of errors in a digital communication system is forward error correction (FEC). Unlike ARQ, FEC not only detects the presence of errors, but also corrects them. This lowers the number of retransmitted erroneous codewords [44]. The

reduction in the number of retransmitted codewords lowers the costs of communication. Avoiding retransmission of packets has the added benefit of saving battery power in mobile devices. The wireless channel is a very noisy channel. When mobile devices transmit packets, the antenna uses the battery power. If the antenna transmits for long periods of time, battery life is shortened. In some cases it is not feasible to retransmit the packets. In real-time systems like television broadcast on mobile devices retransmission will not be possible [51]. In the wireless channel error correction also reduces jamming on the network as fewer packets are retransmitted. Using ARQ would result in a large number of retransmission as the channel is very noisy. FEC on the other hand recovers data that is lost due to interference [52]. In simplex communication systems, there is no return path for the receiving device to send a NAK if the codeword is corrupted. ARQ cannot be used in such communication systems, hence FEC is used [40].

Prior to the discovery of the field of error control coding, it was believed that the noise in the channel prevented error free communication. Claude Shannon proved that noise in the channel does not prevent error free communication rather it limits the rate at which the information is transmitted. Shannon came up with an equation that governed the rate of communication in a channel. The equation for an AWGN channel is as follows:

$$C = B\log_2\left(1+\frac{E_b R_T}{N_o B}\right), \tag{2.14}$$

Where C is the channel capacity in bits per second, B is the bandwidth, $\frac{E_b}{N_o}$ is the signal-to-noise ratio (SNR) of the channel at an instance and R_T is the bit transmission rate. Shannon's equation gives a measure of the maximum number of error free bits per second C, that can be transmitted across a channel within a particular bandwidth B, given the signal to noise to ratio of the channel at that instant. Shannon's theorem stated that as long as $R_T \leq C$, error control codes could be employed to introduce error free transmission. Though Shannon defined the theorem, he did not stated how those codes could be found [53]. His theorem has been used to give a measure of channel performance, as it gives a limit on the capabilities of the channel. Shannon's theorem states that in any communication system there exists a coding scheme that

can be used to achieve error free communication provided the codeword (message bits plus parity bits) is less than the channel capacity [54]. Many coding schemes have been designed since Shannon published his work.

Figure 2.7 shows the general concept of FEC. Redundant bits known as parity bits are added to the message. The message with the parity bits is known as a codeword. The parity bits are used to aid in the correcting of any bits that are received in error. The method of adding the parity bits is known as encoding. The receiving device receives the codeword and computes the original information from that codeword. This is known as decoding. The receiving device uses the parity check to detect any errors that might occur in the transmitted information, and also correct the errors. If the receiving device does not detect the errors, or cannot recover the original information from the codeword, the packet has to be re-sent. The entire process is defined by a coding scheme [41]. FEC enables a receiving device to recover lost data and or correct errors that occur during communication without further interaction with the sending device.

FEC codes are designed to either correct errors or recover erasures or in some cases do both. In [55] an error is defined as a corrupted symbol in an unknown position. If a symbol has been demodulated to the wrong value, then an error has occurred for example a binary **1** is received as **0**. An erasure is a corrupted symbol in a known position. In the case of an erasure, the demodulator is unable to ascertain whether a **0** or a **1** was sent.

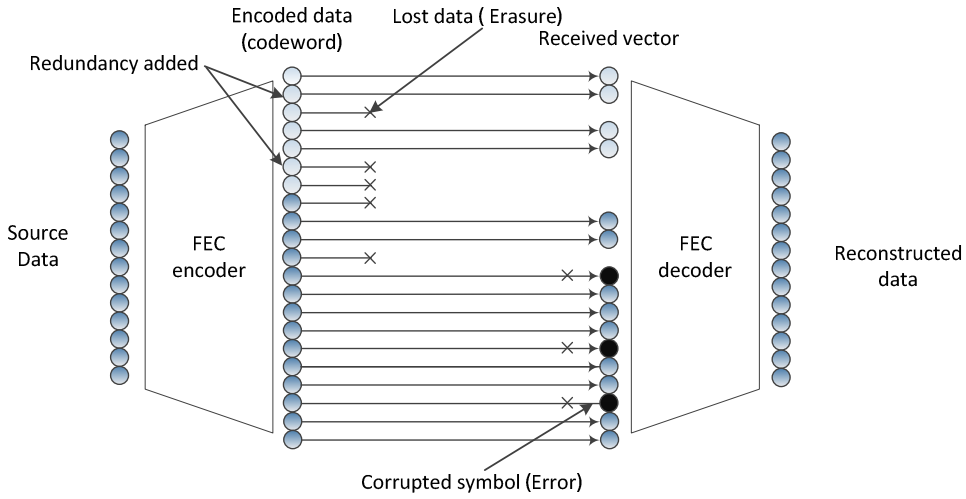

Figure 2.7: Forward error correction encoding and decoding adapted from [55][56]

Code Rate

In general, the higher the number of redundant bits in a code, the better the error correcting capability of that code. Unfortunately there is a compromise with the fact that the redundant bits add overhead and transmission costs in the form of bandwidth and power. It also adds complexity to the decoding of the codeword. Complexity means more processing power will be needed to decode the codeword, and impacts negatively on the battery life of the mobile device. A rate defined as the code rate is used to give a quantitative measure of the redundant bits. The code rate is defined as the ratio of the number of data bits k, compared to the total number of codeword bits n. When encoding, k bits are encoded resulting in a codeword of n bits. Thus the code rate R can be calculated as

$$R = \frac{k}{n}. \tag{2.15}$$

A low code rate indicates that the redundancy is high. The ideal is keeping the code rate as close to 1 as possible yet not compromise on the error correcting capability of the code [41]. It is common practise to use a high error correcting code with a low code rate when designing a coding scheme. In very low SNRs, the coding scheme fails to correct all the errors. In high SNRs, the added redundancy impacts negatively on the overall throughput of the communication system. Variable rate codes are used to keep the throughput of communication systems as high as the SNR permits. In low SNR a lot of redundancy is added to increase the error correcting capability of the code, then at high SNR, higher code rate codes are used to increase the transmission efficiency of the communication system.

Types of codes

Codes can be divided into two subclasses depending on the manner in which the information is encoded. The two subclasses are block codes and convolutional codes. Block coding schemes divide a bit stream into non-overlapping blocks, and each block is encoded independently. They are sometimes referred to as memory-less since each successive information block is encoded independently [30]. This research focuses on block codes. They are presented in greater detail in Section 2.5.

Convolutional codes were first presented by P. Elias in 1955 [57]. Elias was motivated by optimizing the trade-off of performance versus complexity. His goal was to find a class of codes for the binary symmetric channel with as much structure as possible, while keeping error correcting capability high [58]. Whereas block codes work on discrete codes blocks of k input symbols, convolutional codes operate on continuous streams of symbols not partitioned into discrete message blocks. For this reason, convolutional codes are viewed as stream codes [30]. Convolutional codes have proved to be equal or sometimes superior to block codes, but they present a challenge when analysing them [59].

The repetition code is an example of a coding scheme. Assume device A wants to transmit a single bit to device B. The transmitting device appends two parity bits to the single bit unlike in the error detection technique stated earlier. If the device is transmitting a message, a binary 1, it appends two parity bits so that the resultant codeword is 111 and if 0 is being transmitted the codeword would be 000. This repetition code has two valid codewords. If during demodulation, the vector 101 is received, device B detects that there is an error in the codeword. Device B can correct the received vector by comparing it to the two possible codewords and changing it to the one that is the closest match, which is 111 in this particular case. Device B then removes the appended parity bits and concludes that the bit sent was 1. This type of decoding is known as maximum likelihood decoding. If two bits are flipped during transmission, and device B receives 001, device B will assume the original codeword was 000, which is not the case. This is known as a decoding error. This is a result of the fact that the number of errors introduced by the channel is beyond the error correcting capability of the code, which is one in this case.

2.5 Linear block codes

Block codes are always represented as (n, k) codes where k information symbols are encoded into n codeword symbols. For binary linear block codes, 2^k blocks of information are encoded to produce 2^k codewords from a space of 2^n possible vectors. Each block of information maps uniquely on to a single codeword. Codes are desirable if they are linear, as this reduces encoding complexity.

Definition 2.1: A binary block code C of length n with 2^k codewords is called linear, if and only if its 2^k codewords form a k-dimensional subspace of the vector space of all the n-tuples over the field $GF(2)$ [14].

As a consequence of Definition 2.1, a binary block code is linear if and only if the modulo-2 sum of any two codewords results in another codeword [14][30].

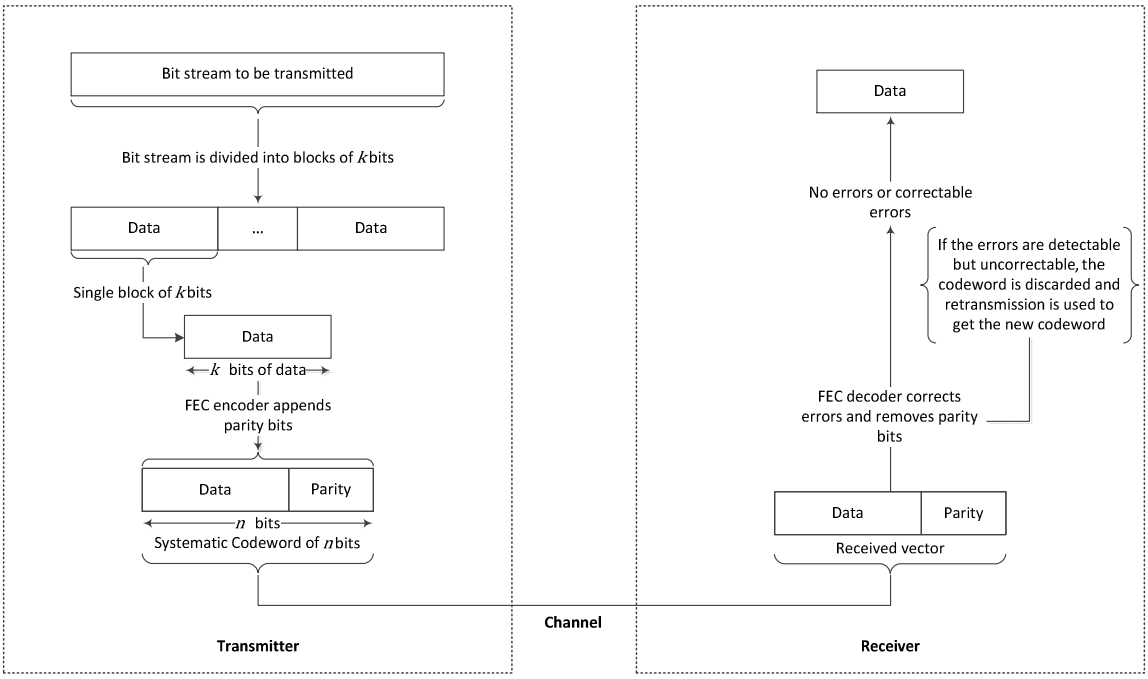

Figure 2.8: Systematic block code transmission adapted from [43]

All $(n,\ k)$ linear block codes satisfy

$$v = uG\ , \tag{2.16}$$

where u is the block of k information symbols, v is the unique codeword of length n information symbols and G is a $k \times n$ matrix called the generator matrix. The generator matrix has linearly independent rows [14]. If the generator matrix can be written as

$$G = [I_k P], \tag{2.17}$$

where I_k is a $k \times k$ identity matrix and P is a $k \times (n-k)$ matrix, the linear block code is systematic. This is a desirable structure in linear block codes. In systematic encoding, the

codeword consists of the k original message symbols and $(n-k)$ parity symbols [30][14]. The systematic codeword is of the form

$$v = uG = u\left[I_k P\right] = \left[u_1, \ldots, u_k, p_1, \ldots, p_{(n-k)}\right]. \tag{2.18}$$

A very important metric of linear block codes is the Hamming distance.

Definition 2.2: The Hamming distance is the number of positions in two equal length sequences at which the corresponding symbols differ [30].

For example, given two binary codewords, 1001101 and 1101100, their Hamming distance is two.

Definition 2.3: The Hamming weight of a codeword is its Hamming distance from the zero codeword. In binary codes, the Hamming weight of any codeword is the number of ones in the word [14].

For example, the binary codeword 1001101 has a Hamming weight of 4. The minimum weight of a code is the Hamming weight of the nonzero codeword with the lowest weight.

Definition 2.4: The minimum Hamming distance d_{min} of a code is the minimum value of the Hamming distances of any two valid codewords.

The error correcting capability t of a linear block code is defined as:

$$t = \left\lfloor \frac{d_{\min} - 1}{2} \right\rfloor, \tag{2.19}$$

where d_{min} is the minimum Hamming distance of the code. The minimum distance of a linear code is equal to the minimum weight of its non-zero vectors [60].

A less commonly used measure of the error correcting capability of codes is the weight distribution.

Definition 2.5: Considering a code C with A_i being the number of codes with a Hamming weight of i. The set $\{A_0, A_i, \ldots\ldots\ldots A_n\}$ is the weight distribution of C [42][61][62]. The weight distribution can be defined using a weight enumerator polynomial:

$$A(x) = A_0 + A_1 x + A_2 x^2 + \ldots\ldots\ldots + A_n x^n \tag{2.20}$$

Given a generator matrix *G*, there exists a $(n - k) \times n$ parity check matrix H. A parity check matrix is used to decode linear block codes. A codeword v is a valid codeword if and only if

$$vH^{\mathrm{T}} = 0. \tag{2.21}$$

This property of linear block codes is used to check and correct the received codewords for errors. If the received vector r' has errors, we write the received codeword as follows

$$r' = v + e \tag{2.22}$$

where e is the error vector representing the codeword symbols corrupted during transmission. The syndrome s of r', is defined as

$$s = r' H^T \tag{2.23}$$

If r' is a valid codeword, then, $s = r' H^T = 0$. The syndrome is equal to an all zero vector, if the received vector is a valid uncorrupted codeword, or is corrupted beyond the minimum Hamming distance such that it is changed into another valid codeword. If the received codeword contains detectable errors, then, $s \neq 0$ [14].

Note that, if G is of the form given by (2.17), then H takes the form

$$H = \left[P^T \; I_{n-k} \right]. \tag{2.24}$$

2.5.1 Repetition code as an example of a block code

Citing the repetition code presented earlier we can give examples of some of the terms that describe a block code. The number of information symbols $k = 1$ since the code transmits one information bit in each codeword. The codeword length is $n = 3$. The Hamming distance between the codewords 000 and 111 is $d_H = 3$ since they differ in all three positions. Because they are the only valid codewords, the minimum Hamming distance of the code is $d_{min} = 3$. The repetition code can be described as a (*3, 1*) code with $d_{min} = 3$. The error correcting capability of the code is 1 from (2.19)**.** The code rate R of the code, from (2.15), is $= 0.333$. The generator matrix of the code is given by

$$G = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix},$$

and the parity check matrix is

$$H = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}.$$

If the received codeword is $v = \begin{bmatrix} 1 & 0 & 1 \end{bmatrix}$, the syndrome will be

$$s = vH^T = \begin{bmatrix} 1 & 0 \end{bmatrix}.$$

From the syndrome, the received vector is determined to have errors. To correct the errors, the received vector is compared to all the possible valid codewords, in this case 111 and 000. The correct vector is the valid codeword which is the closest match to the received vector. The vector 101 differs in one place with 111, and in two places with 000. Since 111 is the closest match, the received vector is corrected by changing the second bit to 1.

2.5.2 Hamming codes

Richard Hamming discovered the first class of linear block codes in 1950 while working at Bell Labs [63]. They were named after him. At the time it was possible to detect errors using parity checks. Hamming required a technique that could be used to enable computers to detect

and correct their own errors. The computers Hamming was working on would stop when they detected an error. He proposed a more complex pattern of parity checks that could not only detect the presence of two errors, but could determine the location of the error if it was a single error [63]. Hamming codes are defined by the parameters $(n,k,3)$. The first Hamming code was the $(7,4,3)$ Hamming code [58]. In general the parameters for a Hamming code are,

Codeword length $$n = 2^m - 1 \quad \textbf{(2.25)}$$

Number of information bits $$k = 2^m - m - 1 \quad \textbf{(2.26)}$$

Number of parity bits $$n - k = m \quad \textbf{(2.27)}$$

where $m \geq 3$. The code rate of Hamming codes increases with an increase in m, but their error correcting capability remains the same. All Hamming codes have $d_{\min} = 3$. Therefore they can only correct a single error. This limits their use. They are used in channels that have a low probability of error. Hamming codes are in a class of codes known as perfect codes.

If a systematic (7, 4) has generator matrix

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

the corresponding parity check matrix would be

$$H = \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Hamming codes can be decoded using syndrome decoding. The received codeword is multiplied with the transpose of the parity check matrix. If the result is 0, then no errors are present in the codeword, or the errors are undetectable. If the syndrome corresponds to the ith column of the parity check matrix, the ith bit in the codeword is changed, assuming only a single error has occurred [64].

2.5.3 BCH codes

Bose, Chaudhuri, and Hocquenghem (BCH) codes fall in a category of block codes known as cyclic codes. BCH and Reed-Solomon (RS) codes are the most commonly used cyclic codes [30]. An (n,k) linear code C is called a cyclic code if every cyclic shift of a codeword in C results in a valid codeword in C. Binary BCH codes were discover by Hocquenghem in 1959. Then in 1960 they were discovered independently by Bose and Chaudhuri. They presented a means of designing codes over GF(2).

RS codes are a non-binary subclass of BCH codes. They were discovered by Reed and Solomon in 1960 independently. In 1961 Gorenstein and Zierler generalized BCH codes to nonbinary codes proving the relationship between BCH codes and Reed-Solomon codes. It was later realised that RS codes are a subclass of BCH codes. The approach used to construct RS codes was independent of the work done to construct BCH codes.

Defining BCH codes

BCH codes are a generalization of Hamming codes. Unlike Hamming codes, they are capable of correcting multiple errors. They are cyclic codes, consequently they can be described in terms of their generator polynomial $g(x)$. The interest of this study is on binary BCH codes. The following parameters describe (n,k,t) binary BCH codes for $m \geq 3$ and $t < 2^{m-1}$

Codeword length $$n = 2^m - 1 \quad \textbf{(2.28)}$$

Parity check bits $$n - k \leq mt \quad \textbf{(2.29)}$$

Minimum distance $$d_{\min} \geq 2t + 1 \quad \textbf{(2.30)}$$

where t is the maximum number of correctable errors.

A t error correcting binary BCH code is capable of correcting t or fewer errors in a codeword of n bits. BCH codes are designed from a specification of n and t. The value of k is not known until the generator polynomial $g(x)$ is found [62].

A t-error correcting BCH code over $GF(2)$ of length n is designed as follows:

1. Determine m such that (2.28) is satisfied.
2. Define a primitive element α, whose primitive polynomial has a degree m.
3. Write down a list of $2t$ consecutive powers of α

$$\alpha, \alpha^2, \alpha^3, \ldots, \alpha^{2t}$$

4. Determine the minimal polynomial of each of these powers. Given an element α in the field $GF(p^e)$, the minimal polynomial of α is the monic polynomial of smallest degree which has coefficients in $GF(p)$ and α as a root.

The generator polynomial $g(x)$ of the BCH code is the lowest-degree polynomial over $GF(2)$ such that $\alpha, \alpha^2, \alpha^3, \ldots, \alpha^{2t}$ are its roots. It is the least common multiple of the minimal polynomials of each α^j term [30].

The value of k is given by the equation

$$k = n - \deg(g(x)) \tag{2.31}$$

where the degree of the generator polynomial is the value of its greatest exponent.

To design a BCH code of length $n = 15$, which can correct 2 errors, we choose α as the primitive element in $GF(2^4)$. We need to find the consecutive minimal polynomials of the elements from α to α^{2t}. Here α, α^2 and α^4 have the same minimal polynomial $g(x) = x^4 + x + 1$ and that of α^3 is $g(x) = x^4 + x^3 + x^2 + x + 1$. Therefore the generator polynomial is:

$$\begin{aligned} g(x) &= (x^4 + x^3 + x^2 + x + 1)(x^4 + x + 1) \\ &= x^8 + x^7 + x^6 + x^4 + 1 \end{aligned}$$

The value of the greatest exponent is 8, therefore the degree of the generator polynomial is 8, and k is

$$k = 15 - \deg(g(x)) = 15 - 8 = 7,$$

where $\deg(g(x))$ is the degree of the generator polynomial $g(x)$.

For product codes, the component codes have to be systematic [1]. The systematic codeword $c(x)$ is obtained from the coefficients of the polynomial resulting from multiplying the message polynomial, $m(x)$ by $x^{(n-k)}$ and adding $rem(x)$ to the result as follows

$$c(x) = \left[rem(x) + m(x)\, x^{(n-k)} \right], \tag{2.32}$$

where

$$rem(x) = \mathrm{mod}\left(\frac{m(x)\, x^{(n-k)}}{g(x)} \right).$$

Using (2.32), we can construct the ith row of the generator matrix G as follows

$$\left[rem(x_i) : m(x)_i \right],$$

where

$$m(x)_i = x^{(n-\deg(g(x))+i}.$$

For the (15,7) BCH code with the generator polynomial $g(x) = x^8 + x^7 + x^6 + x^4 + 1$, the first row of the generator matrix is as follows:

$$\begin{aligned} m(x)_1 &= x^{(n-\deg(g(x))+i} \\ &= x^{(15-8)+1} \\ &= x^8 \end{aligned},$$

and

$$\begin{aligned} rem(x_1) &= \mathrm{mod}\left(g(x) \overline{)m(x)_1} \right) \\ &= \mathrm{mod}(x^8 + x^7 + x^6 + x^4 + 1 \overline{)x^8}). \\ &= x^7 + x^6 + x^4 + 1 \end{aligned}$$

Therefore the first row G_{r1} of the generator matrix is

$$G_{r1} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

and the generator matrix is:

$$G=\begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

The data is encoded using (2.18).

Decoding BCH codes

Any of the decoding algorithms used for cyclic codes can be used for BCH codes. The structure of BCH codes has made it possible to come up with a very efficient decoding algorithm. This algorithm involves three steps. If the transmitted codeword is $v(x)=v_0+v_1x+v_2x^2+\cdots+v_{n-1}x^{n-1}$, the received vector is $r(x)=r_0+r_1x+r_2x^2+\ldots+r_{n-1}x^{n-1}$ and the error pattern is $e(x)$ then

$$r(x)=v(x)+e(x) \tag{2.33}$$

Like other block codes, the first step is to compute the syndrome from the received vector $r(x)$. The syndrome of a *t*-error correcting BCH code consists of $2t$ components.

$$S=\begin{bmatrix} s_1 & s_2 & \ldots & s_{2t} \end{bmatrix}=r\cdot H^{\mathrm{T}} \tag{2.34}$$

where H the parity check matrix can be written as

$$H=\begin{bmatrix} 1 & \alpha & \alpha^2 & \alpha^3 & \alpha^4 & \alpha^5 & \cdots & \alpha^{n-1} \\ 1 & \left(\alpha^2\right) & \left(\alpha^2\right)^2 & \left(\alpha^2\right)^3 & \left(\alpha^2\right)^4 & \left(\alpha^2\right)^5 & \cdots & \left(\alpha^2\right)^{n-1} \\ 1 & \left(\alpha^3\right) & \left(\alpha^3\right)^2 & \left(\alpha^3\right)^3 & \left(\alpha^3\right)^4 & \left(\alpha^3\right)^5 & \cdots & \left(\alpha^3\right)^{n-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \cdots & \vdots \\ 1 & \left(\alpha^{2t}\right) & \left(\alpha^{2t}\right)^2 & \left(\alpha^{2t}\right)^3 & \left(\alpha^{2t}\right)^4 & \left(\alpha^{2t}\right)^5 & \cdots & \left(\alpha^{2t}\right)^{n-1} \end{bmatrix} \tag{2.35}$$

The ith component of the syndrome is given by

$$\begin{aligned} s_i &= r(\alpha^i) \\ &= r_0 + r_1\alpha^i + r_2\alpha^{2i} + \ldots + r_{n-1}\alpha^{i(n-1)}, \end{aligned}$$

(2.36)

where $1 \le i \le 2t$. The syndrome components are elements in the Galois field $GF(2^m)$. Dividing $r(X)$ by the minimal polynomial $\phi_i(X)$ of α^i, the remainder $b_i(X)$ is obtained. The syndrome component S_i can be obtained by evaluating $b_i(X)$ with $X = \alpha^i$ [14][30][65].

