Ollscoil na hÉireann

Simplified Decoding Techniques for Linear Block Codes

A thesis presented to the

National University of Ireland, Cork

for the Degree of

Doctor of Philosophy

by

Shraddha Srivastava

Supervisor: Dr. Emanuel. M. Popovici

Head of Department: Prof. Nabeel Riza

Department of Electrical and Electronic Engineering,

University College Cork, Ireland

2013

Acknowledgment

First and foremost, all thanks are due to God Almighty for the innumerable blessings through my life, one of which is this thesis. I would also like to express my gratitude towards my supervisor, Dr. Emanuel Popovici for giving me the opportunity for pursuing this PhD and also for his guidance and support throughout the PhD. Thanks for giving me this wonderful opportunity. I would also like to thank Dr. Fernando Hernando for his guidance, encouragement and help with the mathematical parts of the thesis. Thanks for your patience and support.

I am very grateful to Dr. Michael O'Sullivan, Dr. Kwankyu Lee, Dr. Christian Spagnol and Dr. Richard McSweeney for their support, suggestions and some wonderful discussions and ideas. Many thanks are given to Gerard Hooton for keeping the entire system running. Also, I would like to thank Dr. Gary Mcguire, Claude Shannon Institute and Science Foundation Ireland (grant number 06/MI/006) for the financial support to this work.

This work would not have been so enjoyable without my friends. I would especially like to thank Rashmi, Nidhi, Richa, Nasim, Dimpy, Tom, Stevan, Chen, Gerard and Brian for making the four years of my stay in Ireland joyful.

In the end, I would like to express my gratitude to my family for their love and encouragement. Especially, I would like to say thanks to my parents and my husband for dealing so calmly with my constant stress. All this work is dedicated to my father C.P. Srivastava, my mother Uma Srivastava, my sister Pooja Srivastava and my husband Saurabh Mathur.

Abstract

Error correcting codes are combinatorial objects, designed to enable reliable transmission of digital data over noisy channels. They are ubiquitously used in communication, data storage etc. Error correction allows reconstruction of the original data from received word. The classical decoding algorithms are constrained to output just one codeword. However, in the late 50's researchers proposed a relaxed error correction model for potentially large error rates known as list decoding.

The research presented in this thesis focuses on reducing the computational effort and enhancing the efficiency of decoding algorithms for several codes from algorithmic as well as architectural standpoint. The codes in consideration are linear block codes closely related to Reed Solomon (RS) codes. A high speed low complexity algorithm and architecture are presented for encoding and decoding RS codes based on evaluation. The implementation results show that the hardware resources and the total execution time are significantly reduced as compared to the classical decoder. The evaluation based encoding and decoding schemes are modified and extended for shortened RS codes and software implementation shows substantial reduction in memory footprint at the expense of latency. Hermitian codes can be seen as concatenated RS codes and are much longer than RS codes over the same alphabet. A fast, novel and efficient VLSI architecture for Hermitian codes is proposed based on interpolation decoding. The proposed architecture is proven to have better than Kötter's decoder for high rate codes.

The thesis work also explores a method of constructing optimal codes by computing the subfield subcodes of Generalized Toric (GT) codes that is a natural extension of RS codes over several dimensions. The polynomial generators or evaluation polynomials for subfield-subcodes of GT codes are identified based on which dimension and bound for the minimum distance are computed. The algebraic structure for the polynomials evaluating to subfield is used to simplify the list decoding algorithm for BCH codes. Finally, an efficient and novel approach is proposed for exploiting powerful codes having complex decoding but simple encoding scheme (comparable to RS codes) for multihop wireless sensor network (WSN) applications.

Contents

List of Figures

List of Tables

List of Publications

The thesis is based in part on the following publications:

- Srivastava, S.; Spagnol, C. and Popovici E.; "Analysis of a set of Error correcting schemes in multihop Wireless Sensor Networks", IEEE PRIME' 2009, pp. 1-4.
- Fernando, H.; O'Sullivan, M.; Popovici, E. and Srivastava, S.; "Subfield Subcodes of Generalized Toric Codes", IEEE International Symposium on Information Theory(ISIT), 2010, pp. 1125-1129.
- Srivastava, S.; McSweeney, R.; Spagnol, C. and Popovici E.; "Efficient Berlekamp-Massey Based Recursive Decoder for Reed-Solomon Codes", IEEE MIEL'2012, pp. 379-382.
- Srivastava, S.; McSweeney, R. and Popovici E.; "Area efficient evaluation based coding of shortened Reed-Solomon codes for memory constrained applications", presented in IET ISSC, 2012.
- Srivastava, S.; Lee, K. and Popovici E.; "Fast parallel implementation of Hermitian decoder based on interpolation", submitted to IET Communications, 2013.

List of Abbreviations

$\mathbb{F}_p$: Finite field, where p is prime

$\mathbb{F}_q$: Extended finite field with $q = p^m$ elements

$\mathbb{F}_{p^s}$: Subfield of $\mathbb{F}_{p^m}$ if $s|m$

p : Characteristic of the field $\mathbb{F}_{p^m}$

α : Primitive element of $\mathbb{F}_q$

C : Code over field $\mathbb{F}_{p^m}$

c : Codeword

$g(x)$: Generator polynomial

G : Generator matrix

H : Parity check matrix

$C^{\perp}$: Dual code of C

$Tr(C)$: Trace code of C

n : Length of code

k : Dimension of code

$d_{min} = d$: Minimum distance of code

D_{BCH} : BCH bound or designed minimum distance

d' : Actual minimum distance

t : Error correction capability

t_{GS} : List error correction capability

n_s : Shortened code length

k_s : Shortened code dimension

$\mathbb{F}[x]$: Ring of polynomials in x with coefficients in $\mathbb{F}$

$\mathbb{F}[x]_{\leq k-1}$: Polynomials in $\mathbb{F}[x]$ with degree $\leq k-1$

m : Message word

e : Error

r : Received word

$\Lambda(x)$: Error locator polynomial

$\Omega(x)$: Error evaluator polynomial

S^r : Received word syndromes

S^e : Error word syndromes

S^c : Codeword syndromes

$\mathbb{F}[x, y]$: $\{x^i y^j : i \geq 0, y \geq 0\}$, the set of bivariate monomials with coefficients in $\mathbb{F}$

D : Code over subfield

$I_{\bar{b}}$: Cyclotomic coset

$f_{I_{\bar{b}}}$: Polynomial with support in cyclotomic coset $I_{\bar{b}}$

$S(w, r)$: Sphere containing all codewords at distance $\leq r$ from w

$S_{=}(w, r)$: Sphere containing all codewords at distance $= r$ from w

E_{enc} : Encoding energy

$\binom{n}{i}$: C_i^n, binomial coefficient

P_e : Probability of error

Chapter 1

Introduction

"The fundamental problem of communication is that of reproducing at one point either exactly or approximately a message selected at another point."

Claude E. Shannon - A Mathematical Theory of Communication

Noise causing data corruption is an inherent part of communication systems. In everyday life, where one wants to transmit a message or some information from the sender's to receiver's end, noisy communication channels cause message distortion during transmission. Hence, a fundamental challenge is to design a strategy which makes it possible for information to be reliably transmitted over a noisy channel. In order to ensure reliable communication, Automatic Repeat Query (ARQ) [1] and Forward Error Correction (FEC) [2] are the two main techniques that are used. ARQ achieves error free communication by resending the lost or damaged packets until all the packets are correctly received. The basic advantage is that the error only needs to be detected and not corrected at the decoder's end. This simplifies the operations at the receiver's end. However, the caveat is that as the Signal to Noise Ratio (SNR) decreases the ARQ must spend more and more time and power in retransmitting the packets. A potential solution is the use of FEC, which corrects the packets having a limited number of errors. This step is computationally heavier at decoder's end in comparison to using ARQ scheme and can be quite demanding if the message is long. Another approach is to combine both schemes in form of Hybrid-ARQ (HARQ) [3] to achieve a communication strategy that best utilizes the resources and meets the bit error rate (BER) requirement of the application.

Error correcting codes are designed to cope with the problem of unreliable communication. The basic idea is to introduce redundancy in the data before transmitting so that the original data can be recovered using the redundancy even if part of the transmitted data (codeword) is in error. Communication is perhaps the most common application for error correcting codes such as transmitting data over the internet, telephone lines, wireless channels and for satellite broadcasting. Although it is not limited just to communication, it is also used widely for data storage such as compact disk players or hard drives where the errors are introduced by scratches, etc [4].

1.1 Historical interlude

The basis of error correcting codes is called coding theory and is closely related to various disciplines like Mathematics, Engineering and Computer Science. The journey of coding theory started in 1948 when Claude Shannon published a landmark paper "A mathematical theory of communication" [5]. He introduced the definition of information and also gave a precise real number called channel capacity such that arbitrarily reliable communication is possible at any rate below the channel capacity. Shannon also proved the converse result that there exist codes for any rate less than the capacity of the channel for which one can have reliable communication. These remarkable results marked the birth of Coding and Information theories. Shortly after Shannon's work, Hamming [6] addressed the problem that was causing a job on a computer machinery to be aborted, with error correction. He defined the Hamming code that can correct up to one bit error. Since then many mathematicians and engineers are trying to build and improve upon the numerous coding schemes for a variety of applications. They attempted to achieve the performance promised by Shannon with a reasonable complexity of implementation.

Shannon modeled the noise probabilistically [5] such that the behavior of the channel is well known. It also ensured that the probability of occurrence of too many or too few errors is low. An undesirable effect of this model is its dependency on how well the noise is modeled. For some cases, it is nearly impossible to model the channel closely and accurately. Therefore, one solution is to consider a channel as an adversary such that the noise can be added randomly with the only constraint on the number of errors.

This channel model was proposed by Hamming [6] and will be used throughout the thesis.

1.2 Basics of communication

The main building blocks of a communication system are [7]:

1. An information source producing the message to be transmitted;
2. A transmitter operating on the message to produce a codeword that can be sent through the channel;
3. A channel performing as a medium to transmit the codeword which is carrying the information needed to be transmitted;
4. A receiver transforming the signal back to the original message;
5. A destination that can be a machine or a person who requires the information;

The problem with this communication system is that the channel is generally noisy and it adds error while sending the data over the channel which results in the modification of the message that has been sent. With all the noise in the channel, whatever information is transmitted from the source might not be received in the same form at receiver. Suppose if a binary string "1100" is transmitted, it can be received as "1101" due to noise. The common structure of a communication system with error correction blocks is shown in Figure 1.1. It shows all the five components of the communication system as mentioned above.

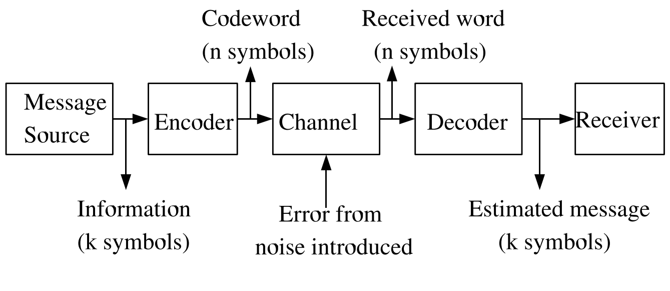

Figure 1.1: Communication system

A symbol is defined as an element of finite field. In this thesis, we will assume that there is a sender who wants to send k message symbols over a channel. The sent message is encoded to n symbols (codeword) before passing it to the channel. At the receiver, we receive n symbols which may not be the same as the transmitted n symbols, so the decoder tries to recover the original message symbol using the $n - k$ redundant symbols added during encoding. The main goals of coding theory are the construction of new codes with simple encoding and decoding schemes and to use them efficiently for communication, computing, storage etc applications.

Some of the important milestones in algebraic coding theory were the discovery of a class of multiple error correcting codes known as BCH codes by Hocquenghem [8] and independently by Bose and Chaudhuri [9] around 1960. Further, the related non-binary Reed-Solomon (RS) codes were found by Reed and Solomon [10] around 1960 as well. Then the search started to find the most efficient encoding and decoding schemes for these codes. Typically, the definition of a code gives the encoding algorithm as well. The decoding procedure is more challenging from the algorithmic and architectural perspectives. The list decoding algorithm, which is a relaxation over the traditional decoding algorithms, outputs either the original message or gives the decoding failure when the number of errors that occurred is more than the error correction capability of the code. In this thesis, we focus more on the decoding algorithms as they are more complex than encoding algorithms and also on some applications of error correction codes in WSN.

1.3 Decoding problem for error correcting codes

A code is a finite subset of a metrical space such as Hamming space [11]. While transmitting the codeword, the channel causing distortion in the codeword can be modeled as additive noise, which means that some random element of the space in consideration gets added to the transmitted codeword. From Figure 1.2, it can be seen that the code construction can be defined as the sphere packing problem where the dots at the center (as shown in the figure of the circles) are the codewords [12].

One of the important algorithmic tasks in coding theory is the decoding function for error correcting codes. Since, the encoding functions are generally easier to perform the main focus of coding theory has always been the decoding function. A code with no algebraic structure can only be defined as an exhaustive list of codewords. The decoding problem will also become much more exhaustive as the received word would need to

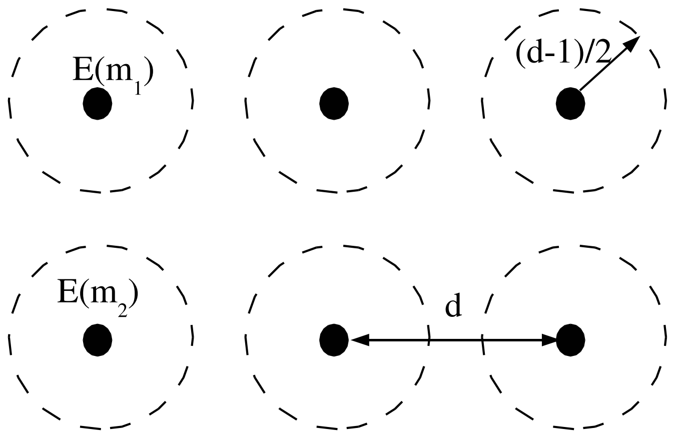

Figure 1.2: Code with minimum distance d

be compared with all the codewords present in the list. This will make the decoding more difficult with the increasing length of the list of codewords. This is not a practical nor an optimum solution. In order to obtain codes that allow efficient encoding and decoding algorithms, coding theorists imposed an algebraic structure on the code. The codes studied in this thesis are linear codes which means that the codes are linear spaces which allows the application of linear algebra in the algebraic coding theory. In the case of linear cyclic codes, we further require that any cyclic shift of a codeword also results in a codeword [11]. Algebraic geometric codes are obtained by evaluating a space of functions in a fixed set of points. The construction of these codes involve choosing the point set for evaluation from an algebraic curve and the codes differ according to the curve in consideration. The positions in a codeword are associated with the points on the curve. Because of their algebraic structure, the codes are much more easier to encode and decode than the non linear codes.

For decoding to be performed in an easier manner, the codes should be designed such that the receiver should be able to identify the actual transmitted word. So, the codewords are assumed to be far apart from each other that means the optimal code is the one with the maximum of minimum distance as d [13]. The classical decoders can correct up to half of the minimum distance i.e. $\frac{d-1}{2}$. But, what will be the case if the number of errors exceeds this bound? The classical decoder reports a decoding failure and does not give any output. Although, it is true when the number of errors exceeds $\frac{d-1}{2}$, error patterns can not be corrected due to the presence of multiple codewords at a distance $\frac{d}{2}$ from received word. However, the sparsity of the codewords gives this a relaxation and it implies that most received words have a unique closest codeword and yet classical (unique) decoding algorithm will result in decoding failure. This limitation stems from the requirement that the decoder should compute a single codeword which should be same as transmitted word. Unique decoding will either compute the actual transmitted word or will result in decoding failure. There is a useful relaxation over unique decoding called list decoding which excluded the condition of unique codeword and thus allowed coding theorists to correct beyond half the minimum distance bound.

The concept of list decoding was introduced by Elias [14] and Wozencraft [15] in late 50's. List decoding allows the decoder to output a list of codewords including the transmitted word instead of just one word as in the case of classical unique decoders. During the communication, if the number of errors exceeds the error correction capability of the code the unique decoder will give a decoding failure. But, what if the decoding failure is not a feasible option? The receiver can use the list decoding algorithm. If the result of list decoding is just one codeword, the receiver accepts it as transmitted codeword and works as a unique decoder. Even if the list contains more than one codeword then it is more desirable as compared to the decoding failure. Especially when information transmission is really costly like space communication then list decoding is much more desirable than a decoding failure. Lets assume a situation where pictures of a planet say Mars are transmitted after encoding via satellite to some space station located on Earth. Due to the noise added by the natural interference in space, the encoded data gets severely distorted. In these cases, the re-transmission will be really costly but with the help of list decoding algorithm we can at least get a list of codewords which contains the transmitted word. The list size can further be reduced with the added information that the decoded information should look something like a planet and therefore the picture can be recovered.

1.4 Wireless sensor networks

As already mentioned, FEC is used to reduce the cost of retransmitting a message. Often this cost translates into power consumption. A fast emerging technology which is affected by power due to low cost is Wireless Sensor Network (WSN). A WSN is a collection of nodes that are organized to form a cooperative system [16]. Each node is capable of processing as they contain microcontrollers or DSP chips, various kinds of memories like flash, program or data, an RF transceiver and a power source. The continuous analog signal produced by the sensors is digitized by an analog-to-digital converter (ADC) for further processing. This new technology consisting of sensors and actuators has an unlimited potential for various application areas like medical, transportation and defense. WSNs are highly dynamic and are prone to faults because of certain environmental conditions or connectivity interruptions. Hence error correction codes are used in order to have reliable communication and for low power.

In this thesis, we target WSN as the end application and we focused on the error control coding side while other researchers in the embedded systems group at UCC were also working on the motes design and implementation side. A sensor node may come in various sizes however motes are of genuine small dimensions. Sensor nodes are great for deployment in hostile areas or even over large regions. The main components of a sensor node as shown if Figure 1.3 are a microcontroller for data processing, flash memory due to cost and storage capacity, power source that is a battery, transceiver and one or multiple sensors. A sensor node should consume the lowest possible energy and should be small in size.

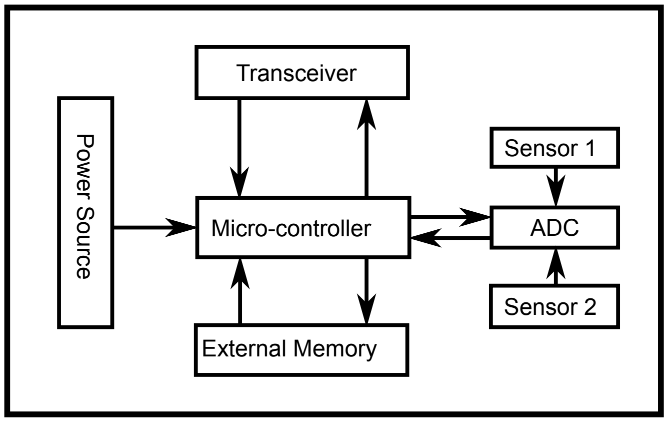

Figure 1.3: Sensor node architecture

One of the biggest constraints on the lifetime and maintenance of WSN is the battery life of sensor nodes. Any operation like computation or transmission can have a significant effect on the battery life of sensor nodes. Certainly, energy consumption at node level has been an important design consideration for WSNs [17]. In this thesis, we made an attempt to reduce the energy consumption at node level while transmitting the data through a WSN. Various codes have been examined for reliable communication of data over multihop WSN. A new model for data transmission has been designed which is different from the previous works involving encoding/decoding at each and every node.

1.5 Thesis contribution

The main focus of this thesis is RS type linear block codes like Hermitian [18], BCH and Toric codes [19]. The various codes discussed in the thesis are similar to RS codes in terms of the encoder's complexity which makes them useful for WSNs. Hermitian codes can be seen as a concatenation of generalized RS codes [20] which simplifies the encoding scheme resulting in an encoder's complexity comparable to RS codes [21]. The simplified encoding scheme for Hermitian codes based on this fact makes it useful for a transmission scheme in multihop WSNs. Also, BCH codes can be seen as subfield subcodes of RS codes whereas Toric codes are the natural extension of RS codes over r dimensions. The main idea is reduce the complexity and enhance the efficiency of unique and list decoding algorithms for BCH, RS and Hermitian codes. In this thesis, we have simplified some of the existing decoding algorithms for BCH and RS codes. The codes are studied from an evaluation perspective (which we will explain later in chapter 3). The main focus is on direct decoding algorithms similar in complexity to Berlekamp-Massey (BM) type algorithm [22], [23], [24] as the BM algorithms is one of the most efficient methods used in decoding of linear block codes. This thesis presents new algorithms and architectures for Hermitian codes and improved decoding algorithm and architectures for RS and BCH codes.

1.6 Thesis organization

As coding theory is a combination of mathematics and engineering, the thesis exploits both the aspects of coding theory. The fundamental objectives of coding theory is code construction (which covers the encoding part as well) and finding an efficient decoding algorithm with simple hardware implementations. RS codes are the most popular algebraic geometric codes which are widely used in industrial applications [25]. To summarize, in this thesis we present a high speed low complexity algorithm and architecture for encoding and decoding RS codes and Hermitian codes based on evaluation. The thesis presents an approach of constructing codes by computing the subfield subcodes of GT Codes which are basically the multidimensional analogues of BCH codes. BCH codes can also be seen as subfield subcodes of RS codes, which provides a mathematical structure that can simplify the list decoding scheme for BCH codes. In the end, we present an approach of utilizing powerful codes for eg. Hermitian, RS (list decoded) etc. with low complexity encoding for multihop WSN.

Chapter 2 introduces the basic concepts of finite field and error correcting codes. It also discusses about the construction of BCH, RS, Hermitian and Toric codes. Chapter 3 discusses the aspects of efficient implementation of the RS decoder. It presents a high speed low complexity algorithm and architecture for encoding and decoding RS codes based on evaluation. The idea is also extended for shortened RS codes and the modified decoding algorithm is also presented for shortened codes.

In chapter 4, we discuss about the efficient architecture of a Hermitian decoder based on a new unique decoding algorithm presented recently in a paper by Lee and O'Sullivan from an interpolation perspective. We present the complete architecture for the Lee-O'Sullivan algorithm. In chapter 5, we study the concept of subfield subcodes and also how BCH codes can be seen as subfield-subcodes of generalized RS codes. We study the subfield subcodes of GT codes over $\mathbb{F}_{p^m}$ which are the multidimensional analogues of BCH codes.

The problem of decoding cyclic codes can be written as an algebraic system of equations where the solutions are closely related to the errors occurred. Based on the algebraic structure for the polynomial generators of BCH codes in chapter 5, the list decoding algorithm for BCH codes via syndrome variety is simplified. Chapter 6 discusses how the use of this algebraic structure reduces the number of variables in the system of equations, which in turn modifies and simplifies the list decoding problem for BCH codes. Also, it is discussed how this structure can be used for simplifying the Roth Ruckenstein factorization step for BCH codes.

Among the codes discussed, RS codes are already widely used in the industry. But, some of the very powerful codes like Hermitian and list decoder RS codes do not find applications due to their complex decoding schemes. Chapter 7 discusses the usage of powerful codes with low complexity encoding as the transmission method in multihop WSN such that the encoding is performed only at the first node and decoding only at the base station.

With Chapter 8 we conclude the thesis. It also discusses the contributions of this thesis along with ideas for future work. In the next chapter, we will introduce the concepts of group, ring and field to understand better the mathematical parts of the thesis. Also, we will discuss about the linear block codes and several classes of it.

Chapter 2

Background

In this chapter, we will review some of the basic definitions relating to the error correcting codes and will define some of the standard notations that will be followed throughout the thesis. We will initially present the basic definition of finite field, linear block codes and then discuss several families of linear block codes. The main focus will be on RS type codes like BCH, RS, Toric and Hermitian on which the thesis chapters are based. The outline of the chapter is as follows. Section 2.1 introduces the basic concepts of finite field algebra. The basics of error correcting codes are defined in section 2.2 and the mathematical definitions related to linear error correcting codes are given in section 2.3. In section 2.4, we discuss about the construction of several codes on which the thesis chapters are based.

2.1 Basic finite field algebra

Many communication and storage systems require a special type of arithmetic over finite fields. In this section, we will review the basic notions and facts about finite field algebra. The notations are followed from [26]. A finite field is also often known as Galois Field (GF), after the French mathematician Pierre Galois. A finite field with q elements will be represented as $\mathbb{F}_q$ of $GF(q)$. A field is a subset of a larger set called a group and the formal definition of group is given as follows:

Definition 2.1.1. *A set G on which a binary operation $*$ is defined is called a group if the following conditions are satisfied.*

1. *The binary operation $*$ is associative.*

2. *G contains an element e such that, for any a in G, $a * e = e * a = a$. This element e is the identity element in G, and is unique for the group G.*

3. *For any element a in G, there exists another element a' in G such that $a * a' = a' * a = e$. The element a' is called an inverse of a, where a is also an inverse of a'. The inverse of a group element is unique in the group.*

A group is also commutative if the binary operation $*$ also satisfies the condition that for any a and b in G, $a * b = b * a$. A subset H of G is called a subgroup if it is a group with respect to the operation $*$ of G. The next definitions introduce the concept of ring and finite field.

Definition 2.1.2. *A ring $\mathbb{R}$ is a set of elements together with the binary operations $+$ and $*$ satisfying the following properties:*

1. *$\mathbb{R}$ is a group under addition '$+$'.*

2. *Closure: For every $a, b \in \mathbb{R}$, $a * b = c \in \mathbb{R}$*

3. *Associativity: For every $a, b, c \in \mathbb{R}$, $(a * b) * c = a * (b * c)$*

4. *Distributivity: Multiplication is distributive over addition; that is for any three elements $a, b, c \in \mathbb{R}$, $a * (b + c) = a * b + a * c$*

Definition 2.1.3. *Let $\mathbb{F}$ be a set of elements on which two binary operations, called addition '$+$' and multiplication '$*$' are defined. The set $\mathbb{F}$ together with the two binary operations '$+$' and '$*$' is a field if the following conditions are satisfied.*

1. *$\mathbb{F}$ is a group under '$+$'. The identity element with respect to addition is called the zero element or the additive identify of $\mathbb{F}$ and is denoted by 0.*

2. *The set of non-zero elements in $\mathbb{F}$ is a group under multiplication '$*$'. The identity element with respect to multiplication is called the unit element or the multiplicative identity of $\mathbb{F}$ and is denoted by 1.*

3. *Multiplication is distributive over addition that is for any three elements $a, b, c \in \mathbb{F}$, $a * (b + c) = a * b + a * c$.*

The ring $\mathbb{R}$ of uni-variate polynomials with coefficients from $\mathbb{F}$ can be denoted as $\mathbb{F}[x]$. The polynomial is called monic if the coefficient of the leading term is 1. An interesting property to be noted is that any polynomial $f(x) \in \mathbb{F}[x]$ of degree at most d can have at most d roots. A subset S of a ring $\mathbb{R}$ is called a subring of $\mathbb{R}$, if S is closed under addition and multiplication, and forms a ring under those operations. The formal definition of subring is as follows:

Definition 2.1.4. *A subring I of a ring R is an ideal if whenever $r \in R$ and $a \in I$ then $ra \in I$ and $ar \in I$.*

If $a_1, a_2, \ldots, a_s \in \mathbb{R}$, $< a_1, a_2, \ldots, a_s$ is denoted as the ideal generated by $\{a_1, a_2, \ldots, a_s\}$ which is the smallest ideal of $\mathbb{R}$ containing $a_1, a_2, \ldots, a_s$.

Definition 2.1.5. *Let $\mathbb{F}$ be a field and let K be a subset of $\mathbb{F}$ such that K is a field under the operations of $\mathbb{F}$. Then K is called the subfield of $\mathbb{F}$ and $\mathbb{F}$ is called an extension field of K and is denoted by $\mathbb{F}/K$ (read as "$\mathbb{F}$ over K") is a field extension to signify that $\mathbb{F}$ is an extension field of K. If $\mathbb{F} \neq K$ then K is called proper subfield of $\mathbb{F}$. A field containing no proper subfields is called a prime field.*

Definition 2.1.6. *A Galois extension is an algebraic field extension $\mathbb{F}/K$ such that $\mathbb{F}$ is a splitting field of a separable polynomial with coefficients in K. A splitting field of a polynomial with coefficients in a field is a smallest field extension of that field over which the polynomial splits or decomposes into linear factors. The term $[\mathbb{F} : K]$ denotes the degree of field extension.*

All finite fields are in the form of either $\mathbb{F}_p$ (p being prime) or an extension of a prime field. Thus, the number of elements in the field is always prime. Also, for any $q = p^m$, where p is a prime number there exists only one finite field. The number p for any field $\mathbb{F}_q$ is known as the characteristic of that field. The group of non-zero elements $\mathbb{F}^*$, also known as a multiplicative group, is cyclic and can be represented as $\mathbb{F}_q^* = \{1, \alpha, \alpha^2, \ldots, \alpha^{q-2}\}$ where $\alpha \in \mathbb{F}_q \setminus 0$ and is called the primitive element of the field.

2.2 Basics of error correcting codes

As already mentioned, error correcting codes provide a systematic method of adding redundancy to a message before sending it. A familiar example to understand the redundancy principle is the Ispell program [27] that checks for the spelling when someone makes some mistakes while typing. The set of all words in English is a small subset of all possible strings, and a large amount of redundancy is built into the valid English words. If someone makes a mistake it will change it to some correct word that is a valid word in English dictionary. In most cases, the misspelled word resembles the correct word and thus Ispell gives us a method to correct the wrong word. The principle of error correcting codes is similar to Ispell and is based on the in-built redundancy.