Since $\alpha, \alpha^2, \alpha^3, \ldots, \alpha^{2t}$ are the roots of each code polynomial, $v\left(\alpha^i\right) = 0$ for $1 \le i \le 2t$. From (2.20) and (2.21), the relationship between the syndrome and the error pattern for $1 \le i \le 2t$ can be defined as follows

$$s_i = e(\alpha^i).$$

(2.37)

This means that the syndrome depends on the error vector only. Suppose the error vector has *l* errors at locations $X^{j_1}, X^{j_2}, \ldots, X^{j_l}$, meaning

$$e(x) = x^{j_1} + x^{j_2} + \ldots + x^{j_l}$$

(2.38)

where $0 \le j_1 < j_2 < \cdots < j_l < n$. From (2.22) and (2.23) we obtain a set of $2t$ equations:

$$s_i = (\alpha^{j_1})^i + (\alpha^{j_2})^i + \ldots + (\alpha^{j_l})^i, \text{ for } 1 \le i \le 2t,$$

(2.39)

where $\alpha^{j_1}, \alpha^{j_2}, \ldots, \alpha^{j_l}$ are unknown. The decoding algorithms of BCH codes solve these equations. Once $\alpha^{j_1}, \alpha^{j_2}, \ldots, \alpha^{j_l}$ have been found, the powers of $j_1, j_2, \ldots, j_l$, give the locations of the errors. $\alpha^{j_1}, \alpha^{j_2}, \ldots, \alpha^{j_l}$ are known as the error locators. Letting

$$x_l = \alpha^{j_q} \text{ for } 1 \le q \le l,$$

we can rewrite (2.39) as

$$s_i = \sum_{q=1}^{l} (x_q)^i = (x_1)^i + (x_2)^i + \ldots + (x_l)^i$$

(2.40)

The $2t$ equations covered in (2.39) are known as the power-sum symmetric functions [30]. They are simultaneous equations in l. An exhaustive search of the equations would give the error locations. However this search would be computationally taxing [14]. A polynomial known as the error-location polynomial,

$$\begin{aligned}\sigma(x) &\triangleq (1+X_1x)(1+X_2x)\cdots(1+X_lx) \\ &= \sigma_0+\sigma_1x+\sigma_2x^2+\ldots+\sigma_lx^l\end{aligned}, \tag{2.41}$$

is defined. The error location polynomial is unknown and its coefficients have to be determined. The process of finding the error location polynomial is the most complex part of the decoding algorithm. The most commonly used algorithm for finding the error location polynomial for both BCH codes and RS codes is the Berlekamp-Massey algorithm. Another algorithm used for BCH codes is the Peterson algorithm. The roots of the error locator polynomial are at the reciprocals of the error location numbers [30]. After finding the error locator polynomial the next step of decoding is finding its roots. One approach would be to examine every element in the finite field to ascertain if it is a root. This is known as the Chien search. If the roots are distinct and all lie in the appropriate field, then we use them to determine the error locations. If they lie outside the field, then the received codeword has errors but they are beyond the error correcting capability of the code [66].

Thus the algebraic algorithm for decoding a BCH code consists of the following steps:

- calculating the syndrome to ascertain if the codeword is received in error,
- computing the $2t$ components of the syndrome,
- determining the error location polynomial,
- finding the roots of the error location polynomial, and
- calculating the error values.

For a binary BCH code, it is unnecessary to calculate the error values [30]. All the bits that are in error are flipped using a bitwise XOR.

2.6 Product codes

2.6.1 Defining product codes

Product codes are serially concatenated codes that were first presented by Elias in 1954. The concept of product codes is a method of combining two or more short constituent codes to produce a code with a longer block length with better distance properties. Considering two linear codes C_1 and C_2 with parameters (n_1,k_1) and (n_2,k_2) respectively, a linear code C_P with parameters (n_1n_2,k_1k_2) can be formed. The codeword is a rectangular array of n_2 rows and n_1 columns in which every row and column is a systematic codeword. This two dimensional code is a product code of the constituent codes C_1 and C_2. Figure 2.9 shows the construction of a two dimensional product code

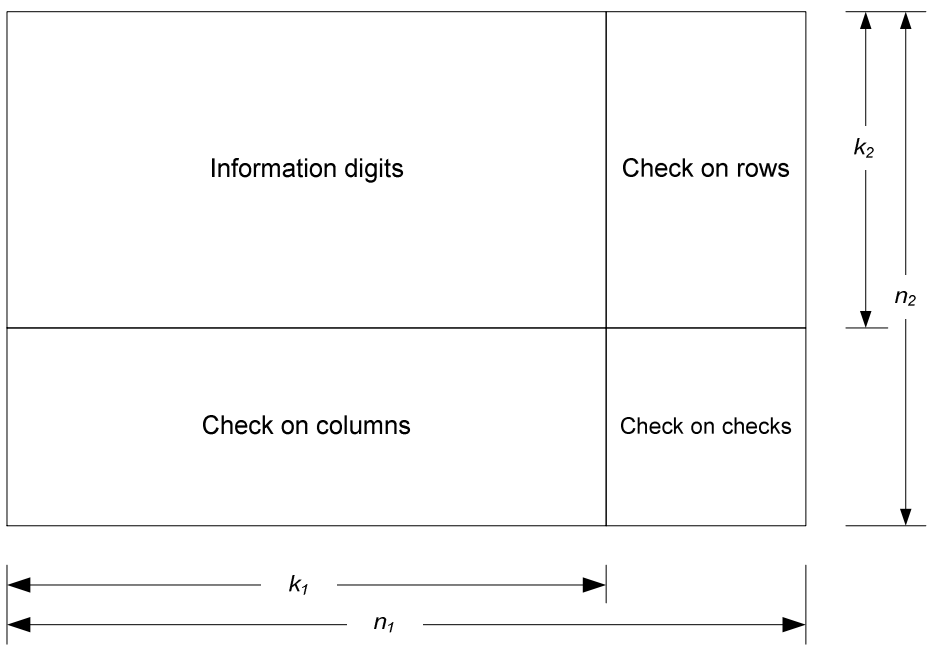

Figure 2.9: Two dimensional product code.

To obtain a product code from the two constituent codes, the following steps are carried out.

1. The information digits are arranged into a $k_1 \times k_2$ array.

2. The rows are encoded using the code C_1 to produce an array with k_2 rows and n_1 columns.
3. The columns are the encoded using the code C_2. The row parity checks are also coded resulting in bits that are checks on checks.

The generator matrix G_p of the product code can be obtained from the generator matrices G_{C_1} and G_{C_2} of the constituent codes by taking the Kronecker product of the two as shown in the following equation:

$$G_p = G_{C_1} \otimes G_{C_2} . \tag{2.42}$$

The codeword c is obtained by multiplying the generator matrix of the code with the information bits arranged in the $k_1 \times k_2$ array. The codeword c can also be obtained by using the following equation

$$c = G_{C_1}{}^{\mathrm{T}} u G_{C_2} , \tag{2.43}$$

where the message u is the data arranged in the $k_1 \times k_2$ matrix.

The minimum distance $d_{\min C_p}$ of a product code is a product of the minimum distances of the constituent codes. It is calculated as follows:

$$d_{\min C_p} = d_{\min C_1} \cdot d_{\min C_2} , \tag{2.44}$$

where $d_{\min C_1}$ is the minimum distance of C_1 and $d_{\min C_2}$ is the minimum distance of C_2. The product code has a larger minimum distance than the constituent codes, but it has a smaller fractional minimum distance than both constituent codes. Let δ_{C_1} and δ_{C_2} be the fractional minimum distance of the codes C_1 and C_2 respectively. Their fractional minimum distances are defined as:

$$\delta_{C_1} \triangleq \frac{d_{\min C_1}}{n_1} , \text{ and} \tag{2.45}$$

$$\delta_{C_2} \triangleq \frac{d_{\min C_2}}{n_2}.$$

(2.46)

The fractional minimum distance of the product code δ_{C_P} is defined as:

$$\delta_{C_P} \triangleq \frac{d_{\min C_1} d_{\min C_2}}{n_1 n_2}$$

(2.47)

or

$$\delta_{C_P} = \delta_{C_1} \delta_{C_2}.$$

(2.48)

Since both δ_{C_1} and δ_{C_2} are less than zero, it follows that their product δ_{C_P} is less than both [67]. This makes product codes less appealing when compared to the other codes. In coding emphasis is put on finding codes with a large fractional minimum distance [58].

It is worth noting that the product code construction can be iterated to produce multi-dimensional product codes. This enable codes of even larger distances to be produced. The number of dimensions lowers the code rate, at the expense of better error correcting capability. Product codes are at times referred to as turbo product codes, because most decoding algorithms developed for product codes are iterative [68].

2.6.2 Decoding product codes

Decoding of product codes can be done using both hard decision and soft decision decoding. In a binary code hard decision decoding, the demodulator output is quantized in two levels, denoted as 0 and 1. This results in a hard decision binary sequence vector. The decoder then processes the hard decision vector. This is referred to as hard-decision decoding. The metric used in Hard-decision decoding is the Hamming distance [14].

In soft decision decoding, the outputs of the demodulator are un-quantized or quantized in more than two levels. The demodulator attaches a confidence value to each symbol it demodulates. This value is passed onto the decoder and is used in error correction. A simple example would be a situation where a symbol is demodulated as a “0” but the demodulator has the least confidence in the value. If the decoder detects an error in the received vector, the value with the lowest confidence value is altered first. This is known as reliability-based soft decision decoding [69][70].

Conventional decoding algorithm

A lot of research on product codes has focused on finding efficient decoding algorithms. Since their introduction by Elias [1], many decoding algorithms have been presented. The first decoding algorithm (the conventional decoding algorithm) was presented by Elias in his original paper. Elias’s decoding algorithm involves a two-step process. First the rows are decoded, then the columns [1]. The rows or columns can be decoded if and only if the numbers of errors in the row or column is within the error correcting capability of the constituent code. The error patterns that are correctable in this algorithm are ones where, after correcting rows, there are correctable patterns in the columns [71]. Iterative decoding is used to improve this decoding algorithm. In iterative decoding, after correcting the errors in the columns, decoding is done column wise and row wise again until there are no errors, or a threshold of iterations is reached. Iterative decoding improves decoding at the expense of decoding delay.

The decoding algorithm proposed by Elias is not an optimum decoding algorithm. A product code has the error correction capability *t* defined by the equation:

$$t = \frac{d_{\min C_1} d_{\min C_2} - 1}{2}. \tag{2.49}$$

The decoding algorithm proposed by Elias is not capable of decoding some error patterns that lie within the error correcting capability of the code. Assuming the constituent codes are both Hamming codes, then the error correcting capability of the product code is four. Assuming the errors are arranged in a manner such that they are at the corners of a rectangular pattern as shown in Figure 2.10.

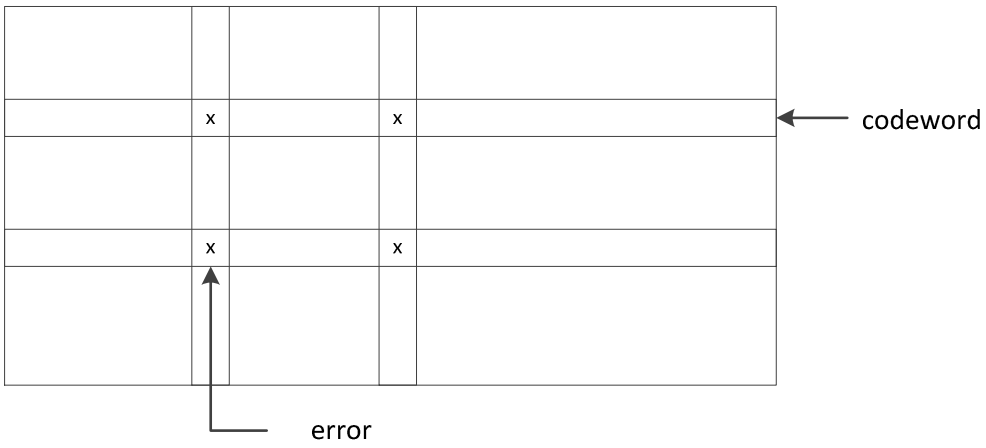

Figure 2.10: A permanent error in a two dimensional product code with Hamming code component codes

Though this is within the error correcting capability of the code, the pattern is not correctable. This is due to the fact that the number of errors in the rows and columns with errors is beyond the error correcting capability of the Hamming code [14]. This decoding algorithm, despite being suboptimal, is effective. Its complexity is approximately the sum of the decoding complexities of the constituent codes.

Viterbi algorithm maximum likelihood decoding

Product codes, like other linear block codes can be decoded using maximum likelihood (ML) decoding. ML decoding lowers the probability of decoding errors. The distance between the received codeword and every other codeword in the code space is computed. The codeword with the smallest distance from the received codeword is assumed to be the sent codeword. This method is very simple, but it is time consuming as all the codewords in the code space have to be compared. Its simplicity is overshadowed by the cumbersome task of comparing with all codewords. As the length of codewords increases, the decoding method becomes unattractive. Product codes are too long for this method of exhaustive searching. For example, for a (31, 26) binary Hamming decoded with the word correlation decoding method, the number of comparisons that have to be carried out is 2^{26}. A compromise has to be made between the simplicity of the decoding algorithm, and the probability of decoding error. A desirable algorithm is one that uses less memory, requires less computing, is easily implemented on equipment and minimizes the probability of error [3].

Andrew J. Viterbi introduced a decoding algorithm for decoding convolutional codes in 1967. The algorithm has since been named after him as the Viterbi algorithm (VA). Viterbi wanted to prove an asymptotic optimum upper bound on the error probability of convolutional codes. The VA algorithm was later proved to be a ML decoding algorithm by Forney and Omura [72][73]. In [74], Bahl *et al* showed that block codes could be represented using a trellis. They proposed a brute force method for decoding block codes based on the trellis. In [75], Wolf defined a trellis as a compact method of cataloguing all the codewords of a linear block code, with each distinct path through the trellis corresponding to a codeword in the codeword space. Wolf showed that the VA could be used for decoding block codes. Convention was that the VA could only be used for convolutional codes.

For a parity check matrix H of a (7,4) Hamming code

$$H = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix},$$

and codewords

$$\begin{aligned} c_i &= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \\ c_{ii} &= \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix} \\ c_{iii} &= \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 & 0 \end{bmatrix} \\ c_{iv} &= \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \end{aligned},$$

the corresponding trellis of the code with paths are represented in Figure 2.11.

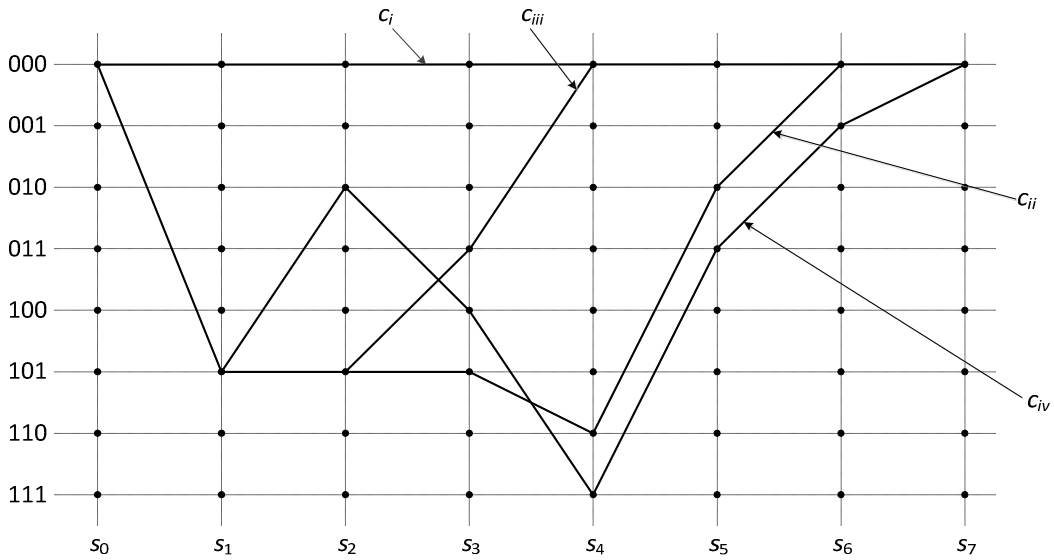

Figure 2.11: The trellis of four codewords of a (7,4) Hamming code

The trellis representation of the Hamming code is referred to as a terminated trellis. The trellis of an (n,k) block code has a depth n. At depth $k=0$, the trellis contains only one node, $s_0(0)$, the all zero state which is a $(n-k)$ tuple. For each $k=0,1,\ldots,(n-1)$, the collection of nodes at $(k+1)$ is obtained from the collection on nodes at depth k. Lines are drawn from nodes at depth k to the nodes at $(k+1)$. For a binary block code, the nodes at depth $(k+1)$ are obtained from the set of nodes at depth k using the equation

$$s_{(k+1)} = s_k + c_{(k+1)} h_{(k+1)},$$

where $c_{(k+1)}$ is the bit at position $(k+1)$ in the received vector, and $h_{(k+1)}$ is the $(k+1)$ column of the H matrix. For example, in the Hamming code above, the two new states at s_1 are

$$\begin{bmatrix}0\\0\\0\end{bmatrix} + 1\cdot\begin{bmatrix}1\\0\\1\end{bmatrix} = \begin{bmatrix}1\\0\\1\end{bmatrix}$$

and .

$$\begin{bmatrix}0\\0\\0\end{bmatrix} + 0\cdot\begin{bmatrix}1\\0\\1\end{bmatrix} = \begin{bmatrix}0\\0\\0\end{bmatrix}$$

Any nodes and paths that do not have a path to the all-zero state at depth n are removed. There is a one-to-one correspondence between the codes in the codeword space and the paths through the trellis. For a binary code, there are 2^k codewords, and therefore 2^k paths.

The objective of the VA is finding the shortest path through a trellis. It can be broken down into three steps:

The first step is calculating the branch metrics.

The second step is to recursively compute the paths through the trellis. Decisions are made from vertex to vertex. If two paths converge at the same vertex, the path with the lower metric is kept as the survivor path. The VA finds the shortest path through the trellis, therefore, where two paths converge, the shortest path is kept. The longer path is disqualified because it will never be the shortest path. The metric of the path at a vertex is a summation of the all the metrics leading to the vertex. This step is known as the add-compare-select (ACS) recursion stage of the algorithm.

For convolutional codes, in the final step, the metrics of all the survivor paths that reach the last vertex are compared. Depending on the metric used, the path with the lowest metric or the highest metric will be used. Assuming we are using a metric of distance between the received bit and the transition bit, we would choose the path with the smallest minimum distance at depth n of the trellis. It has the closest correlation to the received data. It is a unique path that can be traced back in time, and is the most likely signal path with the corrected codeword [76]. In block codes, the only paths that are considered, are the paths which terminate at the all zero state [75]. This is due to the fact that the syndrome of a valid codeword is at zero state. The path with the lowest metric is chosen as corrected codeword.

Wolf showed that soft decision ML decoding could be used for any linear block code (product codes included) using the VA [75]. When compared to the word correlation ML decoding algorithm, the VA offers a significant improvement as some possible codewords are discarded during the search process. The VA is practical for short codes with few vertices, because the number of vertices determines the decoding complexity of the code. The number of edges in the trellis is related to the number of vertices, and each edge is a symbol in the codeword. Long codes will have a large number of edges, and consequently a large number of vertices [30].

Generalized minimum distance (GMD) decoding

This decoding algorithm was presented by Forney in 1965 [77]. This soft decision decoding algorithm uses the reliability information of the received symbols to improve the algebraic decoding of both binary and non-binary codes. The decoding algorithm generates a list of $\left\lfloor \frac{d_{min}+2}{2} \right\rfloor$ candidate codewords based on an error-and-erasure algebraic decoding method and then selects the most likely one from the generated codewords. Assume a binary codeword is received with erasures and errors. The erasure positions can be filled with zeros. This can introduce more errors. The error-and-erasure decoding method can correct the combinations of e errors and ε erasures provided that

$$2e + \varepsilon \leq d_{min} - 1 \tag{2.50}$$

The erasure positions are considered the least reliable positions, and are the most likely positions to be in error. The decoding is carried out in 4 steps as follows:

1. Assign a reliability value to each of the symbols in the received vector v'.
2. Generate a list of $\left\lfloor \frac{d_{min}+2}{2} \right\rfloor$ sequences by modifying v'. If the minimum distance of the code is even, modify v' by erasing the least reliable symbol, then the three least reliable symbols, incrementing in odd numbers until the $d_{min}-1$ least reliable symbol. If the minimum distance is odd modify v' by erasing no symbol, then the two least reliable, and increment in even numbers till $d_{min}-1$.
3. Decode the modified codewords using an error-and-erasure algebraic decoding algorithm to generate the candidate codewords.
4. Compute the soft decision decoding metric for the candidate codewords. The candidate codeword with the best metric is chosen as the decoded solution.

Generally less than $\left\lfloor \frac{d_{min}+2}{2} \right\rfloor$ candidate codewords are generated as some of the sequences may fail at step 3 [14].

BCJR MAP decoding

The maximum a posteriori (MAP) decoding algorithm is one of the most commonly used soft decision decoding algorithms. It is also known as the BCJR algorithm, named after Bahl, Cock, Jelenic, and Raviv who proposed it originally in 1974 [30][45]. The MAP decoding algorithm can be applied to any code for which a coding trellis can be drawn [74]. It works by computing the a posteriori probability of symbols from Markov sources transmitted through discrete memoryless channels. The BCJR algorithm is similar to the Viterbi algorithm in that respect. The difference between the two algorithms is that, the Viterbi algorithm makes a hard decision at each edge in the trellis. The BCJR algorithm computes soft outputs in the form of posterior probabilities for each of the message symbols. The continuous path through the trellis formed by the Viterbi algorithm produces the maximum likelihood codeword. This means there is no way to establish the reliability of the individual symbols in the corrected codeword. The BCJR algorithm produces the most likely sequence of message bits. The sequence of symbols produced by the MAP algorithm may be disjointed. The probabilities of the message bits produced in the BCJR algorithm are used in some iteratively decoded product codes (turbo product codes).

In turbo decoding, the received codeword is MAP decoded row wise. The real values obtained from the row wise decoding are used for the column wise decoding. The new results are then used for the next row wise decoding, and the process is continued until the results concur or a predefined number of iterations is reached. Compared to the Viterbi algorithm, the MAP algorithm is computationally complex. It is also sensitive to SNR mismatch and inaccurate estimation of noise variance. For long block codes, the memory requirements for MAP-based decoding algorithms become extremely large. The complexity for the gain is a hindrance when using MAP decoding. In order to solve the complexity problem other MAP decoding algorithms have been suggested. They include a Log-MAP algorithm, MAX-Log-MAP algorithm and the Chase-Pyndiah MAP decoding algorithm.

2.7 Bit flipping decoding

The objective of this research is to study the performance of a bit flipping decoding algorithm to see if it can be used as a low complexity decoding algorithm for product codes. In this section

we discuss how a bit flipping decoding algorithm works. We base the discussion on the bit flipping decoding algorithm presented by Gallager for Low-density parity-check (LDPC) codes. LDPC codes are codes specified by a parity check matrix containing mainly 0's and few 1's. They are defined as (n, j, k) low-density codes, where n represents the block length of the low-density code matrix, j represents the number of 1's in each column of the matrix and k represents the number 1's in each row [3].

LDPC codes can be presented using Tanner graphs. Tanner graphs are bipartite graphs. This means, Tanner graphs consist of two sets of nodes. The first set of n nodes represents the n bits of a codeword. These are called bit nodes. The second set of $n-k$ nodes is a set of the parity bits of the codeword. These are known as the check nodes. The graph has edges between the bit nodes and the check nodes. A bit node has an edge with a check node if and only if that bit node is used in the computation of the check node. Consequently a Tanner graph is a graphical depiction of the parity check matrix of the code [30].

Given the following parity check matrix for a systematic (7, 4) Hamming code

$$H = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix},$$

The corresponding Tanner graph of the code is presented in Figure 2.12.

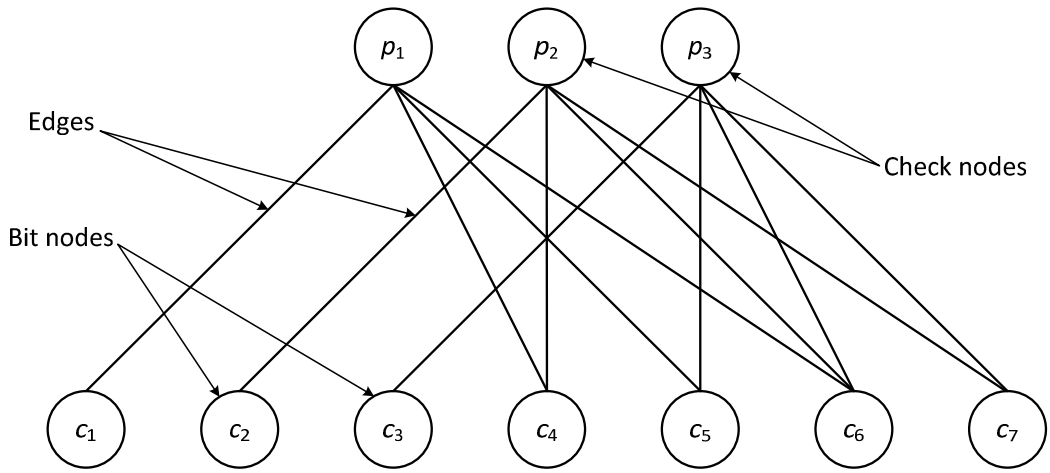

Figure 2.12: Tanner graph of a systematic (7, 4) Hamming code

Three parity equations exist for the Hamming code defined by the *H* matrix above. For this Tanner graph the following set of parity check equations can be obtained:

$$p_1 = (c_1 + c_4 + c_5 + c_6)_2 = 0 \tag{2.51}$$

$$p_2 = (c_2 + c_4 + c_6 + c_7)_2 = 0 \tag{2.52}$$

$$p_3 = (c_3 + c_5 + c_6 + c_7)_2 = 0 \tag{2.53}$$

The parity p_2 is influenced by the codeword bits c_2, c_4, c_6 and c_7.

Several decoding algorithms exist for LDPC codes. The one of most interest in this study is the bit flipping decoding algorithm proposed by Gallager. The decoding algorithm is applicable to binary codes. Gallager proposed it be used in the Binary Symmetric Channel at very low rates. The algorithm's simplicity is a very attractive feature. The decoder computes all the parity checks. Any bit that is contained in more than a defined number of unsatisfied parity check equations is flipped or the bit involved in the most number of parity check equations is flipped. Using the new values of the bits, the parity checks are recomputed and the process repeated until such a time that the parity checks are all satisfied.

Using the Hamming code stated earlier in the section, assuming the transmitted data was 1010, the codeword *c* would be 0011010. Assuming the received vector is 0010010, the parity check equations would be computed as

$$p_1 = (0+0+0+1)_2 = 1$$
$$p_2 = (0+0+1+0)_2 = 1$$
$$p_3 = (1+0+1+0)_2 = 0$$

Assuming a single error has occurred, the next objective is finding the bit that affects both parities p_1 and p_2. From the parity check equations bits c_4, and c_6 both influence parities p_1 and p_2, but c_6 can be ruled out because it also influences p_3. The solution is to flip the bit c_4.

After flipping c_4, the parity checks are computed again. This time all parities check, meaning error correction is complete.

2.8 Conclusion

This chapter provides background information on Information Theory. Section 2.2 of the chapter gives an overview of the digital communication system, showing where error control is introduced in the system. The chapter presents two error control techniques that are used in wireless communication, namely ARQ and FEC, stating the advantages and disadvantages of using them in wireless communication.

Section 2.4.2 defines FEC as a means of recovering lost information (erasures) and corrupted information (errors) from redundant information. The code rate is defined as a measure of the redundancy added to the encoded information. FEC codes can be subdivided into two subclasses block codes and convolutional codes. In this report block codes are used. Block coding schemes divide the information to be transmitted into non-overlapping blocks and each block is encoded separately. Convolutional codes were first presented by Elias in 1955. They use a sliding window when encoding. This allows the same data to be encoded a number of times depending on the size of the sliding window.

Section 2.5 gives a description of three different block codes, namely, Repetition codes, Hamming codes and BCH codes. Repetition codes are simple form of error correction coding where the message vector is repeated at least three times. Hamming codes were first presented by Richard Hamming in 1954. They are perfect codes, only capable of correcting a single code. BCH codes were discovered by Bose, Chaudhuri, and Hocquenghem. They fall into a category of block codes known as cyclic codes. They are the chosen codes for this research because they offer the flexibility of choosing the error correcting capability of the component codes.

Section 2.6 defines product codes as serially concatenated codes. Product codes were first introduced by Elias in 1954. The section defines the different parameters of product codes including the minimum distance, code rate and the fractional minimum distance. Product codes

have a larger minimum distance than their constituent codes, but they have a lower fractional minimum distance.

Section 2.6.2 gives a description of the different decoding algorithms that are used for product codes. The decoding algorithms used for block codes can be used to decode product codes. Most of the decoding algorithms used for product codes encourage decoding the rows then the columns iteratively as Elias had first suggested.

Section 2.7 presents the bit flipping decoding algorithm first presented by Gallager for LDPC codes. The section includes an example on how the algorithm can be used to decode a Hamming code. The simplicity of the algorithm makes it an attractive decoding algorithm for product codes.

Chapter 3 presents a review of the literature survey of the study. When Elias [1] presented product codes, he used the codes that had been discovered at the time, Hamming and Golay codes . He stated that product codes could be obtained from any systematic codes. Chapter 3 presents literature on cyclic product codes and single parity check product codes. The chapter also presents literature on derivatives of product codes, like array codes, augmented product codes and turbo product codes. It also presents literature on the decoding algorithms for product codes.

Chapter 3 - Literature Survey

3.1 Introduction

This chapter presents a review of the literature of the study. The literature traces the development of product codes from the time they were first proposed by P. Elias in 1954, to the single parity check product codes introduced by Battail. It is worth noting that the encoding process of product codes has not varied much from that which was proposed by Elias. Most of the research on these codes has been focused on finding better ways of decoding them. The survey also presents literature on the decoding algorithms that have been used in decoding product codes. We also consider bit flipping, a method that was presented by Gallagher as a hard decision decoding algorithm for LDPC codes. This method may also be used in decoding product codes, and a variation of the algorithm is used in this study.

3.2 Development of product codes

Generally it has been shown that the decoding complexity of a block code is directly dependent on its error correcting capability. To build a powerful block code, one has to use long block codes. It should be clarified that the term powerful in this context refers to a code with

high error correcting capability. Generally increasing the block length of a code, increases its decoding complexity [30]. One of the objectives of forward error correction is to build powerful codes yet keep the decoding complexity low. Elias proposed the use of iterated codes as a means of creating powerful codes while keeping the decoding complexity low. These codes are defined as product codes [1]. Elias proposed using simple codes, like the ones that had been discovered by Hamming and Golay, to build the product codes. Although Elias used Hamming codes when he presented product codes, his paper shows that other codes can also be used. He stipulated that the codes be of the check-digit, or systematic type, so that the check digits could be computed from the preceding information digits and appended onto the message. Elias also stated that it was not necessary to use the same type of code in each dimension of the product code.

The procedure to create a product code is to arrange the data to be transmitted into a multidimensional array, then encode the data in each dimension, encoding the parity bits as well. This creates a systematic code with some bits being checks on checks [1][78]. Initially, as stated, Elias presented product codes using Hamming and Golay codes. Later works show that product codes could be extended to other linear block codes, as long as they were systematic [79][80]. This led to the development of cyclic product codes and single parity check product codes which are discussed in Sections 3.3 and 3.4.

In 1966 Forney introduced concatenated codes. He defined concatenation as a method of building longer, powerful codes from short codes [58]. Algebraic coding focused on building codes with large minimum distances as a means of increasing error correcting capability. Concatenated codes were based on the concept of product codes that had been introduced earlier by Elias [77]. A concatenated code consists of two linear codes. The outer code is a non-binary code like a Reed Solomon code over a finite field $GF(2^k)$. The inner code is typically a binary code. When encoding, the outer code is used first. The bits are arranged into k rows of k_2 bits. The k_2 bits are encoded into RS codewords over the finite field. The resultant k tuples of binary bits are encoded using the binary inner code. Product codes are sometimes defined as serially concatenated codes [29][81]. Some decoding algorithms used for decoding concatenated codes are used for decoding product codes [29].

Another spin-off of product codes is a group of codes named array codes [1][82][83]. Array codes work on the premise of product codes, of simple component codes with low complexity

decoding methods. Array codes, like product codes, are a constructed using component codes. A single parity check product code is also known as a row-and-column (RAC) array code [82][84]. The RAC consists of a rectangular array of information bits with row and column parity checks.

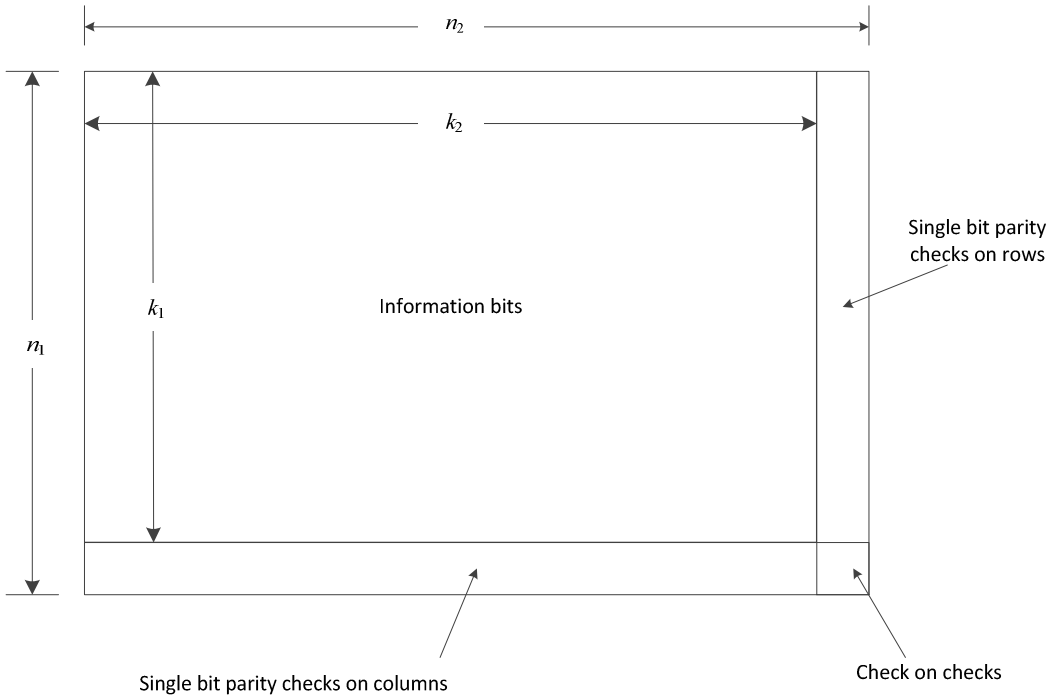

Figure 3.1: Single parity check product code or row-and-column (RAC) array code, adapted from [82]

Unlike product codes, array codes can be non-systematic. Array codes were first presented by Elias in 1954 [1][85]. They are most often binary codes. Array codes are also defined as structured LDPC codes [86].

After Elias introduced product codes in [1], a lot of research was carried out on them. It was noted that, for a given error correcting capability, the number of parity check symbols required by a product code is relatively higher than that of other linear block codes. This is a disadvantage when using product codes in practical data communication systems [87]. Augmented product (AP) codes are used to ensure high code rates while not affecting the minimum distance of the code or reducing it marginally [88][89]. A codeword is augmented by increasing the number of information symbols while keeping the length of the codeword constant. The resultant code has a higher code rate, but it might have a lower minimum distance [45]. Augmented product codes

were first presented by Goldberg in [90]. Goldberg augmented an extended Hamming code with a single parity check code to result in a product code shown in Figure 3.2.

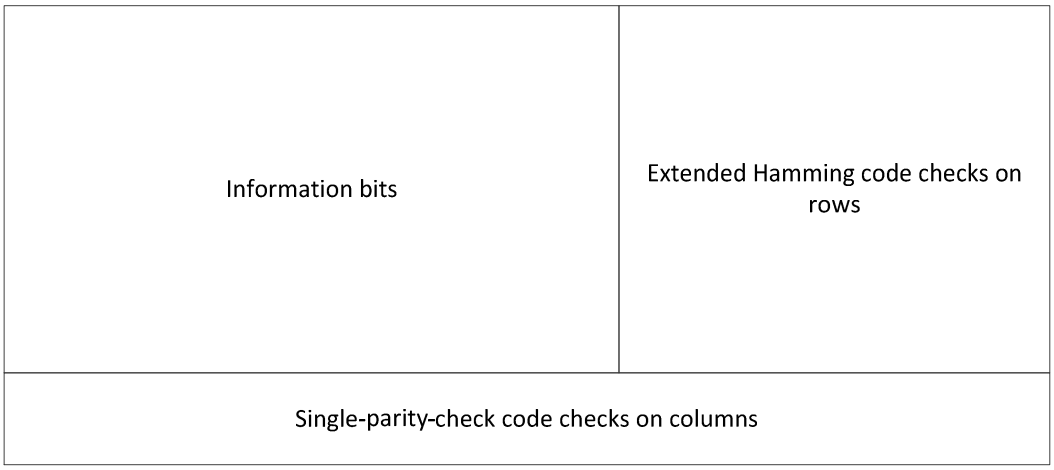

Figure 3.2: An augmented product code [90]

Turbo codes were first presented for convolutional codes. They are sometimes referred to as parallel concatenated codes. Until the introduction of turbo codes, the decoding complexity of codes was hindering the development of codes that could provide performances close to the Shannon limit [91]. Turbo codes can be distinguished from other codes by the fact that they use an iterative decoding algorithm [30]. A product code is only a turbo product code if an iterative decoding algorithm is used to decode it.

The first turbo product code was presented by Lodge *et al.* [81], in 1993 . A filtered signal is fed to the decoders, and the decoders interpret the signal amplitude to a soft decision. The *a priori probabilities* of the input symbols are used, and a soft output based on the reliability of the decision is calculated. The result is iterated between the two decoders until a reliable codeword is obtained. Lodge *et al.* [81] presented a decoding algorithm for product codes that could be generalized for multi-dimensional product codes. In their paper the algorithm is used to decode a two dimensional code. A one dimensional maximum a posterior (MAP) filter is used to obtain a refined set of probabilities row-wise. The new probabilities are then further refined by doing a column-wise decoding. The process is iterated until the probabilities have converged, meaning they are consistent, or the predefined maximum number of iterations is reached [81][30].

3.3 Cyclic product codes

Cyclic product codes were introduced by Burton and Weldon in 1965 [92][93]. Having noted the ease of implementing cyclic codes, they worked on a new sub-class of cyclic codes, namely, cyclic product codes. Cyclic code implementations either have good burst error correcting capability but poor random error correcting capabilities or good random error correcting capability but poor burst error correcting capability. The Fire codes have good random error correcting capability but poor burst error correcting capability. The Bose-Chaudhuri-Hocquenghem (BCH) codes [92] on the other hand have good burst error correcting capability but poor random error correcting capability. Burton and Weldon proposed the use of cyclic product codes as a means to address this problem in cyclic codes. Cyclic product codes could provide a compromise between random and burst-error correcting codes, thereby providing error correction for channels in which both occur like the switched telephone network. The sub-codes of product codes did not necessarily have to be the same code. Cyclic product codes also impart an algebraic structure to a subclass of cyclic codes. Burton also proposed cyclic product codes to construct codes that are powerful but with relatively low decoding complexity [94].

Given two cyclic codes C_1 and C_2, their product is a two dimensional code C whose rows and columns are codewords in C_1 and C_2. This is how Elias defined product codes. The component codes C_1 and C_2 in C maybe cyclic, but it does not necessarily follow that C will be cyclic as well [95]. Cyclic product codes were first defined by Burton and Weldon, and they proved that under certain conditions a two dimensional product code whose sub-codes were cyclic codes, could also be a cyclic code [92]. The first of the two conditions is that the lengths of the sub-codes has to be relatively prime. This was further proved by Tang and Chien in [22]. The second condition is that the mapping between locations in the two dimensional arrays and the terms in the polynomial representation of the cyclic code has to follow a defined function. In the cyclic product codes proposed by Burton and Weldon, the data is not arranged into a matrix before encoding as Elias had proposed. The data is coded, and then a mapping function used to determine the order of sending the data in the cyclic code. In [93], it was shown that there exists more than one mapping that would result in a cyclic product code. Though the cyclic product codes defined by Burton and Weldon are different in structure from those defined by Elias, they

have the same advantage. They are formulated using shorter component codes hence their decoding complexity may be considerably simpler than codes of the same length [94].

In their paper [22] in 1966, Tang and Chien recognized the need to have variable code rates in cyclic product codes. Messages in communication systems vary in importance, so there was a need to add more redundancy to the important ones and less in the ones that were deemed less important. The channel is ever-changing hence in some cases when the SNR is high, less redundancy is required. To cater for this Tang and Chien realized the need to address two problems, the first of which was selecting the codes that offer variable redundancy. The second, though not addressed in their paper, was determining efficient methods to inform the receiver of the change in the code [22]. The objective of the study was selecting efficient, variable redundancy simple codes using cyclic product codes. Burton and Weldon had shown in [92] that the generator polynomial of a two dimensional cyclic product code is obtained by finding the tensor product of the generator polynomials of the two component codes. To vary the redundancy of the product code, Tang and Chien changed the generator polynomials of the component codes. Tang and Chien proposed a two dimensional product code with two sub codes obtained from the generator polynomials $g_1(x)$ and $g_2(x)$. The high redundancy code is generated with the generator polynomial $g(x) = g_1(x) \cdot g_2(x)$. The low redundancy code is generated with the polynomial $g_1(x)$.

In 1967, Abramson showed a method for constructing cyclic product codes by interlacing cyclic codes. The data is arranged into an array, and encoded row-wise, then encoded column-wise as Elias had done. Abramson showed that the resultant code would have a Hamming distance that would be the product of the Hamming distances in each dimension [2]. In his paper Abramson showed that the decoding algorithm proposed by Elias for decoding product codes, where the decoding is done row-wise then column-wise does not decode all error patterns with errors within the error correcting capability of the code [2]. This can be illustrated using a Hamming code. The minimum distance of a Hamming code is 3. A two dimensional product code using Hamming codes as subcodes has a minimum distance of 9. In principle the code should be able to decode up to 4 errors from (2.19). If the received codeword has two errors in two rows and the errors are also in two columns resulting in an error pattern shown by the Figure

2.1, the method proposed by Elias will not correct these errors even though it is within the error correcting capability of the code. The Hamming code can only correct a single error per row or column.

Abramson defined this as a **permanent error** pattern. The decoding strategy proposed by Elias does not take advantage of the interlacing used when constructing the code. Abramson proposed using a cascade decoder. In the conventional method of constructing product codes, the data received from the communication channel is in the form of strings of digits in one dimension. Before the codeword can be decoded, it has to be mapped into the original multi-dimensional array. The original method for mapping the digits is to map the rows sequentially. Abramson proposed constructing the product code by interlacing the column and row codewords. Then the output of the communication channel would be fed directly into an interlaced decoder for the sub codes. The cascade of decoders will then result in decoding first each column then each row. Cascade decoding offers the advantage of distributing any burst errors over successive rows or column codes. This decoding method takes advantage of the structure of product codes, thereby improving the error correcting capability of the code. It allows the decoding of permanent error patterns [2]. Further literature on decoding of product codes is covered in Section 3.5.

3.4 Single Parity Check Product Codes

A single parity check (SPC) code has a single check digit appended at the end of the block. In binary codes, this digit is a modulo 2 summation of all the information digits. These are simple codes that have a minimum distance of two. They cannot correct any errors, rather they can detect the presence of one error. Bahl and Chien introduced SPC product codes in 1971. They used SPC codes as constituent sub-codes for cyclic product codes. The focus of their study was on single and multiple burst error correcting capabilities of cyclic codes [96]. The idea of using SPC codes in product codes was also looked into by Rao and Reddy [79]. Rao and Reddy presented a code that had the highest code rate with a codeword length of 48 and a minimum distance of 8. They accomplished this by combining a (16,11,4) extended Hamming code with a (3, 2, 2) SPC code to produce a (48, 22, 8) augmented product code. The motivation of their

work was finding high rate codes with a high error correcting capability using augmented product codes [89]. Their work inspired other researchers to look into improving the fractional minimum distances of product codes [87].

The desire to create long codes with a low decoding complexity led Battail [97] to investigate the use of the SPC codes as constituent product codes in 1989. Battail used Elias's iterated approach to combine SPC codes to create single parity check product codes [97]. To decode the product code, Battail used successive soft-out decodings. The performance of the code was good for the minimum distance of the code. The minimum distance of a SPC code is two, and a two dimensional product code would have a minimum distance of four. Battail concluded that the minimum distance is not a sufficient criterion to be used for judging the performance of long binary codes, rather to look at the distance distribution. He proposed that long codes with a distance distribution close to random codes would perform well despite the low minimum distance. Battail did not formally prove that the iterated product of SPC codes asymptotically approaches the average distance distribution of random coding [97].

In [97], Carie *et al*. studied further what Battail had done years earlier. Initially Battail proposed using different length SPC codes, but Carie *et al*. used SPC codes of equal length. When decoding the SPC product codes, Battail proposed decoding the shortest dimensions first using weighted-output decoding. The decoded codeword provides both the decisions and reliability estimates for decoding the next longer codeword. This is done until all the codewords had been decoded. The decoding is not iterated though. The reason for decoding the shorter codewords first is because shorter codewords are less susceptible to noise. They will have fewer errors than longer codewords. Decoding the shorter codewords makes it possible to decode the longer codewords [97].

The decoding algorithm proposed by Battail works for product codes whose sub-codes are not equal in length. In [97], Carie *et al*. proposed a new decoding algorithm to cater for single parity check product codes whose were of equal length. The algorithm is based on weighted threshold decoding. The decoding algorithm uses an iterated replication decoding algorithm. This improved the results of the coding scheme to those obtained by Battail. Replication decoding is an iterative symbol-by-symbol MAP decoding algorithm. It is used for redundantly encoded messages. The decoding algorithm is optimal when all symbols are taken into account [98].

The general structure of SPC product codes remained the same until a new structure was proposed by Xu and Takawira in 2004 [37]. In the conventional SPC product code, a single parity bit is appended at the end of each row and column. The resultant codeword has an even number of ones in every row and column (i.e. modulo-2 addition is zero). The conventional structure of a SPC product code is shown in Figure 3.1. To decode the codeword, iterative decoding is used. In [99], Kousa investigated the erasure decoding capability of SPC product codes. To decode erasures, row-wise and column-wise iterative decoding is used. Using parity checking allows for recovery of data from single erasures. If more than one bit is erased in the row or column, the row or column is skipped. Decoding is performed in many iterations until all the erasures have been recovered or the erasure pattern does not permit further decoding [99].

In [37], it was noted that the decoding method proposed by Kousa could not recover errors in a sub-pattern such that every occupied row and every occupied column in that sub-pattern has at least two erasures. The sub-pattern is shown in Figure 3.3(b). This pattern had been identified as a permanent error by Abramson in [2].

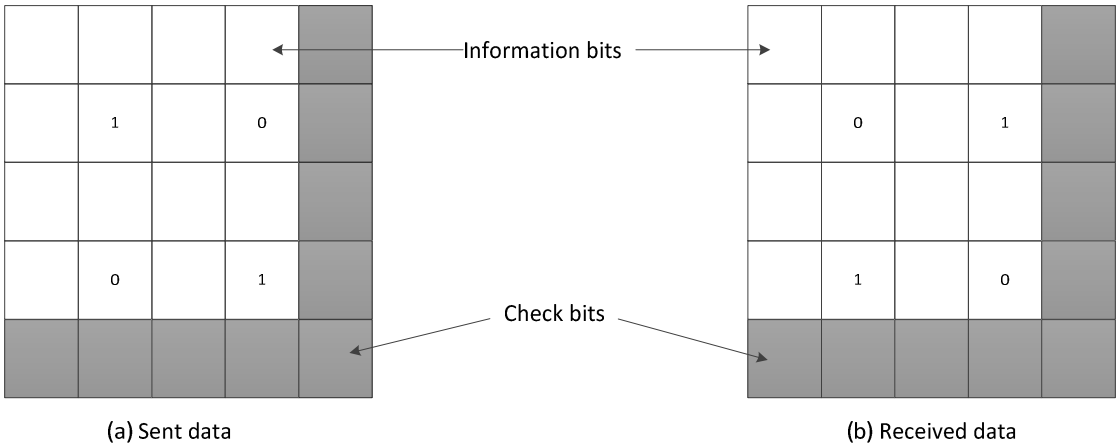

Figure 3.3: An example of a permanent error [37]

In [37], it was noted that the error pattern could be recoverable if more parity check equations were used and the parity equations covered any of the bits in the error pattern. Xu and Takawira were concerned that increasing the number of parity check equations would impact negatively on the code rate of the product code. SPC were developed because conventional product codes have a low code rate compared to linear codes of the same error correcting capability [100]. To ensure the code rate of the system remained the same as a conventional SPC code, more than *n*

information bits could be used to obtain each parity bit. From the message passing algorithm, it is preferable to have a parity bit with a low degree, calculated from few parity bits. In [37], the parity check bit for the product code is generated by using every two rows and every two columns of *n* information bits. The structure of the code is shown in Figure 3.4.

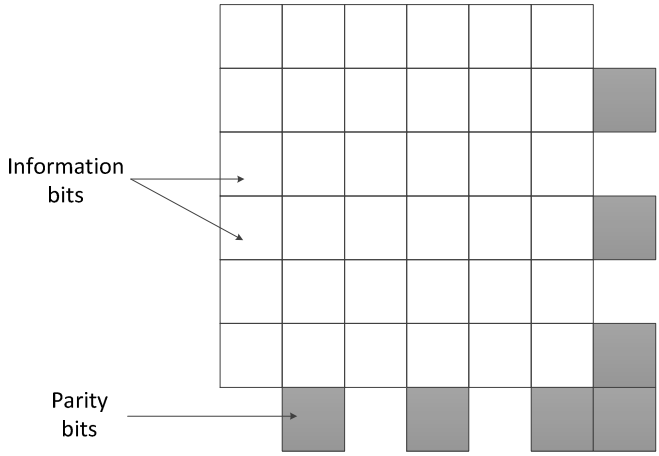

Figure 3.4: SPC-Product code proposed by Xu and Takawira [37]

The SPC product code proposed by Xu and Takawira can only be two dimensional [37].

3.5 Decoding product codes

Decoding is the most complex part of the FEC system. The decoder in the FEC system attempts to determine the location of the errors from the discrepancies between the received vector and the recalculated parity bits [40]. Most of the decoding algorithms used for block codes can be used for product codes. This is because product codes can be decoded by decoding their component codes row-wise and column-wise. Few decoding algorithms look at decoding the entire codeword as a whole , due to the fact that, decoding the constituent codes first ensures the decoding complexity of the system is kept low, which was the objective of Elias presenting product codes [1]. This section provides a literature survey on the decoding algorithms used for product codes.

The first decoding algorithm for product codes, is the algorithm that was proposed by Elias in [1]. The component codes used to encode the rows and columns are used to decode the row and

column codewords as long as the errors are within the error correcting capability of the component codes. This decoding algorithm has been deemed effective by [14], but it is not optimal. The received vector is decodable if and only if the error pattern obtained after decoding the rows is within the error correcting capability of the column component codes. Breaking the decoding into row-wise and column-wise decoding processes makes decoding manageable for large blocks of code [77]. Studies, for instance [2] and [37], have shown that this method of decoding does not always correct errors within the error correcting capability of the code for certain error patterns. One such pattern is the permanent error pattern illustrated by Figure 3.3. A lot of research has been carried out on finding efficient decoding algorithms of product codes. Some of the algorithms are adapted from algorithms proposed for other codes.

3.5.1 Majority-logic decoding

Majority-logic (MLG) decoding is also known as threshold decoding [101]. The first majority logic decoding algorithm was presented by Reed in 1954, for Reed-Muller codes. The simplest of these decoding algorithms is the one used in the replication code [102]. In MLG decoding the multiple estimates of a bit value are obtained, and the decoded value is the value which occurs in the majority of estimates. MLG decoding is an attractive decoding algorithm for cyclic codes because of its simplicity to implement. It became more attractive as more cyclic codes were discovered [103]. MLG has been mainly used in product codes that have cyclic codes as component codes.