The two important functions or concepts involved with the error correcting codes are encoding and decoding. Before going in to any details, we would like to explain these notions.

- Encoding: The encoding function performed at the sender's end maps a message word m having k symbols or message length k to a codeword c containing n symbols where $\sum$ is the alphabet. Here, the data stream m that has to be encoded can be seen as string over some alphabet $\sum$, where the size of alphabet is q. In most of the applications q is considered to be prime power and $\sum$ as finite field $\mathbb{F}_q$. The encoding function can be written mathematically as $E : \sum^k \to \sum^n$, such that $E(m)$ will give us the codeword c.

- Decoding: The encoded message is sent to the receiver via a channel which adds noise resulting in a distorted copy of the actual sent codeword. The goal of the decoding function is just the reverse of encoding, where the decoding function maps an n symbol received word to a k symbol message word at the receiver's end. It can be defined mathematically as $D : \sum^n \to \sum^k$. But, there is a bound on the correction capability of a code. It is not possible to correct any amount of errors that occur during transmission. The bound is related to a code parameter called a 'minimum distance' which is the smallest distance between two distinct codewords of a code and this bound determines how far a code can correct.

The ideal code should be designed such that the possibility of one codeword to be confused with another should be negligible even if errors occur during transmission. This means if the codewords are far apart from each other then after the introduction of errors during transmission, the decoder will be able to provide the original codeword instead of any other codeword. The notion of distance between two codewords was given by Richard Hamming and is now referred to as Hamming distance. He also formally gave the concept of (minimum) distance of a code as the smallest distance between two distinct codewords. Figure 1.2 shows a code with minimum distance 'd'. It also shows the spheres of radius $\frac{d-1}{2}$ around each codeword which are all disjoint and is called as Hamming balls. In this model, the optimal code is the one with the largest Hamming distance among all the codes that have a certain number of codewords. The concept of Hamming and minimum distance will be discussed more mathematically in the next section.

2.3 Definitions

We will present here some of the basic definitions. The number of elements in the finite field is of the form $q = p^m$ where p is a prime number and m is a positive integer. The prime p is called the characteristic of the field and the positive integer m is called the dimension of the field over its prime field. The notations in this subsection are mainly followed from [27].

2.3.1 Basic mathematical definitions for codes

For any integer $q > 1$, we will use $[q]$ to denote the set $\{1, ..., q\}$.

- Code: An error correcting code C is a subset of $[q]^n$ for positive integers q and n. The elements of C are called as the codewords.

- Alphabet size: The number q is referred as to the alphabet size of the code and is generally called as q-ary code. For $q = 2$, it is called as binary code.

- Block length: The integer n is the block length of the code.

- Dimension: For a code defined over $\mathbb{F}_q$, the quantity $k = log_q|C|$ is known as the dimension of the code. This definition makes much more sense in the case of linear block codes which we will discuss in the next section.

- Rate: It is defined as the ratio of dimension and block length of the code and can be simply represented by the quantity $R = k/n$

- Hamming Distance: For any two strings $x, y \in [q]^n$ such that $x =< x_1, x_2, \ldots, x_n >$ and $y =< y_1, y_2, \ldots, y_n >$, the Hamming distance (represented as $\triangle(x, y)$) between them can be defined as the number of coordinates in which they differ i.e. $\triangle(x, y) = |\{i|x_i \neq y_i\}|$.

- Minimum Distance: Minimum distance of a code C is the minimum Hamming distance between any two codewords and can be formally defined as, $dist(C) = \underset{c_1, c_2 \in C, c_1 \neq c_2}{min} \triangle(c_1, c_2)$.

2.3.2 Linear Codes

An important family of codes is known as linear codes. They do not only have simple representations but also have practical advantages in terms of providing efficient encoding and decoding schemes.

Definition 2.3.1. *A linear code of block length n is a linear subspace (over some field $\mathbb{F}_q$) of $\mathbb{F}_q^n$.*

A linear code C defined has q^k elements, where k is the dimension of the code as a vector space over $\mathbb{F}_q$. The important code characteristics to define a code is length n, dimension k, minimum distance d and its alphabet size q. Thus, the parameters of the code is written as $[n, k, d]_q$. For linear codes, the all zero string is always a codeword. Hence the minimum distance for a linear code is the minimum Hamming weight of any non-zero codeword, where the Hamming weight of a vector is the number of positions with non-zero values. For linear codes, one of the most fundamental bounds is the Singleton Bound [28] which combines the three main parameters k, n and d as given in lemma 2.3.2. Another popular bound is the GilbertVarshamov bound [29], [30], which limits the parameters of a code which are not necessarily linear, is given in 2.3.3.

Lemma 2.3.2. *Singleton Bound: Every linear code $[n, k, d]$ code C satisfies $k + d \leq n - 1$.*

Definition 2.3.3. *Gilbert-Varshamov bound: Let $A_q(n, d)$ denote the maximum possible size of $q-ary$ code C with parameters $[n, k, d]$. Then:*

$$A_q(n, d) \geq \frac{q^n}{\sum_{j=0}^{d-1} \binom{n}{j} (q-1)^j}.$$

In block codes, data is encoded in discrete blocks and not continuously unlike convolutional codes [31]. Another important property of some block codes are that they are cyclic in nature which means a cyclic shift of a codeword will also give a codeword. The development of cyclic codes was started by Prange[32] in 1957.

2.4 Basic linear code families and their constructions

The design of error correcting codes should first consider the application and the requirements. Many codes exist that have characteristics and capabilities that are suited to different applications. In an optical communication systems where the main constraint is throughput and error correction capability, the decoder can be complex to achieve high performance. However in the case of short range multihop wireless networks where there is a constraint on memory and power, the design of FEC for this application is a trade-off between architectural complexity and power. Although, a FEC may result in significant reduction in the BER for any given value of transmit power, the additional processing power that is consumed during encoding and decoding must be considered while using FEC for WSN. If the additional processing power is greater than the coding gain then the system is better off without FEC. However, FEC is an incredible asset for WSN if the processing power is smaller than additional transmission power for ARQ [33]. The nodes must be able to encode and decode at every point in the network if it is a full-duplex or a multihop network. In the case of Wireless Body Area Network (WBAN), the communication range of sensor nodes is quite small. Thus, the SNR is presumed to be quite high. In these cases, a large error correction scheme would be a waste of resources (because of the decoder's complexity) and the need is to design codes that are relatively simple to both encode and decode with small error correction capability at high rate and would be able to handle the case of burst errors.

In this section, we will describe basic code families that are important from an algorithmic and architectural perspective for this thesis. All these codes are cyclic linear block codes.

2.4.1 BCH codes

One interesting key feature of BCH codes is that during the design of a code, there is a precise control over the number of symbol errors that can be corrected by the code. This directly implies that we can design a binary BCH code which can correct multiple bit errors. Here, we will mainly discuss about primitive narrow sense BCH codes. Let $q = p^m$. A BCH code defined over $\mathbb{F}_q$ with $n = p^m - 1$ is known as a primitive BCH code. For a given prime p and positive integers m and d such that $d \leq p^m - 1$, a primitive narrow-sense BCH code over the finite field $\mathbb{F}_q$ and minimum distance at least d can be constructed by the following method:

Let α be a primitive element of $\mathbb{F}_q$. For any positive integer i, let $m_i(x)$ be the minimal polynomial of α^i. The generator polynomial of the primitive BCH code is defined as the least common multiple $g(x) = lcm(m_1(x), \ldots, m_{d-1}(x))$. The polynomial $g(x)$ has coefficients in $\mathbb{F}_q$ and divides $x^n - 1$. The polynomial code defined by it is a cyclic code [25]. The consecutive roots of a generator polynomial $g(x)$ may run from $\alpha^c, \ldots, \alpha^{c+d-2}$ but if the consecutive roots of generator polynomials are starting from α^1 $(c = 1)$, it is called a narrow sense primitive BCH code.

2.4.2 Reed-Solomon codes

RS codes introduced in [10] are one of the most important cyclic codes which are well studied and are widely use in industrial applications like compact disk players, high speed modems such as ADSL, xDSL and space communications. Some detailed applications can be found in [34] [35].

Proposition 2.4.1. *A RS code is an* $[n, k, d = n - k + 1]_q$ *code.*

RS codes actually match the singleton bound as mentioned in Lemma 2.3.2. Such codes are called as maximum distance separable (MDS) codes, since they have the maximum possible distance for a given block length and dimension. The MDS property of this algebraic code is advantageous to build several efficient encoding and decoding algorithms. In the thesis. we tried to find some efficient encoding and decoding schemes for RS codes.

A systematic code is any error-correcting code in which the message symbols are embedded in the codeword. However, for a non-systematic code the codeword does not contain the input symbols. Every non-systematic linear code can be transformed into a systematic code with the same properties essentially. Systematic codes have the advantage that the parity data can simply be appended to the source block, and receivers do not need to recover the original source symbols if received correctly. However, in this thesis, we will follow the non-systematic encoding technique for several codes and will prove that the decoding technique based on evaluation has much better performance as compared to the classical decoder.

2.4.2.1 Non-systematic encoder from an evaluation perspective

The message that is needed to be sent is mapped to a univariate polynomial and the codeword is constructed by evaluating that polynomial at several points of the field. Lets say the n distinct field points are $\alpha_1, \alpha_2, \ldots, \alpha_n$. The message space consists of polynomial $m \in \mathbb{F}_q[x]$ with degree at most $k-1$ and a message $'m'$ is encoded as:

$$m \rightarrow \langle m(\alpha_1), m(\alpha_2), ..., m(\alpha_n) \rangle.$$

The message space can be seen as $\langle m_o, m_1, \ldots, m_{k-1} \rangle$ with the polynomial in the form $m_0 + m_1 x + \ldots + m_{k-1} x^{k-1}$.

2.4.2.2 Systematic encoder

For systematic encoding, a codeword polynomial is obtained by concatenating a message polynomial, $m(x)$ with a parity polynomial $p(x)$ such that

$$c(x) = p(x) + x^{n-k} m(x)$$

We can think of shifting a message polynomial $m(x)$, into the rightmost k positions of the codeword and then the last $n-k$ positions at the leftmost side are occupied by the parity polynomial. For computing the parity polynomial $p(x)$, we divide $x^{n-k} m(x)$ with the generator polynomial $g(x)$ and it can be written as:

$$x^{n-k} m(x) = q(x) g(x) + p(x)$$

such that $q(x)$ and $p(x)$ are the quotient and the remainder polynomials respectively. $p(x)$ can also be computed directly as $p(x) = x^{n-k} m(x) \; mod \; g(x)$.

2.4.3 Toric Codes

J. Hansen introduced recently the notion of a Toric code [19] which are also studied in [36], [37], [38]. Toric codes are a class of r-dimensional cyclic codes. They are in a sense a natural extension of RS codes over r dimensions. Let P be a convex polytope such that $P \cap \mathbb{Z}^r$ is properly contained in the rectangular box $[0, q-2]^r$ for some prime power q. A Toric code can then be obtained by evaluating linear combinations of the monomials with exponent vector in $P \cap \mathbb{Z}^r$ at some subset of points or at all the points of $(\mathbb{F}_q^*)^r$. D. Ruano introduced a natural generalization of this family, the so called Generalized Toric Codes (GT) [39], which consist of the evaluation of any polynomial algebra in the algebraic torus. It is clear from his construction that any Toric code is a GT code. The formal definition is taken from [40]:

Definition 2.4.2. *Let $\mathbb{F}_q$ be a finite field with primitive element α. For $f \in \mathbb{Z}^r$ with $0 \leq f_i \leq q-2$ for all i, let $P_f = (\alpha^{f_1}, \dots, \alpha^{f_r})$ in $(\mathbb{F}_q^*)^r$. For each $e = (e_1, e_2, \dots, e_r) \in P \cap \mathbb{Z}^r$, let x^e be the corresponding monomial and write*

$$(p_f)^e = (\alpha^{f_1})^{e_1} \cdots (\alpha^{f_r})^{e_r}$$

The Toric code $C_P(\mathbb{F}_q)$ over the field $\mathbb{F}_q$ associated to P is the linear code of block length $n = (q-1)^r$ with generator matrix $G = ((p_f)^e)$, where the rows are indexed by the $e \in P \cap \mathbb{Z}^r$, and the columns are indexed by $p_f \in (\mathbb{F}_q^*)^r$. In other words, letting $L = Span\{x^e : e \in P \cap \mathbb{Z}^r\}$, we define the evaluation mapping as:

$$ev : L \to \mathbb{F}_q^{(q-1)^r}$$

$$g \mapsto (g(p_f) : f \in (\mathbb{F}_q^*)^r)$$

Then $C_P = ev(L)$. If the field is clear from the context, we will often omit it in the notation and simply write C_P . The matrix G will be called the standard generator matrix for the Toric code.

Substituting $r = 1$ will make P then just a line segment in $[0, q-2] \in \mathbb{R}$ with end points as integers. This clearly implies that $1-$ dimensional Toric Codes will simply correspond to RS codes. From here, we can easily deduce that higher dimensional Toric codes will be the natural extension of RS codes and will thus have several similar properties. One of them is that they are all $m-$dimensional cyclic codes [36]. So, the hope is that Toric codes will have similarly good parameters as RS codes. Several examples showing that Toric codes have very good parameters are given in [38], where the minimum distances are either equal or better than the best known codes for a given code length and dimension.

2.4.4 Hermitian Codes

Hermitian codes are a very interesting subclass of Algebraic Geometric (AG) codes having lots of good properties. The codes constructed by choosing points from a curve and a space of rational functions on this curve are called AG codes. RS codes are a particular case of AG codes where the curve is an affine line [41]. With RS codes, the main disadvantage of RS codes is that it is not possible to create codes with long lengths. For RS codes, the code length is limited to the size of the finite field ($\mathbb{F}_q$) over which they are defined i.e. q. Goppa [42] constructed efficient long block codes using methods from algebraic geometry. Later, Ţsfasman, Vládut and Zink [43] constructed a sequence of codes from algebraic geometry which perform better than the Gilbert-Varshamov bound . Since this discovery, many of the coding theorists have focused their attention on this area which is codes from curves. Hermitian curves are one of the most famous curves in this category as it is a maximal curve. Curves defined over $\mathbb{F}_{q^2}$ whose number of points on the curve reaches the Serre bound [44] (which gives an upper bound on the number of rational points for an algebraic curve) are called maximal curves. The curve has total $q^3 + 1$ points including one point at infinity. Thus, Hermitian codes offer desirable properties over RS codes such as large code lengths over the same finite field, good error correction at high code rates, etc.

We will follow the notations and the construction from [45]. Before going into detail, we recall some definitions from [46]. A homogeneous polynomial is a sum of monomials with all the terms having the same degree. A projective plain curve over $\mathbb{F}_q$ is the set of zeros of a homogeneous monomial. The complexity of the curve is defined by its genus. A rational point P_i on curve is a point having coordinates in $\mathbb{F}_q$. A rational function is given by a quotient of homogeneous polynomials of same degree. A divisor is defined as a formal sum of rational points P_i over the curve. The Hermitian curve $\mathcal{H}$ over $\mathbb{F}_{q^2}$ is described by the homogeneous equation:

$$\mathcal{H} : X^{q+1} = ZY^q + Z^qY.$$

Using the notation in [47], with $x = X/Z$ and $y = Y/Z$ the curve equation can be re written as:

$$\mathcal{H} : x^{q+1} = y^q + y.$$

$\mathcal{H}$ has genus $g = q(q-1)/2$, contains one point at infinity P_∞ (the point where the curve $\mathcal{H}$ intersects the line at infinity) and q^3 non-singular affine rational points. The curve $\mathcal{H}$ is a maximal curve as it attains the Serre upper bound which is a bound on maximum number of points on the curve over a field $\mathbb{F}_q$. The two divisors of the curve are defined as $D = \sum_{i=1}^{n=q^3} P_i$ and $G = mP_\infty$ such that G has disjoint support from D. We restrict our attention to codes with a divisor G as it will allow the maximum code length because it has just one point in its support. The functional code can be defined as:

$$C_L(D, G) = \{(f(P_1), f(P_2), \ldots, f(P_n)) | f \in L(G)\}$$

where, $L(G)$ is the set of rational functions which may have a pole at P_∞ with order at most m.

The residual code is defined as:

$$C_\Omega(D, G) = (C_L(D, G))^\perp.$$

The code is called a residual code since it evaluates residues of differentials at point P_i defining the divisor D [11].

Systematic encoding of Hermitian codes is discussed in [45]. In Chapter 4, we will discuss about the non-systematic evaluation based encoding of Hermitian codes. Also, we will propose an efficient VLSI architecture for Hermitian decoder based on a new decoding scheme from Lee and O'Sullivan.

In this chapter, we presented several linear block codes and their construction. RS codes are industrially most popular codes among all these codes and the next chapter discusses an efficient way on encoding and decoding it. Also, the encoding and decoding algorithms are modified and extended for shortened codes.

Chapter 3

Efficient Berlekamp-Massey based recursive decoder for RS and shortened RS codes

RS codes are one of the most popular AG codes. These codes have great power and utility with common usage in all kinds of industrial applications like computing, communications and storage. RS decoders are used to protect the digital data against the errors that occurred and reduce the signal to noise ratio in the transmission process. Due to the balanced encoding and decoding complexities of RS codes and robust error correction capability, RS codes are chosen to provide reliable and energy efficient communication in most of the cases.

However for a few applications like WSN [48], there is a limitation on the memory and hardware resources. In those cases, it is not always necessary to use the full length RS codes but a shortened code. RS codes used in a systematic context can be shortened to any arbitrary extent by keeping the redundancy fixed which means the minimum distance will remain the same and the values of n and k will be decreased. The process of shortening can produce a smaller code of desired length from a large code over the same field. For example, the widely used code $(255, 223)$ RS code can be converted to $(160, 128)$ code by padding the unwanted portion with zeros and not transmitting them [25]. Shortening a block code will remove the symbols from the message portion. For shortening a $(63, 53)$ code to a $(54, 44)$ code, we can simply put the n and k value as

54 and 44 respectively, in the encoder and decoder block masks. However, if it has to be shortened to a $(28, 18)$ code, it is needed to explicitly specify that the field is $\mathbb{F}_{2^6}$. Otherwise, the RS blocks will assume that the code is shortened from a $(31, 21)$ code. The choice of using an error correcting code depends on the decoding speed and the area consumed by the decoder as well as the amount of the simplicity with which it can be implemented in hardware and software,

3.1 Introduction

Much effort and research in the past has been focused to obtain a high speed and area efficient implementations for RS decoders in order to meet the demand for low power consumption at high data rates. The traditional RS decoder has three blocks i.e. syndrome computation, key equation solver (combined with the computation of error evaluator polynomial) and Chien Search combined with Forney's Formula used for error evaluation. The key equation solver is based on the Euclidean algorithm [49] or Berlekamp Massey Algorithm. In the next definition, we will introduce the concept of shortened codes. Let C be an RS code defined over $\mathbb{F}_q$ with parameters $[n, k, d]$ such that the length of the code is n, dimension is k and the minimum distance is d. A shortened code C_s is constructed by dropping data symbols. This implies the length n_s of shortened code C_s is reduced by dropping out data symbols resulting in a decrease of dimension from k to k_s.

For a shortened code, say C_s, the length of codeword is $n_s < n$ as we do not want to transmit the full length code for some of the applications due to system constraints or by design. The most convenient approach to implement it is to set the last $n - n_s$ symbols to zero and then not transmit them. The dimension of the shortened code will be k_s such that the minimum distance (d) remains the same for both the shortened and full length code. This preserves the error correction capability of the shortened codes. The original message can be recovered if the number of errors occurred ν satisfies the condition, $\nu \leq t = \lfloor \frac{d-1}{2} \rfloor$, where t is the error correction capability of the RS code. The idea is to find an efficient decoding algorithm and architecture for RS and Shortened RS based on recursive use of the discrepancy computation step of the Berlekamp Massey algorithm. In [50], decoding of RS codes directly to the message polynomial (without using the Chien search and Forney's formula) has been discussed in frequency domain. In [51], another algorithm for direct decoding has been discussed based on fast Fourier transforms and the Euclidean algorithm. This algorithm also has the same time complexity as the direct decoding algorithm based on BM algorithm. But, any of these papers did not discuss about the benefits of the direct decoding algorithm in hardware like speed or simple circuitry. In this chapter, we focus on the efficiency of the direct decoding algorithm from a hardware perspective. The scheme proposed in this chapter requires less clock cycles and smaller area as compared to the classical decoder leading to lower energy and area consumption for RS codes. The extended algorithm for shortened RS codes has shown memory savings for the software implementations making it more beneficial for memory constrained WSN applications.

The outline of the chapter is as follows. Section 3.2 describes the encoding scheme based on evaluation for RS codes and a modified version for Shortened RS codes. The classical and the proposed decoder are discussed in Section 3.3 which also discusses about the dual property of evaluation based decoding scheme which explains how message can directly be obtained with the proposed decoder. In section 3.5, the evaluation based decoding scheme is proposed for shortened RS codes. In section 3.6, hardware implementation issues are discussed and also the results for implementations are presented along with the some software implementation results for Shortened RS codes. Finally section 3.7 concludes the chapter.

3.2 Encoder block

3.2.1 Evaluation based RS encoder

In its original construction, RS codes can be viewed as evaluation codes over a Finite Field $\mathbb{F}_q$ [11]. Consider $(1, \alpha_1, ..., \alpha_{n-1})$ to be the elements of the field $\mathbb{F}_q$. A packet of k information symbols $(m_0, m_1, ..., m_{k-1})$ forms a message polynomial $m(x) = m_0 + m_1 x + ... + m_{k-1}x^{k-1}$ over the field $\mathbb{F}_q$ as discussed already. For encoding, $m(x)$ is evaluated at n distinct elements of the field in order to have an RS code with parameters $[n, k, d]$, where $d = n - k + 1$ and $t = \lfloor \frac{d-1}{2} \rfloor$. The codeword can be defined then as

$$c \;=\; (c_0, ..., c_{n-1}) = (m(1), m(\alpha_1), ..., m(\alpha_{n-1}))$$

where, $m(x) \in F[x]$, degree of $m(x) < k$ and $F[x]$ is a polynomial ring over $\mathbb{F}_q$ and $1 \leq k \leq n \leq q$. The encoding from evaluation can also be shown in matrix form:

$$\begin{pmatrix} c_0 & c_1 & \dots & c_{n-1} \end{pmatrix} = \begin{pmatrix} m_0 & m_1 & \dots & m_{k-1} \end{pmatrix} * \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \alpha & \alpha^2 & \cdots & \alpha^{n-1} \\ 1 & \alpha^2 & \alpha^4 & \cdots & \alpha^{2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha^{k-1} & \alpha^{2(k-1)} & \dots & \alpha^{(k-1)(n-1)} \end{pmatrix}$$

Here α is the primitive element of the field and α_i is shown as α^{i-1} for $i = 1, 2, \dots, q-1$. This evaluation based encoder is non-systematic and is shown in Figure 3.1. The encoder is just a simple evaluation of the message polynomial whose coefficients are $(m_0, m_1, ..., m_{k-1})$ which is similar to the syndrome computation block in the classical decoder [21] but for the syndromes, evaluation is done for $2t$ symbols while here it is done for n symbols. This results in higher area as it needs n registers and multipliers for the proposed encoder compared to $2t$ registers and multipliers for the syndrome computation in classical decoder. The number of clock cycles are n for both of them.

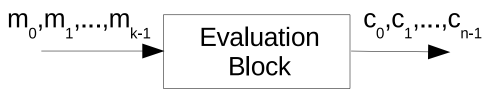

Figure 3.1: RS encoder block

3.2.2 Proposed encoding scheme for shortened RS codes

The message polynomial for shortened code can be written as $m_s(x) = m_0 + m_1x + \dots + m_{k_s-1}x^{k_s-1}$. For the encoding based on evaluation, the message polynomial for shortened code needs to be evaluated at all the points of the field for computing the codeword. If we evaluate this polynomial at all the points of field the resulting codeword will not have $n-n_s$ zeros in the end. Thus, for attaining the required length of shortened code that is n_s, we need to have $n - n_s$ zeros in the codeword. The obtained codeword is the shortened code where only n_s symbols will be transmitted instead of n symbols making it suitable for certain applications.

This section proposes the novel concept of obtaining shortened code from evaluation based encoding scheme. As already discussed, only evaluation of shortened message will not give us shortened code. Now, the question is what can be done to obtain $n - n_s$

zeros in the codeword from an evaluation scheme? One method of achieving this is to multiply the message polynomial with some other polynomial which has $n - n_s$ zeros at the field elements. A potential candidate for this operation would be:

$$P(x) = (x - \alpha^{n_s}) \dots (x - \alpha^{n-1}) \tag{3.1}$$

It has zeros at the last $n - n_s$ elements of the field i.e. $\alpha^{n_s}, \alpha^{n_s+1}, \dots, \alpha^{n-1}$. Evaluating this polynomial at all the field points will automatically give $n - n_s$ zeros in the last $n - n_s$ field elements. Thus, the new message polynomial can then be obtained as:

$$m'(x) = m_s(x)P(x) = (m_0 + m_1 x + \dots + m_{k_s-1} x^{k_s-1})(p_0 + p_1 x + \dots + p_{n-n_s} x^{n-n_s}) \tag{3.2}$$

By computing the degree of the polynomial $m'(x)$, we can easily see that it's degree is same as the degree of full length RS code that is $k_s + n - n_s = k$, which is same as the dimension of the full length RS code. Evaluating $m'(x)$ at all the elements of the field forces the last $n - n_s$ symbols to be zero which is the shortened code.

We can see as $m'(x) = m_s(x)P(x)$, it follows that $c_s = ev(m'(x)) = ev(m_s(x)P(x)) = ev(m_s(x))ev(P(x))$, where ev is the evaluation over the elements of the field. Now, evaluating $m'(x)$ will be a complex job (comparable to the full length code) as the degree of the polynomial is k. This can be simplified as we already know the $P(x)$ that is fixed polynomial for specific code parameters. Thus, $ev(P(x))$ are just constant field elements for those code parameters. Thus, the encoder can be seen only as a syndrome computer with constant multipliers as shown in the Figure 3.2.

Algorithm 1 Evaluation based encoding for shortened RS codes

```
Input: $m_s(x) = m_0 + m_1 x + \dots + m_{k_s-1} x^{k_s-1}$, $P(0), P(1), \dots, P(\alpha^{n_s-1})$
1: for $i = 1 : n_s$ do
2:     $m'(i) = m_s(\alpha^{i-1}) * P(\alpha^{i-1})$
3: end for
4: return $c_s = (c_0, c_1, \dots, c_{n_s-1})$
```

3.3 Classical decoding scheme

In this section, we give a detailed introduction to classical decoding scheme before proceeding to the direct decoding scheme. This section will help to differentiate between

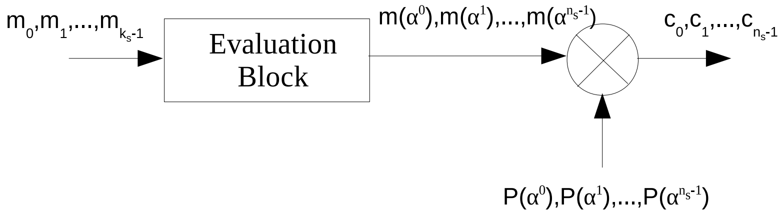
$m_0, m_1, \ldots, m_{k_s-1}$
Evaluation Block
$m(\alpha^0), m(\alpha^1), \ldots, m(\alpha^{n_s-1})$
$c_0, c_1, \ldots, c_{n_s-1}$
$P(\alpha^0), P(\alpha^1), \ldots, P(\alpha^{n_s-1})$

Figure 3.2: Shortened RS encoder block

the two decoding schemes. The codeword $c(x)$ is transmitted over a noisy channel and may get corrupted by an error vector $e(x)$. The result is a received word $r(x) = c(x) + e(x)$ at the receiver. The error vector can be viewed as:

$$e(x) = \sum_{j=1}^{e} Y_j X_j \tag{3.3}$$

where Y_j are the error values that have occurred at error locations X_j in $\mathbb{F}_q$. Since the error locations are in $\mathbb{F}_q$, $X_j = \alpha^{i_j}$. If the number of errors e is less than or equal to the error correction capability t, then the decoder is able to correct all the errors and hence the receiver will have the correct information that has been transmitted. The standard algebraic decoding method calculates the $2t$ syndromes corresponding to the $2t$ roots of $g(x)$ and uses these syndromes to calculate the error locations X_j and the error values Y_j.

3.3.1 Block level architecture for classical decoder

A block diagram of the classical decoding architecture for the RS codes is shown in Figure 3.3. The figure shows that there are four stages (syndrome computation, BM algorithm, Chien search and Forney formula) in the classical decoder while there are only two stages (syndrome computation and BM algorithm) in the proposed decoder. We will discuss each of the four stages in the next part as discussed in [22] [23] [25].