One of the first implementations of MLG decoding for product codes was by Bahl and Chien in 1969 [104]. Bahl and Chien were investigating the multiple burst error correcting capabilities of cyclic product codes. Their cyclic product code is based on SPC codes of relatively prime block lengths. The properties of cyclic product codes had been presented by Burton and Weldon in [92], and Calabi and Haefeli in [105]. Bahl and Chien proved that multi-dimensional cyclic product codes can be decoded using MLG decoding. In [106], Gore used extended Reed-Solomon codes as the constituent codes of the product code. The resultant product code was an extended generalized Reed-Muller code which was MLG decodable [106]. Lin and Weldon in [94], generalized that the product of two MLG decodable cyclic codes is also MLG decodable,

provided that one of the component codes is one-step decodable. In [94], BCH codes were used as the component codes. Lin and Weldon also proposed an algorithm for deriving the majority-logic parity checks of the product code. In [107], the MLG decoding algorithm proposed in [94] was improved. The improvement consisted of using the parity checks of the column code and incorporating them in the other checks for the product code. The decoding algorithm was still restricted to product codes that consisted of only MLG decodable component codes. Reddy [108] proposed an algorithm that decodes product codes using MLG decoding if only one of the component codes is MLG decodable. Reddy presented the decoding algorithm to improve on the decoding algorithm proposed by Elias. Reddy adapted the decoding algorithm from one proposed for some convolutional codes in [109].

3.5.2 Maximum Likelihood Decoding

A true maximum likelihood (ML) decoder works by correlating the received vector sequence to all the possible codewords, and choosing the codeword where the correlation discrepancy is lowest. The computation is done iteratively, thereby achieving an exhaustive search of the code space. For large codes, this can be computationally intensive as the received vector has to be compared to all the possible codewords [110]. ML decoding can be achieved by finding the shortest path through a code trellis. The Viterbi algorithm provides an efficient solution for finding the most likely path through a trellis [111]. The algorithm was proposed in 1967 by Andrew J. Viterbi as a decoding algorithm for convolutional codes [72].

In [75], Wolf showed that the Viterbi algorithm can be used for decoding any (n,k) linear block code over $GF(q)$ with a trellis with no more than $q^{(n-k)}$ states. Wolf showed that a $(n_1 n_2, k_1 k_2)$ product code requires $q^{k_1(n_2-k_2)}$ states. By symmetry, the algorithm requires $q^{k_2(n_1-k_1)}$ states. Wolf stated that if one of the codes was a low-rate code and the other a high-rate code, the number of states required improves on the decoding complexity of the code when compared to algorithms that require $q^{k_1 k_2}$ states [75]. Wolf's paper also shows that for long block codes, the number of states required is large thereby making the decoding complexity too high.

Trellis decoding for product codes was proposed by Lodge *et al*. in [81]. Lodge noted that the symbol-by-symbol MAP algorithm could be used by any code that could be represented by a trellis of finite duration. Decoding trellises existed for convolutional codes as well as for linear block codes [75]. Lodge *et al*. note that creating a single trellis for an *N*-dimensional code requires excessive computation, rather it would be better to use separable digital filters, thereby a trellis for each row or column. One dimensional MAP filters are used sequentially in each dimension of the product code. Considering a two dimensional product code, one dimensional MAP filtering is done across the rows, giving a set of refined probabilities taking into account only the horizontal structure of the code [81]. The probabilities are further refined by performing a one dimensional filtering along the columns. This is a single filtering cycle. This process can be iterated, thereby improving performance. A MAP filter computes the *a posteriori* probabilities of the coded bits whereas a MAP decoder computes the *a posteriori* probabilities of the decoded bits.

3.5.3 Generalized Minimum Distance decoding

Generalized Minimum Distance (GMD) decoding was first presented by Forney in [77] and [112]. The GMD decoder makes use of the fact that an erasure-and-error decoder is capable of correcting twice as many erasures as errors. The algorithm works by deliberately erasing the least reliable symbols then correcting them using an erasure-and-error decoder [61]. A number of candidate codewords are obtained. The candidate codewords are checked to see if they are within the minimum distance of the received word according to a generalized minimum distance criterion. The one that lies within the criterion is the unique required codeword [112].

In 1972, Chase proposed three soft decision decoding algorithms that are also based on the GMD algorithm. They are referred to as the Chase-1, Chase-2 and Chase-3 decoding algorithms. [14][113]. The algorithms Chase presented are based on the GMD proposed by Forney. Chase showed that they could be applied to all block codes [113]. Chase stated that the objective of binary decoding is finding the codeword that differs in the least number of places from the received sequence. The binary decoder will find a unique codeword if the proposed codeword differs from the received sequence by a number of bits within the error correcting capability of

the code. The first step of Chase's algorithm is to determine the positions of the least reliable bits of the received sequence using the channel measurement information. Each bit in the received sequence has a corresponding confidence value on the reliability of that bit. The algorithms differ in the number of least reliable bits selected. In the Chase-1 algorithm, $d_{min}-1$ bits are selected, in the Chase-2, $\left\lfloor \frac{d_{min}}{2} \right\rfloor$ bits are selected. In the Chase-3 algorithm $\left\lfloor \frac{d_{min}}{2}+1 \right\rfloor$ candidate codewords are generated [30].

The Chase-2 algorithm has the best performance of the three. Error patterns are defined with all the possible errors confined to the $\left\lfloor \frac{d_{min}}{2} \right\rfloor$ least reliable bits of the received vector. A modified vector of the received vector is formed by complementation of the received vector for each error pattern. The modified vector is then corrected using an error-correction-only algebraic decoder to produce a candidate codeword. A soft decision decoding metric is then computed for the generated candidate codewords. The codeword with the best metric is selected as the final candidate codeword [14],[29],[113].

In 1994, Pyndiah *et al.* presented an improvement of the Chase algorithm for decoding product codes [29][114]. They noted that large powerful codes like product codes cannot be decoded using one step as it would be computationally intensive. It is easier to decode large product codes by decoding their component codes first. The authors proposed a decoding algorithm for product codes that was based on Chase's decoding algorithm. On receiving the vector sequence, the decoder performs soft decision decoding of the rows. The soft decision decoding is performed using Chase's decoding algorithm. Unlike the Chase algorithm that yields binary values, the Pyndiah algorithm yields a soft output. The soft output is then subtracted from the soft input to yield extrinsic information. A factor of the extrinsic information is added to the original information, and this is used as soft input for the decoding of the columns. This process is iterated [29][114].

3.5.4 The Bit flipping decoding algorithm

In 1962, when Gallager presented LDPC codes, he presented two decoding algorithms for them. The first decoding algorithm was a hard decision decoding algorithm. It could only be applied to a binary symmetric channel at rates far below capacity [3]. In this decoding scheme, the decoder computes all the parity checks. It then changes or flips the digit that is contained in more than a fixed number of unsatisfied parity check equations. The process of computing the parity checks and flipping digits is done iteratively until all the parity checks are satisfied. Researchers agree this bit flipping (BF) decoding algorithm presented by Gallager is simple, but performs poorly when compared to the Belief propagation algorithm. The BF algorithm, though, offers a better trade-off between error performance, complexity and decoding speed or latency [115].

Most research on the BF algorithm has been on improving the performance of the algorithm to make it competitive to the belief propagation algorithm. In 2001, the weighted BF decoding algorithm was presented as a means of improving the performance of the BF algorithm by Kou *et al*. [116]. Reliability information is added to the received vector sequence. An error pattern vector is then created based on a weighted check sum of the code bit. The least reliable bit is flipped. The least reliable bit is the bit with largest weighted checksum. The weighted BF algorithm is a soft decision decoding algorithm [116]. The complexity of the weighted BF is greater than that of the original BF algorithm presented by Gallager, but so is its performance. Further research on the BF decoding algorithm has yielded other BF algorithms like the bootstrap weighted BF algorithm and the modified weighted BF algorithm [31][117]. To further improve the performance of BF algorithms the parallel weighted BF (PWBF) algorithm was proposed in [36]. The earlier BF algorithms flipped only one bit at a time. The PWBF flips a number of bits at the same time to enable faster convergence.

3.6 Conclusion

Section 3.2 covers the development of product codes. The section presents codes that are spin-offs from product codes like array codes, and concatenated codes. The SPC product code is sometimes referred to as an array code. Block product codes are sometimes defined as serially

concatenated codes. The section also presents information on how product code augmentation is been used to improve the code rate of product codes. Product codes have been shunned because of their low fractional minimum distance.

Sections 3.3 and 3.4 present cyclic product codes and SPC product codes. Cyclic product codes were first investigated by Burton and Weldon, because they could offer a compromise between random error correction and burst error correction. Burton and Weldon did not use the method proposed by Elias to encode the data by first arranging it into a matrix. They used mapping functions to order the transmission of the data. Battail combined the simplest codes, single parity check codes, to present SPC product codes. He showed that the weight distribution of the codes was a better measure of the error correcting capability of the codes rather than the minimum distance. Battail's SPC product codes had the added benefit of high code rates.

Section 3.5 presents the different decoding algorithms that can be used for product codes. The decoding algorithms used for block codes can be used for decoding product codes, because decoding of product codes is usually carried out by decoding of the component codes. Many researchers have proved that the decoding algorithm proposed by Elias can fail to decode an error pattern known as a permanent error. Various researchers have proposed different ways for solving the problem. Takawira and Xu proposed changing the structure of the SPC product code to solve the permanent error.

Section 3.5.4 of the chapter presents the development of the bit flipping decoding algorithm first presented by Gallager for LDPC codes. The algorithm has been used mainly for LDPC codes, because of the manner in which they are structured. A single bit affects more than one parity check. This enables the determination of the bit in error by using the parity checks that fail. The algorithm is simple to implement. Little research has been done on using bit flipping for product codes other than single parity check product codes. It is used in LDPC codes because it offers a good compromise between decoding complexity and performance.

Chapter 4 presents the new decoding algorithm for product codes. The algorithm is based on finding the Candidate Error Matrix, a matrix that shows the bits that are probably in error and flipping the bits in the shadow area. The objective of the algorithm is to converge the shadow area. Two types of error patterns were identified when the algorithm was developed, namely the

simple error pattern and the complex error pattern. The simple error pattern is identified by that, when flipping all the bits in the shadow area, the shadow area becomes smaller. The complex error pattern is identified by that, when flipping all the bits in the shadow area, the shadow area remains the same. Chapter 4 explains how the two error patterns are flipped in order to correct the errors in a binary product code.

Chapter 4 – Methodology

4.1 Introduction

This chapter presents the methodology used to carry out the research. The chapter presents a novel bit flipping decoding algorithm for binary product codes. The bit flipping decoding algorithm was first presented by Gallager for LDPC codes. This research is based on binary product codes, hence in this chapter the product codes referred to, are two dimensional binary product codes unless otherwise stated. Every bit in a product code affects the same number of parity checks as the dimensions of the product code. The proposed decoding algorithm takes advantage of this to correct errors using a simple algorithm.

4.2 Proposed decoding algorithm

Generally, the decoding complexity of block codes increases with their increase in length. Let r represent the received vector corresponding to the codeword c. Due to noise in the channel, r may not be identical to c, instead, the received vector is most likely to be equal to c plus an error pattern or error vector $e = (e_0, e_1, e_2, \cdots, e_{n-1})$ caused by the channel noise

$$r = c + e \tag{4.1}$$

where $e_i = 1$ if $r_i \neq c_i$ and $e_i = 0$ if $r_i = c_i$ [41][118]. The objective of decoding is two-fold. The first objective is to determine if the error pattern e is the zero vector. This is known as error detection. The second objective is to find the most probable error pattern e so that the original transmitted codeword can be recovered. This is known as error correction. Error correction is more complex than error detection [61]. Various error decoding algorithms have been proposed. An objective of developing decoding algorithms is correcting all the errors that will occur during communication while keeping the decoding complexity low.

4.2.1 Finding the Candidate Error Matrix

Decoding of block codes can be done using the parity check matrix. Generally this type of decoding consists of two stages. The first stage checks whether the received vector sequence corresponds to a valid codeword. This stage is actually an error detection stage. Error detection involves deciding whether all the parity checks are satisfied in the received sequence. For block codes, error detection can be done using syndrome checking using (2.23). The result is the syndrome, of the received vector sequence. The syndrome is equal to zero if the received sequence is a valid codeword, but not equal to zero if the received sequence has errors. This stage is likely to be the easier of the two stages in decoding [41].

The second stage in decoding is in determining the position and the magnitude of the errors in the received sequence. This stage is generally the more complex of the two stages. Finding a simple algorithm to carry out the second stage could improve the decoding complexity of block codes. In this project we present a simple decoding algorithm that is based on syndrome checking to declare erasures in product codes. In a product code the information is arranged into an array and the rows and columns are encoded separately. The resultant codeword is actually a collection of codewords. Each row or column is a codeword. By using syndrome checking on the rows and columns, erasures can be declared in the product code.

Assume a $(n_1 n_2, k_1 k_2)$ product code, we can define the following terms:

Definition 4.1: Row Parity Check Vector $\boldsymbol{X}$ of dimension $n_2 \times 1$

$$\boldsymbol{X} = \left[x_1, x_2, \ldots, x_{n_2} \right]^{\mathrm{T}}, \text{ where } x_i = \begin{cases} 0, \, if \;\; s_i = 0 \\ 1, \, if \;\; s_i \neq 1 \end{cases},$$

(4.2)

and s_i is the syndrome of the received row i.

Definition 4.2: Column Parity Check Vector $\boldsymbol{Y}$ of dimension $1 \times n_1$

$$\boldsymbol{Y} = \left[y_1, y_2, \ldots, y_{n_1} \right], \text{ where } y_i = \begin{cases} 0, \, if \;\; s_i = 0 \\ 1, \, if \;\; s_i \neq 1 \end{cases},$$

(4.3)

and s_i is the syndrome of the received column i.

From the Row Parity Check Vector and the Column Parity Check Vector, the Candidate Error Matrix can be obtained.

Definition 4.3: The Candidate Error Matrix $\boldsymbol{M}$ is a matrix that shows the possible locations of the errors in the received vector sequence.

The Candidate Error Matrix $\boldsymbol{M}$ of dimensions $n_2 \times n_1$ is defined as

$$\boldsymbol{M} = \boldsymbol{X} \times \boldsymbol{Y} .$$

(4.4)

$$\boldsymbol{M} = \begin{bmatrix} m_{(1,1)} & m_{(1,2)} & m_{(1,3)} & \cdots & m_{(1,n_2)} \\ m_{(2,1)} & m_{(2,2)} & m_{(2,3)} & \cdots & m_{(2,n_2)} \\ m_{(3,1)} & m_{(3,2)} & m_{(3,3)} & \cdots & m_{(3,n_2)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ m_{(n_1,1)} & m_{(n_1,2)} & m_{(n_1,3)} & \cdots & m_{(n_1,n_2)} \end{bmatrix},$$

$$\text{where } m_{(i,j)} = \begin{cases} 1, \text{ if } c_{(i,j)} \text{ is possibly in error} \\ 0, \text{ if } c_{(i,j)} \text{ is not in error} \end{cases}.$$

(4.5)

The ***M*** matrix is obtained by deploying syndrome checking on the rows and columns. Some ambiguity exists in ***M***. In some cases more than one error pattern could result in the same parity checks failing, resulting in the same Candidate Error Matrix ***M***.

Supposing the received sequence is as shown in Figure 4.1

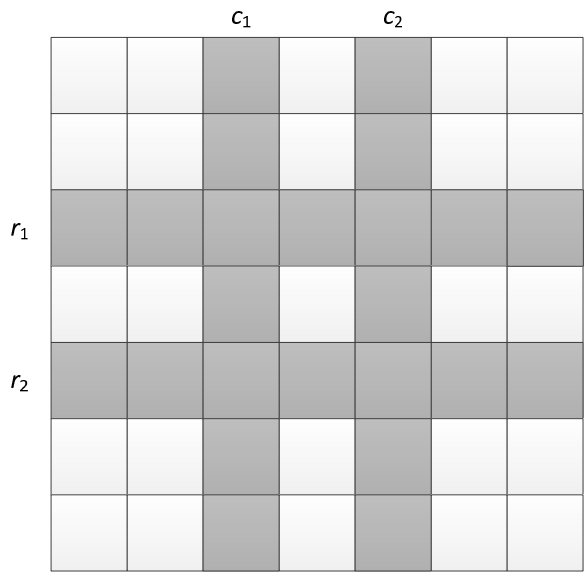

Figure 4.1: Erasure declaration using syndrome checking

The syndromes fail for rows r_1, r_2 and also for columns c_1 and c_2. The values of $\boldsymbol{X}$ and $\boldsymbol{Y}$ are as follows

$$\boldsymbol{X} = [0,0,\ldots,x_{r_1}=1,0,0,\ldots,x_{r_2}=1,0,0,\ldots,0]^T \text{ and}$$

$$\boldsymbol{Y} = \left[0,0,\ldots,y_{c_1}=1,0,0,\ldots,y_{c_2}=1,0,0,\ldots,0\right]$$

For example, if the component codes are (7, 4) Hamming codes, and assuming r_1 is the third row and r_2 is the fifth row, then

$$\boldsymbol{X} = [0 \quad 0 \quad 1 \quad 0 \quad 1 \quad 0 \quad 0]^{\mathrm{T}}$$

and if c_1 is the third column and c_2 is the fifth column, then

$$\boldsymbol{Y} = [0 \quad 0 \quad 1 \quad 0 \quad 1 \quad 0 \quad 0]$$

thus, yielding

$$\boldsymbol{M} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

The ones in $\boldsymbol{M}$ correspond to the positions of the possible errors. The Candidate Error Matrix does not show the exact positions of the errors, rather it shows the possible locations of the errors. In Figure 4.1, the syndromes fail for rows, r_1 and r_2 and for columns c_1 and c_2. This indicates that for the set of indices $\{(r_1,c_1),(r_1,c_2),(r_2,c_1),(r_2,c_2)\}$ two or more errors are present. For the following error combinations, the same parity checks will fail, (yielding the same $\boldsymbol{M}$ matrix):

- for two diagonal error combinations, i.e (r_1,c_1) and (r_2,c_2) or (r_1,c_2) and (r_2,c_1),
- any of the four combinations of three errors, and
- when all four positions are in error.

The last of these cases was defined as a permanent error by Abramson in [2]. He proposed using cascade decoding of product codes to solve for permanent errors. Xu and Takawira proposed a new type of single parity check product code to solve permanent errors in [37].

4.2.2 Error patterns

Having identified the possible positions of the errors, a new decoding algorithm for correcting the errors is proposed. The positions which are suspected to be erroneous are from henceforth referred to as the shadow area.

Definition 4.4: The shadow area is the area indicated in the Candidate Error Matrix $\boldsymbol{M}$ by ones.

For example in Figure 4.1 the shadow area is the four possible positions with errors (r_1, c_1), (r_1, c_2), (r_2, c_1) and (r_2, c_2) shown in Figure 4.2

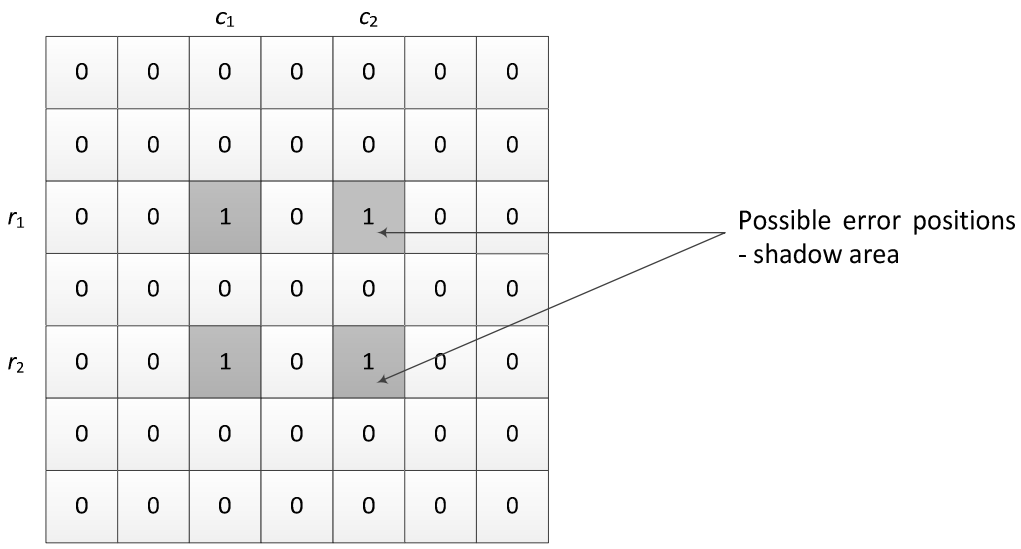

Figure 4.2: Candidate Error Matrix showing shadow area

The objective of the decoding algorithm is to decrease the number or errors by flipping the bits in the shadow area iteratively until the shadow area is zero. The flipping of the bits is dependent on the error pattern. Two types of error patterns exist. Before presenting our decoding algorithm it is necessary to explain these error patterns

Definition 4.5: Simple error pattern, an error pattern in which after flipping all the bits in the shadow area, the new shadow area is smaller. This the first type of error pattern that was identified when developing the decoding algorithm. For example in the case of the Candidate Error Matrix from Figure 4.1, this would be the situations where

- there are three errors in any combination, and
- all four positions have errors.

Assuming we have three errors in our received vector sequence and they are distributed in the manner shown in Figure 4.3a.

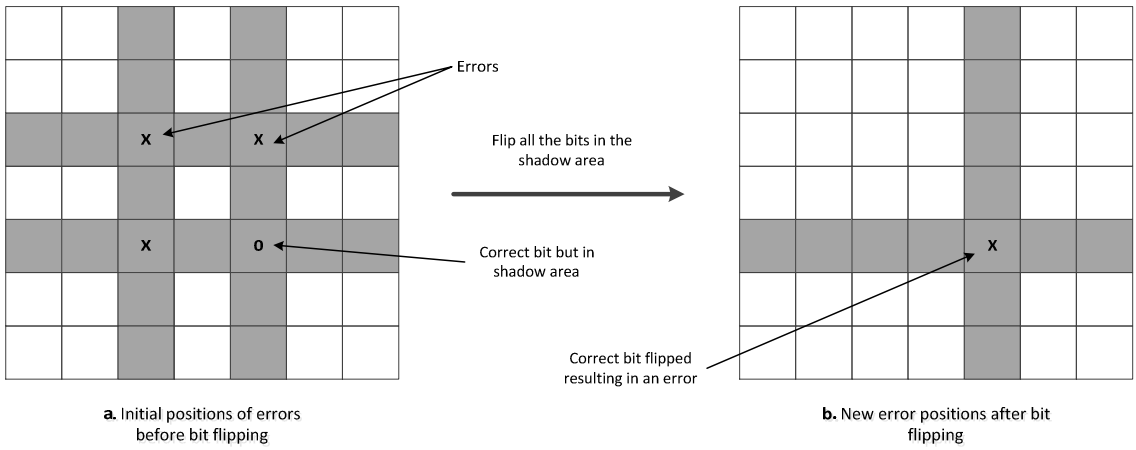

Figure 4.3: A simple error pattern

From syndrome checking we note that for two rows and two columns the parity checks fail. This means we have a minimum of two errors or a maximum of four errors as stated in Section 4.2.1. All the bits in the shadow area are declared as erasures and flipped. The resultant sequence has created a new error as shown by Figure 4.3b, but has corrected the initial three errors. The new pattern is simpler than the initial one, as we have only one row and one column with failed parity checks. Simple patterns are patterns that result in a simpler or smaller pattern after flipping all the bits. Another simple pattern is illustrated in Figure 4.4. After flipping all the bits in the shadow area, all the errors in the received sequence have been corrected. All the row and column parity checks pass, and the new sequence has no errors

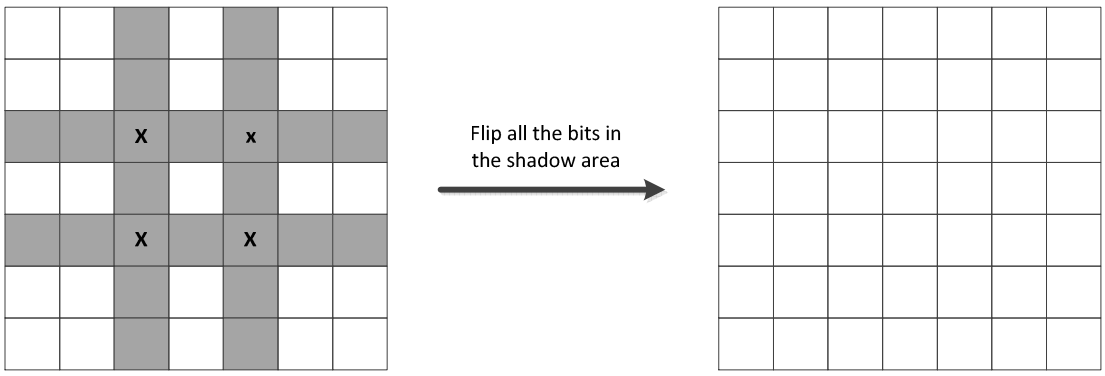

Figure 4.4: A simple error pattern

.

Definition 4.6: Complex error pattern, a pattern where flipping the bits in the shadow area results in a new shadow area which is equal in size to the original one. In the case of the Candidate Error Matrix from Figure 4.1, this would be in situations where the errors are diagonally across each other. This type of error pattern is illustrated in Figure 4.5.

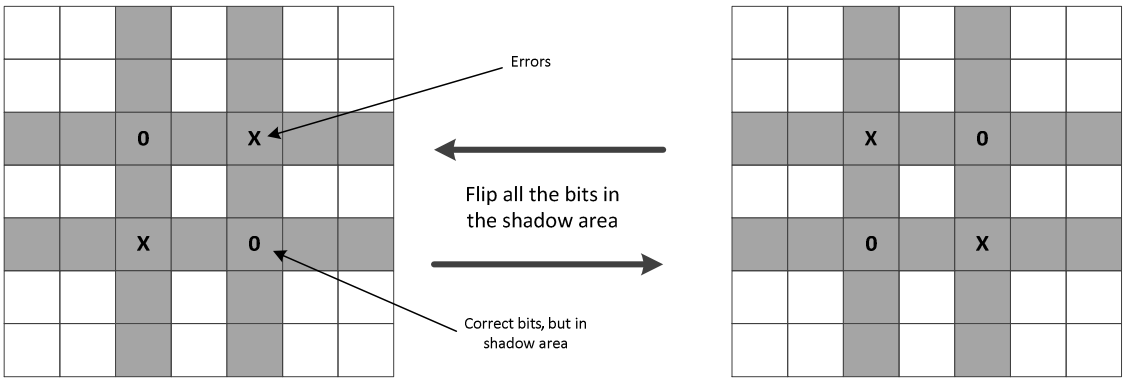

Figure 4.5: A complex error pattern

Flipping all the bits in the shadow area does not decrease the shadow area in both cases, as the cases are a mirror image of each other. The complex error pattern requires a different way of selecting which bits to flip.

To solve the complex error pattern, a combination of a number of rows or columns that have been determined to have errors, is flipped. A value known as the bit flipping complexity is used to keep track of the number of rows or columns being flipped. The bit flipping complexity is incremented from one row or column at a time, to a combination of at most half the total number of rows or columns with errors

Considering the complex error pattern in Figure 4.5, when the bit flipping complexity is 1, the first row is flipped. The result is a simpler pattern covering a smaller shadow area as shown in

Figure **4.6.**

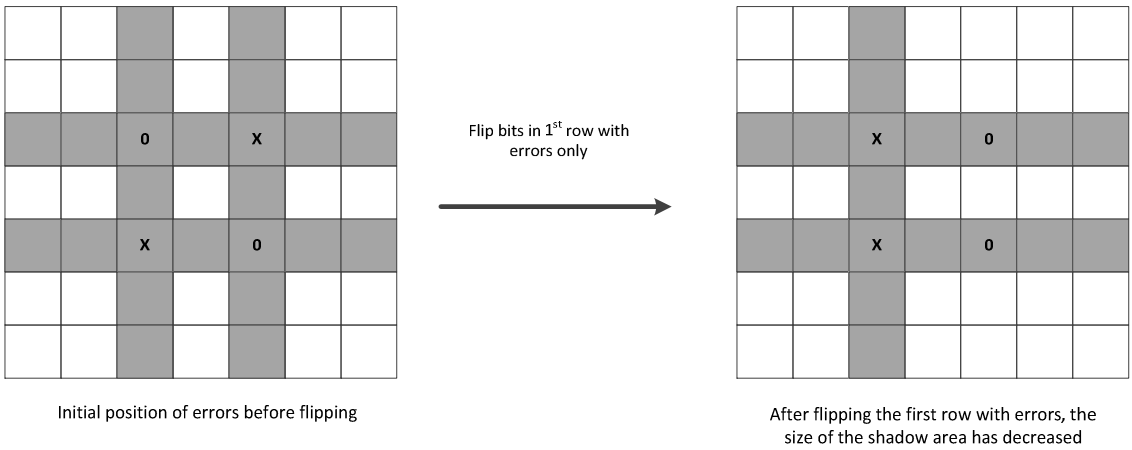

Figure 4.6: Combination flipping for a complex error pattern

4.2.3 Bit flipping algorithm

The objective of the algorithm is to decrease the shadow area until no errors can be detected. At that point, the assumption is, all the errors would have been corrected and the final vector will be identical to the initial codeword that was sent. Figure 4.7, gives an overview of the proposed bit flipping decoding algorithm.

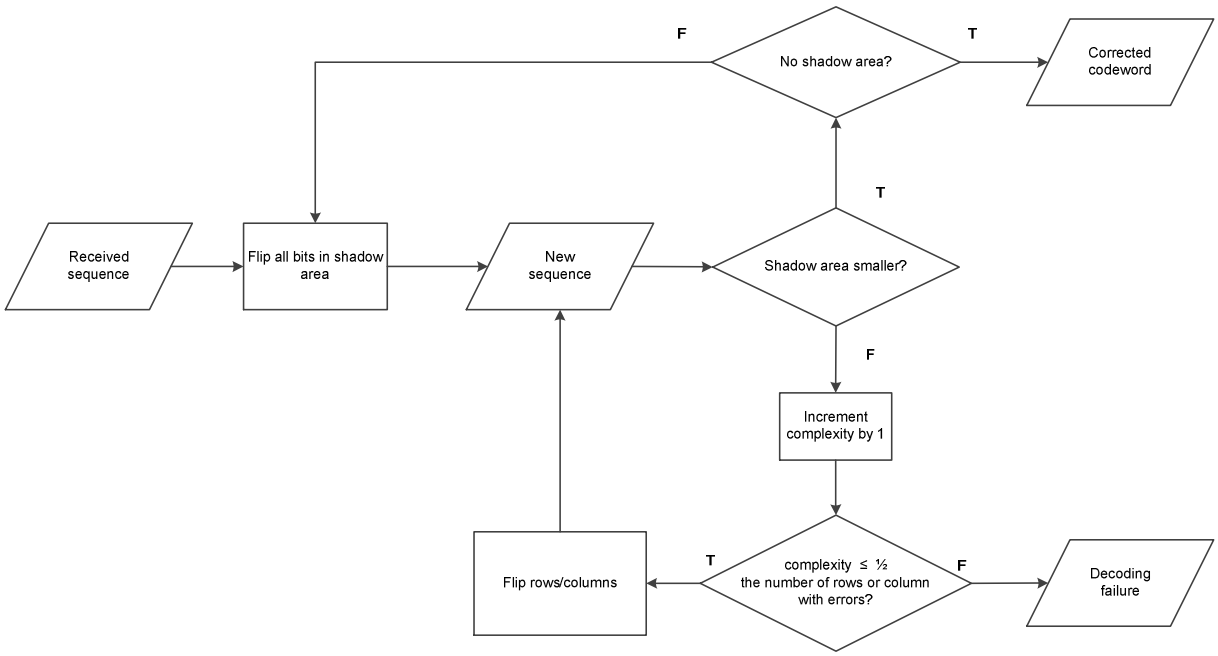

Figure 4.7: Overview of bit flipping decoding algorithm

The syndromes of the rows and columns are checked to compute the Candidate Error matrix and obtain the shadow area. All the bits covered by the Candidate Error matrix in the received vector sequence r are flipped using a bitwise XOR function as follows

$$r' = M \oplus r \tag{4.6}$$

where r' is the new sequence and M is the Candidate Error Matrix for the sequence r. A new Candidate Error Matrix M' is computed from the new sequence. If M' is identical to M it implies that the error pattern in r is a complex error pattern. If M' is not equal to M it implies the error pattern in r is a simple error pattern. If flipping the bits has decreased the shadow area, then the operation is repeated with r' and M' used as input for (4.6). This process is repeated every time M is not equal to M' after flipping the bits.

After flipping the bits, if M' is identical to M, we compare the number of rows to the number of columns that have errors. If the number of rows with errors is less than the number of columns, we flip the bits covered by the shadow area, by flipping a single row at a time. On the other hand, if the number of rows with errors is more than the number of columns, we flip a single column at a time. Every time we flip a row or column we compare M' to M to see if they are identical. If they are, we flip the next row. If they are not, it means the shadow area has decreased, and we start the process afresh by flipping all the bits in the shadow area. If after flipping all the rows and the shadow area has not decreased, we flip the columns one by one checking the shadow area after flipping each column. If the shadow area has remained the same after flipping the column we flip the next. If the shadow area decreases after flipping a column, we start the process from the beginning by flipping all the bits in the new shadow area.

If after flipping one row or column at a time and the shadow area has not decreased, we flip a combination of rows or columns. Initially we flip a combination of two rows or columns and increase the number of rows or columns in the combination while the combination of rows or columns is less than or equal to half the total number of rows or half the total number of columns with errors. After trying all the combinations and the vector still has a shadow area, it means the decoding algorithm has failed to correct the errors. Other means are then used to recover the codeword, like those in Section 2.4.1.

The algorithm can be summarized by the pseudo code in Algorithm 4.1

Algorithm 4.1: Iterative decoding of binary product codes using bit flipping

---

Input: received sequence r,

Initialization: find M from the received vector sequence

While we have a shadow area, and all combinations have not been tried

Flip bits covered by M to get a new vector,

$$r' = M \oplus r$$

Find new Candidate Error Matrix M' based on r'

If M' is not identical to M then

Save the new vector to be the received vector,

$$r = r'$$

Save the new Candidate Error Matrix to be

$$M = M'$$

Else the error pattern is complex

Flip combinations or rows or columns (see Algorithm 4.2)

End if

End while

End simple flipping

The flipping of combinations of rows or columns is covered by the pseudo code in Algorithm 4.2. The algorithm for flipping row or column combinations is the same. Even though the pseudo code is written for the rows, the same applies when flipping the columns.

Algorithm 4.2: Combination flipping of rows or columns

Input: current received vector v and current Candidate Error Matrix M

Initialization: $\mathbb{C} = 0$, initialize the complexity by setting it to zero. The complexity is used to keep track of the number of rows or columns being flipped in combination flipping.

Find $\mathbb{R}$, a vector that stores the indexes of the rows with errors.

While the new shadow area is identical to the current shadow area, ($M' == M$), and $\mathbb{C} \leq$ half the number of rows with errors

Increment the complexity by one, $\mathbb{C} = \mathbb{C} + 1$

Find the possible combinations $\mathbb{Q}$ of the rows with errors based on $\mathbb{C}$ and $\mathbb{R}$

$$\mathbb{Q} = \binom{\mathbb{R}}{\mathbb{C}}$$

Use the next combination and M to create a Flip Matrix, F, to be used for flipping the combination of rows or columns

Flip the bits using F,

$$r' = F \oplus r$$

Find new Candidate Error Matrix M' based on r'

End while

If $M' \neq M$ then

Save the new vector to be the received vector,

$$r = r'$$

Save the new Candidate Error Matrix to be

$$M = M'$$

Else the error pattern cannot be corrected

Set flag to show all possible combination have been tried but the received vector still has errors

End if

End complex flipping

4.3 Numerical stepwise example of bit flipping algorithm

Assume an all zero codeword with (15,7,2) BCH component codes is sent, and the error sequence e showing the grid of the errors is as follows:

$$e = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \end{bmatrix},$$

where the positions marked by 1s are the errors. For demonstration purposes we track changes in the positions of the errors in e. It must be noted that, in reality the actual positions of the errors in the received sequence are unknown. The first step of the algorithm is finding the current Candidate Error Matrix M of the current received vector. M is found using (4.4). Assuming the rest of the received vector has no errors besides the ones in e, from (4.5), the Row Check Vector will be

$$X = \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix}^T,$$

and the Column Check Vector will be

$$Y = \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix},$$

and the Candidate Error Matrix will be

$$\begin{aligned} M &= X \cdot Y \\ &= \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix}. \\ &= \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \end{aligned}$$

Flipping all the bits using M yields a new error sequence

$$e' = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix},$$

with a new Candidate Error Matrix

$$M' = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}.$$

Because M' is identical to M, the flip is discarded and the original error sequence is used for the next step. The algorithm proceeds by flipping a combination of rows or a combination of the columns. A value to keep track of the number of columns or rows being flipped known as the complexity is set to 1. We then create the Flip Matrix, F, based on the complexity, and the indices of the rows with errors. The Flip Matrix is still based on (4.4), but we change X or Y to reflect the rows or columns we are flipping.

Assuming the complexity is 2 and we are flipping the first 2 rows, then

$$X_i = \begin{bmatrix} 1 & 1 & 0 & 0 \end{bmatrix}^T.$$

Note that Y remains the same, since we are only flipping based on a combination of the rows, therefore

$$F = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

In our case, the complexity is 1, therefore,

$$X_i = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}^T, \text{ and}$$

$$F = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

The new error sequence is obtained by flipping using;

$$\begin{aligned} e' &= F \oplus e \\ &= \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \end{bmatrix}. \end{aligned}$$

Two new errors have been introduced by the flipping, but the last column no longer has errors.

$$M' = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \end{bmatrix}.$$

The new error sequence is saved for the next stage of flipping. The process restarts by setting

$$M = M'$$
$$e = e'$$

All the bits are flipped using M as follows

$$\begin{aligned} e' &= M \oplus e \\ &= \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}. \end{aligned}$$

The new error vector has no errors in the first column, and the shadow area is reduced to

$$M' = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \end{bmatrix},$$

therefore, $M = M'$ and $e = e'$. Again all the bits are flipped using M. After flipping, $M' \equiv M$, so the flip is discarded and flipping using combinations of the rows or columns is used. Since the number of columns with errors is less than the number of rows with errors, combination flipping is done using a combination of the columns with errors. The complexity is set to 1 and a single column is used, resulting in the following Flip Matrix,

$$F = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}.$$

The flipping is done as follows

$$
\begin{aligned}
e' &= F \oplus e \\
&= \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.
\end{aligned}
$$

The corresponding Candidate Error Matrix for the new error sequence is,

$$
M' = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.
$$

Since $M \neq M'$, so $M = M'$ and $e = e'$. Due to the fact that the shadow area has become smaller, the algorithm flips all the bits in the shadow area using M resulting in

$$
\begin{aligned}
e' &= M \oplus e \\
&= \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.
\end{aligned}
$$

The decoding is complete when there is no shadow area, M is all-zero, or combination flipping does not yield a new error vector with a different shadow area. While running testing the algorithm in simulations, it was discovered that it was possible to have no shadow area, but still have errors in the corrected vector. This was defined as the Hidden error pattern, and is discussed in Section 5.2.3. In such a case, the conventional algorithm is used to correct the errors in the rows with errors until a shadow area is obtained, then the bit flipping algorithm is used to correct the rest of the errors.

4.4 Analysis of bit flipping decoding algorithm

The major reason for proposing the development of the bit flipping algorithm is the simplicity of the bit flipping algorithm presented by Gallager for LDPC codes. Hard decision decoding

algorithms are simple, fast and easy to implement in hardware. Of the hard decision decoding algorithms, bit flipping has been touted as one of the simplest to implement on hardware. The turbo decoding algorithms for product codes, though they approach the Shannon limit, are complex and computationally intensive to implement. The development of our decoding algorithm means we use the advantages of bit flipping to decode product codes by using a low complexity decoding algorithm.

For a two dimensional product code, the bit flipping algorithm makes use of the row and column parities to correct errors. The conventional decoding algorithm makes use of one parity at a time, either the row parity when decoding the rows or the column parity when decoding the columns. Sometimes the conventional algorithm introduces new errors in the columns when decoding the rows. The bit flipping algorithm on the other hand ensures that if new errors are introduced, they are only introduced in columns or rows that already have errors. This is done by ensuring the shadow area never becomes bigger when flipping. Only bits in columns and rows with errors are changed.

The bit flipping algorithm and the conventional algorithm both use syndrome checking. They have to first determine whether a row or column has errors. The difference in complexity between the two algorithms is down to their differences in the methods they use for error correction. The conventional algorithm uses the respective base codes to correct the rows then the columns. For a BCH code, this is the combined complexity of the Berlekamp-Massey algorithm and the Chien search. The bit flipping algorithm uses the less complex method of bitwise XOR. The simplicity of bit flipping is well documented. The algorithm has been used for decoding LDPC codes, and is attractive as it offers an effective compromise between error correcting capability and decoding complexity. We show in Chapter 6 that the complexity of the bit flipping decoding algorithm is comparable to the conventional decoding algorithm at the higher SNRs.

Turbo decoding also uses the multi-dimensional parities of product codes to achieve performance close to the Shannon limit. The method uses a soft-in soft-out approach. The decoding of the rows results in reliability values of the row information symbols, which are taken into consideration when decoding the columns. The process is iterated with the values from decoding the columns taken into consideration when decoding the rows, until the values of the

rows and columns concur [119]. Soft decision decoding generally outperforms hard decision decoding, in some cases having a gain of up to 3dB over hard decision decoding. The major disadvantage of soft decision decoding is the complexity as soft decision decoding algorithms are generally more complex than hard decision decoding algorithms. Likewise turbo decoding may offer better performance over our bit flipping decoding algorithm, but it is more complex than the conventional decoding algorithm, and therefore expected to be more complex than our algorithm.

4.5 Conclusion

Section 4.2 presents a new bit flipping decoding algorithm. The algorithm can be used for error correction of binary systematic product codes. It works with the premise of finding a shadow area. This is done by using syndrome checking of the rows and columns in the received vector. In most decoding algorithms the process of detecting errors presents less of a challenge, when compared to the process of finding the positions of the errors, and correcting them. The decoding algorithm uses the simplicity of syndrome detection to find the likely positions of the errors by mapping out a shadow area. Using iterative bit flipping, for different types of error patterns, the errors in the received vector are corrected.

Section 4.3 gives a numerical stepwise example of the bit flipping algorithm. The example clarifies the workings of the algorithm by showing how the bits are flipped in a vector received with errors. It shows that in some instances the algorithm introduces new errors, but the shadow area never increases in size. This is an advantage of this algorithm because it leaves room for the development of an ARQ technique that could have only the rows or columns that have errors resent instead of the entire codeword, thereby saving on bandwidth and power. The conventional decoding algorithm at times increases the number of rows and columns with errors.

Chapter 5 presents the results from simulations run to compare the developed bit flipping algorithm, and the conventional decoding algorithm that was presented by Elias for product codes. Coding systems can be compared using the block error rate (BLER) or word error rate (WER), symbol error rate (SER) and the bit error rate (BER). In Chapter 5 the decoding

algorithms are compared in terms of the WER and BER performance. A comparison of the decoding complexities of the algorithms is done.

Chapter 5 - Results and Analysis

5.1 Introduction

This chapter presents the results that were obtained during the study. The results were obtained from simulations carried out on Matlab[1].

5.2 Error Patterns

The error pattern of the received sequence greatly influences the decoding complexity. The error patterns were classified into four groups according to the distribution of errors in the Error Matrix, namely

- Simple error pattern
- Complex error pattern
- Hidden error pattern
- Un-decodable error pattern

[1] All results in Chapter 0 are based on the (15,7) BCH code, unless otherwise stated. The Berlekamp-Massey decoding algorithm was used as the conventional decoding algorithm.

5.2.1 Simple error pattern

In this error pattern, the number of errors in one or more rows and or columns is equal to the width or the height of the Candidate Error Matrix, the shadow area. It should be noted that the width of the Candidate Error Matrix is the number of columns denoted as having errors by the Column Parity Check Vector, and the height is the number of rows denoted as having errors by the Row Parity Check Vector.

In a simple error pattern, if all the bits in the Candidate Error Matrix are flipped using bitwise XOR, the size of the shadow area will decrease.

After sending an all zero codeword, if the received sequence r has errors (denoted by 1s) distributed as follows

$$r = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

this is a simple error pattern. The width of the shadow area is 2, and the height of the shadow area is 2. Row has 2 errors and column 2 also has two errors.

After flipping a simple error pattern, the resultant error pattern could be either a complex error pattern or another simple error pattern. The simple error patterns and complex error patterns occur anytime during the course of decoding using the bit flipping algorithm irrespective of the SNR. They are dependent on the distribution of the errors.

5.2.2 Complex error pattern

In a complex error pattern, the size of the shadow area will remain the same after a bitwise XOR of all the bits in the Candidate Error Matrix is done. The number of errors in all rows and columns is less than the height or the width of the area under shadow.

To decrease the size of the shadow area, a bitwise XOR of a combination of one or more rows or columns has to be done. If it is a combination of rows, it has to be such that there exist one or more columns with errors only in the selected row or rows. If it is a combination of columns, there has to be one or more rows where there are errors only in the selected column or columns. It should be noted that as bit flipping is done, the successive error pattern could be either be complex or simple. Assuming an all zero codeword was sent, and the 1s depict errors, the following sequence depicts a complex error pattern;

$$v = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Flipping all the bits in the shadow area will result in a sequence with the same size shadow area. Flipping a combination of rows or columns using the stated criteria has to be used to decrease the shadow area. For example combinations of rows 3, 4, and 6 or of 3 and 5 or of 4 and 6 could be used to decrease the shadow area. If the combination of rows 3 and 5 is taken into account, the combination works because they are the only rows with errors in columns 2 and 4. The combination of rows 3 and 6 does not work because, the only column with errors they have in common has an error in row 4. The rest of their columns do not have errors in common. Flipping a combination of rows decreases the number of columns with errors and vice versa.

5.2.3 Hidden error pattern

The hidden error pattern is a result of the short comings of syndrome checking. If the number of errors in a row or column is equal to or more than the minimum distance of the component code, this can result in the received row or column vector becoming a different but valid codeword. The Candidate Error Matrix is a result of the multiplication

$$M = X \times Y$$

where M is the Candidate Error Matrix, X is a transpose of the Row Parity Check Matrix and Y is the Column Parity Check Matrix. Assuming the hidden error pattern is a row, the Column Parity Check Matrix will have ones for the columns with errors, but the Row Parity Check Matrix will be all zero. The resultant Candidate Error Matrix will be all zero, giving the impression there are no errors in the received vector.

Hidden errors are prevalent at low SNRs. A product code with $(15,7,2)$ BCH codes as component codes was used to illustrate the Hidden error pattern. The minimum distance of each of the component codes was 5. After sending an all zero codeword, the first six rows of the received sequence were:

$$r = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix},$$

where the 1s depict errors. The corresponding rows of the Candidate Error Matrix illustrate the Hidden error pattern,

$$M = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

Despite the fact that there are errors in row two of the received vector, the Candidate Error Matrix does not have errors in row two. This is because row two of the received vector has errors such that the row becomes another valid codeword. Hidden errors can be detected by the presence of errors in rows but none in the columns or errors in the columns but none in the rows. If the errors are in the rows, the conventional decoding algorithm is used to decode row-wise in

the rows with errors until a non-zero Candidate Error Matrix is obtained. If the errors are visible column-wise then the conventional decoding is done column-wise.

Hidden errors also occur after decoding using the bit flipping algorithm. After sending an all zero codeword for a product code with $(7,4,3)$ BCH constituent codes, the received sequence with errors r was:

$$r = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

and the corrected vector r' was

$$r' = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

X' and Y', the Row Parity Check and Column Parity Check vectors for the corrected vector were:

$$\begin{aligned} X' &= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T \\ Y' &= \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 & 0 \end{bmatrix} \end{aligned}.$$

X' does not show any errors in the corrected sequnce, whereas Y' shows that three columns have errors. The Candidate Error Matrix for the corrected sequence was:

$$M' = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

5.2.4 Un-decodable error pattern

The un-decodable error pattern is prevalent at low SNRs. This error pattern occurs when the height and or the width of the shadow area is larger than the minimum distance of the component codes. Flipping all the bits covered by the Candidate Error Matrix may result in hidden errors.

To avoid hidden errors, we decrease the size of the shadow area to an area such that the number of the rows and the columns with errors is less than the minimum distances of their respective constituent codes before using the bit flipping algorithm. Two methods can be used to decrease the size of the shadow area. The first method uses the conventional decoding algorithm to decrease the number of rows and columns with errors. If the number of rows with errors is greater than the minimum distance of the component code, but the number of columns is less, then the conventional decoding algorithm is used to decode the rows otherwise it is used decode the columns. If both the number of rows and columns with errors are greater than the minimum distances of the respective component codes, conventional decoding is done row-wise then column-wise. This is done iteratively until the numbers of both are less than the minimum distance or one of them is. If only one of them is less than the minimum distance, then only row-wise or column-wise decoding is done depending on the one that is less, as stated.

The second method flips a maximum of t bits in the rows or columns with errors to decrease the size of the shadow area. This method is less likely to decrease the number of rows with or columns with errors more-so at low SNRs where the number of bits with errors may be more than the error correcting capability of the component codes. The method may also require a large number of flips to correct the errors in a row or column. It has the advantage of being less likely to increase the number of errors in the received vector.

To decrease the number of rows or columns with errors we used the first method of using the conventional decoding algorithm to decrease the number or rows or columns with errors. We noted that at very low SNRs, while using the conventional decoding algorithm to decrease the number of rows with errors, errors could be introduced in the columns. For example, an all zero codeword was sent through the channel at SNR of 1 dB. The first six rows of the received sequence were as follows:

$$r = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}.$$

After using the conventional decoding algorithm to decrease the number of rows with errors, new errors were introduced in the columns. The new errors are:

$$e' = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 1 & \mathbf{1} & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & \mathbf{1} & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & \mathbf{1} & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \mathbf{1} & 1 & 0 & 1 & 1 & \mathbf{1} & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 1 & 0 \end{bmatrix}.$$

The conventional decoding algorithm uses the Berlekamp-Massey algorithm. If the number of errors in the row or column is beyond the error correcting capability of the code, it is likely to introduce more errors. This is prevalent and expected at low SNR. The method is further complicated when we decrease the number of columns with errors after decreasing the rows. The columns introduce errors in rows. Though the method fails at very low SNR, it works when the total number of errors in the received sequence is within the error correcting capability of the product code.

5.3 Performance of coded communication systems

The performance of a coded communication system can be measured by its probability of decoding error, called the error probability, and its coding gain over an un-coded system transmitting information at the same rate [14]. There are three types of error probability, probability of word (or block) error, probability of symbol error and probability of bit error. The probability of word error is the probability that a decoded word or block at the decoder is in error. This is known as the block error rate (BLER) or word error rate (WER). The BLER is not dependent on the number of symbols or bits in error. The symbol error rate (SER) is defined as the ratio of erroneously decoded information symbols (constellation points) to the total number of transmitted symbols. The bit error rate (BER) is the probability that a decoded information bit at the output of the decoder is in error. BER is the ratio of error-bits received to the total bits sent.