3.3.2 Syndrome computation

The first step in decoding RS codes is the syndrome computation. Assuming that the code is built using the generator polynomial approach, the received polynomial $r(x)$ is evaluated at the roots of the generator polynomial $g(x)$, which are $\alpha, \alpha^2, \alpha^3, \ldots, \alpha^{2t}$. Since the codeword polynomial $c(x)$ is a multiple of $g(x)$, it follows that $c(\alpha^i) = 0$ for i

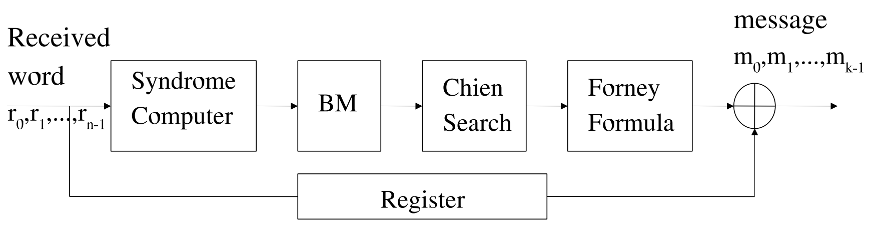

Figure 3.3: RS classical decoder

$= 0, 1, \ldots, 2t-1$. The syndromes are defined as,

$$S_i = r(\alpha^i) = e(\alpha^i) = \sum_{j=1}^{e} Y_j (X_j \alpha^i) = \sum_{j=1}^{e} Y_j X_j^i \tag{3.4}$$

The $2t$ syndromes form the syndrome polynomial $S(x) = S_0 + S_1 x + \ldots + S_{2t-1} x^{2t-1}$. This polynomial is dependent only on the error vector $e(x)$. If all the syndromes are zero, then $c(x) = r(x)$ which means that no errors have occurred.

3.3.3 Error locations

Let us define an error locator polynomial $\Lambda(x)$ of degree e and an error evaluator polynomial $\Omega(x)$ of degree $e-1$ as follows [50],

$$\Lambda(x) = \prod_{j=1}^{e} (1 - X_j x) = 1 + \lambda_1 x + \lambda_2 x^2 + \ldots + \lambda_e x^e \tag{3.5}$$

$$\Omega(x) = \sum_{i=1}^{e} Y_i X_i^{m0} \prod_{j=1, j\neq i}^{e} (1 - X_j x) = \omega_0 + \omega_1 + \ldots + \omega_{e-1} x^{e-1}. \tag{3.6}$$

From equations 3.3, 3.5 and 3.6 that the syndromes, error locator polynomial and the error evaluator polynomial are related in the following manner as stated in the key equation [50],

$$\Lambda(x) S(x) = \Omega(x) \ mod \ x^{2t}. \tag{3.7}$$

Solving the key equation then allows us to find X_j and Y_j. Several algorithms such as extended Euclidean algorithm [52], Berlekamp-Massey algorithm [53] etc can be used to solve this problem. The algorithms produce a solution as long as $e \leq t$, or else a decoding error occurs. In this work, the Berlekamp-Massey algorithm is presented.

3.3.4 Berlekamp-Massey algorithm

The Berlekamp-Massey algorithm is an iterative process for generating $\Lambda(x)$ and $\Omega(x)$ polynomials from $S(x)$ polynomial using the relation as mentioned in Equation 3.7. The algorithm uses the key equation in an iterative manner such that,

$$\Lambda(x,r)S(x) = \Omega(x,r) \ mod \ x^r \qquad r = 1, 2, \ldots, 2t.$$

The $\Lambda(x)$ and $\Omega(x)$ polynomials are initialized in the beginning to $\Lambda(0,x) = 0$ and $\Omega(0,x) = 0$, and after $2t$ iterations, it gives the polynomials $\Lambda(2t,x)$ and $\Omega(2t,x)$. The algorithm uses two update polynomials, $B(x)$ for $\Lambda(x)$ and $H(x)$ for $\Omega(x)$, which are initialized as $B(0,x) = 1$ and $H(0,x) = -1$.

Berlekamp-Massey algorithm is an alternative procedure to solve the set of algebraic linear equations described in Reed-Solomon Peterson decoder [54], which can be seen as:

$$\sum_{j=0}^{t} \Lambda_j S^r_{i-j} = 0 \qquad i = t+1, \ldots, 2t. \tag{3.8}$$

The goal of the algorithm is to find the error locator polynomial as mentioned in Equation 3.5. The idea is to determine the minimal degree $\Lambda(x)$ which satisfies Equation 3.8. The algorithm starts with $L(x) = 1$ and number of errors initialized to 0. $2t$ is the total number of syndromes and and r is the iterator running from 0 to $2t - 1$. $B(x)$ is the copy of the last $\Lambda(x)$ since its degree was updated. Each iteration computes a discrepancy. Each iteration computes a discrepancy,

$$\delta(r) = S_r\lambda_0(r) + S_{r-1}\lambda_1(r) + \ldots + S_{r-t}\lambda_t(r)$$

for the r^{th} iteration. It $\delta(r) = 0$, the algorithm assumes that current $\Lambda(x)$ is correct for the moment and increases r. If $\delta(r) \neq 0$, the algorithm adjusts $\Lambda(x)$ to ensure that $\delta(r) = 0$. After $2t$ iterations, the algorithm outputs the coefficients of $\Lambda(x)$ and $\Omega(x)$. The pseudo code for the algorithm is shown in Algorithm 2 following the notations from [50].

3.3.5 Chien Search and Forney's formula

The most common method used in finding the roots of the error locator polynomial $\Lambda(x)$ is a brute force approach known as Chien search [55]. It consists of testing all

Algorithm 2 The Berlekamp-Massey Algorithm

```
Input: : $S_i$, $i = 0, 1, \ldots, 2t-1$
 1: Initialization: $\lambda_0(0) = b_0(0) = 1$, $\lambda_i(0) = b_i(0) = 0$, for $i = 1, 2, \ldots, 2t$, $k(0) = 0$
 2: for $r = 0; r < 2t; r++$ do
 3:     $\delta(r) = S_r\lambda_0(r) + S_{r-1}\lambda_1(r) + \ldots + S_{r-t}\lambda_t(r)$
 4:     $\lambda_i(r+1) = \lambda_i(r) - \delta(r)b_{i-1}(r)$, $(i = 0, 1, \ldots, t)$
 5:     if $\delta(r) \neq 0 \& k(r) \geq 0$ then
 6:         $b_i(r+1) = \lambda_i(r)\delta^{-1}(r)$
 7:         $k(r+1) = -k(r) - 1$
 8:     else
 9:         $b_i(r+1) = b_{i-1}(r)$, $(i = 0, 1, \ldots, t)$
10:         $k(r+1) = k(r) + 1$
11:     end if
12: end for
13: for $i = 0; i < t; i++$ do
14:     $\omega_i(2t) = S_i\lambda_0(2t) + S_{i-1}\lambda_1(2t) + \ldots + S_0\lambda_i(2t)$
15: end for
Output: $\lambda_i(2t)$, $i = 0, 1, \ldots, t$. $\omega_i(2t)$, $i = 0, 1, \ldots, t-1$
```

the values of the field. The search is a cyclic procedure that evaluates the polynomial through all the elements in the finite field until e number of roots are found, where e is the degree of $\Lambda(x)$. Once the roots of $\Lambda(x)$ have been found, the decoder proceeds to calculate the error values, $Y_j, j = 1, 2, \ldots, e$. This is achieved through a Lagrange interpolation method known as the Forney's algorithm [56]. The last step consist of correcting the received message, knowing the error location with the computed error values.

3.4 Proposed decoder based on evaluation

In order to make a fair comparison, we discussed about the classical decoding algorithm for RS codes which involved the Chien Search and Forney's formula for computing the error locations and error values respectively in the last section. We will now present the proposed decoding algorithm in detail. In this section, we will discuss the different blocks of the proposed decoder. The proposed decoder is discussed from frequency domain perspective in [50], however we will discuss it from evaluation perspective. Also, we will discuss how the algorithm can be made more efficient in hardware implementation based on the proposed decoding algorithm. The proposed scheme does not require the extra steps like Chien Search and Forney's algorithm for computing the error locations and error values respectively. The algorithm directly calculates the message that has been transmitted from the source.

3.4.1 Block level architecture for the proposed decoder

The blocks and circuitry required for the proposed decoder are shown in Figure 3.4. The multiplexer is used to decide whether it is required to use both the discrepancy computation step and polynomial update step (for initial $2t$ iterations) or just the discrepancy computation step (for the last $n - 2t$ iterations).

3.4.2 Direct decoding algorithm

The syndrome calculation for the direct decoding algorithm is equivalent to evaluating the received message polynomial $r(x)$ in all the field elements. The syndromes of the

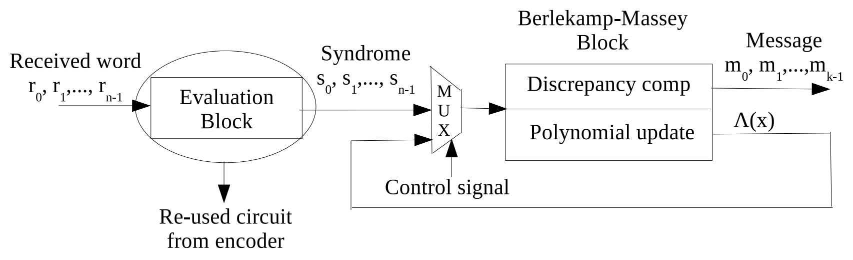

Figure 3.4: RS proposed decoder

received message r are given by:

$$r(\alpha^j) = S_j^r = \sum_{i=1}^{n} \alpha^{ij} r_i \qquad j = 1, 2, ..., 2t \tag{3.9}$$

Let the syndromes of the codeword be denoted as S_j^c and the syndromes for the errors be denoted as S_j^e. As $r(x) = c(x) + e(x)$, it follows that $r(\alpha^j) = c(\alpha^j) + e(\alpha^j)$ which means that $S_j^r = S_j^c + S_j^e$. From the construction of RS codes, $c(\alpha^j) = 0$ for $j = 1, 2, ..., 2t$ (proved in section 3.4.3), or in frequency domain terms it can be said that the parity frequencies have spectral components equal to zero [50]. Therefore, $S_j^r = S_j^e$ for $j = 1, 2, \ldots 2t$.

The error locator polynomial $\Lambda(x)$ is defined as the polynomial with the inverse of error location as its roots as discussed in previous section. In other words, $\Lambda(\alpha^{-j}) = 0$ when $e_j \neq 0$. Let

$$S^e(x) = S_0^e + S_1^e x + \ldots + S_{n-1}^e x^{n-1}$$

is a polynomial of degree $n-1$ with coefficients as S_j^e. For any α^j, which is a root of $\Lambda(x)$, we can equate the coefficients to 0 in the equation $\Lambda(x)S^e(x) = 0$. Therefore,

$$\sum_{j=0}^{t} \Lambda_j S_{i-j}^e = 0 \qquad i = 0, \ldots, n-1, \tag{3.10}$$

where, $S_{i-j}^e = 0$ if $i-j < 0$. This linear recursion can also be proved from error spectrum as explained in [50]. The sum is from 0 to t because the degree of $\Lambda(x)$ is at most t. This is an algebraic system of n equations. The unknowns are t, the coefficients of $\Lambda(x)$ and $n-2t$ components of S^e. As mentioned above, $2t$ components of S^e are equal to S^r and hence t equations as shown in Equation 3.11 involve only the components of $\Lambda(x)$.

$$\sum_{j=0}^{t} \Lambda_j S_{i-j}^r = 0 \qquad i = t+1, \ldots, 2t \tag{3.11}$$

Equation 3.11 is solvable and will give the correct error locator polynomial using BM algorithm if less than or equal to t errors have occurred.

With the use of Equation 3.10, Equation 3.12 can be easily derived to compute the remaining S^e in an iterative fashion. This method is also mentioned is [50]. After running the BM algorithm for $2t$ syndromes, we already know the values of Λ_j for $j = 1, 2, \ldots, 2t$ and all the previous error syndromes are known. Thus, by using the Equation 3.12, we can easily and efficiently compute the rest of S_k^e from $k = 2t + 1, \ldots, n - 1$. This part is same as the discrepancy computation part of the BM algorithm as discussed before.

$$S_k^e = -\sum_{j=0}^{t} \Lambda_j S_{k-j}^e \qquad k = 2t + 1, \ldots, n - 1 \tag{3.12}$$

In other words, the first $2t$ error syndromes ($S^e = S^r$) are used to compute $\Lambda(x)$ (from BM algorithm), then $\Lambda(x)$ and the known S^e are used to compute the remaining error syndromes. Once all error syndromes S^e are known, the codeword syndromes can be computed by subtraction (symbolwise-XOR). With the presented decoding process, the message can be found in the last k positions of the codeword syndromes, which will be proved in the next section. Syndromes for the codeword S^c are again the evaluation of the codeword polynomial $c(x)$ in $\mathbb{F}_q$ which gives the coefficients of message polynomial i.e. $m(x)$ in reverse order. Hence S^c results in the recovered original message. The detailed explanation for these based on evaluation can be found in Section 3.4.3. The direct decoding algorithm is mentioned in Algorithm 3. Also, we can see the algorithm starts outputting the message after $n+2t$ cycles as m_{k-1} will be computed at $n+2t+1^{st}$ clock cycle. This type of strategy is most effective if the next block accepts serial data, and is not dependent on other data elements in the packet.

3.4.3 Duality of evaluation

In this section, the dual nature of evaluation will be explained for clear understanding of algorithm 3. The syndromes for the codeword S^c are again the evaluation of the codeword polynomial c in $\mathbb{F}_q$ which will actually give the coefficients of message polynomial i.e. $m(x)$ in reverse order. The message can be found in the last k components of S^c. This operation can be explained by examining the matrix multiplications.

For encoding or obtaining the codeword, we have $c = m * A$, where

Algorithm 3 Evaluation based decoding algorithm for RS codes

```
Input: $r(x) = r_0 + r_1 x + \ldots + r_{n-1} x^{n-1}$
1:  for $j = 1 : n$ do
2:      $S_j^r = \sum_{i=1}^{n-1} \alpha^{ij} r_i$
3:  end for
4:  for $i = 1 : 2t$ do
5:      BM algorithm to obtain $\Lambda(x)$.
6:      $S_i^e = S_i^r$
7:  end for
8:  $\delta_1 = 0$
9:  for $i = 2t + 1 : n$ do
10:     $S_i^e = -\sum_{j=0}^{t} \Lambda_j S_{i-j}^e$
11:     $m_{n-i} = S_i^c = S_i^r + S_i^e$
12: end for
13: return $(m_0, m_1, \ldots, m_{k-1})$
```

$$A = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \alpha & \alpha^2 & \cdots & \alpha^{n-1} \\ 1 & \alpha^2 & \alpha^4 & \cdots & \alpha^{2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha^{k-1} & \alpha^{2(k-1)} & \ldots & \alpha^{(k-1)(n-1)} \end{bmatrix}$$

After encoding the codeword is computed. It is transmitted and the errors get added such that the received word is $r = c + e$. For the computation of syndromes, $S_r = r * B$, where

$$B = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \alpha & \alpha^2 & \cdots & \alpha^{n-1} \\ 1 & \alpha^2 & \alpha^4 & \cdots & \alpha^{2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha^{n-1} & \alpha^{2(n-1)} & \ldots & \alpha^{(n-1)(n-1)} \end{bmatrix}$$

Assuming that we have no errors while transmitting the codeword, $r = c$ and thus

$S_r = S_c = c * B = m * A * B.$

$$A * B = \begin{bmatrix} 1 & 0 & \cdots & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & \cdots & 0 & \cdots & 0 & 1 & 0 \\ 0 & 0 & \cdots & 0 & \cdots & 1 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & \cdots & 0 & 0 & 0 \end{bmatrix}$$

Therefore the coefficients of the message polynomial $m(x) = [m_0\ 0\ 0....0\ m_{k-1}...m_1]$ can be directly obtained. In the next section, we will give an example to explain the direct decoding algorithm for RS codes.

Example 3.4.1. *Consider C be a RS code with parameters $[31, 23, 9]$ defined over $\mathbb{F}_{2^5}$. The error correction capability is 4. Suppose, we choose a message word as:*
$m = [\alpha^{22}\ \alpha^{30}\ \alpha^{29}\ \alpha^{9}\ \alpha^{18}\ \alpha^{29}\ \alpha^{18}\ \alpha^{20}\ \alpha^{6}\ \alpha^{14}\ \alpha^{25}\ \alpha^{3}\ 1\ 0\ \alpha^{10}\ \alpha^{26}\ \alpha^{25}\ \alpha^{17}\ \alpha^{11}\alpha^{2}\ \alpha^{7}\ \alpha^{14}\ \alpha^{25}]$.
The codeword is computed by evaluating the polynomial $m(x)$ with $m(i)$ as coefficients at all the points of the field $\mathbb{F}_{2^5}$:
$c = [\alpha^{29}\ \alpha^{5}\ 0\ \alpha^{11}\ \alpha^{16}\ \alpha^{14}\ \alpha^{25}\ \alpha^{4}\ \alpha^{12}\ \alpha^{13}\ \alpha\ \alpha^{23}\ \alpha^{23}\ \alpha^{11}\ \alpha^{9}\ \alpha^{8}\ \alpha^{13}\ \alpha^{19}\ \alpha^{27}\ \alpha^{3}\ \alpha^{8}\ \alpha^{26}$
$\alpha^{28}\ \alpha^{24}\ \alpha^{16}\ \alpha^{4}\ \alpha^{24}\ \alpha^{22}\ \alpha\ \alpha^{23}\ \alpha^{26}]$.
Suppose the errors introduced is:
$e = [0\ 0\ 0\ 0\ 0\ \alpha^{19}\ 0\ 0\ \alpha^{19}\ \alpha^{7}\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]$.
Received word is computed as $r = c + e$:
$r = [\alpha^{29}\ \alpha^{5}\ 0\ \alpha^{11}\ \alpha^{16}\ \alpha^{3}\ \alpha^{3}\ \alpha^{4}\ \alpha^{12}\ \alpha^{13}\ \alpha\ \alpha^{23}\ \alpha^{23}\ \alpha^{11}\ \alpha^{9}\ \alpha^{8}\ \alpha^{14}\ \alpha^{19}\ \alpha^{27}\ \alpha^{3}\ \alpha^{8}\ \alpha^{26}$
$\alpha^{28}\ \alpha^{24}\ \alpha^{16}\ \alpha^{4}\ \alpha^{24}\ \alpha^{22}\ \alpha\ \alpha^{23}\ \alpha^{26}]$.
Syndromes for received word S^r is computed by evaluating polynomial with coefficients in received word $r(x)$ at all the points of the field:
$S^r = [0\ \alpha^{22}\ \alpha\ \alpha^{14}\ \alpha^{27}\ \alpha^{10}\ \alpha^{13}\ \alpha^{21}\ \alpha^{8}\ \alpha^{23}\ \alpha^{28}\ \alpha^{18}\ \alpha^{14}\ \alpha^{7}\ \alpha\ \alpha^{3}\ \alpha^{7}\ \alpha^{20}\ \alpha^{10}\ \alpha^{7}\ \alpha^{30}\ \alpha^{13}\ \alpha^{18}\ \alpha^{22}$
$\alpha^{12}\ \alpha\ \alpha^{18}\ \alpha^{15}\ \alpha^{11}\ \alpha^{5}\ \alpha^{24}]$ ..
By first $2t$ (in our case from 2 to $2t+1$) syndromes, error locator polynomial is computed as:
$\Lambda = [1\ \alpha^{29}\ \alpha^{8}\ \alpha^{6}\ \alpha^{7}]$.
By using the discrepancy computation step iteratively, S^e is computed as:
$S^e = [\alpha^{22}\ \alpha^{22}\ \alpha\ \alpha^{14}\ \alpha^{27}\ \alpha^{10}\ \alpha^{13}\ \alpha^{21}\ \alpha^{8}\ \alpha^{28}\ \alpha^{27}\ \alpha^{26}\ \alpha^{25}\ \alpha^{6}\ \alpha^{10}\ \alpha^{10}\ \alpha^{18}\ \alpha^{14}\ \alpha^{10}\ \alpha^{22}\ \alpha^{9}\ \alpha^{5}\ \alpha^{24}$
$\alpha^{15}\ \alpha\ 1\ \alpha^{6}\ \alpha^{13}\ \alpha^{14}\ \alpha^{20}\ \alpha^{20}]$.

Therefore, codeword syndromes are computed as $S^c = S^r + S^e$*:*

$S^c = [\alpha^{22}\ \alpha^5\ 0\ \alpha^{11}\ \alpha^{16}\ \alpha^{16}\ \alpha^{25}\ \alpha^4\ 0\ \alpha^{25}\ \alpha^{14}\ \alpha^7\ \alpha^2\ \alpha^{11}\ \alpha^{17}\ \alpha^{25}\ \alpha^{26}\ \alpha^{10}\ 0\ 1\ \alpha^3\ \alpha^{25}\ \alpha^{14}\ \alpha^6\ \alpha^{20}$ $\alpha^{18}\ \alpha^{29}\ \alpha^{18}\ \alpha^9\ \alpha^{29}\ \alpha^{30}]$.

The decoded message word can be computed from last k *symbols of* S^c *in reverse order:*

$m_{dec} = [\alpha^{22}\ \alpha^{30}\ \alpha^{29}\ \alpha^9\ \alpha^{18}\ \alpha^{29}\ \alpha^{18}\ \alpha^{20}\ \alpha^6\ \alpha^{14}\ \alpha^{25}\ \alpha^3\ 1\ 0\ \alpha^{10}\ \alpha^{26}\ \alpha^{25}\ \alpha^{17}\ \alpha^{11}\alpha^2\ \alpha^7\ \alpha^{14}\ \alpha^{25}]$

, which is same as the transmitted message.

3.5 Evaluation based decoding for shortened RS codes

The decoding of shortened RS codes is performed using the algorithm as defined for the RS codes in section 3 with some modifications which are discussed in this section. It starts by computing the n_s syndromes for the received word using Equation 3.13.

$$S_j^r = \sum_{i=1}^{n_s-1} \alpha^{ij} r_i \qquad j = 1, 2, ..., n_s \tag{3.13}$$

As discussed in section 3.4, using Equation 3.12 allows the $n_s - 2t$ syndromes for the error to be computed. During the computation of the error syndromes, the polynomial updating step is not needed. Based on our proposed encoding algorithm for shortened codes, the direct decoding algorithm will yield the coefficients of $m'(x)$ instead of $m_s(x)$. This result has also been confirmed computationally by MATLAB results. Since the degree of $m'(x)$ is more than $m(x)$, we need to compute more coefficients. This will result in more clock cycles. Moreover, instead of getting the original message polynomial $m_s(x)$, the modified version of the message polynomial is obtained i.e. $m'(x)$. This decoding method will also require a polynomial division at the end in order to extract the original message polynomial i.e. $m_s(x) = \frac{m'(x)}{P(x)}$ which is quite costly in terms of hardware or software. The most efficient method is to avoid the polynomial division and get the coefficients of m_s in the algorithm itself. We modified the direct decoding algorithm for shortened codes in order to obtain directly the coefficients for $m_s(x)$. As already mentioned, $m'(x) = m_s(x)P(x)$, therefore by comparing the coefficients of both sides with respect to x-degree, we can obtain the equations discussed below:

$$m'_{k-i} = m_{k_s-i} + \delta(i) \qquad i = 1, 2, \ldots, k_s \tag{3.14}$$

where, $\delta(i) = \sum_{j=1}^{i-1} m_{k_s-j} P_{n-n_s-i+j}$, for $i = 2, 3, ..., k_s$ and for $i = 1$, $\delta(1) = 0$.

We can use these iterative equations in order to obtain the coefficients of $m_s(x)$ without any polynomial division. This can be performed during the BM algorithm calculating the complete $m_s(x)$ by using an in-line division scheme as defined in Equation 3.15, 3.16 and 3.17. Thus, it will not require any extra polynomial division or extra clocks for obtaining original shortened message.

$$\delta(1) = 0 \tag{3.15}$$

$$\delta(i) = \sum_{j=1}^{i-1} m_{k_s-j} P_{n-n_s-i+j}, i = 2, 3, ..., k_s \tag{3.16}$$

$$m_{k_s-i} = m'_{k-i} - \delta(i) \qquad i = 1, 2, \ldots, k_s \tag{3.17}$$

3.5.1 Algorithm

By using the above equations, all the coefficients for $m_s(x)$ can be calculated after k_s clock cycles using an Linear Feedback Shift Register (LFSR) circuit [57]. The decoding algorithm is summarized in Algorithm 4.

Algorithm 4 Evaluation based decoding for shortened RS codes

```
Input: $r(x) = r_0 + r_1 x + \ldots + r_{n_s-1} x^{n_s-1}$, $P(x) = (p_0 + p_1 x + \ldots + p_{n-n_s} x^{n-n_s})$
1: for $j = 1 \ : \ n_s$ do
2:     $S_j^r = \sum_{i=1}^{n_s-1} \alpha^{ij} r_i$
3: end for
4: for $i = 1 \ : \ 2t$ do
5:     Run BM algorithm to obtain the error locator polynomial $\Lambda(x)$.
6: end for
7: $\delta_1 = 0$
8: for $i = 2t + 1 \ : \ n_s$ do
9:     $S_i^e = -\sum_{j=0}^{t} \Lambda_j S_{i-j}^e$
10:    $m'_{n-i} = S_i^c = S_i^r + S_i^e$
11:    $m_{n_s-i} = m'_{n-i} - \delta_{i-2t}$
12:    $\delta_{i-2t} = \sum_{j=1}^{i-2t} m_{n_s-2t-j} P_{n-n_s-i+2t+j}$
13: end for
14: return $(m_{s_0}, m_{s_1}, \ldots, m_{s_{k_s-1}})$
```

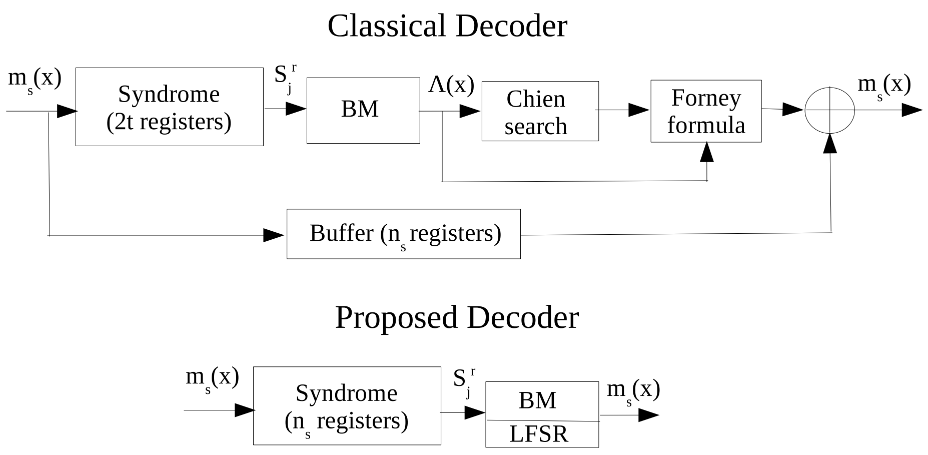

Figure 3.5: Shortened RS decoder block

3.5.2 Block level architecture

Block diagrams of the classical and the proposed decoding architectures for the shortened RS codes are shown in Figure 3.5. The difference in the decoder for shortened codes from full length RS codes is that it needs one more LFSR for computing the message polynomial $m_s(x)$ from $m'(x)$ as discussed in section 3.5.

3.5.3 Example for encoding and decoding shortened RS code

Consider a shortened RS code with parameters $[12, 4, 9]$ shortened from $[15, 7, 9]$ RS code. The code has error correction capability, $t = 4$. Suppose the message polynomial is $m_3x^3 + m_1x^2 + m_1x + m_0$, $P(x) = (x + b_0)(x + b_1)(x + b_2)$, where b_0, b_1, b_2 are just $\alpha^{12}, \alpha^{13}, \alpha^{14}$. $P(x)$ is a constant polynomial for fixed code parameters and $m'(x) = m(x)P(x) = m_3x^6 + ... + b_0b_ob_1b_2$. The codeword obtained by evaluating $m'(x)$ at all field points $\mathbb{F}_q - \{0\}$ is $c = (c_0, c_1, ..., c_{n_s}, 0, 0, 0)$. The last 3 symbols are forced to zero because of $P(x)$ and only 12 symbols are transmitted through the channel. Suppose the error introduced is $e = (e_0, e_1, 0, 0, e_4, 0, 0, 0, 0, e_9, 0, 0)$. The $n_s = 12$ syndromes are calculated and with $2t = 8$ syndromes the error locator polynomial $\Lambda(x)$ is calculated and the rest $n_s - 2t$ syndromes are used to calculate the message polynomial. During the message polynomial calculation the obtained result is the answer for $m'(x)$ instead of $m(x)$, so the algorithm is modified slightly in order to get $m_s(x)$ directly in n_s clock cycles. In this example $k_s = 4$; which means k_s coefficients for the message polynomial need to be computed. Without using an in-line division method, coefficients for $m'(x)$

are obtained which is $m(x) * P(x)$. By applying the equations Equation 3.15, 3.16 and 3.17: $m_3 = m'_6$;

$m_2 = m'_5 - m_3(b_0 + b_1 + b_2)$;

$m_1 = m'_4 - m_2(b_0 + b_1 + b_2) - m_3(b_0 * b_1 + .. + b_2 * b_0)$

$m_0 = m'_3 - m_1 * (b_0 + b_1 + b_2) - m_2(b_0 * b_1 + .. + b_2 * b_0) + a_0 b_0 b_1 b_2$

Thus, the equations are used to modify the algorithm to directly obtain the original message polynomial $m_s(x)$ by using n_s syndromes and n_s clock cycles.

Example 3.5.1. *Consider C be a shortened-RS code with parameters $[28, 20, 9]$ defined over $\mathbb{F}_{2^5}$. The error correction capability is 4. Suppose, we choose a shortened message word as:*

$m_s = [1\ \alpha^{28}\ \alpha^{14}\ \alpha^{2}\ \alpha^{20}\ \alpha\ \alpha^{8}\ \alpha^{9}\ \alpha^{20}\ \alpha^{11}\ \alpha^{6}\ \alpha^{26}\ \alpha^{7}\ \alpha^{14}\ \alpha^{25}\ \alpha^{12}\ \alpha^{17}\ 1\ \alpha^{4}\ \alpha^{12}]$.

If we evaluate $m_s(x)$ polynomial, we will not get the last 3 symbols as zero. Thus, we define $P(x) = (x - \alpha^{28})(x - \alpha^{29})(x - \alpha^{30})$ and $m'(x) = m_s(x)P(x)$. The shortened codeword is computed by evaluating the polynomial $m(x)$ with $m_s(i)$ as coefficients at $n_s = 28$ points of the field $\mathbb{F}_{2^5}$. Here, we have done the computation at all the points of the field in order to show that the last 3 symbols are zero and we do not need to transmit them:

$c = [\alpha\ \alpha^{18}\ \alpha^{29}\ \alpha^{26}\ \alpha^{18}\ \alpha^{30}\ \alpha^{17}\ \alpha^{14}\ \alpha^{27}\ \alpha^{5}\ \alpha^{29}\ \alpha^{4}\ \alpha^{21}\ \alpha^{5}\ \alpha^{20}\ \alpha^{12}\ \alpha^{20}\ 0\ \alpha^{2}\ \alpha^{12}\ \alpha\ 0\ \alpha^{21}\ \alpha^{30}\ \alpha^{16}\ \alpha^{4}\ \alpha^{9}\ \alpha^{12}\ 0\ 0\ 0]$.

Suppose the errors introduced is:

$e = [0\ 0\ 0\ 0\ 0\ 0\ \alpha^{11}\ 0\ \alpha^{8}\ \alpha^{9}\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ \alpha^{5}\ 0\ 0]$.

Received word is computed as $r = c + e$:

$r = [\alpha\ \alpha^{18}\ \alpha^{29}\ \alpha^{26}\ \alpha^{18}\ \alpha^{30}\ \alpha^{7}\ \alpha^{14}\ \alpha^{19}\ \alpha^{15}\ \alpha^{29}\ \alpha^{4}\ \alpha^{21}\ \alpha^{5}\ \alpha^{20}\ \alpha^{12}\ \alpha^{20}\ 0\ \alpha^{2}\ \alpha^{12}\ \alpha\ 0\ \alpha^{21}\ \alpha^{30}\ \alpha^{16}\ \alpha^{22}\ \alpha^{9}\ \alpha^{12}]$.

Syndromes for received word S^r is computed by evaluating polynomial with coefficients in received word $r(x)$ at n_s points of the field:

$S^r = [\alpha^{20}\ \alpha^{25}\ \alpha^{2}\ \alpha^{8}\ \alpha^{25}\ \alpha^{11}\ 1\ \alpha^{25}\ \alpha^{11}\ \alpha^{23}\ \alpha^{23}\ \alpha^{19}\ \alpha^{17}\ \alpha^{10}\ \alpha^{28}\ 1\ \alpha^{7}\ \alpha^{14}\ \alpha^{10}\ \alpha^{16}\ \alpha^{30}\ \alpha^{13}\ \alpha^{8}\ \alpha^{23}\ \alpha^{29}\ \alpha^{15}\ \alpha^{26}\ \alpha^{14}]$ ..

By first $2t$ (in our case from 2 to $2t+1$) syndromes, error locator polynomial is computed as:

$\Lambda = [1\ \alpha^{2}\ \alpha^{13}\ \alpha^{15}\ \alpha^{17}]$.

By using the discrepancy computation step iteratively, S^e *is computed as:*
$S^e = [0\ \alpha^{25}\ \alpha^{2}\ \alpha^{8}\ \alpha^{25}\ \alpha^{11}\ 1\ \alpha^{25}\ \alpha^{11}\ 1\ \alpha^{17}\ \alpha^{6}\ \alpha^{28}\ \alpha^{5}\ \alpha^{22}\ \alpha^{13}\ \alpha^{24}\ \alpha^{15}\ \alpha\ 1\ \alpha^{9}\ \alpha\ 1\ \alpha^{7}\ \alpha^{6}\ \alpha^{14}\ \alpha^{16}\ \alpha^{12}]$.
Therefore, codeword syndromes are computed as $S^c = S^r + S^e$*:*
$S^c = [\alpha\ \alpha^{18}\ \alpha^{29}\ \alpha^{26}\ \alpha^{18}\ \alpha^{30}\ \alpha^{7}\ \alpha^{14}\ 0\ \alpha^{12}\ \alpha^{13}\ \alpha^{20}\ \alpha^{5}\ \alpha^{7}\ \alpha^{18}\ \alpha^{14}\ \alpha^{6}\ \alpha\ \alpha^{17}\ \alpha^{9}\ \alpha^{3}\ \alpha^{24}\ \alpha^{20}\ \alpha^{16}\ \alpha^{18}\ \alpha\ \alpha^{20}\ \alpha^{17}]$.
The decoded message word $m'(x)$ *can be computed from last* k_s *symbols of* S^c *in reverse order but by applying in-line division scheme developed by us, we can obtain* $m_{dec} = m_s$ *in the same step:*
$m_{dec} = [1\ \alpha^{28}\ \alpha^{14}\ \alpha^{2}\ \alpha^{20}\ \alpha\ \alpha^{8}\ \alpha^{9}\ \alpha^{20}\ \alpha^{11}\ \alpha^{6}\ \alpha^{26}\ \alpha^{7}\ \alpha^{14}\ \alpha^{25}\ \alpha^{12}\ \alpha^{17}\ 1\ \alpha^{4}\ \alpha^{12}]$
,which is same as transmitted message.

3.6 Hardware and software implementation results

The classical decoding scheme for RS uses n cycles for the syndromes computation, $2t$ for the BM algorithm, n for the Chien search and $2t$ for the Forney's formula which results $2n + 4t$ cycles in total. The new evaluation based decoding scheme needs n cycles for the syndromes calculation, $2t$ for the BM algorithm and k for the message polynomial calculation resulting in $2n$ cycles. The proposed decoding algorithm also allows the serial usage of data only after $n + 2t$ clock cycles compared to the classical decoder which starts outputting the message after $2n + 4t$ clock cycles. Thus, the proposed algorithm can be useful for communications. In terms of area, the new encoding and the syndrome computation share the same hardware and the decoder needs only a BM block instead of the classical decoder (which includes BM, Chien search and Forney formula). The circuitry is much more simple and small as compared to the classical decoding algorithm because we do not need any extra hardware for the Chien search and Forney's formula. This not only reduces the circuit area but also makes the circuit operate at much higher frequency as compared to the classical circuit.

The hardware implementations are done for commercially available $65nm$ Application Specific Integrated Circuit (ASIC) process. The implementation results for total number of cells and maximum clock rate of RS code over $\mathbb{F}_{256}$ with the error correction capability of $t = 2$, $t = 4$, $t = 6$ and $t = 8$ are shown in Figure 3.6 and Figure 3.7 [58]. The proposed decoder is compared to the classical decoder. Both the classical and the proposed decoder is implemented utilizing the same conventional architecture of Berlekamp-Massey algorithm from an efficiency point of view.

The total number of cells needed against the error correction capability (t) plot is shown in Figure 3.6 and it can be seen that for any t the total number of cells is smaller for the proposed decoder. Also, an important point to be noticed is that with the increasing error correction capability the total number of cells increases drastically (resulting in a very high slope) for the classical decoder. This means that with the increasing t, the required circuitry size is increasing very rapidly for the classical decoder and for $t = 8$, there is a total area reduction of 57.6%. As t increases, the classical decoder needs more circuitry for all the blocks including the syndrome, key equation, Chien Search and Forney's Formula while for the proposed decoder circuitry only increases due to the key equation block. That is why the area is increasing drastically in Figure 3.6 with increasing t, for the classical decoder as compared to the proposed decoder.

The maximum clock rate against the error correction capability (t) plot is shown in Figure 3.7 and it can be clearly seen that the proposed decoder can be operated at higher clock rates. For $t = 8$, there is a total decoding time reduction of 53.07% which is highly significant.

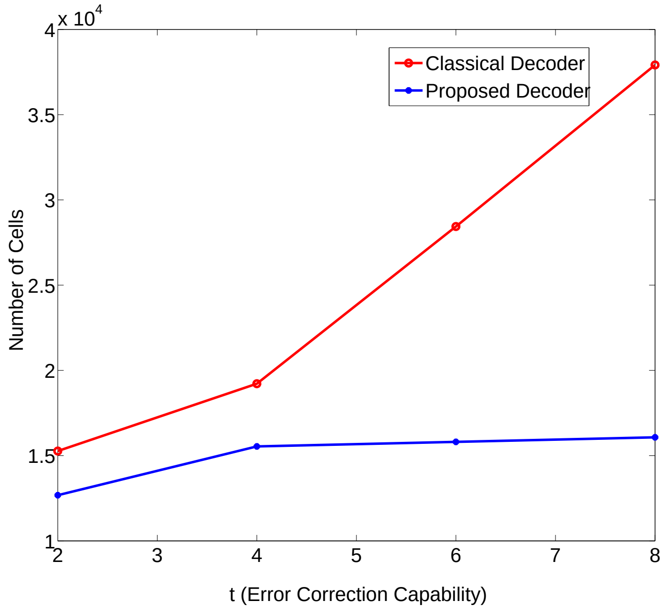

Figure 3.6: ASIC implementation results for area

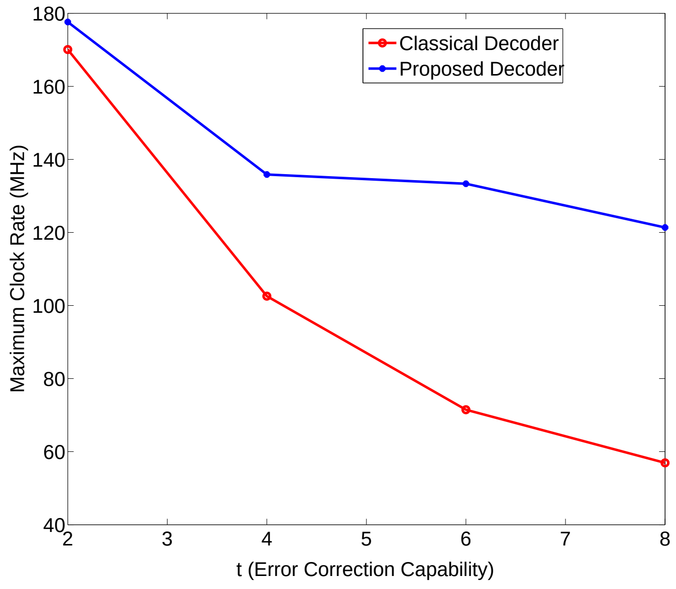

Figure 3.7: ASIC implementation results for maximum clock rate

Software implementation results for a resource constrained WSN application on a ATMEL Atmega 128 microcontroller are shown in Table 3.1 for RS $(32, 24)$ [59]. RS $(32, 24)$ is quite commonly used for Wireless Body Area network applications as it satisfies the code rate which should be around 0.8 and has packet length that is acceptable by transmitter [60]. The new decoder and encoder is 52% and 72% in memory footprint of the traditional method respectively. Significant reduction in memory is achieved at the expense of decoding time because it is sufficient to run the discrepancy computation loop longer and there is no need to code for Chien search and Forney's formula. It is ideal for applications requiring small software FEC codes operating at low data rates.

3.7 Conclusion

A new efficient evaluation based encoding and decoding scheme has been proposed for full length RS codes which can also be adapted for shortened RS codes. The hardware circuitry for the proposed decoder is simple as compared to the classical decoding algorithm since it requires only the syndrome calculation and BM algorithm blocks. The hardware implementations on a commercially available $65nm$ ASIC process shows that

Table 3.1: Memory and decoding time implementation results for RS(32, 24)

	Program Size (Bytes)	Clock cycles (10^3)
Classical Encoder	450	10.024
Proposed Encoder	324	48.68
Classical Decoder	2744	30.016
Proposed Decoder	1426	65.208

the proposed decoder is area efficient as well as faster in comparison to the classical one. For an RS(255, 239) code, the proposed decoder has a total cell count of 16075 which gives a reduction of 57.6% in hardware area as compared to the classical decoder. Also, a 53.07% reduction in decoding time is obtained with the proposed decoder which is a significant improvement.

Figure 3.6 shows that the rate of increase in area for the classical decoder is very high in comparison to the proposed decoder. It clearly indicates that the area for the classical decoder increases significantly even for small variation in error correction capability or with the dimension of the code. But, for the proposed decoder it does not vary much which is a significant improvement in terms of hardware area as compared to classical one.

Significant memory savings for software implementations were observed at the expense of processing time, throughput and latency for shortened RS codes. RS (32, 24) for resource constrained WSN application on a ATMEL Atmega 128 microcontroller shows that the new decoder and encoder is 52% and 72% in memory footprint with respect to the traditional method respectively.

In this chapter, we discussed about the efficient evaluation based algorithm and architecture for RS codes. But, one of the restrictions of RS codes is that code lengths are limited to the size of the chosen alphabet, that is, the finite field. Goppa generalized RS codes to algebraic geometry (AG) codes which allowed the code length to be much longer than the size of the codeword alphabet [42]. A subclass of these codes called Hermitian codes is one of the popular AG codes defined over Hermitian curve. Hermitian codes offer desirable properties over RS codes such as large code lengths over the same finite field and good error correction capability at high code rates. Hermitian codes are closely related to RS codes and can be seen as their concatenation with the added advantage of larger length over same alphabet. However, the RS codes are still among the most extensively used error correcting codes with many industrial applications because of the highly complex Hermitian decoders. In the next chapter, we will propose an efficient architecture for Hemrmitian decoders.

Chapter 4

Architecture for Interpolation Decoding of Hermitian Codes

4.1 Introduction

The performance of a block code is measured in terms of probability of decoding error which can be improved by increasing the length of the code relative to the codeword alphabet [61]. The main problem in making Hermitian codes industrially more applicable is to find a computationally simpler and area efficient method for the decoding algorithm so that it satisfies the resource and throughput constraints imposed by the application. For a long time, researchers were trying to find more efficient decoding algorithms and architectures for AG codes[62][63].

To improve the efficiency and simplicity of unique decoding techniques for AG codes is a classical research subject for a long time. A computationally efficient decoding algorithm for AG codes was first described by Justesen *et al.* [64]. In [65], Sakata *et al.* extended the algorithm to decode up to half the minimum distance. The running time of the algorithm was estimated to be $An^{7/3}$, where A is large enough to impact the term. There is a considerable amount of literature on efficient decoding of AG codes [66], [67], [68], [69], [70]. Until the advent of Guruswami and Sudan's list decoding algorithm based on interpolation which was a relaxation over unique decoding, the syndrome decoding was the only choice for decoding AG codes. But, due to the high computational demands and the architectural complexity of Hermitian codes, the unique decoding algorithm is much more interesting from implementation perspective, and researchers are still trying to optimize time and area complexity. The list decoding algorithm for AG codes is discussed using Gröbner bases in [71]. In [72], it is observed that by setting the parameters list size and multiplicity as 1, the list decoding algorithm of [71] will just be reminiscent of Kötter's implementation of the Berlekamp-Massey syndrome decoding algorithm [73] without the majority voting enhancement. In [72], the majority voting part which makes the algorithm decode up to half of the order bound was devised.

In this chapter, we will present an efficient VLSI architecture for the Lee-O'Sullivan interpolation based decoding algorithm [72] explicitly for Hermitian codes. We developed the architecture for the Lee-O'Sullivan algorithm based on interpolation and we proved that the hardware implementation complexity for the widely used high rate codes, the time complexity will be $O(q^5)$ which is q times faster than Kötter's and the space complexity will be of $O(q^4)$ which is same as Kötter's algorithm. The new algorithm computes the message directly from the received word under evaluation encoding without computing the error locations and the values unlike Kötter's algorithm. All the comparisons in this chapter are made with just the key equation part of Kötter's algorithm with minor modifications, and without taking into account the extra memory and extra clock cycles, required by Chien Search and Forney's Formula, which follow the key equation part in Kötter's algorithm.

4.1.1 Introduction to Gröbner basis

We present some introductory definitions for Gröbner basis [74] before going into the details of the Lee-O'Sullivan algorithm. Let $\mathbb{F}$ be a field and $A = \mathbb{F}[x_1, x_2, \ldots, x_n]$. An element $x^{i_1} \ldots x^{i_n}$ is called a term. If T is the set of all terms, we can define an order relation $<$ if:

- For any pair of terms m and n we have $m < n$ or $n < m$ or $m = n$
- if $m < n$ and $n < p$ then $m < p$
- if $m < n$ then $pm < pn$ for any term p

With such ordering, $f \in A$ can be written uniquely as sum of monomials:

$$f = c_1 m_1 + c_2 m_2 + \ldots + c_k m_k$$

such that $m_1 > m_2 > \ldots > m_k$. We define the leading term of f to be $lt(f) = m_1$ and corresponding leading monomial as $lm(f) = c_1m_1$. The definition of Gröbner basis is as follows:

Definition 4.1.1. *A set of non-zero polynomials $G = \{g_1, g_2, \ldots, g_n\}$ contained in an ideal I, is called a Gröbner basis of I iff for all $f \in I$ such that $f \neq 0$ there exists $i \in \{1, 2, \ldots, t\}$ such that $lt(g_i)$ divides $lt(f)$.*

Hence, $f \in I$ iff f has remainder 0 under division by Gröbner basis of I, where division means successive reduction of f by multiples of generators based on comparison of the leading terms of Gröbner basis with the leading terms of dividend/remainder. In other words, the leading term of any polynomial in I is divisible by the leading term of some polynomial in the basis G.

4.1.2 Introduction to list decoding and interpolation problem

As we discussed, the unique decoding problem can be solved in two parts, that is first to find the error locator polynomial and then to compute the error locations and error values. Based on the problem of the computing error locator polynomial being seen as a problem of finding a plane curve interpolating points, Sudan developed list decoding algorithm [75]. Similarly, the list decoding algorithm consists of two steps: the interpolation step and the root finding step. Both of these problems can be solved in multiple ways. Since, the interpolation problem can be solved by finding a solution of a system of linear equations over a field, it left the researchers to search for an efficient interpolation algorithm. There are two important perspectives to look at for the interpolation problem. Several authors including Kötter's algorithm (as presented by McEliece in [76]), formulated the interpolation problem as a problem of finding the minimal polynomial with respect to weighted monomial order of the ideal of polynomials interpolating certain points. The algorithm was based on the point by point approach by building the Gröbner basis of the ideal for points $\{P_1, P_2, \ldots, P_n\}$ by recursively computing the Gröbner basis of the ideal for points $\{P_1, P_2, \ldots, P_i\}$ while i increases from 1 to n.

Another approach is also based on the Gröbner basis but the strategy is different. The algorithm starts with a set of generators of the module induced from the ideal for $\{P_1, P_2, \ldots, P_n\}$ and converts the generators to a Gröbner basis of the module, in which the minimal polynomial is found. This results in an efficient algorithm for solving the interpolation problem. The Lee-O'Sullivan algorithm discussed in this chapter is based on this approach and will be the direct decoding algorithm as discussed for RS codes. The algorithm will output the message word directly. We will discuss the detailed algorithm in next section.

The structure of this chapter is as follows. Section 4.2 introduces the Hermitian codes and their evaluation based encoding. In section 4.3, we discuss the Lee-O'Sullivan decoding algorithm based on interpolation. The difference between the Kötter and the Lee-O'Sullivan decoder architectures is pointed out in section 4.5 at block level. Section 4.6 presents the architectures for the computational units required in the Lee-O'Sullivan decoder. In Section 4.7, we discuss increasing the efficiency of the Lee-O'Sullivan decoder further with the addition of a division block. Finally, section 4.8 concludes the chapter.

4.2 Hermitian code construction

4.2.1 Hermitian curve

The codes constructed by choosing points from a curve and a space of rational functions on this curve are called AG codes as discussed in Chapter 2. In this chapter, we will discuss about the encoding from the evaluation perspective.

4.2.2 Encoding by evaluation

Hermitian curve has $q^3 + 1$ points with a unique point P_∞ at infinity with a unique valuation of ν_{P_∞} associated with it. Let $\delta(x) = q$ and $\delta(y) = q + 1$. A function in the coordinate ring $\mathbb{F}[x, y]$ can be written as a unique $\mathbb{F}$-linear combination of monomials $x^i y^j$ with $i \geq 0$ and $0 \leq j < q$. The numerical semigroup of $\mathbb{R}$ at P_∞ can be defined as [71]:

$$S = \{s_0, s_1, \ldots\} = \{\delta(f) | f \in \mathbb{R}\} = \{\delta(x^i y^j) | i \geq 0, 0 \leq j < q\}$$
$$= \{qi + (q+1)j | i \geq 0, 0 \leq j < q\}$$

and is known as non-gaps. As already mentioned in Chapter 2, for each non-gap there is a unique monomial $x^i y^j$ with $0 \leq j < q$ such that $\delta(x^i y^j) = qi + (q+1)j = s$.

Let us denote the monomial as φ_s. Let $\mathbb{F}^n$ be the Hamming space over $\mathbb{F}$. Hamming space is a mathematical space in which words of some given length (here n) are situated. The evaluation map $ev : R \to \mathbb{F}^n$ is defined by

$$\varphi \mapsto (\varphi(P_1), \varphi(P_2), \ldots, \varphi(P_n))$$

which is a linear map over $\mathbb{F}$.

Hermitian codes are obtained by evaluation of functions f in the linear span of $\{x^i y^j : iq + j(q+1) \leq u, 0 \leq i, 0 \leq j < q\}$ for a fixed positive integer u.

A message $m = (m_0, m_1, \ldots, m_k) \in \mathbb{F}^k$ is encoded $ev(\mu) \in C_u$ where,

$$\mu = \sum_{s=1}^{k} m_s \varphi_s \in f$$

and C_u is hermitian code with parameter u. Note that it is a non-systematic encoding. For systematic encoding, the reader is referred to [21].

4.3 Lee-O'Sullivan decoder based on interpolation

This section presents an algorithm of the general Lee-O'Sullivan algorithm tailored for Hermitian codes. The Lee-O'Sullivan algorithm first constructs a set of generators for the module induced from the ideal of polynomials passing through all the interpolation points. Then the set of generators are converted to a Gröbner basis. It does not involve any complex computations like discrepancy computation as in [73]. Hence, it can be useful for a much more efficient implementation. The detailed algorithm is discussed in [71]. We will explain the algorithm briefly.

4.3.1 Unique decoding by interpolation

In this section, we will explain the theory behind the Lee-O'Sullivan algorithm. Assuming a codeword $c \in C_u$ is sent through noisy transmission channel and $v = c + e$ is received. Then $c = ev(\mu)$ for a unique:

$$\mu = \sum_{s \in S, s \leq u} w_s \varphi_s, \qquad w_s \in \mathbb{F}$$

Under evaluation encoding, the vector $(w_s | s \in S, s \leq u) \in \mathbb{F}^k$ is the message encoded to codeword c. The decoding problem essentially is then to find out w_s for all non-gaps

$s \leq u$ from the given received word v. For $s \geq u$, let $v^{(s)} = v$, $c^{(s)} = c$ and $\mu^{(s)} = \mu$. For non-gap $s \leq u$,

$$\mu^{(s-1)} = \mu^{(s)} - w_s \varphi_s,$$

$$c^{(s-1)} = c^{(s)} - ev(w_s \varphi_s),$$

$$v^{(s-1)} = v^{(s)} - ev(w_s \varphi_s),$$

and for gap $s \leq u$, $v^{(s-1)} = v$, $c^{(s-1)} = c$ and $\mu^{(s-1)} = \mu$. Note that, $\mu^s \in L_s$, $c^{(s)} \in C_s$ and $v^{(s)} = c^{(s)} + e$ for all s. Hence, the decoding problem can be re written as iteratively figuring out w_s for each non-gap s from u to 0.

A polynomial in $\mathbb{R}[z]$ defines a function on the product surface of Hermitian curve and affine line and can be evaluated at a point (P, v) such that P is a point on curve and $v \in \mathbb{F}$. Hence, the module of $z-$ linear polynomials over $\mathbb{R}$ that interpolates the point (P_i, v_i) is given by:

$$I_v = \{f \in \mathbb{R}z \oplus \mathbb{R} | f(P_i, v_i) = 0, 0 \leq i \leq n-1\}.$$

The Lee-O'Sullivan decoding algorithm works with the Gröbner basis of $I_{v^{(s)}}$ with respect to the monomial order $>_s$ on $\mathbb{R}z \oplus \mathbb{R}$. The monomial $x^i y^j z^k$ of $\mathbb{R}[z]$ is given the weight $o(x^i y^j) + sk$. The monomial order $>_s$ orders the monomials of $\mathbb{R}z \oplus \mathbb{R}$ by their weighted degrees. For any $f \in \mathbb{R}z \oplus \mathbb{R}$, the notations $lt_s(f)$, $lm_s(f)$ and $lc_s(f)$ denote the leading term, leading monomial and leading coefficient of f with respect to $>_s$ and $deg_s f$ is the weighted degree of $lt_s(f)$. Let M is a submodule of $\mathbb{R}z \oplus \mathbb{R}$. A subset B of M is called a Gröbner basis with respect to $>_s$ if the leading term of every element of M is divided by the leading term of some element of B. We can write,

$$B = \{G_i, F_j\}$$

with i and j as some index sets assuming $lt_s(G_i) \in \mathbb{R}$ and $lt_s(F_j) \in \mathbb{R}z$. Suppose $B^{(s)}$ is a Gröbner basis of $I_{v^{(s)}}$ with respect to $>_s$ then the majority voting procedure makes a guess of $w^{(s)}$ of w_s from $B^{(s)}$.

4.4 Unique decoding algorithm

The unique decoding algorithm is presented in Algorithm 5.

Algorithm 5 Unique decoding algorithm

```
Input: Received word $v$
 1: INITIALIZATION: Compute $h_v$ and $N = o(h_v)$
 2: for $i = 0 \ : \ n-1$ do
 3:     if $s$ is a non-gap $\leq u$ then
 4:         Make a guess $w^{(s)}$ for $w_s$ and let $B_1 = \{G_i(z + w^{(s)}\varphi_s), F_j(z + w^{(s)}\varphi_s\}$.
 5:     else
 6:         $B_1 = B^{(s)}$.
 7:     end if
 8:     Compute $B^{(s-1)}$ from $B^{(s)}$.
 9: end for
10: return $w^{(s)}$ for non-gaps $s \leq u$
```

Before going through the detailed algorithm for Hermitian codes, we will introduce some notations. Let q be a prime power. Let

$n = q^3$	number of points on the curve used for encoding
u	parameter that determines the dimension of the code
$\mathbb{F}$	finite field of q^2 elements
$\mathbb{F}[x, y]$	bivariate polynomial ring over $\mathbb{F}$
$\deg_x a$	x-degree of polynomial a in x
$a[x^k]$	coefficient of the term x^k in polynomial a
$s \bmod q$	integer remainder of s divided by q
s/q	integer quotient of s divided by q
$a \bmod y^q + y - x^{q+1}$	polynomial a reduced by $y^q = x^{q+1} - y$
$\mathrm{lc}(a)$	leading coefficient of polynomial a
$\tau = \lfloor (d_u - 1)/2 \rfloor$	with order bound d_u

For $1 \leq i \leq n$, let (α_i, β_i) denote the i-th point on the Hermitian curve.

4.4.1 Precomputation

For computing the initial Gröbner basis, the Lagrange interpolation polynomial needs to be computed. The Lee-O'Sullivan paper discussed the computation of the Lagrange polynomial h_v for the received word v. The Lagrange polynomial can be computed as $h_v = \sum_{i=1}^{n} v_i h_i$, where h_i is Lagrange basis for points corresponding to $P_1, \ldots, P_n$. Also, h_i's can be computed in advance as shown in Equation 4.1.

$$h_i = -\frac{(x^{q^2} - x)(y^q + y - \beta_i^q - \beta_i)}{(x - \alpha_i)(y - \beta_i)} \in \mathbb{F}[x, y]. \tag{4.1}$$

For Hermitian curve, $P_i = (\alpha_i, \beta_i)$ are the points on the curve, which we will discuss in details in section 4.4.2. In the proposed architecture, we will compute h_i's along with the computation of h_v as it will reduce the storage space dramatically and is more appropriate from circuit perspective. The details will be discussed later in section 4.6.1.