The SER depends on the modulation scheme being used for transmission. For a BPSK modulated signal, the BER and the SER are equal because there is a single bit per symbol. In QPSK, each symbol has two bits. The performance of coded systems can be studied through the analysis of their BLER, SER and BER.

5.3.1 Performance comparison of Hamming product codes

The performance of a one coded communication system can be measured by its probability of decoding error, called the error probability, and its coding gain over another coded system transmitting information at the same rate. In our case, since the same encoding method is used, it means the difference in the two systems is down to the differences in their decoding algorithms.

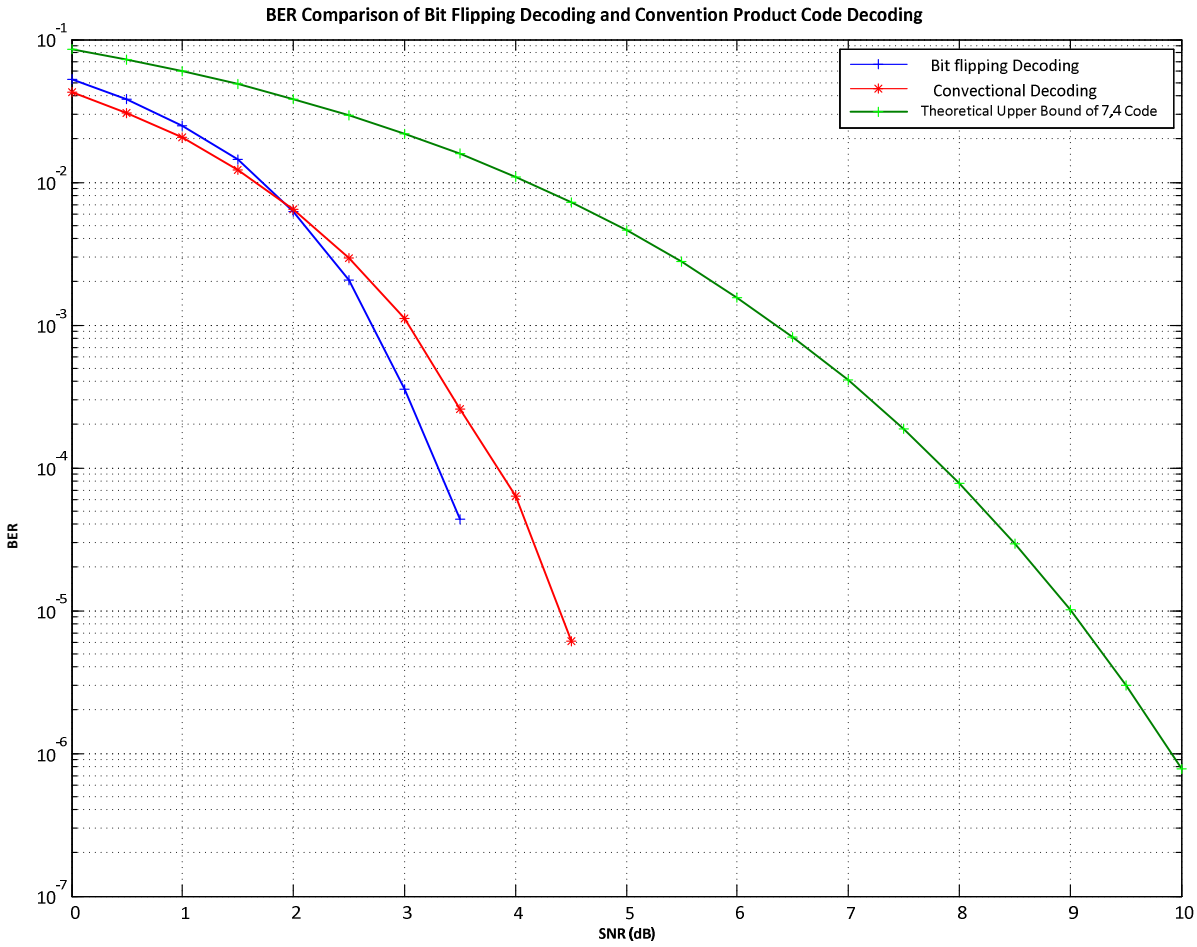

Figure **5.1** shows a comparison of the bit flipping decoding algorithm and the conventional decoding algorithm. The component code used was a (7, 4) Hamming code. At 1.9 dB, the bit flip decoding algorithm has the same BER as the conventional decoding algorithm. After 1.9 dB, the bit flipping decoding algorithm outperform the conventional algorithm at same SNRs. At a BER of 10^{-4}, the bit flipping decoding algorithm has a 0.4 dB gain over the conventional decoding algorithm. The bit flipping decoding algorithm exhibits a steeper gradient curve showing a better BER for a smaller increase in signal strength.

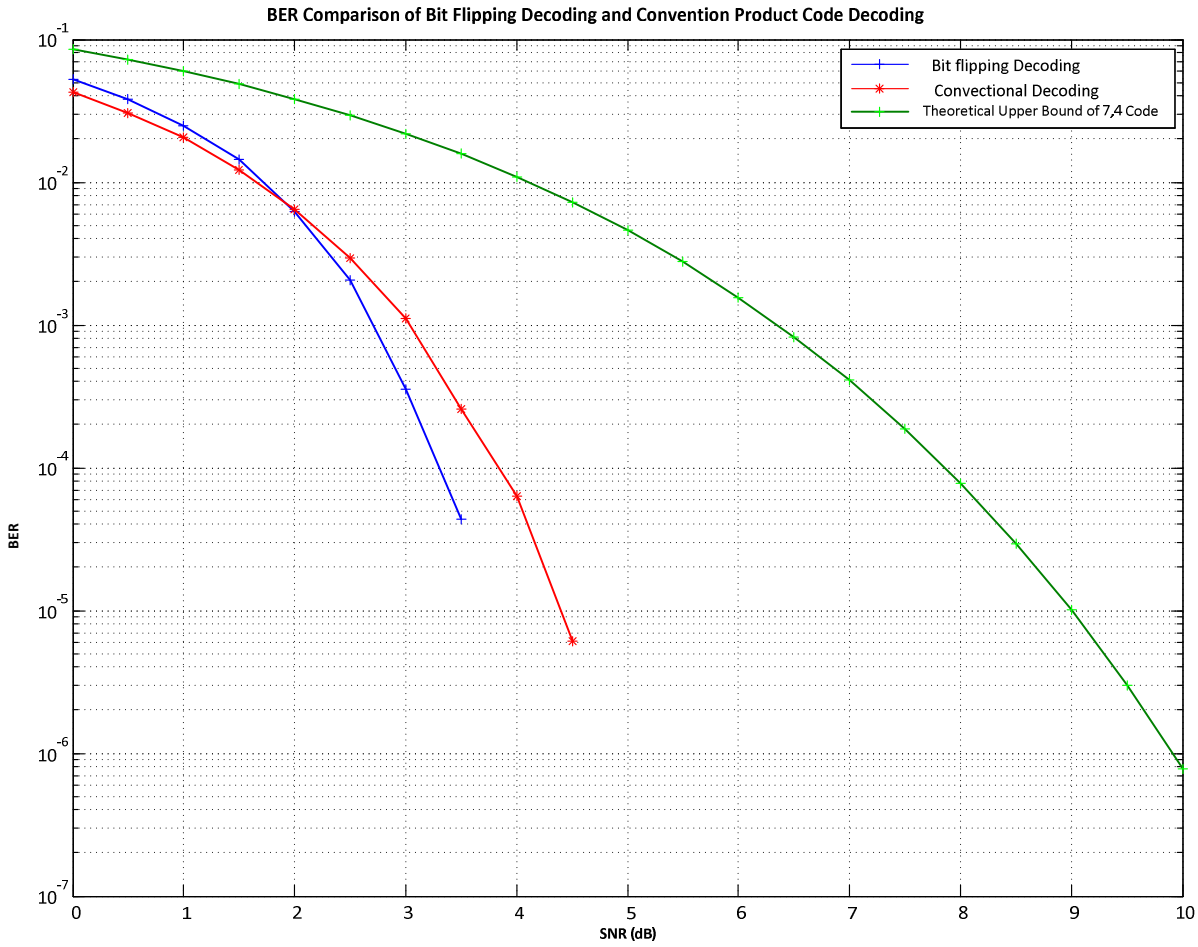

Figure 5.1: BER comparison of bit flipping and conventional decoding algorithms - (7, 4) Hamming constituent code

Previous studies have shown that the conventional decoding algorithm fails to decode some error patterns within the error correcting capability of the code [37]. The results from the performance comparison of the conventional decoding algorithm and bit flipping decoding algorithm were not as expected. The expected results were that the bit flipping decoding algorithm would always outperform the conventional decoding algorithm. The BER curves give the impression that in low SNR channel conditions, the conventional decoding algorithm outperforms the bit flipping decoding algorithm. Both algorithms perform better than the upper bound of the constituent code.

Simulations were run to compare the WER of the bit flipping and conventional decoding algorithm. From Figure 5.2, it can be noted that the bit flipping decoding algorithm corrects more codewords than the conventional decoding as expected.

The bit flipping decoding algorithm being an iterative decoding algorithm, as expected out performs the conventional decoding algorithm, which only decodes using a single cycle of row decoding followed by column decoding.

The reason the conventional decoding algorithm outperforms the bit flipping decoding algorithm at lower SNRs when considering the BER, is because of the very nature of the algorithm. The algorithm relies on syndrome detection. If the bit flipping introduces a number of errors greater than the minimum distance, this can result in a decoding failure for the row or column. At low SNR, this occurrence is prevalent as the bit flipping algorithm introduces more errors by flipping entire rows or columns. The decoded data tends to have more errors in the bit flipping decoding algorithm than in the conventional decoding algorithm.

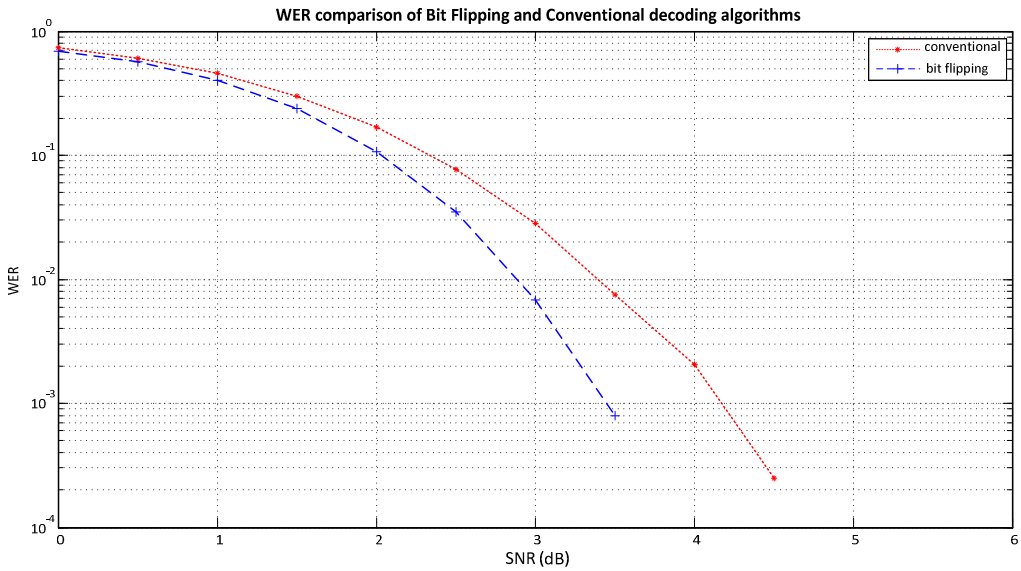

Figure 5.2: WER comparison of bit flipping and conventional decoding algorithms – (7, 4) Hamming constituent code (AWGN channel)

This is illustrated by Figure 5.3. To plot the curves in Figure 5.3, the total number of errors in each of the decoded blocks was counted. A comparison of the block with the greater number of errors was done. Figure 5.3 shows the number of times each of the decoding algorithms has a larger number of errors after decoding per block for each SNR. At low SNRs, the bit flipping decoding algorithm has a greater total number of blocks that have more errors than the conventional decoding algorithm. This number accounts for the poor performance in terms of the BER of the bit flipping decoding algorithm when compared to the conventional decoding algorithm at low SNR.

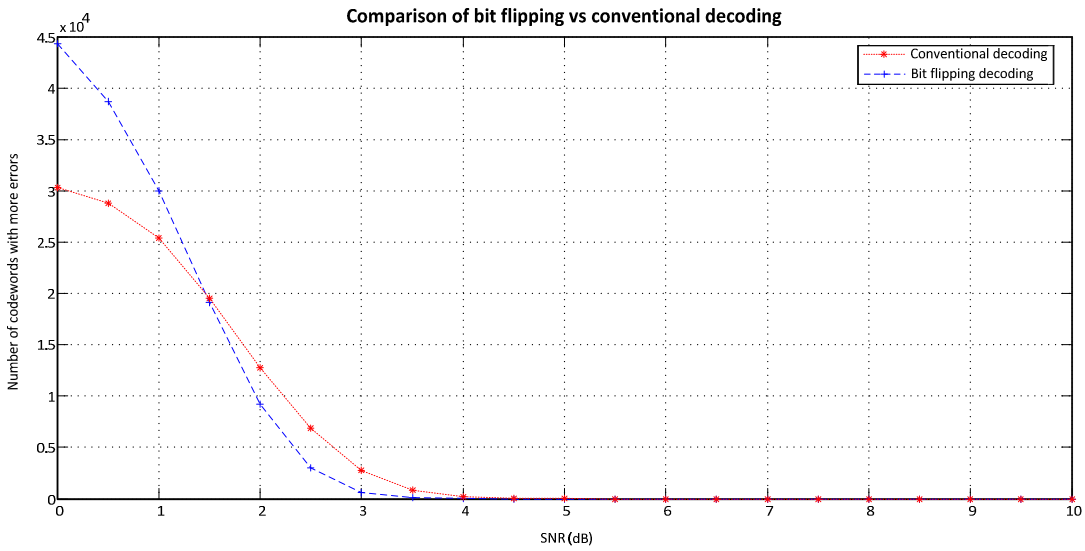

Figure 5.3: Comparison of total number of errors per codeword – (7, 4) Hamming constituent code (AWGN channel)

5.3.2 Performance comparison of BCH product codes

Hamming codes are perfect codes. They all have a minimum distance of three. For any two dimensional product code with Hamming codes as the constituent codes, the minimum distance will be nine. This is in spite of the block length. BCH codes offer the flexibility of being able to change the error correcting capability of the product code. The component codes can be chosen with their error correcting capability in mind. A comparison of the performance of the conventional decoding algorithm and the bit flipping decoding algorithm was done.

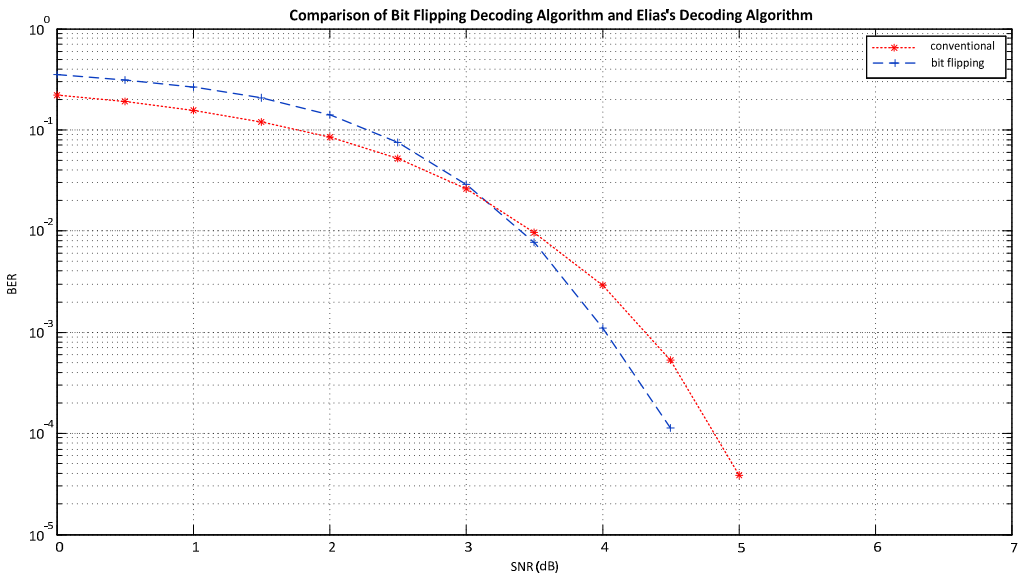

Figure 5.4 : BER comparison of bit flipping and conventional decoding algorithm - (15, 7) BCH constituent code (AWGN channel)

Figure 5.4 shows the BER curves for the conventional decoding algorithm and the bit flipping decoding algorithm for a two dimensional product code with a (15, 7, 2) BCH code as the component code.

The results of the BER comparison of the two codes are consistent with those of the Hamming product code. The conventional decoding algorithm outperforms the bit flipping algorithm at low SNRs. At about 3.2 dB, the bit flipping algorithm starts to outperform the conventional algorithm. This is a little later than in the Hamming product code.

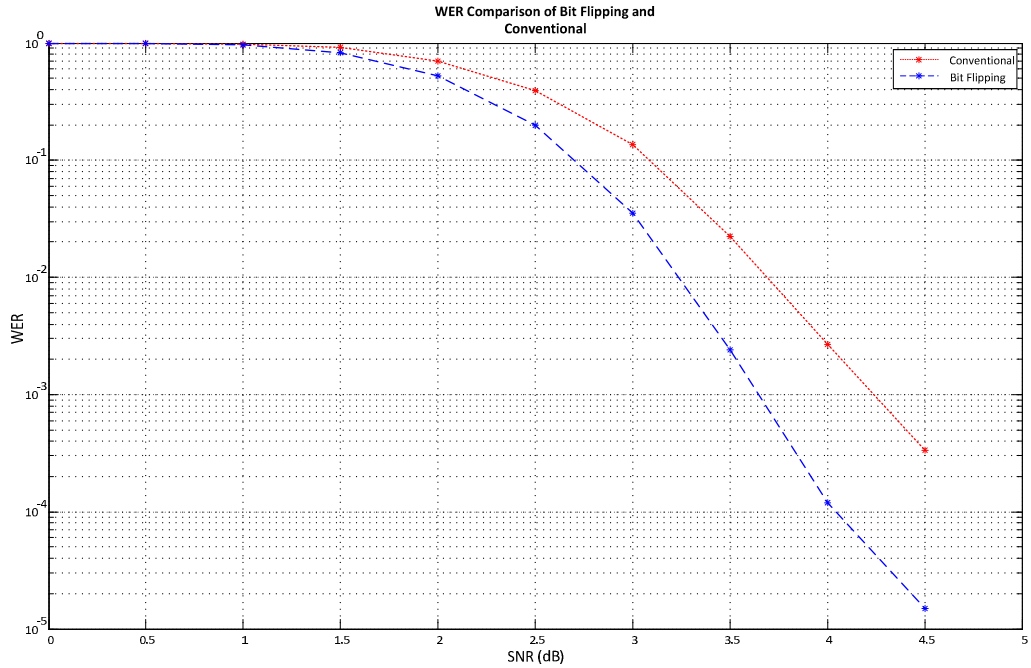

Figure 5.5: WER comparison of bit flipping and conventional decoding algorithm - (15, 7) BCH constituent code

The results of the WER comparison are also consistent with those of the Hamming product code. The small difference in WER between the two algorithms at low SNR accounts for the later change in performance in the BER curves. At 10^{-3} WER, the bit flipping decoding algorithm has a gain of 0.75dB.

5.4 Impact of SNR on shadow area

The size of the shadow area has an impact on the BER curve. At low SNRs, the shadow area covers more than the error correcting capability of the code.

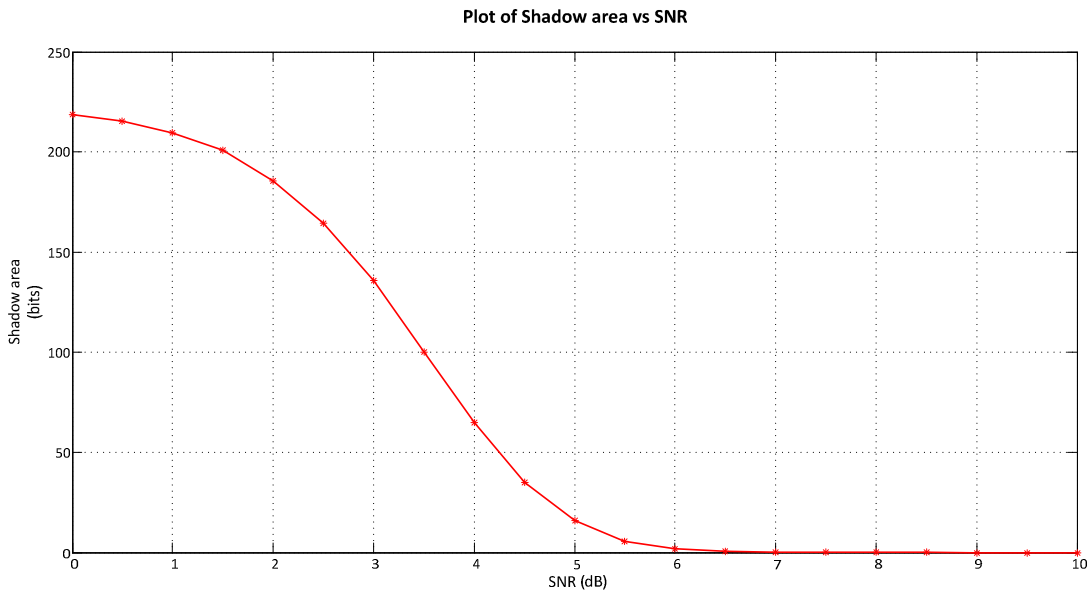

Figure 5.6: Plot of shadow area vs SNR - (15,7) BCH constituent code (AWGN Channel)

In Figure 5.6, at 0.5 dB, the shadow area is 210 bits. The error correcting capability of the code is 12. This impacts on the BER curve of the code, as shown in Figure 5.7. At low SNR, the BER is at its highest.

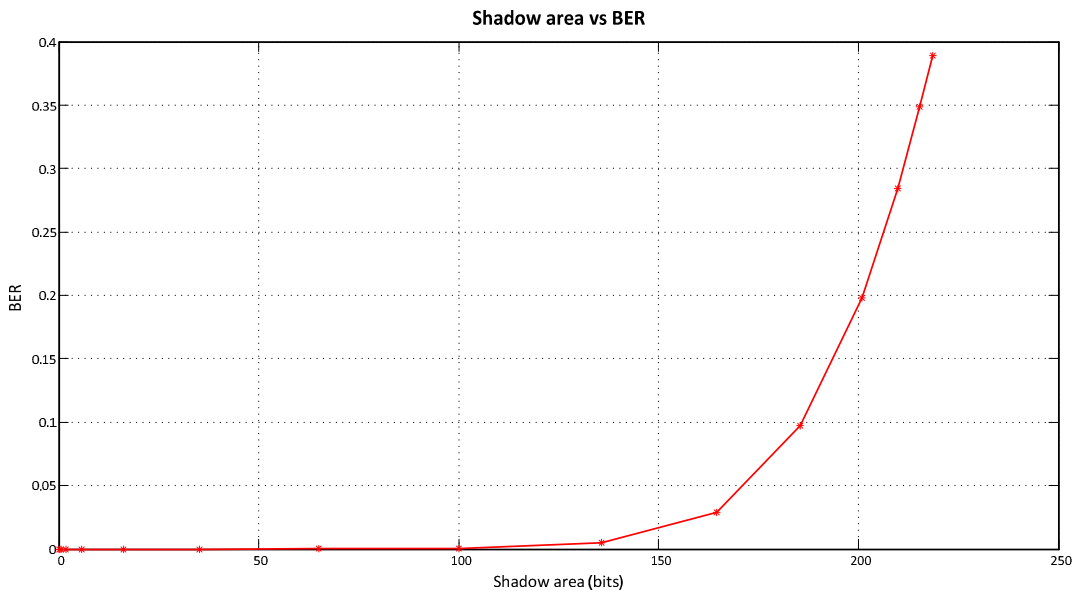

Figure 5.7: Plot of BER vs shadow area – (15, 7) BCH constituent code (AWGN channel)

As the shadow area decreases, the BER decreases exponentially. When the shadow area is less than 50, there are no errors in the decoded sequence. This is consistent with the results that can be inferred from Figure 5.4 and Figure 5.6.

5.5 Rayleigh fading channel results

Fading is a multiplicative change in the amplitude of the signal. In wireless communication, unlike wired communication, the signal is subject to multiple reflections. The reflectors and the transmitter may also be moving with respect to each other. In wireless communication, in the urban environment, a signal may reflect off of buildings and cars before reaching the intended recipient. The received signal becomes a superposition of the reflected signals. The signals travel different distances to get to the receiver, increasing the likelihood of them arriving at the receiver out of phase with each other. If the signals are out of phase, they will add destructively, and thus fading occurs [30].

Different fading channel models have been proposed for studying wireless communication. The Rayleigh fading channel model is used for simulating multipath propagation of the signal. This makes it suitable for studying the behaviour of coding systems in mobile environments where there are many obstacles that reflect the signal sent from the base station to the cell phone. The channel model can also be used when studying high frequency ionospheric radio wave propagation where refractions occur at many points within the ionosphere. It can also be used for studying tropospheric radio wave propagation. To study the behaviour of the bit flipping decoding algorithm in wireless communication, we chose the Rayleigh fading channel as it is used to model a wide range of wireless telecommunications systems. For this study we chose a flat fading Rayleigh channel, modelled as a single tap filter with a Rayleigh fading response with zero mean and unit variance, and where the noise contributed by AWGN is Gaussian distributed with zero mean and unit variance.