4.4.2 Points on the curve

The q^3 rational points representation (on Hermitian curve) is taken from [77]. The $n = q^3$ points which are represented in the form (α_i, β_i) on the curve are given as $(\alpha^{s_i}, \alpha^{s_i(q+1)+1} + \gamma_j)$ where $i = 0, 1, \ldots, q^2 - 1$ and $j = 0, 1, \ldots, q - 1$ such that $s_i = i - 1$ if $i \geq 1$ and $s_0 = -\infty, \alpha^{-\infty} = 0$. Also, $\{\gamma_0, \gamma_1, \ldots, \gamma_{q-1}\}$ are the q solutions to the equation $y^q + y = 0$ in $\mathbb{F}_{q^2}$ and α is the primitive element in $\mathbb{F}_q^2$. The total q^3 points are shown below:

$$\begin{array}{cccc}
P_0 = (0, \gamma_0) & P_1 = (1, \alpha + \gamma_0) & \cdots & P_{q^2-1} = (\alpha^{q^2-2}, \alpha^{(q^2-2)(q+1)+1} + \gamma_0) \\
P_{q^2} = (0, \gamma_1) & P_{q^2+1} = (1, \alpha + \gamma_1) & \cdots & P_{2q^2-1} = (\alpha^{q^2-2}, \alpha^{(q^2-2)(q+1)+1} + \gamma_1) \\
\vdots & \vdots & \ddots & \vdots \\
P_{q^2(q-1)} = (0, \gamma q - 1) & P_{q^2(q-1)+1} = (1, \alpha + \gamma_{q-1}) & \cdots & P_{q^3-1} = (\alpha^{q^2-2}, \alpha^{(q^2-2)(q+1)+1} + \gamma_{q-1})
\end{array}$$

4.4.3 Decoding

Let $v = [v_i]$ be the received vector in $\mathbb{F}^n$. Let s be a non-gap such that $s \leq u$. Suppose $\{g_i^{(s)}, f_i^{(s)} | 0 \leq i < r\}$ is a Gröbner basis of the module of $z-$linear polynomial that interpolates the point (P_i, v_i) with respect to $>_s$. For further details, please refer to

[72]. The polynomials f_i, g_i can be interpreted as following throughout the algorithm:

$$\begin{aligned} g_i &= \sum_{0\le j<q} c_{i,j} y^j z + \sum_{0\le j<q} d_{i,j} y^j \\ f_i &= \sum_{0\le j<q} a_{i,j} y^j z + \sum_{0\le j<q} b_{i,j} y^j \end{aligned} \tag{4.2}$$

with $a_{i,j}, b_{i,j}, c_{i,j}, d_{i,j} \in \mathbb{F}[x]$ during the execution.

Initialization The Lagrange polynomial h_v can be computed from (4.3) using the received word v and the pre-computed values of h_i.

$$h_v = \sum_{i=1}^{n} v_i h_i. \tag{4.3}$$

The initialization of Gröbner basis can be done as: For $0 \le i < q$, set

$$\begin{aligned} g_i &= y^i (x^{q^2} - x) \\ f_i &= y^i (z - h_v) \bmod y^q + y - x^{q+1} \end{aligned} \tag{4.4}$$

The iterations will be performed from $s = \delta(h_v)$ to $s = 0$, where $\delta(h_v)$ is the weighted degree of h_v. Further simplification will result in, $s = \delta_x * deg_x h_v + \delta_y * deg_y h_v = q * (q^2 - 1) + (q+1)(q-1) = q^3 + q^2 - q - 1$ (which is same as the initialization value of s) to $s = 0$. The decoding will be done from majority voting and hence for computing the message word $[w^{(s)}]_{0\le s\le u, s_x \ge 0}$, majority voting will be performed from $s = u$ to $s = 0$.

Pairing Set

$$s_y = s \bmod q$$

$$s_x = \frac{(s - s_y)}{q} - s_y$$

For $0 \le i < q - s_y$, set

$$k_i = \deg_x a_{i,i} + s_x$$

$$i' = i + s_y$$

For $q - s_y \le i < q$, set

$$k_i = \deg_x a_{i,i} + s_x + q + 1$$

$$i' = i + s_y - q$$

For $0 \le i < q$, set

$$c_i = \deg_x d_{i',i'} - k_i.$$

Voting If $s > u$ or $s_x < 0$, then set $w = 0$, and for i with $k_i < 0$, set

$$w_i = -b_{i,i'}[x^{k_i}], \quad \mu_i = 1$$

and for i with $k_i \geq 0$, set

$$w_i = 0, \quad \mu_i = 1.$$

If $s_x \geq 0$, then for $0 \leq i < q$, set

$$\mu_i = \mathrm{lc}(a_{i,i}), \quad w_i = -\frac{b_{i,i'}[x^{k_i}]}{\mu_i}$$

and let $\bar{c}_i = \max\{c_i, 0\}$, and let w be the element of $\{w_i\}$ with the largest

$$\sum_{w=w_i} \bar{c}_i.$$

Finally if $s \leq u$ and $s_x \geq 0$, set $w^{(s)} = w$.

Rebasing If $w \neq 0$, then for $0 \leq i < q$, set

$$\begin{aligned} g_i &= g_i(z + wx^{s_x}y^{s_y}) \bmod y^q + y - x^{q+1} \\ f_i &= f_i(z + wx^{s_x}y^{s_y}) \bmod y^q + y - x^{q+1} \end{aligned} \tag{4.5}$$

For $0 \leq i < q$, do the following. If $w_i \neq w$ and $c_i > 0$, then set (saving f_i in a temporary variable)

$$\begin{aligned} f_i &= x^{c_i} f_i - \frac{\mu_i(w - w_i)}{\nu_{i'}} g_{i'} \\ g_{i'} &= f_i \end{aligned} \tag{4.6}$$

and set $\nu_{i'} = \mu_i(w - w_i)$. If $w_i \neq w$ and $c_i \leq 0$, then set

$$f_i = f_i - \frac{\mu_i(w - w_i)}{\nu_{i'}} x^{-c_i} g_{i'}. \tag{4.7}$$

If $s > 0$, then set $s = s - 1$ and go back to Pairing, otherwise go to *Output*.

Output Output $[w^{(s)}]_{0 \leq s \leq u, s_x \geq 0}$.

This is the basic description of the algorithm defined in the paper of Lee-O'Sullivan. We will present an efficient VLSI architecture for this algorithm, where we simplified the equations by using some mathematical tricks which will be shown in the later parts of the chapter. We have also compared our architecture with the best existing algorithm for Hermitian decoding. The differences at block level between both the algorithms are covered in next section.

4.5 Difference between the architectures of Kötter's and Lee-O'Sullivan decoding algorithm at block level

As we already know that the decoding problem in coding theory is split into two sub-problems. One is to find the support of the error vector that computes the the error locator polynomial and then computing the roots of the polynomial to determine the error locations. Second is to find the error values corresponding to the error locations. Kötter's algorithm works on the same decoding principle.

The various blocks of Kötter's decoding scheme are shown in Figure 4.1. The syndrome computation block where computes the syndromes from the received word. The syndromes are then passed to the key equation block to compute the error locator polynomial. There is a feedback loop for the majority voting block because without using the majority voting block, the decoding algorithm can not correct up to the full error correction capability of Hermitian codes. Thus, with the use of the majority voting block, some extra syndromes can be computed and those extra syndromes are again fed back to the key equation block. With the known error locator polynomial, the roots of the polynomial (error locations) can be determined using a Chien Search and the error values can be determined using the Forney's Formula. The received word is stored in a register which can be added at the end of the decoding process with the error values at specific error locations in order to determine the originally transmitted codeword.

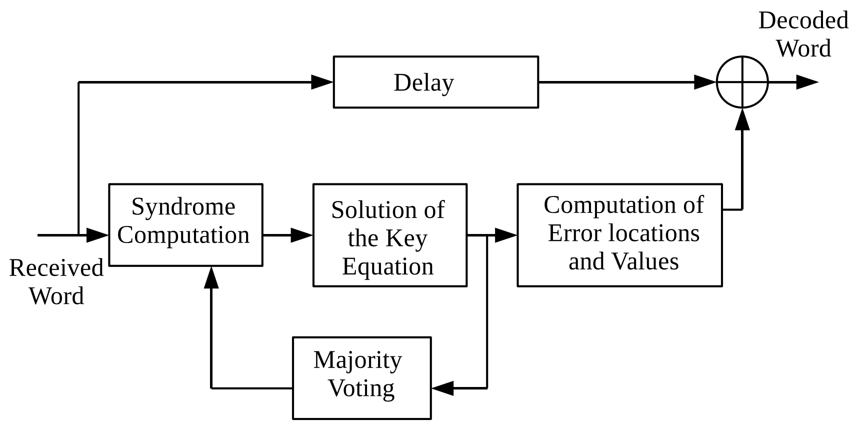
Decoded Word
Delay
Syndrome Computation
Solution of the Key Equation
Computation of Error locations and Values
Received Word
Majority Voting

Figure 4.1: Kötter decoder

Following the description of the algorithm in the previous section, the different blocks for the Lee-O'Sullivan decoding or the interpolation decoding algorithm are shown in Figure 4.2. Like the syndrome decoding algorithm, the Lee-O'Sullivan decoding algorithm corrects errors of up to half of the order bound and is also suitable for deriving a parallel hardware architecture. It computes the message vector directly from the received vector under evaluation encoding, which is a distinctive feature of the list decoding algorithm. The Lee-O'Sullivan decoding algorithm uses the theory of Gröbner basis of modules which allows simple and regular structure. The detailed architecture for each block will be discussed in later sections of the chapter. The first block computes the Lagrange polynomial that is h_v which is needed to compute the initial Gröbner basis with respect to (wrt) s that is the initialization of the polynomials f_i and g_i for $i = 0, \dots, q-1$. At the rebasing block, the polynomials f_i, g_i for $0 \leq i \leq q-1$ are updated and thus will give the new Gröbner basis with respect to $s-1$. When s reaches u, the algorithm will start outputting the message word as $w^{(s)}_{0 \leq s \leq u, s_x \geq 0}$.

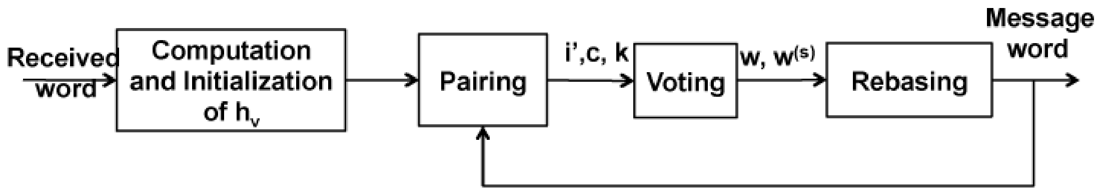

Figure 4.2: Lee-O'Sullivan decoder

4.6 Architecture for the Lee-O'Sullivan decoder

We developed the complete VLSI architecture for various blocks of Lee-O'Sullivan decoder and also simplifeid some of the parts mathematically to make the architecture more efficient. In the following subsections, the algorithm and the architecture for all the blocks shown in Figure 4.2 will be covered in detail.

4.6.1 Architecture for the computation of h_v

The set of h_i is called the Lagrange basis for the points $P_1, P_2, \dots, P_n$ on the curve and $v = [v_i]$ is the received word. In the Lee-O'Sullivan algorithm, the h_i's are precomputed but from the hardware point of view it requires significant space for storing $n = q^3$ different h_i's. It takes q^3 registers for storing each of the h_i, thus in total it needs

$q^3 * q^3 = q^6$ registers for storing all h_i's which is a large storage space. We developed a method to compute h_v simultaneously with h_i's as shown in Figure 4.3, instead of pre-computing and storing it. We need q replicas of this same circuit, for the computation of complete h_v.

For practical usage, the field used is of the form $\mathbb{F}_{2^m}$ which means all the subtractions is treated as bitwise xor and we will use it throughout the chapter.

These h_i as shown in (4.1) can be rewritten as the polynomial given below under the condition $\alpha_i \neq 0$:

$$h_i = y^{q-1}(x^{q^2-1} + \ldots + \alpha_i^{q^2-2}x) + \ldots + (\beta_i^{q-1} + 1)(x^{q^2-1} + \ldots + \alpha_i^{q^2-2}x) \qquad (4.8)$$

and we write all these h_i's as a two dimensional matrix in x and y for our particular case where codes are defined over $\mathbb{F}_{q^2}$.

$$h_i = \begin{bmatrix} 0 & \alpha_i^{q^2-2}(\beta_i^{q-1}+1) & \alpha_i^{q^2-3}(\beta_i^{q-1}+1) & \cdots & (\beta_i^{q-1}+1) \\ 0 & \alpha_i^{q^2-2}\beta_i^{q-2} & \alpha_i^{q^2-3}\beta_i^{q-2} & \cdots & \beta_i^{q-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \alpha_i^{q^2-2}\beta_i^{0} & \alpha_i^{q^2-3}\beta_i^{1} & \cdots & \beta_i^{0} \end{bmatrix}$$

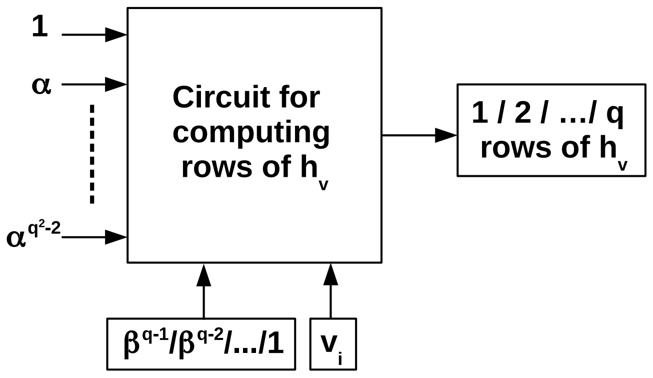

Figure 4.3: Architecture for computation of h_v at block level

If $\alpha_i = 0$ then the first column of h_i will be $[1\ 0\ \ldots\ 0\ 1]$. The first and last row values of the first column of h_v will be just the addition of the v_i corresponding to the zero values of α_i.

For the ease of computation, the q^3 values for the terms $\beta_i^{q-1} + 1, \beta_i^{q-2}, \ldots, \beta_i$ are pre-computed and are initially saved in the dual port memories which can be reused for

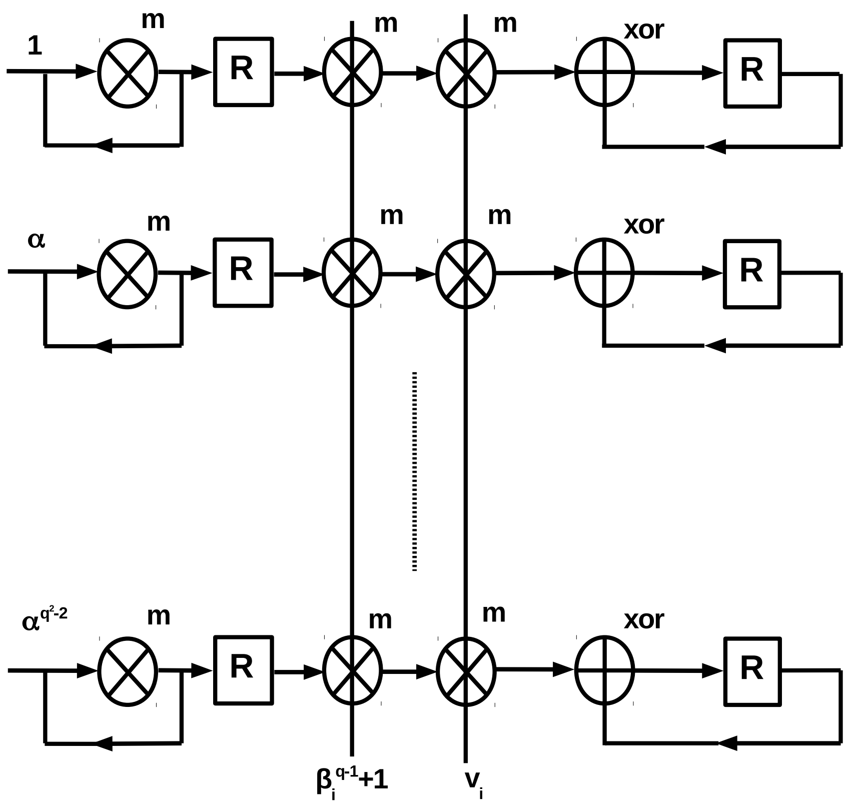

Figure 4.4: Circuit for computing h_v

the storage of the polynomials f and g later in the algorithm after the h_v computation as they are not needed for the rest of the algorithm. The architecture for the computation of the rest $q^2 - 1$ columns of h_v (except the 1^{st} column) based on the matrix structure discussed above is shown in Figure 4.3. The circuity inside the box is shown in Figure 4.4. In Figure 4.3, with the changing values of inputs $\beta_i^{q-1} + 1$, β_i^{q-2}, …, 1 will compute the 1^{st} ,2^{nd}, …, $(q-1)^{th}$ rows of h_v respectively. Figure 4.4 shows only the the computations for the first row elements of h_v matrix that is the multiplication term is $\beta_i^{q-1}+1$. In the first row of h_v, if we focus on the 2^{nd} element that is $\alpha_i^{q^2-2}(\beta_i^{q-1}+1)$, which is represented by the last row in Figure 4.4, we can easily see that at each clock cycle $\alpha_i^{q^2-2}(\beta_i^{q-1}+1)$ is computed for various (α_i, β_i) and gets added to the previous value stored in register R. This will continue for $n = q^3$ clock cycles and h_v will be obtained. There can be $q-1$ more circuits similar to this with the multiplication values set as β_i^{q-2}, β_i^{q-3}, …, 1. The computation of h_v requires q^3 registers for storing the h_v. It also needs $3q^3$ multipliers, q^3 adders, q^3 registers and in terms of time it is performed in q^3 clocks. Also, if we don't use q of this circuits, it would take a bit longer for computing the h_v that is q^4 clocks but then only $3q^2$ multipliers and q^2 adders are needed which actually keeps the product of area and time as a constant. This step is used only once and the received word does not need to be stored after computing h_v.

The structure for storing f_i's and g_i's for $0 \leq i \leq q-1$ is shown in Figure 4.5. The memory used is dual port RAM so that it will allow multiple reads or writes to occur at the same time. The coefficients are stored as shown in the figure as $1, x, \ldots, x^{df}$ on the columns and $1, y, \ldots, y^{q-1}, z, yz, \ldots, y^{q-1}z$ on $2q$ rows.

4.6.2 Intialization

The polynomials f_i and g_i can be initialized according to (4.4). The upper bounds on the degree of f and g can be given as:

$$\deg_x f_i \leq (q^3 + 2q^2 + 4q - 5 - u + (u \bmod q))/2$$
$$\deg_x g_i \leq q^2 + q - 1$$

For storing each of the polynomial of f_i and g_i, $2q$ dual port memory blocks of size $deg_x f$ and $deg_x g$ are used. For total storage, we need $2q^2(deg_x f + deg_x g)$ which results in total as $O(q^5)$ in terms of memory. But, for the high rate codes where $n = q^3 \simeq u$ makes the x-degree of f as $O(q^2)$ and so the space complexity is $O(q^4)$ which is the

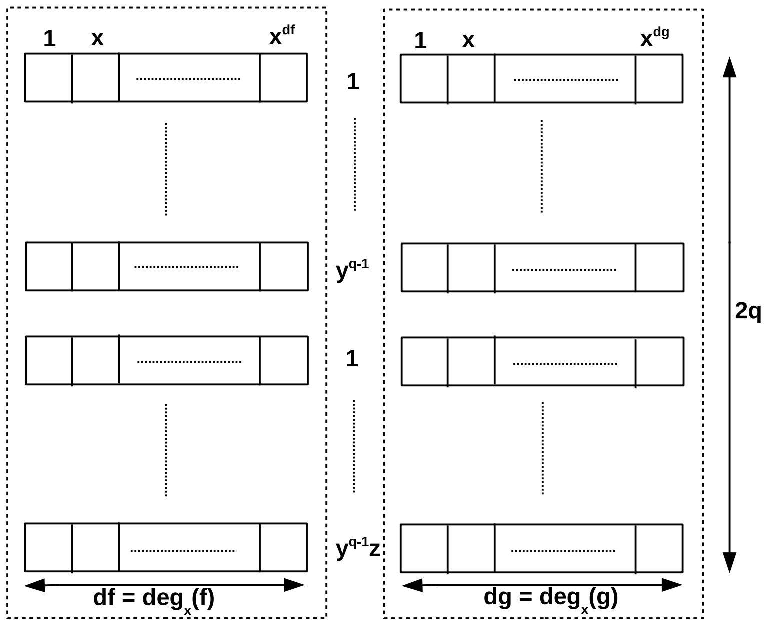

Figure 4.5: Storage structure for f_i and g_i in dual port memories

same as Kötter's. For convenience in the implementation of various blocks, the same x degrees is used for both f and g which makes them of the same size. This means we need the memory blocks of size $deg_x f$. For doing the modulo $y^q + y - x^{q+1}$ polynomial part, the same circuitry from a part of rebasing can be used with all the w_i's taken as 1. The architecture for modulo is discussed later in the rebasing step in section 4.6.4. The steps discussed below that is pairing, voting and rebasing are iterated for $s = q^3 + q^2 - q - 1$ times but the voting is performed only after $s = u$.

4.6.3 Voting

For the ease of implementation in hardware, the voting algorithm as discussed in section 4.4.3 is rewritten as Algorithm 6.

The circuitry for the voting part described in Algorithm 6 is discussed in Figure 4.6 and is mentioned in [21]. The voting circuit requires q multipliers and $4q^2$ registers along with 2 multiplexers.

4.6.4 Rebasing

We developed the architecture for rebasing and updating the polynomial which is discussed in this section. The complete rebasing part is mentioned in (4.5), (4.6) and

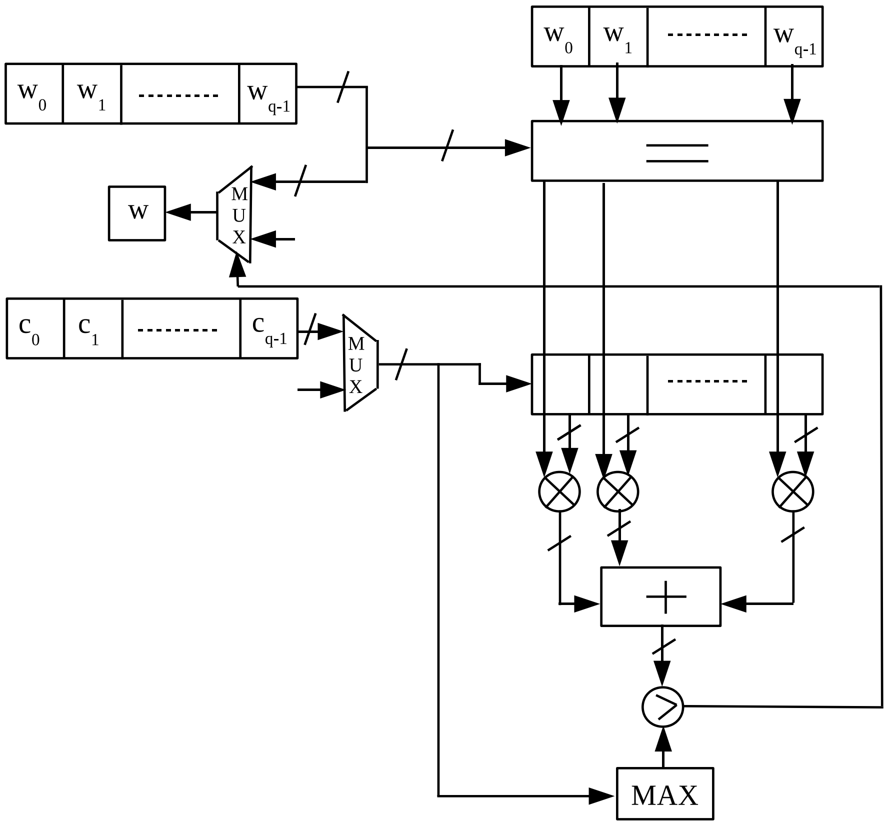

Figure 4.6: Voting

Algorithm 6 Voting

```
Input: $\{w_i\}, \{c_i\}$
1: $max = \sum_{i=0}^{q-1} c_i$
2: for $k = 0 \ : \ q-1$ do
3:     for $j = 0 \ : \ q-1$ do
4:         if $w_k == w_j$ then
5:             if $c_j \geq 0$ then
6:                 $Sum = Sum + c_j$
7:             end if
8:         end if
9:     end for
10:    if $Sum > max/2$ then
11:        $w^{(s)} = w_k$
12:    end if
13: end for
14: return $w^{(s)}$
```

(4.7).

The term with z is replaced by $(z + wx^{s_x}y^{s_y})$ and a modulo $y^q + y - x^{q+1}$ after that as mentioned in (4.5). As shown in Figure 4.7, the terms from f or g are read columnwise and is multiplied with w_i and are stored in registers $D_1, \ldots, D_q$. At the same time the address for them are read as $a_1 = iq + j(q+1)$, where i is column number and j is row number and for taking care for the multiplication by $x^{s_x}y^{s_y}$ (having address as $a_2 = qs_x + (q+1)s_y$), the new address is generated as $a_1 + a_2$ if (a_1 mod q)+(a_2 mod q) $\leq q$ or $a_1 + a_2$ and $a_1 + a_2 - q^2 + 1$ two addresses are generated otherwise. The address generation comes from polynomial multiplication $((x^{s_x}y^{s_y})(x^iy^j))$ modulo $y^q + y - x^{q+1}$. For details please refer to [73]. The same addresses are read in the q upper memory blocks and the values stored in registers $R_1, R'_1, \ldots, R'_q$ and the values stored in $D_1, \ldots, D_q$ are added to the values present at the registers $R_1, \ldots, R'_q$ depending upon the addresses and is written to the same address at the same clock as the memories are dual port, thus every column will take 1 clock cycle. This would effectively take $deg_x f$ clocks which will be q^2 for high rate codes. It requires $2q$ replicas of this circuit for all of f_i and g_i. Each circuit needs $5q$ registers, $2q$ adders, q multipliers and 1 multiplexer.

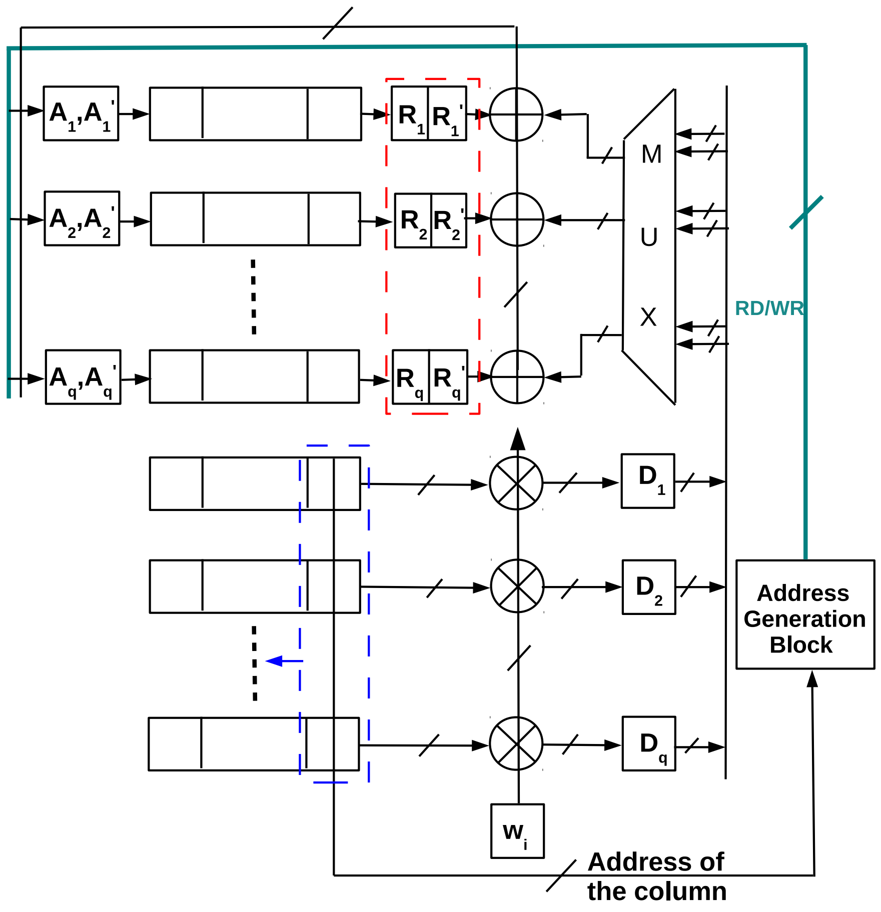

Figure 4.7: Rebasing

Thus, in total we need $10q^2$ registers, $4q^2$ adders, $2q^2$ multipliers and $2q$ multiplexers.

The updating of the polynomials part is given by the set of equations mentioned in (4.6) and (4.7). The architecture is shown in Figure 4.8. For updating, the two columns of f_i should be read at the same time. Suppose for the first iteration, the last non-zero column say nz, then $(nz+c_i)^{th}$ column and nz^{th} column should be read and stored in registers as shown at the same time. The coefficients stored in $(nz+ci)^{th}$ column of $g_{i'}$ should also be read simultaneously and multiplied with $\frac{\mu_i(w-w_i)}{\nu_{i'}}$ and stored in the register. The addition of these values should be written on the last non-zero column read of f_i and the values of $(nz+ci)^{th}$ column of f_i should be written on $(nz+ci)^{th}$ column of g_i. The next iteration will be done for $(nz+c_i-1)^{th}$ column and so on. This effectively takes $deg_x f$ clock cycles which is q^2 for high rate codes. It requires $2q$ replicas of this circuit for all of f_i and g_i. Each circuit needs $6q$ registers, q adders and q multipliers. Thus, in total we need $12q^2$ registers, $2q^2$ adders and $2q^2$ multipliers.

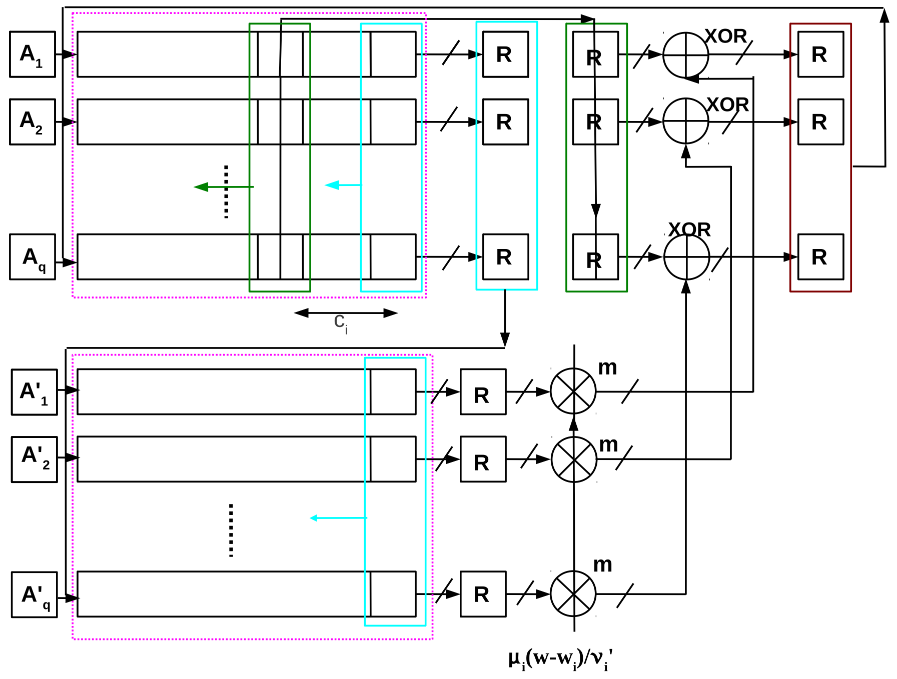

Figure 4.8: Updating

The total number of iterations that will include the rebasing and the updating steps are q^3+q^2-q-1 which is of $O(q^3)$. Thus, the total clock cycles needed effectively for the decoder is of $O(q^3 * deg_x f)$ which is maximum of $O(q^5)$ in any case for high rate codes.

Example 4.6.1. *Consider a Hermitian curve* $x^3 = y^2 + y$ *and a code defined over it with* $u = 3$*. The error correction capability is* $t = 2$ *and* α *is the primitive element of* $\mathbb{F}_{2^2}$*. There are* $q^3 = 8$ *points on the curve and can be written as* $[(0,0),(0,\alpha),(1,\alpha),(1,\alpha^2),(\alpha,\alpha),(\alpha,\alpha^2),(\alpha^2,\alpha),(\alpha^2,\alpha^2)]$*. The message needs to be transmitted is* $w = [0\ \alpha\ \alpha^2]$*. The set of non-gaps is* $S = \{0,2,3,4,5,6,7,8\}$*.*

The codeword is computed by evaluating the message polynomial at all the points of the curve. The message polynomial in our case is $a^2y + ax$*. The codeword is:* $c = (0\ \alpha^2\ \alpha^2\ 0\ \alpha\ 1\ 0\ \alpha^2)$*.*

Suppose the error introduced is: $e = (0\ 0\ 0\ \alpha\ 0\ \alpha\ 0\ 0)$*.*

The received word is $r = c + e$*:* $r = (0\ \alpha^2\ \alpha^2\ \alpha\ \alpha\ \alpha^2\ 0\ \alpha^2)$

Now, the decoding starts by computing the Lagrange interpolation polynomial h_v which is $q * q^2$ matrix.

$$h_v = \begin{bmatrix} 0 & \alpha^2 & \alpha & 0 \\ \alpha^2 & \alpha^2 & 1 & 0 \end{bmatrix}$$

The upper bound on the x degree for f_i and g_i can be computed as, $deg_x f_i = deg_x g_i = 8$. The Gröbner basis initialization:

$$f(:,:,0) = \begin{bmatrix} 0 & \alpha^2 & \alpha & 0 & 0 & 0 & 0 & 0 \\ \alpha^2 & \alpha^2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$f(:,:,1) = \begin{bmatrix} 0 & 0 & 0 & \alpha^2 & \alpha^2 & 1 & 0 & 0 \\ \alpha^2 & 0 & \alpha^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$g(:,:,0) = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$g(:,:,1) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

The iteration will start from $s = q^3 + q^2 - q - 1 = 9$. For the first iteration, after pairing:

$$s_x = 3 \qquad s_y = 1$$

$$k_0 = 3 \qquad i_0^{'} = 1 \qquad c_0 = 1$$

$$k_1 = 6 \qquad i_1^{'} = 0 \qquad c_1 = -2.$$

After the voting step:

$$w_0 = 0 \qquad \mu_0 = 1$$

$$w_1 = 0 \qquad \mu_1 = 1.$$

There will not be any substitution of z to $z + wx^{s_x}y^{s_y}$ for $s > u$. After the rebasing step, the polynomials will be updated:

$$f(:,:,0) = \begin{bmatrix} 0 & \alpha^2 & \alpha & 0 & 0 & 0 & 0 & 0 \\ \alpha^2 & \alpha^2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$f(:,:,1) = \begin{bmatrix} 0 & 0 & 0 & \alpha^2 & \alpha^2 & 1 & 0 & 0 \\ \alpha^2 & 0 & \alpha^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$g(:,:,0) = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$g(:,:,1) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

The procedure will continue similarly until $s = 3$ where $s = u$. For the $s = 3$ iteration, after pairing:

$$s_x = 0 \qquad s_y = 1$$

$$k_0 = 2 \qquad i_0' = 1 \qquad c_0 = 0$$

$$k_1 = 3 \qquad i_1' = 0 \qquad c_1 = 1.$$

the majority voting for outputting the message word will start for $s \leq u$. After the voting step:

$$w_0 = 1 \qquad \mu_0 = 1 \qquad \bar{c}_0 = 0$$

$$w_1 = \alpha^2 \qquad \mu_1 = 1 \qquad \bar{c}_1 = 1$$

Therefore, selected w is a^2.

There will be substitution of z to $z + wx^{s_x}y^{s_y}$ for $s > u$ followed by polynomial updating. The polynomials will be updated:

$$f(:,:,0) = \begin{bmatrix} 0 & \alpha^2 & 1 & \alpha & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \alpha & \alpha^2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$f(:,:,1) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \alpha & 0 & 0 & 0 & 0 & 0 & 0 \\ \alpha^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$g(:,:,0) = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$g(:,:,1) = \begin{bmatrix} 0 & \alpha^2 & \alpha & 0 & 0 & 0 & 0 & 0 \\ 0 & \alpha^2 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

The decoding process will continue similarly till $s = 0$ and the obtained message word is $w = [0\ \alpha\ \alpha^2]$.

4.7 Optimized Lee-O'Sullivan algorithm

The Lee-O'Sullivan algorithm can be further optimized in terms of time. It turns out that the interpolation-based unique decoding algorithm may benefit more from the list decoding, as the central concept of the Guruswami-Sudan list decoding, namely the Q-polynomial can be used along with the majority voting. The idea is that the computationally expensive Gröbner basis computation is iterated only until a Q-polynomial is found, and then the root of the Q-polynomial reveals the rest of the sent message. This simple idea boosts the decoding speed significantly. After the pairing step, if the $qk_i + (q+1)i' + \tau < q^3$ is satisfied which means we have the Q polynomial now then the algorithm directly goes to the division block and now the message is obtained only by division as shown in Figure 4.9.

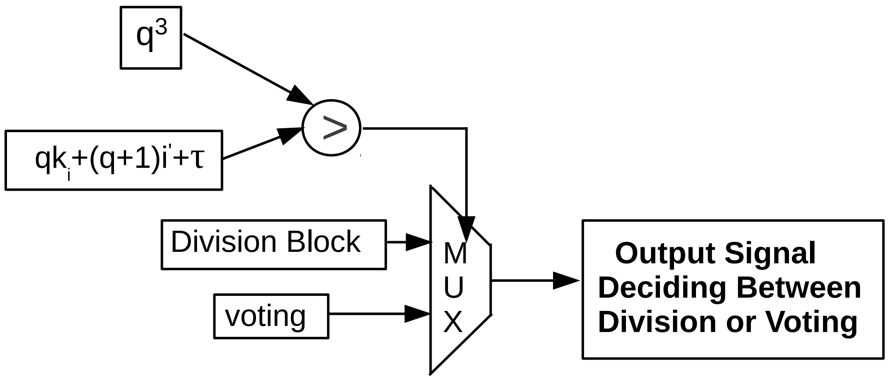

Figure 4.9: Conditional circuit for inclusion of division block

This results in skipping the voting and the polynomial updating part of the rebasing step which effectively saves up to q^3 clocks (q^2 clocks for high rate codes) at every iteration. This makes the algorithm 2 times faster than the above algorithm. The modified Lee-O'Sullivan algorithm decoder circuit at block level is shown in Figure 4.10. The division block does not require any extra circuitry. It just uses the circuitry from pairing and part of the rebasing circuit. The division block arithmetic is discussed below and is taken from [78]. After the initialization and the pairing blocks depending upon the control unit for the de-multiplexer, the decoding procedure can proceed either with the blocks of voting and rebasing or simply through the division block which re-use the circuitry from pairing block and the rest is a part of the rebasing block. The final output is the message word $w^{(s)}_{1\leq s\leq u, s_x\geq 0}$.

Division Set

$$s_y = s \bmod q$$

$$s_x = (s - s_y)/q - s_y$$

and if $i < q - s_y$, set

$$k_i = \deg_x(a_{i,i}) + s_x$$

$$i' = i + s_y$$

and if $i \geq q - s_y$, set

$$k_i = \deg_x(a_{i,i}) + s_x + q + 1$$

$$i' = i + s_y - q$$

If $s \leq u$ and $s_x \geq 0$, then set

$$w = -\frac{b_{i,i'}[x^{k_i}]}{\mathrm{lc}(a_{i,i})}$$

and set $w^{(s)} = w$, and if $w \neq 0$, then set

$$f_i = f_i(z + w x^{s_x} y^{s_y}) \bmod y^q + y - x^{q+1}$$

If $s > 0$, then set $s = s - 1$ and go back to Division, otherwise go to *Output*.

In the next chapter, we will discuss about the Generalized Toric codes that are also closely related to RS codes. They are the extension of RS codes over r-dimensions. With the Toric codes, we can construct longer codes as compared to RS codes over the same alphabet that can in turn improve the performance of the code in terms of probability of decoding error. We construct the subfield subcodes of GT codes which are essentially the multidimensional analogues of BCH codes and compute their parameters. Also, we will show with the help of several examples that this method can be used as a method to construct some optimal codes.

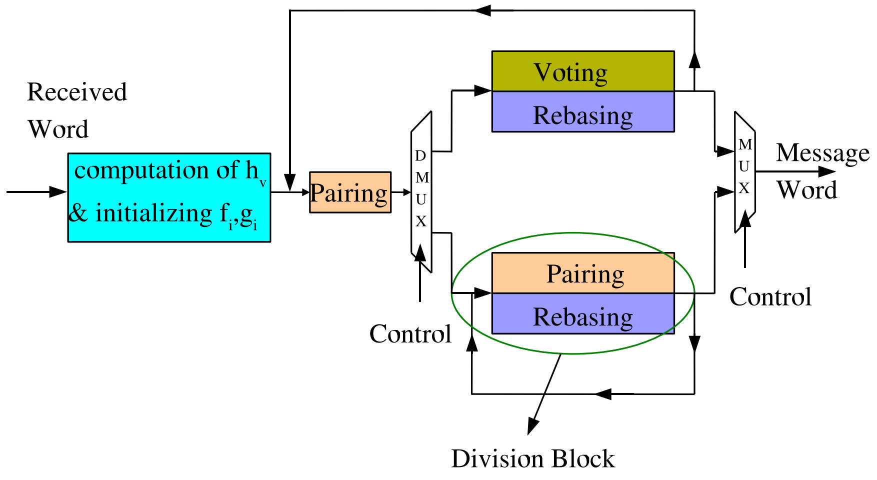

Figure 4.10: Lee-O'Sullivan decoder with division block

4.8 Conclusion

We developed and presented a novel and efficient architecture for the implementation of Lee-O'Sullivan decoder in this chapter. We computed the complexity of the algorithm from the hardware perspective in terms of area and decoding time and found that it is similar to the Kötter's algorithm. The decoding time complexity is $O(q^5)$ which is q times faster than the Kötter's algorithm for high rate codes and the space complexity in terms of registers, memories, adders and multipliers is of $O(q^4)$ which is same as Kötter's algorithm. Also, Lee-O'Sullivan appended the division block which makes the circuitry 2 times faster than the original circuit implementations of Lee-O'Sullivan algorithm. We presented the control circuit for faster Lee-O'Sullivan algorithm.

Chapter 5

Subfield Subcodes

In the previous chapters, we have already discussed about RS and Hermitian codes. In this chapter, we will discuss about the extension of RS codes over r dimensions that is known as Generalized Toric codes and then about its subfield subcodes. Similar to Hermitian codes, GT codes are also much longer than RS codes over same alphabet. The chapter describes a method to construct several optimal codes which is an important problem for coding theorists. The subfield subcodes of Generalized Toric codes is the multidimensional analogue of BCH codes which are essentially the subfield subcodes of RS codes. It is a known fact that BCH codes, Goppa codes and more generally alternant codes [79] can be defined as subfield subcodes. Subfield subcodes can also be used as a method of constructing new codes from old codes. Assuming $q = p^s$, where p is a prime number. A very useful method of constructing codes over $\mathbb{F}_p$ is to restrict the codes which are defined over $\mathbb{F}_{p^s}$. In this chapter, we study subfield-subcodes of Generalized Toric (GT) codes over $\mathbb{F}_{p^s}$. These are the multidimensional analogues of BCH codes, which may be seen as subfield-subcodes of generalized RS codes [80], [81], [82], [83], [84].

5.1 Subfield-Subcodes

Definition 5.1.1. *Let C be a linear code of length n over $\mathbb{F}_{p^s}$, the subfield-subcode of C, say D, is the set of the codewords $c \in C$ such that $c \in \mathbb{F}_p^n$, i.e., $D = C \cap \mathbb{F}_p^n$.*

Many authors have been interested in computing the dimension of subfield-subcodes. Delsarte studied in [81] the subfield-subcodes of modified RS codes. Stichtenoth improved the bound on dimension in [84] and Shibuya et al gave a better lower bound on dimension [83]. Later on Hattori, McEliece and Solomon [85] gave a lower bound on the dimension of subspace-subcodes of RS codes. Finally Jie and Junying [86] generalized the previous bound for Generalized RS codes.

There is a method of defining a code over $\mathbb{F}_p$ if a code over $\mathbb{F}_{p^s}$ is given [11]. This construction uses the trace mapping defined in Equation 5.1. The class of cyclic codes can be naturally represented as trace codes. Consider the field extension $\mathbb{F}_{p^s}/\mathbb{F}_p$, which is a Galois extension of degree $[\mathbb{F}_{p^s} : \mathbb{F}_p] = s$. The trace map Tr, takes the elements of $\mathbb{F}_{p^s}$ to $\mathbb{F}_p$ and can be denoted as:

$$\mathrm{Tr} : \mathbb{F}_{p^s} \longrightarrow \mathbb{F}_p$$

For $a = (a_1, a_2, \ldots, a_n) \in \mathbb{F}_{p^s}^n$, we define:

$$\mathrm{Tr}\,(a) := (\mathrm{Tr}\,(a_1), \mathrm{Tr}\,(a_2), \ldots, \mathrm{Tr}\,(a_n)) \in \mathbb{F}_p^n. \tag{5.1}$$

This is the method to obtain the $\mathbb{F}_p$-linear map $Tr : \mathbb{F}_{p^s}^n \to \mathbb{F}_p^n$.

Definition 5.1.2. $\mathrm{Tr}\,(C) := \{\mathrm{Tr}\,(C) | c \in C\} \subseteq \mathbb{F}_p^n$ *is called the trace code of* C.

Definition 5.1.3. *Dual code of a linear code* $C \subset \mathbb{F}_q^n$ *is the linear code defined as* $C^{\perp} = \{x \in \mathbb{F}_q^n \mid \langle x, c\rangle = 0 \ \forall \ c \in C\}$, *where* $\langle x, c\rangle = \sum_{i=1}^{n} x_i c_i$ *is a scalar product.*

An important thing to be noted is both subfield subcode and trace code of a code $C \in \mathbb{F}_{p^s}^n$ are codes over $\mathbb{F}_p$ having length n. In particular Delsarte provides the following result (Theorem 5.1.4) which gave a relation between subfield subcodes and trace codes [81]:

Theorem 5.1.4.

$$(C \cap \mathbb{F}_p^n)^{\perp} = \mathrm{Tr}\,(C^{\perp})$$

where $\mathrm{Tr} : \mathbb{F}_{p^s} \to \mathbb{F}_p$ *sending* x *to* $x + x^p + \cdots + x^{p^{s-1}}$.

5.1.1 Bounds for the dimension of subfield subcodes and trace codes

There are obvious bounds on the dimensions of subfield subcodes and trace codes:

$$dim\ C|_{\mathbb{F}_p} \leq dim(C) \tag{5.2}$$

$$dim\,\mathrm{Tr}\,(C) \leq s\ dim(C) \tag{5.3}$$

Equation 5.2 follows from the fact that a basis of $C|_{\mathbb{F}_p}$ over $\mathbb{F}_p$ is also linearly independent over $\mathbb{F}_{p^s}$. Equation 5.3 follows since $\mathrm{Tr}\,: C \to \mathrm{Tr}\,(C)$ is a surjective $\mathbb{F}_p$-linear mapping and the dimension of C, regarded as a vector space over the field $\mathbb{F}_p$, is $s\ dim(C)$.

5.2 BCH codes as subfield-subcode of a RS code

In this section, we are going to introduce a new approach for BCH codes (the authors presented this approach in [87]) which is also discussed in [11]. The main idea is that BCH codes are subfield-subcodes of RS codes. The latter is a well known family of codes which may be described as an evaluation codes. This together with elementary properties of Galois extensions allow us to describe BCH codes as an evaluation code as well.

The next result is provided in [11] although it is possibly known before.

Proposition 5.2.1. *A BCH code D over $\mathbb{F}_p$ of length $n = p^s - 1$ is a subfield-subcode of a RS code C over $\mathbb{F}_{p^s}$, and therefore $d(D) \geq d(C)$.*

In this chapter, we will extend the concept of subfield-subcodes to multidimensional analogues of RS codes that is Toric codes.

5.3 Generalized Toric codes

Toric codes are algebraic geometry codes over Toric varieties. These codes were introduced by J.P. Hansen [37], see also [88]. Let M be an integral lattice and P be a convex polytope in $M \otimes \mathbb{R}$. The Toric code C_P over $\mathbb{F}_q$ associated to P is the evaluation code generated by the monomials x^α where $\alpha \in P \cap M$ at the points of the algebraic torus $T = (\mathbb{F}_q^*)^r$. A lower bound for the minimum distance is estimated in [89] using intersection theory and mixed volumes, extending the methods of J.P. Hansen for plane polytopes.

D.Ruano introduces a natural generalization of this family, the so called Generalized Toric Codes [39], which consist of the evaluation of any polynomial algebra in the

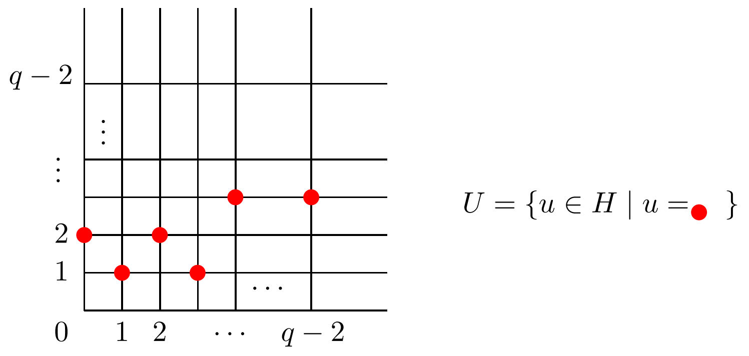

Figure 5.1: $U \subseteq \{0, \ldots, q-2\}^r$

algebraic torus. More precisely, one may consider any subset $U \subseteq \{0, \ldots, q-2\}^r$ which is pictorially shown in Figure 5.3.

The corresponding vector space $\mathbb{F}_q[U] = \langle \{x^u = x_1^{u_1} \cdots x_r^{u_r} \mid u = (u_1, \ldots, u_r) \in U\} \rangle \subset \mathbb{F}_q[x_1, \ldots, x_r]$, thus the Generalized Toric code, C_U, is the image under the $\mathbb{F}_q$-linear map, $ev : \mathbb{F}_q[U] \to \mathbb{F}_q^n$, $ev(f) = (f(t))_{t \in T}$, $n = (q-1)^r$. It is clear from his construction that any Toric code is a GT code.

Proposition 5.3.1. *Let $H = \{0, \ldots, q-2\}^r$ and $n = (q-1)^r$. The $\mathbb{F}_q$-linear map*

$$ev : \mathbb{F}_q[H] \to \mathbb{F}_q^n, \quad f \to (f(t))_{t \in T}$$

is an isomorphism [11].

Corollary 5.3.2. *In particular, ev restricted to $\mathbb{F}_q[U]$ is injective, so dim$(C_U) = |U|$ [11].*

The next result may be found in [90] and [39].

Proposition 5.3.3. *For $u \in H$, let $\hat{u} \in H$ be defined by $\hat{u}_i = 0$ if $u_i = 0$ and $\hat{u}_i = q - 1 - u_i$ if $u_i \neq 0$. Let C be the GT code defined by $U \subset H$, then $C^{\perp}$ is the GT code defined by $U^{\perp} = \{\hat{u} : u \in U\}$.*

5.4 Subfield subcodes of Generalized Toric codes

Let R be $\mathbb{F}_{p^s}[y_1, \ldots, y_r]/\langle y_1^{p^s-1} - 1, \ldots, y_r^{p^s-1} - 1 \rangle$. The r-dimensional vector $y_1, y_2, \ldots, y_r$ is represented as $\underline{y}$. We are looking for $f \in R$ such that $f(t) \in \mathbb{F}_p, \forall t \in T$. If this occurs we say that f is a polynomial evaluating to $\mathbb{F}_p$. The idea is to find out first all those polynomials evaluating to $\mathbb{F}_p$ in R and then restrict this set to $\mathbb{F}_{p^s}[U]$.

Proposition 5.4.1. *$ev(f) \in \mathbb{F}_p^n \Leftrightarrow f(t) = (f(t))^p \ \forall t \in T \Leftrightarrow f^p = f$ in R.*

Proof. According to Proposition 5.3.1 we know that $ev(f)^p = ev(f^p)$ then it is clear that:

$ev(f) \in \mathbb{F}_p^n \Leftrightarrow f(t) = (f(t))^p \forall t \in T \Leftrightarrow ev(f) = ev(f)^p \Leftrightarrow ev(f) = ev(f^p) \Leftrightarrow ev(f - f^p) = 0 \Leftrightarrow f - f^p \in ker(ev) \Leftrightarrow f^p(\underline{y}) = f(\underline{y})$ in R. □

5.4.1 Cyclotomic cosets and their properties

Consider $G = Gal(\mathbb{F}_{p^s} \mid \mathbb{F}_p) = \{g_0, \ldots, g_{s-1}\}$ the Galois group, where g_i maps α to α^{p^i}. Looking at exponents, we may consider G to act on $\mathbb{Z}_{p^s-1}$ by multiplying by p and this may be naturally extended to $\mathbb{Z}_{p^s-1} \times \cdots \times \mathbb{Z}_{p^s-1}$ by multiplying by p coordinate wise. The orbits of G on $\mathbb{Z}_{p^s-1} \times \cdots \times \mathbb{Z}_{p^s-1}$ are called cyclotomic cosets, i.e., for every $\underline{b} \in (\mathbb{Z}_{p^s-1})^r$ we define the cyclotomic coset $I_{\underline{b}}$ by $\{\underline{b}, p\underline{b}, \ldots, p^{n_{\underline{b}}-1}\underline{b}\}$ where $n_{\underline{b}}$ is the smallest positive integer such that $\underline{b} = \underline{b}p^{n_{\underline{b}}}$. The integer $n_{\underline{b}}$ is the cardinal of $I_{\underline{b}}$.

Some known properties of cyclotomic cosets [11]:

Proposition 5.4.2.

(i) $I_{\underline{b}} = \{\underline{b}, p\underline{b}, p^2\underline{b}, \ldots, p^{n_{\underline{b}}-1}\underline{b}\}$ *is closed under multiplication by* p.

(ii) The cardinal of $I_{\underline{b}}$ *is either* s *or a divisor of it.*

(iii) $I_{\underline{b}}$ *and* $I_{\underline{b}'}$ *are either identical or they don't intersect. Thus* $\mathcal{B} = \{I_{\underline{b}} : \underline{b} \in (\mathbb{Z}_{p^s-1})^r\}$ *partitions* $(\mathbb{Z}_{p^s-1})^r$.

5.4.2 Computation of evaluation polynomial for subfield subcodes of GT codes

In this section, we will compute the basis for subfield subcode of GT codes and in the process, we developed various theorems. Based on those theorems, we computed the dimension of subfield subcodes as we already know the basis of subfield subcode. If $\theta : R \to R$ is an isomorphism and f evaluates to $\mathbb{F}_p$. Then so does $\theta(f)$. This is because $\theta(f)^p = \theta(f^p) = \theta(f)$. So it is worthwhile cataloging some isomorphisms of R.

Proposition 5.4.3. *(i) For any i coprime with $p^s - 1$, the map θ_i fixing F_{p^s} and taking $f(y_1, \ldots, y_r) \to f(y_1^i, \ldots, y_r^i)$ is an isomorphism of R.*

(ii) *For any $\underline{\alpha} \in \mathbb{F}_{p^s}^* \times \cdots \times \mathbb{F}_{p^s}^*$, the map $\theta_{\underline{\alpha}}$ fixing $\mathbb{F}_{p^s}$ and taking $f(y_1, \ldots, y_r) \to f(\alpha_1 y_1, \ldots, \alpha_r y_r)$ is an isomorphism of R.*

(iii) *The Frobenious map on $\mathbb{F}_{p^s}$ combined with $y_i \mapsto y_i$ for $i = 1, \ldots, r$ is an isomorphism of R.*

Let $f(\underline{y}) = \sum a_{i_1,\ldots,i_r} y_1^{i_1} \cdots y_r^{i_r} \in R$, we denote *supp*$(f) = \{\underline{i} \mid a_{\underline{i}} \neq 0\}$ as the support of f. If $I_{\underline{b}}$ is a cyclotomic coset, we denote $f_{I_{\underline{b}}} = \sum_{\underline{i} \in I_{\underline{b}}} \underline{y}^{\underline{i}}$ as the polynomial having *supp*$(f) = I_{\underline{b}}$ and coefficients equal to one. It is easy to verify that $f_{I_{\underline{b}}}$ evaluates to $\mathbb{F}_p$. Note that $\theta_{\underline{\alpha}}(f_{I_{\underline{b}}}) = \sum_{\underline{i} \in I_{\underline{b}}} \alpha_1^{i_1} y_1^{i_1} \cdots \alpha_r^{i_r} y_r^{i_r}$ is the polynomial with support $I_{\underline{b}}$ and coefficients determined by $\underline{\alpha}$. Since $\theta_{\underline{\alpha}}$ is an isomorphism, $\theta_{\underline{\alpha}}(f_{I_{\underline{b}}})$ evaluates to $\mathbb{F}_p$.

Let $l = |\mathcal{B}|$ be the number of cyclotomic cosets and let $J = \{\underline{b}_1, \ldots, \underline{b}_l\}$, be a set of representatives, so $\mathcal{B} = \{I_{\underline{b}_1}, \ldots, I_{\underline{b}_l}\}$. From now on we will denote by $f_{I_{\underline{b}},\beta}$ the polynomial with support I_b and leading coefficient β, i.e., $f_{I_{\underline{b}},\beta} = \beta \underline{y}^{\underline{b}} + \beta^p \underline{y}^{\underline{b}p} + \cdots + \beta^{p^{n_{\underline{b}}-1}} \underline{y}^{\underline{b}p^{n_{\underline{b}}-1}}$. Note that $f_{I_{\underline{b}},\beta}$ evaluates to $\mathbb{F}_p$ if and only if $\beta \in \mathbb{F}_{p^{n_{\underline{b}}}}$.

Proposition 5.4.4. *Let f be a function that evaluates to $\mathbb{F}_p$ with supp$(f) = I_{\underline{b}}$ and let β be a primitive element of $\mathbb{F}_{p^{n_{\underline{b}}}}$. Then, f is a linear combination of $f_{I_{\underline{b}}}, f_{I_{\underline{b}},\beta}, \ldots, f_{I_{\underline{b}},\beta^{n_b-1}}$.*

Proof. Since *supp*$(f) = I_{\underline{b}}$ and $f^p = f$ there is some α such that $f = \alpha \underline{y}^{\underline{b}} + \alpha^p \underline{y}^{\underline{b}p} + \cdots + \alpha^{p^{n_{\underline{b}}-1}} \underline{y}^{\underline{b}p^{n_{\underline{b}}-1}}$. Moreover $\alpha^{p^{n_{\underline{b}}}} = \alpha$, which implies that $\alpha \in \mathbb{F}_{p^{n_{\underline{b}}}}$.