Figure 5.8 shows a performance comparison between the bit flipping decoding algorithm and the conventional decoding algorithm in a Rayleigh fading channel.

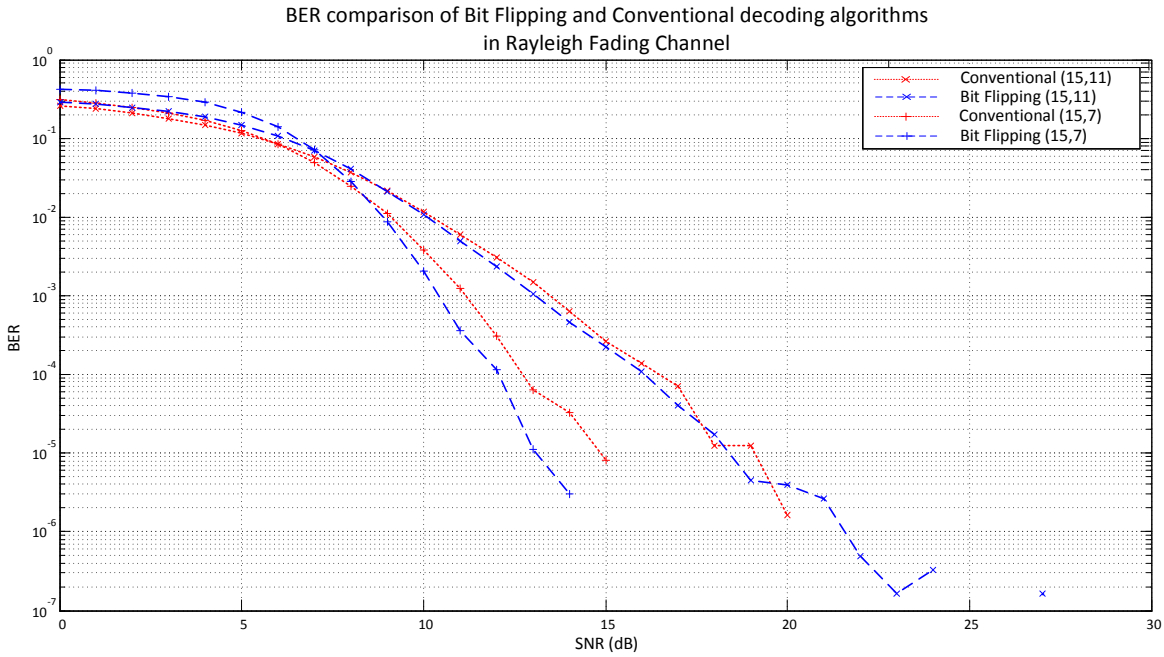

Figure 5.8: BER comparison of bit flipping decoding algorithm and conventional decoding algorithm in Rayleigh fading channel –Method Two

The BER curve in Figure 5.8 was obtained using the single bit flipping method to cater for the Un-decodable error pattern and the Hidden error pattern. To compare the method of using the conventional decoding algorithm to decrease the size of the shadow area and the second method of flipping bits, simulations using the method of using the conventional algorithm were run. The results of the simulations are portrayed in

Figure **5.9**. Using the conventional decoding algorithm to reduce the size of the shadow area improves the BER of the bit flipping decoding algorithm. This is evident from comparing Figure 5.8 and

Figure **5.9**. The (15,7) BCH code for the code using the conventional decoding has a gain of over 4 dB at the BER of 10^{-4} over the single bit flipping method.

The results from

Figure **5.9**, differ from the results of the other simulations. The bit flipping decoding algorithm outperforms the conventional decoding algorithm at the low SNRs. This is because the method incorporates the conventional decoding algorithm, which improves the performance of the

algorithm at the lower SNRs. To reduce the size of the shadow area, the conventional decoding was used iteratively. Iterative decoding of product codes has a better decoding gain than the conventional method used by Elias [76]. The resultant decoding algorithm has better performance than using single bit flipping to reduce the shadow area. The conventional decoding has been proven to be a sub-optimal decoding algorithm. A simulation was run to compare the WER of the two algorithms. The WER simulation shows that bit flipping decoding algorithm outperforms the conventional decoding algorithm from 1dB. Between -1 and 1dB, both decoding algorithms fail to correct all the errors in all the received vectors.

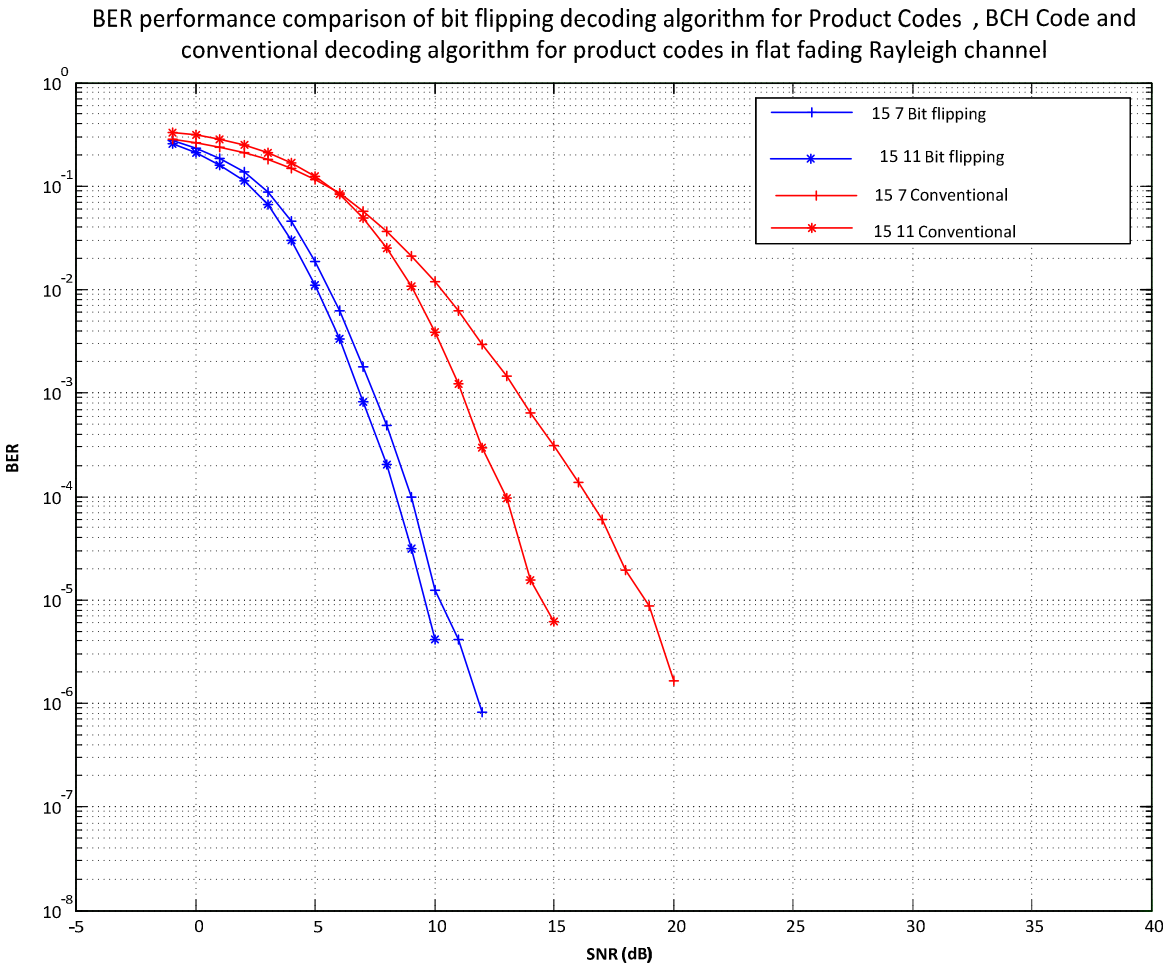

Figure 5.9: BER comparison of bit flipping decoding algorithm and conventional decoding algorithm in Rayleigh fading channel – Method One

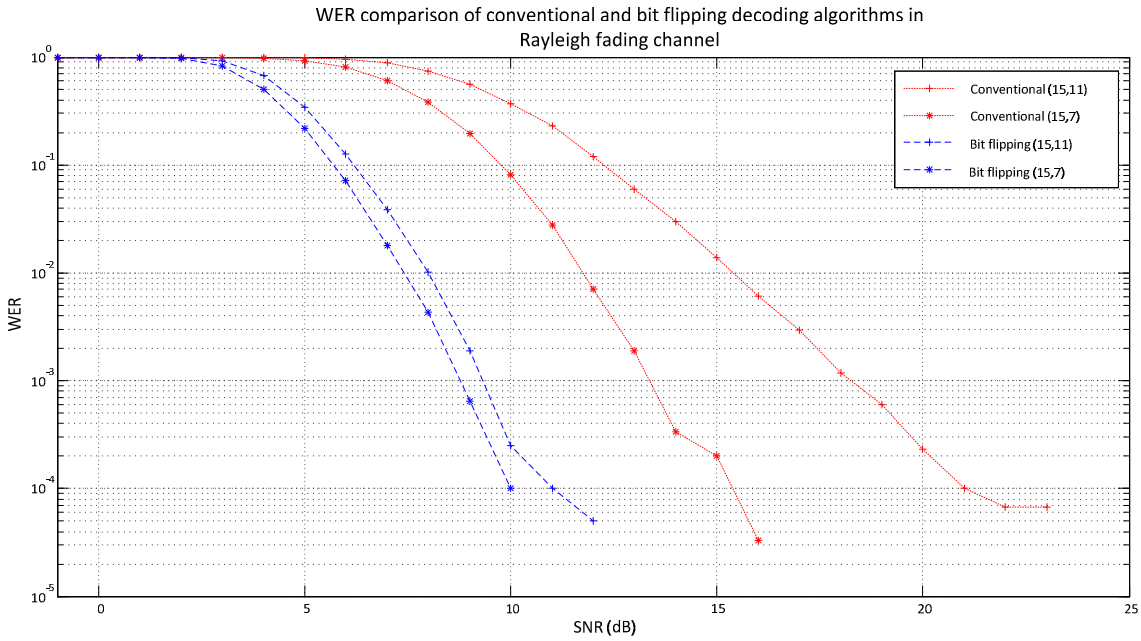

Figure 5.10: WER comparison of bit flipping and conventional decoding algorithm - (15, 7) BCH constituent code in Rayleigh Fading channel – Method One

5.6 Varying the code rate

Using a single code rate for a communication channel with varying channel conditions, compromises on the payload of the communication system. To ensure a high throughput, digital communication systems use varying code rates. The results in Figure 5.11 show the BER curves of different code rate two dimensional product codes using the bit flipping decoding algorithm in an AWGN channel. The longer codes tend to outperform the shorter codes at higher SNRs as expected.

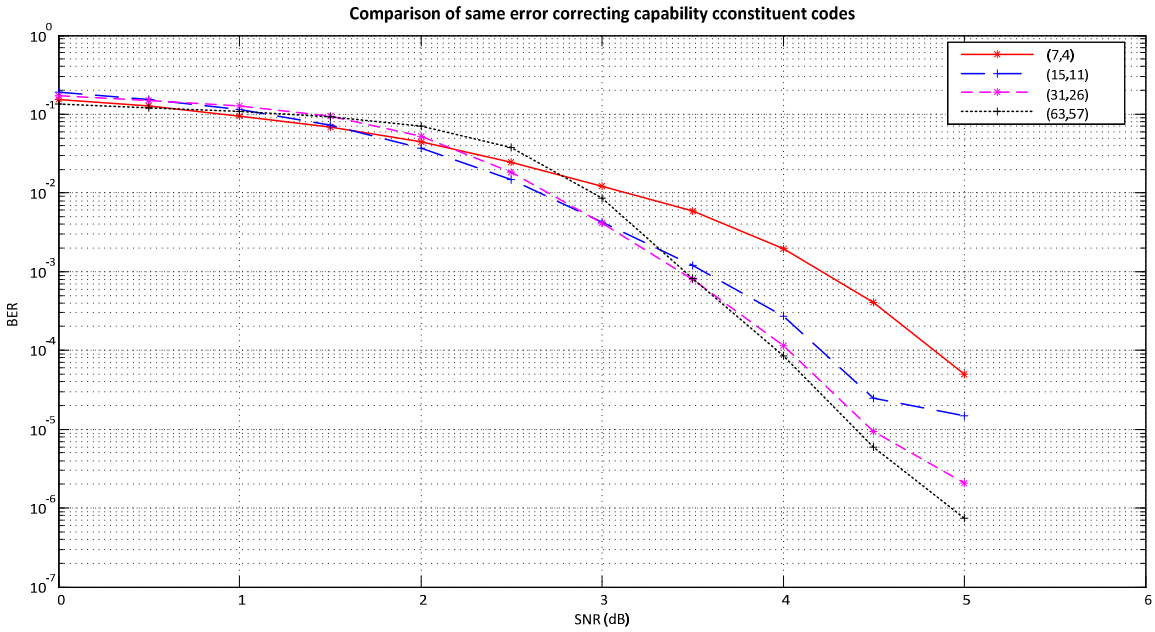

Figure 5.11: BER comparison of different code rate codes ($d_{\min} = 9$) – BCH constituent codes (AWGN channel)

5.7 Decoding complexity

5.7.1 Complexity of conventional algorithm

The generator polynomials of BCH codes have as roots $2t$ consecutive powers of an element of order n. They are attractive codes because they can be designed to a desired error correcting capability, and efficient decoding algorithms exist for their decoding. The decoding procedure for BCH codes consist of four steps.

The first step is the evaluation of the received vector's syndrome. If $r(x)$ is the received vector, and $\alpha^i, i = 1, 2, \ldots, 2t$, are the roots of the generator polynomial, then the syndromes are given by (2.46). The syndromes can be calculated by diving $r(x)$ by the minimal polynomials of the roots and then evaluating the remainder polynomials at the roots. The complexity of the step is

$$\sim 2t(n-1).$$

(5.1)

The second step of decoding a BCH code is determining the Error-locator polynomial. This step is the most complex step in the decoding of BCH codes. Two popular methods for computing the error-locator polynomial are the Berlekamp-Massey algorithm and the Euclidean algorithm [120]. The Berlekamp-Massey algorithm has complexity

$$C_{\mu} = 4t^2 + 2te - e^2 + 10t + e,$$

(5.2)

where the number of errors $e \le t$, the error correcting capability of the code [120]. The Euclidean algorithm is less efficient than the Berlekamp-Massey algorithm. It has a complexity given by the following equation

$$C_{\mu} = 8te - \frac{1}{2}e^2 + \frac{13}{2}e.$$

(5.3)

Once the error-locator polynomial is known, the third step in the algorithm is determining its roots. This is done using the Chien search. The complexity of the Chien search is

$$C_{\mu} = e(n-i),$$

(5.4)

where $1 \le i \le n$ and α^i is root, and i is the index of the least root [120]. The Chien procedure searches, in the order $\alpha^n, \alpha^{n-1}, \ldots$, hence its complexity is determined by the root with the smallest index [120]. The error positions are the inverses of the roots of the error-locator polynomial.

The fourth step is calculating the magnitude of the error. For Binary BCH codes, the error magnitude is always one. Flipping the bits in error corrects the errors. For non-binary BCH codes, the error magnitudes have to be determined. Forney's algorithm can be used to determine the error values [121]. In this study we do not use non-binary BCH codes, hence we will not look into the complexity of Forney's algorithm.

The complexity of decoding the product code using the conventional algorithm is:

$$T(n) = O(n^4)$$

5.7.2 Complexity of bit flipping algorithm

The decoding complexity of the bit flipping algorithm is dependent on two operations. The first is the complexity of computing the syndromes of all the rows and columns. The second is the complexity of the actual bit flipping.

On receiving a codeword sequence, the syndromes of the rows and columns are computed. To ensure a lower computational complexity, the syndromes of the rows and columns are stored in two lookup tables, one for the rows and the other for the columns. Each lookup table is structured as follows

$$L = [S_1 \quad S_2 \quad S_3 \quad \dots \quad S_n],$$

where S_i is the syndrome of row or column i. For example, if an all zero codeword was sent, and the received sequence was:

$$r = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix},$$

with the parity check matrix of the constituent codes as

$$H = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix},$$

then the row lookup table L_r would be

$$L_r = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

and that of the column lookup table L_c would be

$$L_c = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 \end{bmatrix},$$

where each column in the lookup table is the syndrome of the corresponding row or column of the received sequence. From syndrome decoding, changing the ith bit in a row changes the syndrome of the row in the manner

$$S_{new} = (h_i + S_r)_2 \ , \tag{5.5}$$

where S_{new} is the new syndrome of the row, h_i is the ith column of H and S_r is the current syndrome of the row. Instead of recalculating the syndromes of the rows and columns using (2.23), the syndromes of the rows and columns in which the flipped bits lie are updated in the respective lookup tables using (5.5). For example the syndrome of the 1st row is

$$S_{r_1} = \begin{bmatrix} 1 & 1 & 0 \end{bmatrix}.$$

If the 2nd bit in the row is flipped, the new syndrome of the row would be

$$S = \left(\begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \right)_2 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}^T .$$

Addition is computationally less complex than recalculating the syndrome using multiplication. This lowers the complexity of the bit flipping decoding algorithm, as the syndrome has to be calculated once for each row and column.

The complexity of the bit flipping decoding algorithm c_β can be defined as the complexity of finding the initial syndromes of the rows and columns plus the complexity of correcting errors using the bit flipping. This can be summarized by the following equation

$$c_\beta = c_s + 2c_e$$

(5.6)

where c_s is the complexity of finding the syndrome and c_e is the complexity of error correction. For a product code c_s has a complexity

$$T(n) = O(n^3).$$

The complexity of error correction, c_e, is the total number of bits that have to be flipped to correct the received sequence. Every bit flipped is a modulo 2 addition, and for every bit flipped, the lookup table has to be updated using a modulo 2 addition. Not that c_e is dependent on the total number of errors, and also the distribution of the errors. The distribution of the errors plays a vital role in the number of flipping operations that will be carried out. For example, if we send an all zero codeword, and the received sequence has four errors distributed in the manner,

$$e = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix},$$

then in this case flipping all the bits would require one operation, and only four additions are required. If the errors are distributed in the manner,

$$e = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}.$$

the number of bits that need to be flipped in order to correct the errors are sixteen. The complexity of correcting errors is also dependent on the SNR of the channel. At low SNRs, more errors are introduced in the received sequence, therefore, the bit flipping algorithm may prove to be more complex than the conventional decoding algorithm as more bits need to be flipped.

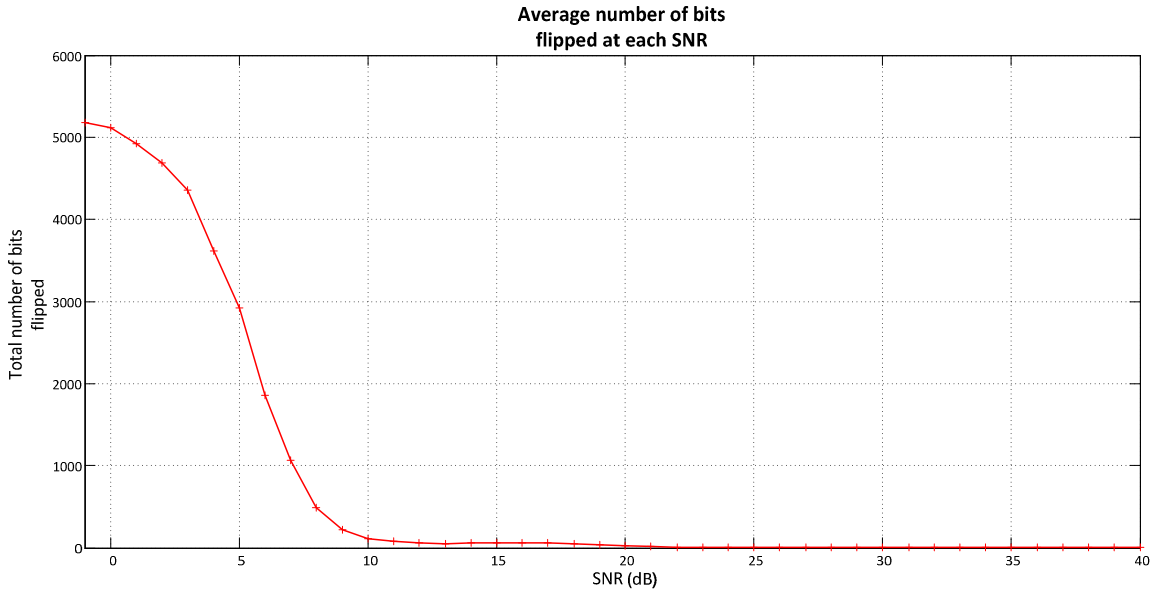

Figure 5.12: Complexity of bit flipping at each SNR (Rayleigh fading channel)

In Figure 5.12, the average number of bits flipped was computed for each SNR. At lower SNRs, a larger number of bits were flipped. Though the complexity of decoding using the bit flipping algorithm is lower than that of using the conventional algorithm, because of the large number of bits that are flipped at the low SNRs of less than 10 dB, the bit flipping decoding algorithm maybe more complex. At higher SNRs, upwards of 10 dB, the bit flipping algorithm because of the lower complexity order could be less complex than the conventional algorithm.

5.8 Conclusion

This chapter presents the results of simulations carried out. The first section of the chapter presents the error patterns that we came across during the development of the algorithm. Four classes of error patterns were identified in the development of the decoding algorithm namely, the hidden error pattern, the simple error pattern, complex error pattern and the un-decodable error pattern. Two methods were proposed for solving the hidden error pattern and the un-decodable error pattern. Single bit flipping or the conventional decoding algorithm could be used to solve both problems. Results from simulations using both showed that using the conventional decoding method to decrease the shadow area into one that could be managed by the bit flipping algorithm yielded a better BER performance.

The BER performance results showed that the conventional decoding algorithm outperformed the bit flipping algorithm at low SNRs, and the bit flipping algorithm outperformed the conventional algorithm at higher SNRs. This was not expected. A WER comparison was done, and it showed that the bit flipping algorithm corrected more block errors than the conventional. The reason for the conventional algorithm outperforming the bit flipping at low SNRs can be attributed to the fact that the bit flipping changes more bits when decoding. When there is a decoding failure, more errors are introduced by the bit flipping algorithm. This was proved in Figure 5.3, where a comparison of the frequency of each algorithm having more erroneous bits after decoding. At lower SNRs, the bit flipping decoding algorithm was more likely to have more errors after decoding than the conventional.

Section 5.7 discusses the decoding complexities of the two algorithms. The Section presents a breakdown of the decoding complexity of the conventional decoding algorithm according to the major steps carried out in decoding a BCH code. The section then presents the complexity of decoding using the bit flipping decoding algorithm as the summation of the complexity of finding the row and column syndromes and the complexity of correcting the errors. The complexity of finding the syndromes is the same for the bit flipping algorithm and the conventional decoding algorithm. The difference in complexity lies in correcting errors. The complexity of the bit flipping algorithm is dependent on the number of errors and the error pattern, thereby the SNR. At low SNRs, the number of required flips could increase the complexity of the algorithm such that it is more complex than the conventional algorithm. At high SNRs, the number of required flips is so low such that the bit flipping algorithm would be lower in complexity when compared to the conventional algorithm.

Chapter 6 – Conclusion

6.1 Introduction

Most product code decoding algorithms decode the rows initially then use the information obtained from decoding the rows when decoding the columns. This method was presented by Elias [1] when he first presented product codes. The method presented by Elias does not take advantage of the fact that each bit in a binary product affects the same number of parity checks as the dimensions of the code. A new decoding algorithm is presented in this study which takes advantage of the multidimensional nature of product codes. Using the fact that a single bit in a product code affects more than one parity a new algorithm based on the simple bit flipping algorithm presented by Gallager for LDPC codes in [3] is developed. This chapter presents the concluding remarks on the novel algorithm.

6.2 Concluding Remarks

FEC has been identified as a means to improve the reliability of the wireless communication channel, by correcting errors that may occur. The factors that influence the development of a FEC system include [53]:

- the error correcting capability of the decoding algorithm,
- the delay introduced by the decoding algorithm,
- and the complexity of the decoding of the algorithm.

In this study we present a novel decoding algorithm for product codes, that addresses the complexity of the decoding algorithm by using the bit flipping algorithm commonly used for LDPC codes and the error correcting capability by ensuring the decoding algorithm addresses the permanent error pattern identified by Abramson in [2]. The bit flipping algorithm has been shown to be a simple algorithm.

The error patterns that were introduced when information was transmitted through an AWGN channel were studied. Four types of error patterns were identified namely, simple error patterns, complex error patterns, un-decodable error patterns and hidden error patterns. Grouping the error patterns was done by identifying a shadow area (a region showing the possible error positions), and checking whether the shadow area became smaller after flipping all the bits in the shadow area. Simple error patterns resulted in a smaller area, complex error patterns resulted in the same shadow area, and hidden error patterns did not yield a shadow area despite the presence of errors. From the error patterns a decoding algorithm was developed. The aim of the algorithm is to reduce the shadow area until syndrome checking of the rows and columns would result in all the syndromes checking.

The un-decodable error pattern has a shadow area that is wider and higher than the minimum distance of the component codes. Two methods are proposed to reduce the width and height of the shadow area for un-decodable error patterns, before the actual bit flipping algorithm can be used. The first method, which yields better BER performance results, uses the conventional decoding algorithm to decode the rows and/or columns with errors until the height and the width of the shadow area are less than the minimum distances of the component codes. The second method uses flipping of a combination of bits in the rows or columns with errors. This method is more complex, and BER performance results showed that it is outperformed by the first.