We know that $\{1, \beta, \ldots, \beta^{(n_{\underline{b}}-1)}\}$ is a basis of $\mathbb{F}_{p^{n_{\underline{b}}}}$ over $\mathbb{F}_p$, so $\alpha = a_0 + a_1\beta + \cdots + a_{n_{\underline{b}}-1}\beta^{(n_{\underline{b}}-1)}$, with $a_i \in \mathbb{F}_p$ for all i. Therefore,

$$\begin{aligned}
f &= \sum_{i=0}^{n_{\underline{b}}-1} \alpha^{p^i} \underline{y}^{\underline{b}p^i} \\
&= \sum_{i=0}^{n_{\underline{b}}-1} \underline{y}^{\underline{b}p^i} \Big(\sum_{j=0}^{n_{\underline{b}}-1} a_j \beta^j \Big)^{p^i} \\
&= \sum_{j=0}^{n_{\underline{b}}-1} a_j \sum_{i=0}^{n_{\underline{b}}-1} \beta^{jp^i} \underline{y}^{\underline{b}p^i} \\
&= \sum_{j=0}^{n_{\underline{b}}-1} a_j f_{I_{\underline{b}},\beta^j}
\end{aligned}$$

□

Proposition 5.4.5. *$f_{I_{\underline{b}}}, f_{I_{\underline{b}},\beta}, \ldots, f_{I_{\underline{b}},\beta^{n_{\underline{b}}-1}}$ are linearly independent over $\mathbb{F}_p$.*

Proof. Suppose it is not true. Thus, $a_0 f_{I_{\underline{b}}} + a_1 f_{I_{\underline{b}},\beta} + \cdots + a_{n_{\underline{b}}-1} f_{I_{\underline{b}},\beta^{n_{\underline{b}}-1}} = 0$. The smallest monomial in the left hand side is $(a_0 + a_1\beta + \cdots + a_{n_b-1}\beta^{(n_{\underline{b}}-1)})\underline{y}^{\underline{b}}$ which has to be zero. This is true if β is a root of $p(z) = a_0 + a_1 z + \cdots + a_{n_b-1} z^{n_b-1}$, but this is not possible because the minimal polynomial of β has degree $n_{\underline{b}}$. ☐

Theorem 5.4.6. *A basis for the set of polynomials evaluating to $\mathbb{F}_p$ is:*

$$\bigcup_{I_{\underline{b}}\in\mathcal{B}} \{f_{I_{\underline{b}},\beta^j} : j \in \{0,\ldots,n_{\underline{b}}-1\}, \beta \text{ primitive in } \mathbb{F}_{p^{n_{\underline{b}}}}\}.$$

Proof. If $I_{\underline{b}}$ and $I_{\underline{b}'}$ are two different cosets then $f_{I_{\underline{b}},\beta}$ and $f_{I_{\underline{b}'},\beta'}$ have different degrees. So, there is no way to generate one from the other which proves that different classes are linearly independent. Moreover within the set of polynomials with the same support, say I_b, we know from Corollary 5.4.5 that the only linearly independent are $\{f_{I_b,1}, f_{I_b,\beta}, \ldots, f_{I_b,\beta^{n_b-1}}\}$. So, the only part left is to see that it is a system of generators.

Consider the smallest monomial in f, say $\beta^{j_1}\underline{y}^{\underline{b}}$ then $f_{I_{\underline{b}},\beta^{j_1}} = \sum_{k=0}^{n_b-1}(\beta^{j_1}\underline{y}^{\underline{b}})^{p^k}$ must appear in f. Since $\beta^{j_1}\underline{y}^{\underline{b}}$ is the smallest monomial in f, therefore $\underline{b}$ must be one of the leaders in $J = \{\underline{b}_1, \ldots, \underline{b}_l\}$. Assume without loss of generality that $\underline{b}_1 < \underline{b}_2 < \cdots < \underline{b}_l$ and $\underline{b} = \underline{b}_1$.

Consider $f_1 = f - f_{I_{\underline{b}_1},\beta^{j_1}}$ and the first monomial on it, say $\beta^{j_2}\underline{y}^{\underline{b}'}$. Again, the polynomial $f_{I_{\underline{b}'},\beta^{j_2}} = \sum_{k=0}^{n_b-1}(\beta^{j_2}\underline{y}^{\underline{b}'})^{p^k}$ must appear in f_1. We may assume that $\underline{b}' = b_2$ and consider $f_2 = f_1 - f_{I_{\underline{b}_2},\beta^{j_2}}$.

In at most l-steps, we can finish the process obtaining that $f = a_1 f_{I_{\underline{b}_1},\beta^{j_1}} + \cdots + a_l f_{I_{\underline{b}_l},\beta^{j_l}}$, which concludes the proof. ☐

For the next result we introduce an $\mathbb{F}_p$ linear mapping on R extending the trace map, $T : R \to R$ is given by $g \mapsto g + g^p + \ldots g^{p^{s-1}}$ for all $g \in R$.

Corollary 5.4.7. *The image of T is exactly the set of $f \in R$ that evaluate to $\mathbb{F}_p$.*

Proof. Let $f = T(g) = g + g^p + \cdots + g^{p^{s-1}}$. Since $g^{p^s} = g$ in R we have $f^p = f$. Thus any image of the map T evaluates to $\mathbb{F}_p$.

For the converse, it is sufficient, by Proposition 5.4.4, to show that each $f_{I_{\underline{b}_l},\beta}$ is in the image of T for β an element of $\mathbb{F}_{p^{n_{\underline{b}}}}$. Let $\gamma \in \mathbb{F}_{p^s}$ be such that $\mathrm{Tr}_{\mathbb{F}_{p^s}/\mathbb{F}_{p^{n_{\underline{b}}}}}(\gamma) = \beta$.

Then

$$T(\gamma \underline{y}^{\underline{b}}) = \sum_{i=0}^{s-1} \gamma^{p^i} \underline{y}^{\underline{b}p^i}$$
$$= \sum_{j=0}^{\frac{s}{n_{\underline{b}}}-1} \sum_{i=0}^{n_{\underline{b}}-1} \gamma^{p^{i+jn_{\underline{b}}}} \underline{y}^{\underline{b}p^{i+jn_{\underline{b}}}}$$

Since $\underline{b}p^{n_{\underline{b}}} = \underline{b}$,

$$= \sum_{i=0}^{n_{\underline{b}}-1} \underline{y}^{\underline{b}p^i} \Big(\sum_{j=0}^{\frac{s}{n_{\underline{b}}}-1} \gamma^{(p^{n_{\underline{b}}})^j} \Big)^{p^i}$$

The term in parentheses is $\mathrm{Tr}_{\mathbb{F}_{p^s}/\mathbb{F}_{p^{n_{\underline{b}}}}}(\gamma) = \beta$, so

$$T(\gamma \underline{y}^{\underline{b}}) = f_{I_{\underline{b}_l},\beta}$$

□

This provides us a constructive method of producing all those polynomials which evaluate to $\mathbb{F}_p$. In particular, if we restrict to those polynomials with support in U, we trivially have a formula for the dimension of a subfield-subcode.

Theorem 5.4.8. *Let $U \subseteq \{0\ldots,q-2\}^r$ and let $D_U = C_U \cap \mathbb{F}_p^n$.*

$$D_U = ev\Big(T(\mathbb{F}_{p^s}[H]) \cap \mathbb{F}_{p^s}[U]\Big)$$

A basis for D_U is:

$$\bigcup_{I_{\underline{b}}:I_{\underline{b}}\subseteq U} \{f_{I_{\underline{b}},\beta^j} : j \in \{0,\ldots,n_{\underline{b}}-1\}, \beta \textit{ primitive in } \mathbb{F}_{p^{n_{\underline{b}}}}\}$$

Moreover it has dimension

$$\dim D_U = \sum_{I_{\underline{b}}:I_{\underline{b}}\subseteq U} n_{\underline{b}}$$

.

Proof. From Theorem 5.4.6, the basis for the set of polynomials evaluating to $\mathbb{F}_p$ is:

$$\bigcup_{I_{\underline{b}}\in\mathcal{B}} \{f_{I_{\underline{b}},\beta^j} : j \in \{0,\ldots,n_{\underline{b}}-1\}, \beta \text{ primitive in } \mathbb{F}_{p^{n_{\underline{b}}}}\}.$$

Now, for ev restricted to $\mathbb{F}_p[U]$, will restrict the degree of monomials to the set U and since we know that for the polynomials evaluating to $\mathbb{F}_p$ their support should lie in cyclotomic cosets. Therefore, basis for D_U will be a subset of U and can be written as a union of polynomial with support in cyclotomic coset which is also a subset of U:

$$\bigcup_{I_{\underline{b}}:I_{\underline{b}}\subseteq U} \{f_{I_{\underline{b}},\beta^j} : j \in \{0,\ldots,n_{\underline{b}}-1\}, \beta \text{ primitive in } \mathbb{F}_{p^{n_{\underline{b}}}}\}$$

Then, the dimension is obvious from Corollary 5.3.2 and is equal to $\sum_{I_{\underline{b}}:I_{\underline{b}}\subseteq U} n_{\underline{b}}$.

$\square$

Remark 5.4.1. *When $r = 1$ and $U = \{0, 1, 2, \ldots, k-1\}$ the GT code is a RS code with parameters $[p^s - 1, k, p^s - k]$.*

This provide us a constructive way of producing all those polynomials which maps $\mathbb{F}_{2^r} \to \mathbb{F}_2$. A possible algorithm is shown in Algorithm 7.

We have built a function in MAGMA for the above algorithm and confirmed that every subfield-subcode of a Reed-Solomon code over $\mathbb{F}_{64}$ (except two cases) is the best known linear code over $\mathbb{F}_2$. Same is true for $\mathbb{F}_{128}$ as well. For example the subfield-subcode of the Reed-Solomon Code $[63, 42, 19]$ is $[63, 16, 23]$.

Example 5.4.9. *Let C be an $[n, k, d]$ RS code with $q = 2^4$, $n = 15$, i.e. we evaluate all the polynomials of degree less than or equal to $k - 1$, at all the points of $\mathbb{F}_{16}^*$. Let D be the subfield-subcode of C, that is, $D = C \cap \mathbb{F}_2^{15}$. What are the functions $f : \mathbb{F}_{16} \to \mathbb{F}_2$ we have to evaluate to get D?*

The different cosets are $I_0 = \{0\}$, $I_1 = \{1, 2, 4, 8\}$, $I_3 = \{3, 6, 12, 9\}$, $I_5 = \{5, 10\}$, $I_7 = \{7, 14, 13, 11\}$. Depending on the value of k we have:

- *From $1 \le k \le 8$, the only function is $f = 1$ corresponding to the coset I_0, so the code D is $[15, 1, 15]$.*

- *If $k = 9$, $C = [15, 9, 7]$ then we have $T_r(x) = f_{I_1}, f_{I_1,\alpha}, f_{I_1,\alpha^2}, f_{I_1,\alpha^3}$ and $f_{I_0} = 1$. Then D is a $[15, 5, 7]$ code.*

- *If $k = 10$ nothing new.*

- *If $k = 11$, $C = [15, 11, 5]$ we consider I_0, I_1 and I_5. That is $f_{I_5} = x^5 + x^{10}$ and $f_{I_1,\alpha} = \alpha^5 x^5 + \alpha^{10} x^{10}$ in addition to the previous functions. Therefore, $D = [15, 7, 5]$.*

- *If $k = 12$ nothing new.*

Algorithm 7 BCH codes as evaluation codes

Input: Dimension of the code = k ; Field = $\mathbb{F}_{2^r}$.

Output: Generator Matrix of the resultant subfield subcode of RS(n,k), where $n = 2^r - 1$.

```
1: $G = \{0, 1, 2, ..., 2^r - 1\}$; $H = \{1, 2, 4, ..., 2^{(r-1)}\}$ ;
2: while $G \neq \varphi$ do
3:     $CH_{k,1} = G(k) * H(0)\ mod\ (2^r - 1)$;
4:     for $i = \{2, ..., r\}$ do
5:         if $CH_{k,i} \neq CH_{k,1}$ then
6:             $CH_{k,i} = G_k * H_i\ mod\ (2^r - 1)$;
7:             $G = G - CH_{k,i}$;
8:         end if
9:     end for
10: end while
11: if $G_{row,column} > k - 1$ then
12:     $p = row$;
13:     $G^{new}$ = After Removing $p^{th}$ row from $G$;
14: end if;
15: for $i = \{1, ..., size(G^{new})\}$ do
16:     for $j = \{1, ...r\}$ do
17:         if $G^{new}_{j,i} \neq 0$ then
18:             $P = P + x^{G^{new}_{j,i}}$;
19:         end if;
20:     end for;
21:     $c = 1$;
22:     for $m = \{1, ..., r\}$ do
23:         if $G^{new}[j, m] \neq 0$ then
24:             $P_c(x)$ =Evaluate($P, \alpha^{(m-1)}x$);
25:             $c = c + 1$;
26:         end if;
27:     end for;
28: end for;
29: Generator Matrix =Evaluate($\forall P(x), \forall$ elements of $\mathbb{F}_{2^r} - \{0\}$);
```

- *If $k = 13$, $C = [15, 13, 3]$ we consider I_0, I_1, I_5 and I_3. That is $f_{I_3,\alpha^i} = \alpha^{3i}x^3 + \alpha^{6i}x^6 + \alpha^{9i}x^9 + \alpha^{12i}x^{12}$ in addition to the previous functions, for $0 \leq i \leq 3$. Therefore, $D = [15, 11, 3]$.*
- *If $k = 14$ nothing new.*
- *If $k = 15$, $C = [15, 15, 1]$ and $D = [15, 15, 1]$ with the 4 new functions corresponding to I_7: $f_{I_7,\alpha^i} = \alpha^{7i}x^7 + \alpha^{11i}x^{11} + \alpha^{13i}x^{13} + \alpha^{14i}x^{14}$ for $0 \leq i \leq 3$.*

5.5 Dual of Subfield-Subcodes

Theorem 5.1.4 together with Theorem 5.4.8 motivate this section.

Let $U \subseteq \{0, \ldots, q-2\}^r$ and let $C_U = ev(\mathbb{F}_{p^s}[U])$ and $D_U = C_U \cap \mathbb{F}_p^n$. From Proposition 5.3.3, we know that $C_U^\perp$ is the GT code defined by $U^\perp$. From Delsarte's Theorem we have

$$D_U^\perp = \mathrm{Tr}\,(C_{U^\perp}) = \mathrm{Tr}\,(ev(\mathbb{F}_{p^s}[U^\perp])) = ev(T(\mathbb{F}_{p^s}[U^\perp]))$$

The last equality follows from $ev \circ T = \mathrm{Tr} \circ ev$, which is easily verified. Clearly, $T(\mathbb{F}_{p^s}[U^\perp])$ is spanned by $T(\gamma y^{\underline{b}})$ for $\underline{b} \in U^\perp$ and $\gamma \in \mathbb{F}_{p^s}$. For $\underline{b}$ fixed and varying γ we get exactly the set of $f_{I_{\underline{b}},\beta}$ for $\beta \in \mathbb{F}_{p^{n_{\underline{b}}}}$. Thus we have a basis for $D_U^\perp$.

Theorem 5.5.1. *$D_U^\perp$ has the basis*

$$\bigcup_{I_{\underline{b}}: I_{\underline{b}} \cap U^\perp \neq \emptyset} \{f_{I_{\underline{b}},\beta^j} : j \in \{0, \ldots, n_{\underline{b}} - 1\}, \beta \text{ primitive in } \mathbb{F}_{p^{n_{\underline{b}}}}\}$$

We therefore have

$$\dim D_U^\perp = \sum_{I_{\underline{b}}: I_{\underline{b}} \cap U^\perp \neq \emptyset} n_{\underline{b}}.$$

Proposition 5.5.2. *Let $\hat{U} = \{supp(h) \mid h = \mathrm{Tr}\,(y^{\underline{b}}), \underline{b} \in U^\perp\} = \{I_{\underline{b}} \mid \underline{b} \in U^\perp\} = \{p^i\underline{b} \mid \underline{b} \in U^\perp, i = 0, 1, \ldots, n_{\underline{b}} - 1\}$ Then $D_U^\perp = C_{\hat{U}} \cap \mathbb{F}_p^n = D_{\hat{U}}$.*

Corollary 5.5.3. *One can always decode $D^\perp$ up to $t = \lfloor d(C_{\hat{U}}) - 1/2 \rfloor$ with the decoding algorithm for $C_{\hat{U}}$.*

5.6 Computations

From the practical point of view it makes sense to choose U to be the union of different cyclotomic cosets, otherwise the evaluation will not be in $\mathbb{F}_p^n$.

We have written a function in MAGMA for computing the subfield-subcode of a GT code and we have found a number of optimal codes. Consider first the field $GF(2^3)$ and $r = 2$ so T is the Toric surface. In each of the following cases we give a subset U of $(\mathbb{Z}_7)^2$ and the parameters of $D = D_U$ and $D^\perp = D_U^\perp$, the subfield-subcode of C_U and its dual.

i) $U = [[1,0],[2,0],[4,0],[0,1],[0,2],[0,4]]$.
D is $[49,6,24]$ and $D^\perp$ is $[49,43,3]$.

ii) $U = [[6,3],[5,6],[3,5],[3,1],[6,2],[5,4],[6,1],[5,2],[3,4]]$.
D is $[49,9,20]$ and $D^\perp$ is $[49,39,3]$.

iii) $U = [[2,1],[4,2],[1,4],[3,1],[6,2],[5,4],[4,1],[1,2],[2,4],[0,0]]$.
D is $[49,10,20]$ and $D^\perp$ is $[49,39,4]$. If we consider,

$$U' = U \cup \{[[1,0],[2,0],[5,0],[6,0],[1,1],[2,2]\},$$

we get a new Toric code, $C_{U'}$, with parameters $[49,16,18]$, i.e, the minimum distance drops by 2 (with respect to C_U) and the subfield-subcode $D_{U'}$ is equal to D_U. The previous is an example of a subfield-subcode $D_{U'}$ of a GT code $C_{U'}$ where $d(D_{U'}) > d(C_{U'})$.

iv) $U = [[1,0],[2,0],[4,0],[2,3],[4,6],[1,5],[0,1],[0,2],[0,4],[6,3],[5,6],[3,5],[6,1],[5$
$,2],[3,4]]$.
D is $[49,15,16]$ and $D^\perp$ is $[49,34,6]$.

v) $U = [[1,0],[2,0],[4,0],[0,1],[0,2],[0,4],[1,1],[2,2],[4,4],[2,1],[4,2],[1,4],[3,1],[6$
$,2],[5,4],[4,1],[1,2],[2,4],[1,3],[2,6],[4,5]]$.
D is $[49,21,12]$ and $D^\perp$ is $[49,28,7]$. We use again the same strategy of adding points: consider $U' = U \cup \{[3,0],[6,0],[6,1],[5,2]\}$, we obtain the GT code $C_{U'}$ with parameters $[49,25,9]$ where the minimum distance drops by 3 and the subfield-subcode $D_U = D_{U'}$.

vi) $U = [[6,3],[5,6],[3,5],[1,0],[2,0],[4,0],[3,0],[6,0],[5,0],[2,1],[4,2],[1,4],[3,1],[6,$
$2],[5,4],[4,1],[1,2],[2,4],[5,1],[3,2],[6,4],[1,3],[2,6],[4,5],[2,3],[4,6],[1,5],[3,3]$
$,[6,6],[5,5],[4,3],[1,6],[2,5]].$
D is $[49,33,6]$ and $D^{\perp}$ is $[49,16,7]$.

vii) $U = [[6,3],[5,6],[3,5],[0,0],[0,1],[0,2],[0,4],[1,1],[2,2],[4,4],[3,1],[6,2],[5,4],[5,$
$1],[3,2],[6,4],[6,1],[5,2],[3,4],[0,3],[0,6],[0,5],[2,3],[4,6],[1,5],[3,3],[6,6],[5,5]$
$,[4,3],[1,6],[2,5],[5,3],[3,6],[6,5]].$
D is $[49,34,6]$ and $D^{\perp}$ is $[49,15,12]$.

viii) $U = [[6,3],[5,6],[3,5],[0,0],[1,0],[2,0],[4,0],[3,0],[6,0],[5,0],[1,1],[2,2],[4,4]$
$,[2,1],[4,2],[1,4],[4,1],[1,2],[2,4],[5,1],[3,2],[6,4],[6,1],[5,2],[3,4],[0,3],[0,6]$
$,[0,5],[1,3],[2,6],[4,5],[3,3],[6,6],[5,5],[4,3],[1,6],[2,5],[5,3],[3,6],[6,5]].$
D is $[49,40,4]$ and $D^{\perp}$ is $[49,9,14]$.

ix) $U = [[0,0],[1,0],[2,0],[4,0],[3,0],[6,0],[5,0],[0,1],[0,2],[0,4],[1,1],[2,2],[4,4]$
$,[2,1],[4,2],[1,4],[3,1],[6,2],[5,4],[4,1],[1,2],[2,4],[5,1],[3,2],[6,4],[6,1],[5,2]$
$,[3,4],[0,3],[0,6],[0,5],[1,3],[2,6],[4,5],[2,3],[4,6],[1,5],[3,3],[6,6],[5,5],[4,3]$
$,[1,6],[2,5],[5,3],[3,6],[6,5]].$
D is $[49,46,2]$ and $D^{\perp}$ is $[49,3,28]$.

Notice that $p = 2 \nmid s = 3$ thus from Theorem 5.5.1 we know that the dual of a subfield-subcode is again the subfield-subcode of another Toric code. In each example the code D is the best known code for a fixed length and dimension. Also in each example, except vi),vii) and viii) the dual code has the same correction capability as the best known code for a fixed length and dimension.

From now on we will denote by D the subfield-subcode of the GT codes over $GF(3^2)$ and $r = 2$. In each of the following cases we give a subset U of $(\mathbb{Z}_8)^2$ and the parameters of $D = D_U$ and $D^{\perp} = D_U^{\perp}$, the subfield-subcode of C_U and its dual.

i) $U = [[5,0],[7,0],[5,5],[7,7]]$
D is $[64,4,42]$ and $D^{\perp}$ is $[64,60,2]$.

ii) $U = [[5,1],[7,3],[0,0],[0,0],[7,1],[5,3],[1,2],[3,6],[2,1],[6,3]]$
D is $[64,9,36]$ and $D^{\perp}$ is $[64,55,4]$.

iii) $U = [[7,1],[5,3],[5,0],[7,0],[0,1],[0,3],[1,5],[3,7],[2,1],[6,3],[6,2],[2,6]]$.
D is $[64,12,30]$ and $D^{\perp}$ is $[64,52,4]$.

iv) $U = [[0,0],[4,0],[0,4],[4,4],[5,0],[7,0],[0,1],[0,3],[1,1],[3,3],[2,1],[6,3],[3,1],[1,3],[4,1],[4,3],[5,1],[7,3],[6,1],[2,3],[1,2],[3,6],[2,2],[6,6],[3,2],[1,6],[4,2],[4,6],[5,2],[7,6],[6,2],[2,6],[7,2],[5,6],[1,4],[3,4],[2,4],[6,4],[0,5],[0,7],[5,4],[7,4],[1,5],[3,7],[2,5],[6,7],[3,5],[1,7],[7,5],[5,7]]$
D is $[64,50,5]$ and $D^{\perp}$is $[64,14,27]$. Consider $U' = U \cup \{[1,0],[6,0],[6,5],[7,7],[4,7]\}$ the new GT code $C_{U'}$ has parameters $[64,55,4]$ where the minimum distance drops by 1 but $D_U = D_{U'}$.

In all the examples the code D has the same correction capability to the best known codes for a fixed length and dimension.

5.7 Conclusion

This chapter presents the construction of subfield-subcodes of GT codes over $\mathbb{F}_p$. We first discussed about the subfield subcodes of RS codes i.e. BCH codes and how it can be defined as evaluation codes. We extended it to more generic scenario by constructing the subfield subcodes of GT codes. The subfield subcodes of GT codes are the multidimensional analogues of BCH codes. We defined the subfield subcodes of GT codes as evaluation codes, identified the evaluation polynomials and determined the dimensions and obtain bounds for the minimum distance. Several examples of binary and ternary subfield-subcodes of GT codes are given that are the best known codes of a given dimension and length. This also gave us a new method to construct the best known codes for a given length and minimum distance.

From this chapter, we know that the polynomial evaluating to $\mathbb{F}_p$ should have support in cyclotomic cosets. Based on this fact, we make an attempt to simplify the existing decoding algorithms for BCH codes which can be seen as subfield subcodes of RS codes. In the next chapter, we will demonstrate the usage of cyclotomic cosets for improving the already existing decoding algorithms of BCH codes.

Chapter 6

Efficient decoding of BCH codes beyond the error correction capability of the code

BCH codes are the subfield subcodes of RS codes and form an interesting sub-class of cyclic codes as discussed in chapter 5. The BCH code with designed minimum distance d_{BCH} which is same as the BCH bound can be constructed with generator polynomial $g(x) = (x - \alpha^b)(x - \alpha^{b+1}) \ldots (x - \alpha^{b+d_{BCH}-2})$. Here $g(x)$ is a minimal degree polynomial in $\mathbb{F}_q[x]$ such that $g(\alpha) = \ldots = g(\alpha^{b+d_{BCH}-2}) = 0$. The BCH bound or designed minimum distance d_{BCH} is only a bound on minimum distance d_{min}, such that $d_{min} \geq d_{BCH}$ [91]. While constructing an arbitrary cyclic code, an exhaustive computer search is often needed to find the actual minimum weight of the codewords and thereby the minimum distance. Classical algorithms for decoding BCH codes (Berlekamp-Massey, Extended Euclidean algorithm,...) can decode up to the half of the designed minimum distance, d_{BCH}. If the true minimum distance, lets say d' of a BCH code is greater than d_{BCH}, it implies we are not using the whole capacity of the code with the above mentioned algorithms. Some algebraic algorithms which decode beyond the half of the BCH bound have been proposed by various researchers. In [92] Rifa J. can correct one more error if d_{BCH} is even. In [93] Gui-Liang F. and Tzeng, K. can decode cyclic codes up to the Hartmann-Tzeng bound and Roos bound. Bours *et al* in [94] and Duursma I. together with Kötter R. in [95] provide different algorithms for decoding binary cyclic codes beyond the half of the BCH bound. In [96] Nilsson J. computes all the error patterns of weight at most $d' - 1/2$.

The list decoding algorithm is a relaxed decoding over unique decoding and it decodes beyond half the minimum distance bound. The motivation behind this chapter is to use the subfield subcode and cyclotomic cosets concept for simplifying the list decoding algorithms of BCH codes. We first find a list decoding technique for BCH codes by utilizing the already existing list decoding technique of RS codes. The advantage to use RS list decoder is that it may be able to decode BCH codes uniquely even beyond half the minimum distance. Pellikaan, R. already *et al.* used this idea for decoding Reed-Muller codes [97]. This is supported by the fact that the subfield-subcodes of RS codes over $\mathbb{F}_{p^m}^n$ is a primitive narrow sense BCH code (section 2.4.1) over $\mathbb{F}_{p^s}^n$ such that $s|m$ (converse is also true) as mentioned in chapter 5. The idea here is to use the list decoder (LD) for RS codes for either unique decoding or list decoding of BCH codes depending upon the inequality that exists between the actual minimum distance of BCH codes and list error correction capability of RS codes. In most of the cases it can uniquely decode up to the actual minimum distance of BCH code. We present a probabilistic analysis for having more than one BCH code in the list when we use RS LD for list decoding of BCH codes. Also, with the use of algebraic approach to BCH codes provided in [87], we improve the Roth-Ruckenstein (RR) [98] [76] factorization step in the list decoding algorithm.

In addition to improving the RR factorization algorithm for BCH codes, the list decoding algorithm via syndrome variety is also modified and simplified by reducing the number of variables in the system of equations. The problem of decoding cyclic codes can be written as an algebraic system of equations where the solutions are closely related to the error that occurred. The idea has been explored in many of the previous papers where researchers have shown that the computation of the Gröbner basis of this system of equations enables us to decode beyond the true minimum distance. Because of the large number of variables, the complexity of previously discussed algorithms are quite high and it becomes difficult to compute the Gröbner basis even for the codes of small length. The idea here is to propose a computationally efficient list decoding algorithm for BCH codes. From the subfield-subcode perspective, the polynomial evaluating to the codeword in the subfield, i.e. evaluation polynomial for BCH codes, has some special algebraic properties based on its cyclotomic cosets structure as mentioned in Section 5.4 by putting $r = 1$. These algebraic properties can be used to improve the list decoding via syndrome variety. We show an example of $[63, 24, 15]$ BCH code where total number of variables has been reduced by 73%.

6.1 Introduction

The list decoding algorithm was introduced by Elias [14] and Wozencraft [15]. In 1997, Madhu Sudan [99] discovered a polynomial time algorithm for decoding low rate RS codes beyond the classical $\frac{d}{2}$ bound. Later, Guruswami *et al.* [100] discovered a significantly improved version of the list decoder which can correct codes of any rates. The list decoder is a relaxation over unique decoding that allows the decoder to produce a list of codewords as answers. It can uniquely decode beyond half of the minimum distance in some cases or else produces a list of codewords. The list decoding problem for a code $C : \mathbb{F}_q^k \longrightarrow \mathbb{F}_q^n$ is defined as : Given a received word $r \in \mathbb{F}_q^n$, find a list of the messages say L such that the Hamming distance between r and the elements of L is at most τ, where τ is the number of errors we are trying to correct using the list decoder. If $\tau = \lfloor \frac{d-1}{2} \rfloor$, it will result in unique decoding. The number of errors we can decode with a list decoder depends on parameters like list size and multiplicity of its zeros [100]. The list decoding algorithm is a polynomial time algorithm for decoding up to t_{GS} errors where, $t_{GS} = \tau_{max} = n - 1 - \lfloor \sqrt{(k-1)n} \rfloor$ that means t_{GS} is the maximum limit which could be achieved with a quite high multiplicity and list size depending upon the various cases. Generally, for the simulation processes we use a multiplicity of 2.