The algorithm was compared to the conventional decoding algorithm proposed by Elias. BER comparisons show that in the Rayleigh fading channels and AWGN channels, the conventional decoding algorithm outperforms the bit flipping algorithm at low SNRs. These results are not as

expected, as the bit flipping decoding algorithm can correct error patterns the conventional decoding algorithm cannot. The results from a WER comparison show that the bit flipping decoding algorithm outperforms the conventional decoding algorithm. WER results are a better measure of the performance of a communication system because they show the performance of a network in terms of the number of successfully decoded codewords. The reason the conventional decoding algorithm is outperforming the bit flipping algorithm in terms of the BER can be attributed to the fact that when the bit flipping algorithm fails, it introduces more errors to the received sequence than the conventional algorithm does.

At high SNRs, the bit flipping algorithm outperforms the conventional decoding algorithm. The BER comparison carried out in the Rayleigh fading channel, show that the bit flipping algorithm has a 4 dB gain over the conventional decoding algorithm at a BER of 10^{-4} when using the conventional algorithm to reduce the size of the shadow area when un-decodable error patterns occur.

Simulation results show that the complexity of the algorithm is high at low SNRs. At a SNR of 10 dB, the complexity of the algorithm improves considerably. The results show that the complexity of the algorithm could be better than those of the conventional decoding algorithm which uses the computational intensive Berlekamp-Massey algorithm to find the error locator polynomial for the row and column codewords in a BCH code.

6.3 Further work

Parts of the study can be improved or extended in the future. In this section we present some possible ways the study can be further improved upon.

6.3.1 The Rician Channel

Two mathematical models are used when simulating wireless communication, the Rayleigh fading channel and the Rician fading channel. The Rayleigh fading channel is used to simulate wireless channels where there is no direct line of sight between the transmitter and the receiver. For this study only the Rayleigh fading channel was used. The study can be further enhanced by

carrying out simulations on the Rician channel. The Rician channel is used for simulations of wireless channels where there is direct line of sight between the communicating devices. Direct line of sight wireless communication is used in some backhaul access technologies like Wi-Max. The performance of the decoding algorithm in line of sight communication can be ascertained by studying its performance using the Rician channel.

6.3.2 Soft decision decoding

The complexity of the algorithm could be greatly improved if the reliability of the bits in the shadow area could be used when deciding which bits to flip. This could also reduce the occurrences of decoding errors. The current decoding algorithm relies on reducing the shadow area into an area that has a number of rows with errors which is less than the minimum distance and a number of columns with errors which is less than the minimum distance, before using the bit flipping algorithm. This is done to ensure the bit flipping algorithm does not have decoding failures and introduce hidden errors. Soft decision decoding can be used to improve the complexity of this process. The least reliable bits will be flipped first. Using reliability information, enhances the algorithm and moves it closer to finding an algorithm that decodes product codes without using iterative decoding of the rows then the columns, and still offer good performance in the wireless environment.

6.3.3 Implementation on a communication channel

This study has been carried out using Matlab based simulations. Simulating on software offers the advantage of being able to manage the attributes of the communication channel and testing the performance of the algorithm for given conditions of the channel. The simulation software also gives tractable conditions for developing the algorithm. Live networks have ever changing conditions. It would be ideal for the algorithm to be tested on a live network. This will give a clear indication of the performance of the decoding algorithm in a live setup. Testing on a live wireless network could be done to also validate the results obtained from simulations.

6.3.4 Generalization of the algorithm

Chapter 3 of the thesis shows that product codes have been developed into other codes like augmented codes and concatenated codes. The algorithm can prove to be beneficial if it can be generalized and applied to other codes that have a structure similar to that of product codes, like generalized concatenated codes.

References

[1] P. Elias, “Error-free coding,” *IEEE Transactions on Information Theory*, vol. 4, no. 4, pp. 29–37, 1954.

[2] N. Abramson, “Cascade decoding of cyclic product codes,” *IEEE Transactions on Communication Technology*, vol. C, no. x, pp. 398–402, 1968.

[3] R. Gallager, “Low-density parity-check codes,” *IRE Transactions on Information Theory*, 1962.

[4] Y. Kwok and V. Lau, “A novel channel-adaptive uplink access control protocol for nomadic computing,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 13, no. 11, pp. 1150–1165, 2002.

[5] W. Stallings, *Wireless communications & networks*, 2nd ed. New Jersey, USA: Pearson Prentice Hall, 2005.

[6] I. Guvenc, S. Gezici, Z. Sahinoglu, and U. Kozat, *Reliable communications for short-range wireless systems*. New York, USA: Cambridge University Press, 2011.

[7] B. Alfonsi, “Wi-Fi certification program aims to boost battery life,” *IEEE Distributed Systems Online*, vol. 7, no. 1, p. 4, Jan. 2006.

[8] B. P. Crow, I. Widjaja, J. G. Kim, and P. T. Sakai, “IEEE 802.11 Wireless local area networks,” *IEEE Communications Magazine*, vol. 35, no. 9, pp. 116–126, 1997.

[9] R. A. Powers, “Batteries for low power electronics,” *Proceedings of the IEEE*, vol. 83, no. 4, pp. 687–693, Apr. 1995.

[10] J. Dielissen, S. Sawitzki, and K. van Berkel, “Multistandard FEC decoders for wireless devices,” *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 55, no. 3, pp. 284–288, Mar. 2008.

[11] D. Feldmeier, “Fast software implementation of error detection codes,” *IEEE/ACM Transactions on Networking*, vol. 3, no. 6, pp. 640–651, 1995.

[12] R. Anand, K. Ramchandran, and I. V. Kozintsev, “Continuous error detection (CED) for reliable communication,” *IEEE Transactions on Communications*, vol. 49, no. 9, pp. 1540–1549, 2001.

[13] W. Peterson and D. Brown, “Cyclic codes for error detection,” *Proceedings of the IRE*, vol. 49, no. 1, pp. 228–235, 1961.

[14] S. Lin and D. J. Costello, *Error Control Coding*, 2nd ed. Upper Saddle River: Pearson Prentice Hall, 2004.

[15] A. Salomon and O. Amrani, "Product construction of codes for Rayleigh-fading channels," *IEEE Transactions on Information Theory,*, vol. 53, no. 1, pp. 416–428, 2007.

[16] M. Baldi, G. Cancellieri, and F. Chiaraluce, "Interleaved product LDPC Codes," *IEEE Transactions on Communications*, vol. 60, no. 4, pp. 895–901, Apr. 2012.

[17] E. Kurtas, K. R. Narayanan, and C. N. Georghiades, "On the performance of turbo product codes over partial response channels," *IEEE Transactions on Magnetics*, vol. 37, no. 4, pp. 1932–1934, Jul. 2001.

[18] D. L. Schilling, "Wireless communications going into the 21st century," *IEEE Transactions on Vehicular Technology*, vol. 43, no. 3, pp. 645–652, 1994.

[19] J. Hagenauer and E. Lutz, "Forward error correction coding for fading compensation in mobile satellite channels," *IEEE Journal on Selected Areas in Communications*, vol. 5, no. 2, pp. 215 – 225, 1987.

[20] G. Forney, "Burst-Correcting Codes for the Classic Bursty Channel," *IEEE Transactions on Communications*, vol. 19, no. 5, pp. 772–781, Oct. 1971.

[21] K. Sripimanwat, *Turbo Code Applications: a journey from a paper to realization*. Dordrecht, Netherlands: Springer, 2005.

[22] D. Tang and R. Chien, "Cyclic product codes and their implementation," *Information and Control*, 1966.

[23] J. Yuan and W. Ye, "A novel block turbo code for high-speed long-haul DWDM optical communication systems," *Optik - International Journal for Light and Electron Optics*, vol. 120, no. 15, pp. 758–764, Oct. 2009.

[24] M. Schwartz, P. H. Siegel, and a. Vardy, "On the asymptotic performance of iterative decoders for product codes," *Proceedings. International Symposium on Information Theory, 2005. ISIT 2005.*, pp. 1758–1762, 2005.

[25] M. Kasahara and Y. Sugiyama, "New classes of binary codes constructed on the basis of concatenated codes and product codes," *IEEE Transactions on Information Theory,*, vol. 22, no. 4, pp. 462–468, 1976.

[26] J. M. Rulnick and N. Bambos, "Mobile power management for maximum battery life in wireless communication networks," *Proceedings of IEEE INFOCOM '96. Conference on Computer Communications*, vol. 2, pp. 443–450, 1996.

[27] K. Kang and H. Shin, “Reduced data rates for energy-efficient Reed–Solomon FEC on fading channels,” *IEEE Transactions on Vehicular Technology,*, vol. 58, no. 1, pp. 176–187, 2009.

[28] J. Sarkar and S. Sengupta, “Differential FEC and ARQ for radio link protocols,” *IEEE Transactions on Computers,*, vol. 55, no. 11, pp. 1458–1472, 2006.

[29] R. Pyndiah, “Near-optimum decoding of product codes: Block turbo codes,” *IEEE Transactions on Communications,*, vol. 46, no. 8, pp. 1003–1010, 1998.

[30] T. Moon, *Error correction coding: mathematical methods and algorithms*. New Jersey, USA: John Wiley & Sons, Inc., 2005.

[31] A. Nouh and A. Banihashemi, “Bootstrap decoding of low-density parity-check codes,” *IEEE Communications Letters,*, vol. 6, no. 9, pp. 391–393, 2002.

[32] A. Chan and F. Kschischang, “A simple taboo-based soft-decision decoding algorithm for expander codes,” *IEEE Communications Letters,*, vol. 2, no. 7, pp. 183–185, 1998.

[33] K. Ovadya and O. Amrani, “On soft decision decoding of product codes,” *2004 23rd IEEE Convention of Electrical and Electronics Engineers in Israel, 2004. Proceedings.*, pp. 92–95, 2004.

[34] M. Fossorier, K. Kobara, and H. Imai, “Modeling bit flipping decoding based on nonorthogonal check sums with application to iterative decoding attack of McEliece cryptosystem,” *IEEE Transactions on Information Theory,*, vol. 53, no. 1, pp. 402–411, 2007.

[35] M. Shan, C. Zhao, and M. Jiang, “Improved weighted bit-flipping algorithm for decoding LDPC codes,” *Communications, IEE Proceedings*, no. 2003, pp. 2003–2006, 2005.

[36] X. Wu, C. Zhao, and X. You, “Parallel Weighted Bit-Flipping Decoding,” *IEEE Communications Letters*, vol. 11, no. 8, pp. 671–673, Aug. 2007.

[37] H. Xu and F. Takawira, “A new structure of single parity check product codes,” *AFRICON, 2004. 7th AFRICON Conference in Africa*, vol. 2, pp. 67–70, 2004.

[38] F. Wang, Y. I. Tang, and F. A. N. Yang, “The iterative decoding algorithm research of Turbo Product Codes,” *The 2010 International Conference on Apperceiving Computing and Intelligence Analysis Proceeding*, no. 2, pp. 97–100, Dec. 2010.

[39] E. Zehavi, “8-PSK trellis codes for a Rayleigh channel,” *IEEE Transactions on Communications*, vol. 40, no. 5, pp. 873–884, May 1992.

[40] H. O. Burton and D. D. Sullivan, “Errors and Error Control,” *Proceedings of the IEEE*, vol. 60, no. 11, pp. 1293–1301, 1972.

[41] P. Sweeny, *Error control coding: from theory to practice*. West Sussex: John Wiley & Sons, Ltd, 2002.

[42] J. Moreira and P. Farrell, *Essentials of error-control coding*. West Sussex: John Wiley & Sons Ltd, 2006.

[43] W. Stallings, *Data and computer communications*, Eighth. New Jersey: Pearson Prentice Hall, 2007.

[44] H. Liu, H. Ma, M. El Zarki, and S. Gupta, "Error control schemes for networks: An overview," *Mobile Networks and Applications*, vol. 2, pp. 167–182, 1997.

[45] J. G. Proakis and M. Salehi, *Digital Communications*, Fifth Edit. New York: McGraw-Hill, 2008.

[46] A. Bhattacharya, *Digital Communication*. New Delhi: Tata McGraw-Hill, 2006.

[47] D. Tse and P. Viswanath, *Fundamentals of wireless communication*. New York, USA: Cambridge University Press, 2005.

[48] B. Sklar, "Rayleigh fading channels in mobile digital communication systems Part I. Characterization," *IEEE Communications Magazine*,, vol. 35, no. 7, pp. 90–100, 1997.

[49] S. Krishna, "Rayleigh multipath channel model," 2008. [Online]. Available: http://www.dsplog.com/2008/07/14/rayleigh-multipath-channel/. [Accessed: 19-Sep-2013].

[50] M. Zorzi, R. Rao, and L. Milstein, "ARQ error control for fading mobile radio channels," *IEEE Transactions on Vehicular Technology*,, pp. 1–9, 1997.

[51] C. Leung and A. Lam, "Forward error correction for an ARQ scheme," *IEEE Transactions on Communications*,, vol. 29, no. 10, pp. 1514–1519, 1981.

[52] M. Tsai and C. Shieh, "Forward-looking forward error correction mechanism for video streaming over wireless networks," *IEEE Systems Journal*,, vol. 5, no. 4, pp. 460–473, 2011.

[53] S. Ling and C. Xing, *Coding theory: a first course*. New York: cambridge university press, 2004.

[54] C. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, pp. 379–423, 1948.

[55] A. Nafaa, T. Taleb, and L. Murphy, "Forward error correction strategies for media streaming over wireless networks," *IEEE Communications Magazine*,, vol. 46, no. 1, pp. 72–79, 2007.

[56] L. Rizzo, “Effective erasure codes for reliable computer communication protocols,” *ACM SIGCOMM Computer Communication Review*, vol. 27, pp. 24–36, 1997.

[57] A. Viterbi, “Convolutional codes and their performance in communication systems,” *IEEE Transactions on Communication Technology*,, vol. 19, no. 5, pp. 751–772, 1971.

[58] D. J. Costello and G. D. Forney, “Channel Coding: The Road to Channel Capacity,” *Proceedings of the IEEE*, vol. 95, no. 6, pp. 1150–1177, Jun. 2007.

[59] G. D. Forney, “Convolutional codes I: Algebraic structure,” *IEEE Transactions on Information Theory*,, vol. 16, no. 6, pp. 720–738, 1970.

[60] W. Peterson and E. Weldon, *Error-correcting codes*, 2nd ed. Cambridge, Massachusetts, US: The MIT Press, 1972.

[61] R. H. Morelos-zaragoza, *The Art of Error Correcting Coding*, 2nd ed. West Sussex: John Wiley & Sons Ltd, 2006.

[62] R. E. Blahut, *Algebraic Codes for Data Transmission*. New York, USA: Cambridge University Press, 2003.

[63] R. Hamming, “Error detecting and error correcting codes,” *Bell System technical journal*, vol. 29, no. 2, 1950.

[64] D. Bertozzi, L. Benini, and G. De Micheli, “Error control schemes for on-chip communication links : The energy –reliability tradeoff,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*,, vol. 24, no. 6, pp. 818–831, 2005.

[65] W. E. Ryan and S. Lin, *Channel Codes: Classical and Modern*. New York, USA: Cambridge University Press, 2009.

[66] E. Krouk and S. Semenov, *Modulation and Coding Techniques in Wireless Communications*. Chichester, UK: John Wiley & Sons, Ltd, 2011.

[67] O. Al-Askary, “Iterative decoding of product codes,” 2003.

[68] P. Adde, C. Leroux, and C. Jego, “Full-parallel architecture for turbo decoding of product codes,” *Electronics Letters*, vol. 42, no. 18, pp. 1052–1053, 2006.

[69] B. G. Dorsch, “A decoding algorithm for binary block codes and J-ary output channels (Corresp.),” *IEEE Transactions on Information Theory*,, vol. vol.20, no. no.3, pp. 391–394, 1974.

[70] W. Jin and M. P. C. Fossorier, “Reliability-Based Soft-Decision Decoding With Multiple Biases,” *IEEE Transactions on Information Theory*, vol. 53, no. 1, pp. 105–120, Jan. 2007.

[71] A. A. Al-Shaikhi and M. A. Kousa, “Modified algorithm for hard decision decoding of product codes,” *2004 IEEE Wireless Communications and Networking Conference, 2004. WCNC.*, vol. 3, pp. 1760–1763, 2004.

[72] A. Viterbi, “Error bounds for convolutional codes and an asymptotically optimum decoding algorithm,” *IEEE Transactions on Information Theory,*, pp. 260–269, 1967.

[73] G. D. Forney, “The viterbi algorithm,” *Proceedings of the IEEE*, pp. 302–309, 1973.

[74] L. Bahl and J. Cocke, “Optimal decoding of linear codes for minimizing symbol error rate (Corresp.),” *IEEE Transactions on Information Theory*, no. January 1972, pp. 284–287, 1974.

[75] J. Wolf, “Efficient maximum likelihood decoding of linear block codes using a trellis,” *IEEE Transactions on Information Theory*, vol. 24, no. 1, pp. 76–80, Jan. 1978.

[76] S. B. Sukhavasi, S. B. Sukhavasi, H. Khan, and C. Pilla, “Performance evaluation of turbo codes using hard decision Viterbi algorithm in VHDL,” *International Journal of Engineering Research and Applications*, vol. 2, no. 3, pp. 2849–2861, 2012.

[77] G. D. Forney, “Concatenated codes,” Cambridge, Massachusetts, 1965.

[78] D. Slepian, “Some further theory of group codes,” *Bell System technical journal*, vol. 39, pp. 1219–1252, 1960.

[79] V. V. Rao and S. M. Reddy, “A (48, 31, 8) linear code,” *IEEE Transactions on Information Theory*, vol. 19, no. 5, pp. 709–711, 1973.

[80] M. Sablatash, “An error-correcting coding scheme for teletext and other transparent data broadcasting,” *IEEE Transactions on Broadcasting,*, vol. 36, no. 1, 1990.

[81] J. Lodge, R. Young, and P. Hoeher, “separable map filters for the decoding of product and concatenated codes,” *, 1993. ICC 93. Geneva.*, no. 4, pp. 1740–1745, 1993.

[82] P. G. Farrell, “A survey of array error control codes,” *European Transactions on Telecommunications*, vol. 3, no. 5, pp. 441–454, Sep. 1992.

[83] P. Farrell and S. Hopkins, “Burst-error-correcting array codes,” *Radio and Electronic Engineer*, vol. 52, no. 4, pp. 188–192, 1982.

[84] J. Li, K. R. Narayanan, and C. N. Georghiades, “Product accumulate codes : A class of codes with near-capacity performance and low decoding complexity,” *IEEE Transactions on Information Theory*, vol. 50, no. 1, pp. 31–46, 2004.

[85] B. Honary, G. Markarian, and P. Farrell, “Generalised array codes and their trellis structure,” *Electronics Letters*, vol. 29, no. 6, pp. 541–542, 1993.

[86] O. Milenkovic, N. Kashyap, and D. Leyba, “Shortened array codes of large girth,” *IEEE Transactions on Information Theory*, vol. 52, no. 8, pp. 3707–3722, Aug. 2006.

[87] O. Amrani, “Nonlinear codes: The product construction,” *IEEE Transactions on Communications*, vol. 55, no. 10, pp. 1845–1851, Oct. 2007.

[88] A. J. Salomon and O. Amrani, “Augmented product codes and lattices : Reed – Muller codes and Barnes – Wall lattices,” *IEEE Transactions on Information Theory,*, vol. 51, no. 11, pp. 3918–3930, 2005.

[89] V. Rao and S. Reddy, “A (48, 31, 8) linear code (Corresp.),” *IEEE Transactions on Information Theory*, vol. 19, no. 5, 1973.

[90] M. Goldberg, “Augmentation techniques for a class of product codes,” *IEEE Transactions on Information Theory*, vol. 19, no. 5, pp. 666–672, Sep. 1973.

[91] C. Berrou, A. Glavieux, and P. Thitimajshima, “Near Shannon limit error-correcting coding and decoding: Turbo-codes. 1,” *IEEE International Conference on Communications, 1993. ICC ’93 Geneva. Technical Program, Conference Record,*, no. 1, 1993.

[92] H. O. Burton and E. J. Weldon, “Cyclic product codes,” *IEEE Transactions Information Theory,*, 1965.

[93] Y. L. Lee and M. C. Cheng, “Cyclic mappings of product Codes,” *IEEE Transactions on Information Theory,*, vol. 21, no. 2, pp. 233–235, 1975.

[94] S. Lin and E. J. Weldon, “Further results on cyclic product codes,” *IEEE Transactions on Information Theory*, vol. 16, no. 4, pp. 452–459, Jul. 1970.

[95] B. S. Rajan, H. S. Madhusudhana, and M. U. Siddiqi, “On cyclic product codes,” in *1993 IEEE International Symposium on Information Theory, 1993. Proceedings.*, 1993, no. 1059, p. 400.

[96] L. Bahl and R. Chien, “Single- and multiple-burst-correcting properties of a class of cyclic product codes,” *IEEE Transactions on Information Theory*, vol. 17, no. 5, pp. 594–600, Sep. 1971.

[97] G. Caire, G. Taricco, and G. Battail, “Weight distribution and performance of the iterated product of single-parity-check codes,” in *IEEE Global Telecommunications Conference, 1994. Communications Theory Mini-Conference Record, 1994 IEEE GLOBECOM.,*, 1994, pp. 206–211.

[98] G. Battail, M. C. Decouvelaere, and P. Godlewski, “Replication decoding,” *IEEE Transactions on Information Theory*, vol. 25, no. 3, pp. 332–345, 1979.

[99] M. A. Kousa, “A novel approach for evaluating the performance of SPC product codes under erasure decoding,” *IEEE Transactions on Communications*, vol. 50, no. 1, pp. 7–11, 2002.

[100] O. Al-askary, “Low complexity maximum-likelihood decoding of product codes,” in *IEEE International Symposium on Information Theory, 2000. Proceedings.*, 2000, vol. 18, no. January 1972, p. 87.

[101] J. Bredeson and S. Hakimi, “Decoding of graph theoretic codes (Corresp.),” *IEEE Transactions on Information Theory,*, vol. 13, no. 2, pp. 348, 349, 1967.

[102] J. L. Massey, “Advances in threshold decoding,” *Advances in Communication Systems*, vol. 3, pp. 91–114, 1968.

[103] T. Kasami and S. Lin, “On the construction of a class of majority-logic decodable codes,” *IEEE Transactions on Information Theory,*, vol. 17, no. 5, pp. 600–610, 1971.

[104] L. R. Bahl and R. T. Chien, “Multiple-burst-error correction by threshold decoding,” *Information and Control*, vol. 15, no. 5, pp. 397–406, Nov. 1969.

[105] L. Calabi and H. Haefeli, “A class of binary systematic codes correcting errors at random and in bursts,” *IRE Transactions on Information Theory,*, vol. 5, no. 5, pp. 79–94, 1959.

[106] W. Gore, “Further results on product codes,” *IEEE Transactions on Information Theory,*, vol. 16, no. 4, pp. 446–451, 1970.

[107] N. Duc, “On the Lin-Weldon majority-logic decoding algorithm for product codes (Corresp.),” *IEEE Transactions on Information Theory,*, vol. 19, no. 4, pp. 581–583, 1973.

[108] S. Reddy, “On decoding iterated codes,” *IEEE Transactions on Information Theory,*, vol. I6, no. 5, pp. 624–627, 1970.

[109] S. Reddy and J. Robinson, “A decoding algorithm for some convolutional codes constructed from block codes,” *Information and Control*, vol. 13, no. 5, pp. 492–507, 1968.

[110] C. Nill and C.-E. W. Sundberg, “List and soft symbol output Viterbi algorithms: extensions and comparisons,” *IEEE Transactions on Communications*, vol. 43, no. 2,3,4, pp. 277–287, 1995.

[111] G. D. Forney, “Convolutional codes II. Maximum-likelihood decoding,” *Information and control*, vol. 25, no. 3, pp. 222–266, 1974.

[112] G. D. Forney, “Generalized minimum distance decoding,” *IEEE Transactions on Information Theory,*, vol. 12, no. 2, pp. 125–131, 1966.

[113] D. Chase, “A class of algorithms for decoding block codes with channel measurement information,” *IEEE Transactions on Information Theory*, vol. 18, no. 1, pp. 170–182, 1972.

[114] R. Pyndiah, A. Glavieux, A. Picart, and S. Jacq, “Near optimum decoding of product codes,” *IEEE Global Telecommunications Conference, 1994. GLOBECOM ’94. Communications: The Global Bridge.,*, vol. 1, pp. 339 – 343, 1994.

[115] J. Zhang, J. S. Yedidia, and M. P. C. Fossorier, “Low-latency decoding of EG LDPC codes,” *Journal of Lightwave Technology*, vol. 25, no. 9, pp. 2879–2886, Sep. 2007.

[116] Y. Kou, S. Lin, and M. Fossorier, “Low-density parity-check codes based on finite geometries: a rediscovery and new results,” *IEEE Transactions on Information Theory,*, vol. 47, no. 7, pp. 2711–2736, 2001.

[117] J. Zhang and M. P. C. Fossorier, “A modified weighted bit-flipping decoding of low-density parity-check codes,” *IEEE Communications Letters,*, vol. 8, no. 3, pp. 165–167, 2004.

[118] Y. Jiang, *A practical guide to error-control coding using MATLAB*. Artech House, 2010.

[119] C. Jego and P. Adde, “Row-column parallel turbo decoding of product codes,” *Electronics Letters*, vol. 42, no. 5, pp. 5–6, 2006.

[120] J. Hong and M. Vetterli, “Simple algorithms for BCH decoding,” *IEEE Transactions on Communications,*, vol. 43, no. 8, pp. 2324–2333, 1995.

[121] D. G. Forney, “On decoding BCH codes,” *IEEE Transactions on Information Theory,*, vol. 11, no. 4, pp. 549–557, 1965.