Also, the list decoding algorithm via syndrome variety for binary cyclic codes is discussed in [101] using an algebraic system of equations and computing the Gröbner basis of this algebraic system. Using the algebraic system of equations, an ideal is generated from the system of equations and a syndrome variety [102] associated to it which is closely related to the error occurred. The proposed approach is similar to [101], but the main novelty here is to reduce the number of variables in the system of equations which in turn makes the computation of the Gröbner basis much more easier. Like [101], the Gröbner basis technique is used to compute the solutions (instead of using brute force which is an exhaustive method) with the significantly reduced number of variables in our case. This decreases the complexity of the computation of the Gröbner basis.

6.2 List Decoder of RS codes for decoding BCH codes

The list decoding error correction capability is much more than that of actual error correction capability. As we have already stated that the $d' = d_{min}(D) \geq d = d_{min}(C)$, we can apply a list decoder for C in order to decode D which is the subfield subcode of C over $\mathbb{F}_{p^s}$. If we apply our list decoder to decode C we can decode maximum up to t_{GS}. We might get just one codeword or a list of codewords depending upon the number of errors and the distribution of codes. Now, if we apply the same list decoder to the subfield subcode D it would be the same. Thus our goal is to find out a bound on the error correction capability of list decoder applied to D such that in the list of final output codewords there is only one element having its coordinates in the subfield ($\mathbb{F}_{p^s}$). In [97], they have used the same idea in order to decode Reed-Muller codes.

There are two possible cases:

1. $d = d'$: For this case, a unique RS decoder can be used for decoding subfield subcodes.

2. $d' > d$: In this case the list decoder of C, when used for decoding of D, will give only one codeword in subfield under the condition which has been discussed later in Theorem 6.2.3. There can be two possible interesting sub-cases which are discussed below:

   (a) $\lfloor \frac{d-1}{2} \rfloor \leq \lfloor \frac{d'-1}{2} \rfloor \leq t_{GS}$: Here, the list decoder error correction capability is more than actual error correction capability of BCH codes. This case is discussed in this section.

   (b) $\lfloor \frac{d-1}{2} \rfloor < t_{GS} \leq \lfloor \frac{d'-1}{2} \rfloor$: In this case, the actual error correction capability of BCH code is more than the list decoder error correction capability. This case has been discussed in section 5.

Definition 6.2.1. *Let D be a $[n, k]$ cyclic code over $\mathbb{F}_{p^s}$ with generator polynomial $g(x) \in \mathbb{F}_p[x]$. Let $\mathbb{F}_{p^m}$ be the decomposition field of $g(x)$. We define the BCH bound and we denote it by d_{BCH} to [103]:*

$$d_{BCH} = max\{l \mid g(\alpha) = \cdots = g(\alpha^{l-1}) = 0 : \alpha \in \mathbb{F}_{p^m}\}.$$

If D is a $[n, k', d']$ primitive narrow sense BCH code it is well known [11] that $D = C_1 \cap \mathbb{F}_{p^s}^n = C_2 \cap \mathbb{F}_{p^s}^n = \cdots = C_x \cap \mathbb{F}_{p^s}^n$ where $C_1, \ldots, C_x$ are RS codes with parameters $[n, k_i, d_i]$ for $i = 1, \ldots, x$. Assume without lose of generality that $d_1 > d_2 > \cdots > d_x$. Since D is a subcode of all of them $d' \geq d_1$. From now on, if $D = C \cap \mathbb{F}_{p^s}^n$, we are assuming that $C = C_1$, $k = k_1$ and $d = d_1$. A natural question is to ask about the relation between d', d and the BCH bound d_{BCH}.

Let C be a RS code with $d_{min}(C) = d$ and $D = C \cap \mathbb{F}_{p^s}$ with $d_{min}(D) = d'$, then $d' \geq d_{BCH} \geq d$, where d_{BCH} is the BCH bound for D.

Example 6.2.2. *Consider a RS code C with parameters $[127, 99, 29]$ over $\mathbb{F}_{2^7}$. The subfield subcode D with parameters $[127, 43, 31]$ [104] over $\mathbb{F}_2$ has generator polynomial,* $g(x) = x^{84} + x^{83} + x^{80} + x^{79} + x^{77} + x^{72} + x^{70} + x^{69} + x^{65} + x^{64} + x^{59} + x^{57} + x^{53} + x^{51} + x^{50} + x^{49} + x^{45} + x^{43} + x^{42} + x^{41} + x^{35} + x^{34} + x^{32} + x^{28} + x^{26} + x^{25} + x^{22} + x^{21} + x^{19} + x^{18} + x^{17} + x^{16} + x^{11} + x^{10} + x^{6} + x^{4} + x^{3} + x^{2} + 1.$

$g(x)$ is completely decomposable over $\mathbb{F}_{2^{14}}$. For calculating the BCH bound, we need to know the maximum number of consecutive roots that occurred. Two big sets of consecutive roots are $\{a, a^2, ..., a^{28}\}$ and $\{a^{32}, ..., a^{38}\}$, where $a = \epsilon^{129}$ and ϵ is the primitive element of $\mathbb{F}_{2^{14}}$. Thus, the maximum number of consecutive roots are 28 in this case. It clearly implies that the generator polynomial of D has 28 consecutive roots i.e. $d_{BCH} = 28 + 1 = 29$ by definition 6.2.1. This is the first clear and interesting example where $d' > d_{BCH} = d$.

Theorem 6.2.3. *Let C be a RS code with parameters $[n, k, d] \in \mathbb{F}_{p^m}^n$ and D a subfield subcode with parameters $[n, k', d'] \in \mathbb{F}_{p^s}^n$. Assume we receive a word $r \in \mathbb{F}_{p^s}$, where $r = c' + e$ such that $c' \in D$ and $wt(e) \leq min\{t_{GS}, \lfloor \frac{d'-1}{2} \rfloor\}$. Applying a list decoder for RS code C (with list error correction capability of t_{GS}) for decoding its subfield subcode D, can uniquely decode up to the $\tau = min\{t_{GS}, \lfloor \frac{d'-1}{2} \rfloor\}$.*

Proof. Since $r = c' + e$ with $c' \in D$ and $wt(e) \leq \tau = min\{t_{GS}, \lfloor \frac{d'-1}{2} \rfloor\}$ we have that $c' \in B(r, \tau) = \{u \in \mathbb{F}_{p^m}^n \mid d(r, u) \leq \tau\}$. Since, τ is the minimum of list error correction capability and half of the actual minimum distance (d'), therefore we can easily assume that $\tau <= t_{GS}$ for all the cases. Therefore, we can guarantee that the RS list decoding algorithm for C applied to r always contains the right codeword as list of the codewords always contain the actual transmitted word if number of errors are less than list error correction capability.

For proving that list will contain only one word that is equivalent to uniue decoding. Suppose that the list contains two codewords c_i and c_j such that they have coordinates in the subfield. Since, τ is the minimum of list error correction capability and half of the actual minimum distance (d'), therefore we can easily assume that $\tau <= \frac{d'1}{2}$ for all the cases. This clearly states that with in distance d' of received word there exists two codewords. But, due to linear property of code, this clearly implies that the minimum distance between two codewords in D is $< d'$, which is a contradiction. Thus, there is only one codeword within distance d' having coordinates in $\mathbb{F}_{p^s}$.

$\square$

Example 6.2.4. *Consider a RS code C with parameters $[127, 99, 28]$. The subfield subcode is D with parameters $[127, 43, 31]$ [104] while d_{BCH} is 29. Since the BCH bound is 29, the BCH decoder can correct up to 14 errors but the true minimum distance of BCH code is 31 which implies we can correct 15 errors. If we apply list decoder of C, having error correction capability $t_{GS} = n - 1 - \sqrt{n(k-1)} = 127 - 1 - \sqrt{127 * 98} = 15$, to D we should be able to decode up to the true minimum distance of D theoretically with the multiplicity 15 of the list decoder.*

In the previous theorem, we stated that a list decoder for C can decode D up to $\tau = min\{t_{GS}, \lfloor \frac{d'-1}{2} \rfloor\}$. If $\tau = \lfloor \frac{d'-1}{2} \rfloor$, then we will always be able to uniquely decode up to the true minimum distance of BCH codes. If this is not the case we can not use list decoder for the unique decoding. So the most interesting case here is when $\lfloor \frac{d'-1}{2} \rfloor \leq t_{GS}$.

The second interesting case is when $\lfloor \frac{d'-1}{2} \rfloor > t_{GS}$. In this case the list decoder for RS code will act as a list decoder for our narrow sense primitive BCH codes as well instead of unique decoder. So, in this case there is a possibility that we will get two codewords in the subfield. But, then the question will be why to use the list decoder for RS code to decode BCH when there is already a list decoder for BCH codes present. This could be answered in terms of complexity since the complexity for RS is $O(n^6(1-\sqrt{1-d})^8)$ while for narrow sense BCH it is $O(n^6(1-\sqrt{1-2d})^8)$ with reference to [105]. Actually, recent results by Lee-O'Sullivan [106] and Beelen-Brander [107] compute RS list decoders which run in $O(l^4 sn^2)$ and$O(s^4 l^4 n \; log^2 n \; log(log \; n))$ respectively, where l denotes the designed list-size and s the multiplicity parameter. We can clearly see that there is a bit of advantage here in terms of complexity which makes this case interesting to study. But, the list error correction capability for BCH code is $\frac{n}{2}(1-\sqrt{(1-2D)})$ while for the RS case is $n(1-\sqrt{(1-D)})$, where $D \simeq \frac{d}{n}$. The list error correction capability of BCH code is always a bit larger than list error correction capability of RS codes. Here, we gained a bit in terms of the complexity but we are paying in terms of the list error correction capability.

6.2.1 Algorithm

In this section, we will discuss about the list decoding algorithm for RS codes in order to decode BCH codes. The algorithm actually uses a RS unique decoder for the case $d' = d$ and list decoder for cases $d' > d$. Since, BCH codes are subfield subcodes of RS codes, so we only output the list of those codewords with coordinates lying in the subfield and all the other codewords will be removed from the list.

Algorithm 8 Unique Decoding ($\lfloor \frac{d'-1}{2} \rfloor < t_{GS}$)/List Decoding ($\lfloor \frac{d'-1}{2} \rfloor > t_{GS}$) for BCH codes using RS list decoder

```
Input: $n = p^m - 1$, multiplicity($s$), received word($\mathbf{r}$) = $(r_1, r_2, ..., r_{n-1}) \in \mathbb{F}_{p^s}^n$, Degree($k$)
1: Compute the parameter $t_{GS} = n - 1 - \sqrt{n(k-1)}$, $t = \lfloor \frac{d-1}{2} \rfloor$ and $t' = \lfloor \frac{d'-1}{2} \rfloor$.
2: Compare $t$ and $t'$.
3: if $t' = t$ then
4:    Apply the unique decoder of RS code for decoding.
5: else
6:    if $t' > t$ then
7:       Apply the list decoding algorithm for RS codes to obtain a list of codewords $L$.
8:    end if
9: end if
10: Compute $L' = L \cap \mathbb{F}_{p^s}$.
11: return ($L'$)
```

6.3 Probability of multiple candidates in the list

If the list returned contains more than one BCH word, it might be a problem for a number of applications. In this section, we do a probabilistic study for having more than one BCH codeword in the list. Technically it is a list decoder but it is highly unlikely that the list contains more than one codeword.

For calculating the actual probability of having more than one codeword we will be following the notations and results from [108]. Let A_u be $|\{c \in D | wt(c) = u\}|$, where $wt(c)$ is the weight of u. The weight distribution of BCH codes can be calculated with the help of MAGMA. Let $S(w, r)$ be the sphere $\{u \in \mathbb{F}_{p^s}^n | d(w, u) \leq r\}$ and $S_=(w, r)$ be the surface of sphere, i.e., $S_=(w, r) = \{u \in \mathbb{F}_{p^s}^n | d(w, u) = r\}$. Suppose that w is a received word and D with parameters $[n, k', d']$ is in use. Then decoding up to τ errors from w can be specified as calculating the following set $dec_\tau(w) = S(w, \tau) \cap D$.

Assuming that all error vectors with weight at most τ have the same probability of occurrence regardless of the weight of the error vector. Let $v \in D$ be the transmitted word and at most τ errors occured in $\mathbb{F}_{p^s}$. If w is the received word then $P(|dec_\tau(w)| > 1)$ will be given by the fraction of words in $S(v, \tau)$ which are also within distance τ from another codeword. Since the code is linear therefore studying the case $v = 0$ will be sufficient. The exact count on the number of words in $S(0, \tau)$ within distance τ from another codeword is very difficult so we give an upper bound i.e. $M(\tau) = \sum_{c \in D \setminus \{0\}} |S(0, \tau) \cap S(c, \tau)|$. The formula for $M(\tau)$ can be directly given as:

$$M(\tau) = \sum_{u=d}^{min\{2\tau, n\}} A_u \sum_{a=u-\tau}^{\tau} \sum_{i=u-a}^{\tau} |S_=(0, a) \cap S_=(c, i)| \tag{6.1}$$

$$|S_=(0, a) \cap S_=(c, i)| = \sum_{j=0}^{l} \binom{u}{u-a+j} \binom{n-u}{j} (p^s-1)^j \binom{a-j}{i-(u-a)-2j} (p^s-2)^{i-(u-a)-2j} \tag{6.2}$$

$$l = min\{\lfloor \frac{i-(u-a)}{2} \rfloor, n-u\} \tag{6.3}$$

Thus the probability that we will obtain more than one word in the list can be given by:

$$P(|dec_\tau(w)| > 1) \leq \frac{M(\tau)}{S|0, \tau|}.$$

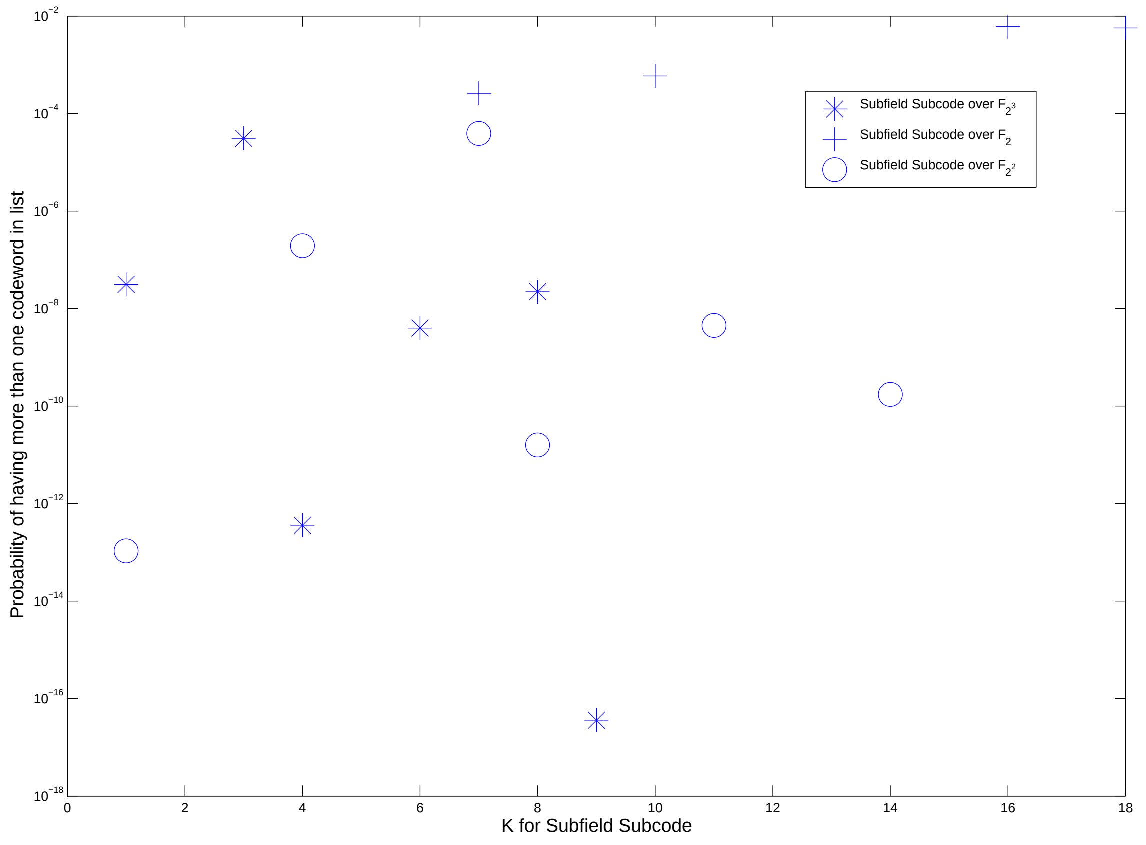

Figure 6.1: Upper bound on probability of having more than one candidate in the list for the subfield subcodes of parent code over $\mathbb{F}_{2^6}$

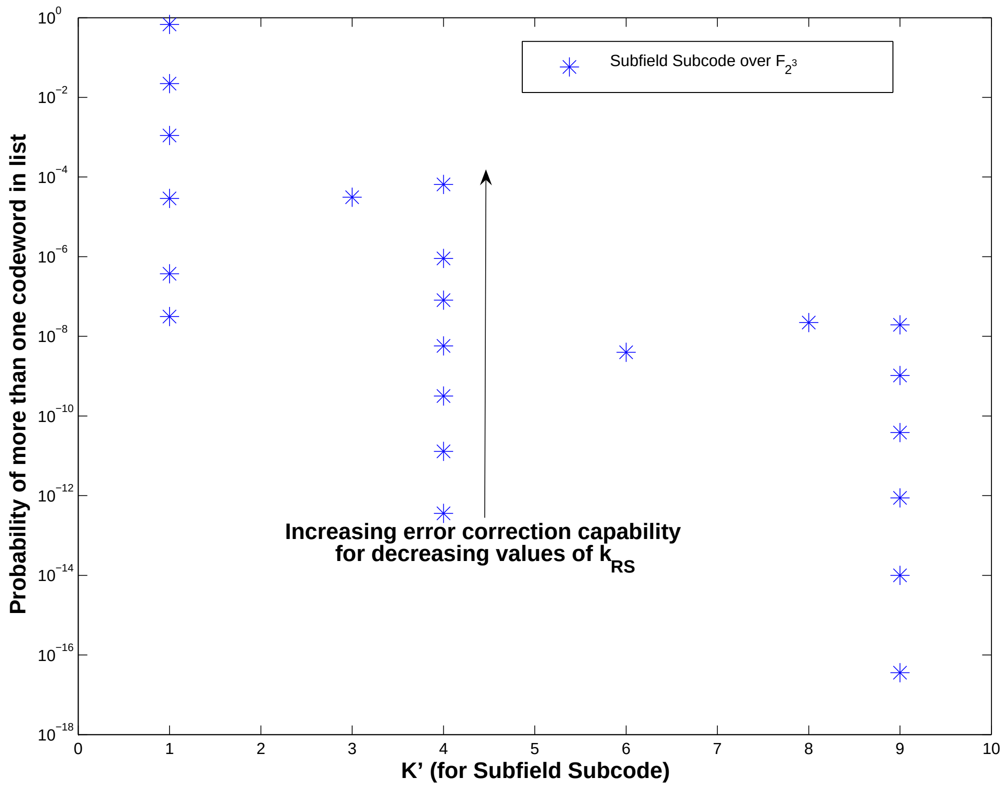

Figure 6.2: Probability of having more than one codeword for subfield subcodes over $\mathbb{F}_{2^3}$

Remark 6.3.1. *We have computed the probability of having more than one BCH codeword in the list in the following case: we took the RS code over field $\mathbb{F}_{2^6}$ and the subfield subcodes over $\mathbb{F}_2$, $\mathbb{F}_{2^2}$ and $\mathbb{F}_{2^3}$. As shown in figure 6.1, the probability of having more than one codeword in the list are quite small for the codes over subfield $\mathbb{F}_{2^3}$ while it is comparatively larger over subfield $\mathbb{F}_2$. But, we can also see that the probability for having more than once codeword in the list is really small for all the cases which is the main goal of this plot. This means there is a very low probability of having more than one codeword up to list decoder error correction capability in this case. Thus, it makes the list decoder of RS code worth using for BCH decoding beyond the designed minimum distance. Also, while calculating the probabilities we had more than one value of probability for same subfield subcodes. This is because for the same subfield subcode, we have more than one RS code as parent code, as discussed in chapter* 5*. For this plot, we have chosen the minimum of all the probabilities.*

In Figure 6.2, we have plotted the probability of having more than one codeword against k' but for all the values of k. Since, in our technique we are using the RS list decoder for BCH list decoding which implies that for calculating the value of τ we are using k of RS code. But, as discussed in earlier, we have same k' for several values of k. So, we can see in the figure that correcting more errors result in higher probability. This is because of the limitation of the list decoder. There is a trade off between number of errors corrected and the possibility of having more than one codeword in the list and vice-versa. Correcting a higher number of errors, will result in a higher number of codewords in the list.

Also, in Table 6.1, we have shown how the probability is decreasing with the decreasing error correction capability or increasing k. The minimum distance of BCH code in this case is 54 which allows us to correct 26 errors with a unique decoder. Now, we can see from the table that we can correct around 5 more errors with a very small probability of having more than one codeword in the list.

Table 6.1: Probability values for varying $k = 10, ..., 16$ and constant $k' = 4$

k for RS code	k' for BCH	τ for list decoder of RS	d of RS	P
10	4	38	54	$6.5 * 10^{-5}$
11	4	36	53	$9.06 * 10^{-7}$
12	4	35	52	$8.07 * 10^{-8}$
13	4	34	51	$5.76 * 10^{-9}$
14	4	33	50	$3.18 * 10^{-10}$
15	4	32	49	$1.29 * 10^{-11}$
16	4	31	48	$3.6 * 10^{-13}$

6.4 Introduction to Roth Ruckenstien factorization for RS codes

We are interested in providing a modified version of the Roth Ruckenstein (RR) algorithm focusing it into the BCH case. So we are going to describe along this section the main results represented in [76].

Given a RS codeword $c \in C$ with parameters $[n, k, d]$ is transmitted and a word r is received. Let $F_{p^m}[x]$ be the ring of polynomials in x and $F_{p^m}[x, y]$ be the ring of polynomials in x and y. Given the received word r_i and the point on the field α_i, the interpolation step consturts a two variable polynomial $Q(x, y)$. $Q(x, y)$ has a zero of certain multiplicity at each of the points (r_i, α_i) and has minimum weighted degree wrt $(1, k-1)$. After applying the interpolation algorithm, we obtain a bivariate polynomial $Q(x, y) \in F_{p^m}[x, y]$. Given $Q(x, y)$, the factorization step is to find all polynomials in $F_{p^m}[x]$ of degree $\leq k - 1$ such that $(y - f(x))|Q(x, y)$. Alternatively, we can say that $Q(x, f(x)) = 0$ which makes $f(x)$ as $y-$root of $Q(x, y)$. We will discuss the RR algorithm which is a method to find the $y-$roots.

If $Q(x, y)$ is a bivariate polynomial such that $x^m|Q(x, y)$, but $x^{m+1} \nmid Q(x, y)$, then define

$$\ll Q(x, y) \gg = \frac{Q(x, y)}{x^m}$$

Suppose $f(x) = a_0 + a_1 x + ... + a_{k-1}x^{k-1}$ is a $y-$ root then we can pick the coefficients one by one. Initially we will see how to determine a_0. The results below are taken from [76].

Lemma 6.4.1. *If $y-f(x)|Q(x,y)$ then $y = f(0) = a_0$ is a root of equation $Q_0(0,y) = 0$, where $Q_0(x,y) = \ll Q(x,y) \gg$.*

Now, proceeding by induction, we define three sequences of polynomials $f_j(x), T_j(x,y)$ and $Q_j(x,y)$, for $j = 1, ..., k-1$ as follows: Initially, $f_0(x) = f(x)$, $Q_0(x,y) = \ll Q(x,y) \gg$. For $j \geq 1$ define

$$f_j(x) = \frac{f_{j-1}(x) - f_{j-1}(0)}{x} = a_j + ... + a_{k-1}x^{k-1-j} \tag{6.4}$$

$$T_j(x,y) = Q_{j-1}(x, xy + a_{j-1}) \tag{6.5}$$

$$Q_j(x,y) = \ll T_j(x,y) \gg \tag{6.6}$$

We are stating the main results from [76] without making the proofs.

Theorem 6.4.2. *Given $f(x) = a_0 + ... + a_{k-1}x^{k-1} \in F_{p^m}[x]$ and $Q(x,y) \in F_{p^m}[x,y]$, define the sequences for $f_j(x)$ and $Q_j(x,y)$ as defined above. Then for $j \geq 1$, $(y - f(x))|Q(x,y)$ if and only if $(y - f_j(x))|Q_j(x,y)$.*

Corollary 6.4.3. *If $(y-f(x))|Q(x,y)$ then $y = a_j$ is a root of the equation $Q_j(0,y) = 0$, for $j = 0, ..., k-1$.*

Corollary 6.4.4. *If $y|Q_k(x,y)$, i.e., if $Q_k(x,0) = 0$, then $f(x) = a_0 + a_1x + ... + a_{k-1}x^{k-1}$ is a $y-$ root of $Q(x,y)$.*

6.5 Improved Roth Ruckenstein (RR) factorization algorithm for subfield subcodes of RS codes

A list decoding algorithm has two important steps: interpolation and factorization. The motivation behind this section is to simplify the factorization step using algebraic properties of the polynomial generators described in the chapter 5. We have made some improvements on the stopping rule by using the fact that the polynomials evaluating to $\mathbb{F}_p$ should have their support lying in cyclotomic coset. Using this property, we have improved the stopping rule for factorization algorithms. The details will be discussed in this section.

The result from interpolation is a bivariate polynomial $Q(x, y)$ which can be factorized as $(y - f_1(x))...(y - f_L(x))$. The polynomials $f_i(x)$ should be of the form $\sum_{i=1}^{k-1} a_i x^i$. As we have seen in Theorem 5.4.8 if $f(x)$ is an evaluation polynomial for a BCH code then $f(x) = \sum_{b \in J} f_{I_b,\beta}$. We can alter the algorithm by introducing a few changes that will reduce the complexity of the factorization part. They are discussed as follows.

1. The input of the algorithm is a bivariate polynomial $Q(x, y)$. The output is the set of polynomials $f(x) \in \mathbb{F}_{p^m}[x]_{\leq k-1}$ (polynomials of degree $\leq k-1$ in $\mathbb{F}[x]$) with $Q(x, f(x)) = 0$ and support in $\mathcal{B} = \{I_{b_1}, \ldots, I_{b_l}\}$ as indicated by Theorem 5.4.6.

   A typical factor of $Q(x, y)$ is of the form $y - f(x)$ where $f(x) = \sum_{i=0}^{k-1} a_i x^i$.The task of the factorization algorithm is to find out the coefficients $a_0, ..., a_{k-1}$.

2. Following Lemma 6.4.1 the candidates to be a_0 are the roots of $Q(0, y)$, therefore we consider $A_0 = RootsOf(Q(0, y))$. Since *supp*$(f(x)) \subseteq \mathcal{B} = \{I_{b_1}, \ldots, I_{b_l}\}$ then $a_0 \in \mathbb{F}_p$, thus we compute $A_0 = A_0 \cap \mathbb{F}_p$.

3. For each element $a_0 \in A_0$ we compute: $A_0 = A_0 \setminus \{a_0\}$ (remove a_0 from set A_0), $T_1(x, y) = Q(x, xy + a_0)$ and $Q_1(x, y) = \ll T_1(x, y) \gg$.

4. From Corollary 6.4.3 we know that the candidates to be a_1 are the roots of $Q_1(0, y)$. So we compute $A_1 = RootsOf(Q_1(0, y))$. No special condition is imposed over A_1.

5. Assume we have already computed $a_0, a_1, \ldots, a_{i-1}$ coefficients of $f(x)$. We compute $T_i(x, y) = Q_{i-1}(x, xy + a_{i-1})$ and $Q_i(x, y) = \ll T_i(x, y) \gg$.

6. From Corollary 6.4.3 we know that the candidates to be a_i are the roots of $Q_i(0, y)$. So we compute $A_i = RootsOf(Q_1(0, y))$. Since *supp*$(f(x)) \subseteq \mathcal{B} = \{I_{b_1}, \ldots, I_{b_l}\}$ then $a_i \in I_{i/p}$ if $i/p \in \mathbb{Z}_{\geq 1}$, i.e., $a_i = a_{i/p}^p$.

7. If $i/p \in \mathbb{Z}_{\geq 1}$ consider the unique element $a_i \in A_i$ verifying that $a_i = a_{i/p}^p$. If there exists such a_i we go as in step 5. with $\{a_0, \ldots, a_i\}$, otherwise we break the process for this particular i and go to the next element in A_j with $j = max\{k : k < i, k/p \notin \mathbb{Z}_{\geq 1}, A_k \neq \emptyset\}$.

8. If $i/p \notin \mathbb{Z}_{\geq 1}$, for each $a_i \in A_i$ compute $A_i = A_i \setminus \{a_i\}$ (remove a_i from set A_i) and proceed as in step 5. with $a_0, \ldots, a_i$.

9. We stop this process when we have computed a_m, with $m = max\{b_1, \ldots, b_l\}$ (assuming b_i is the leader of I_{b_i}). All other coefficients a_j with $m < j \leq k-1$ are computed as p-powers of $a_0, \ldots, a_m$, that is, consider $A = \{a_j : j \leq m, j/p \in \mathbb{Z}_{\geq 1}\}$ then $f(x) = \sum_{a \in A} f_{I,a}$.

10. If $y - f(x) \mid Q(x, y)$ return $f(x)$ otherwise go to the next element in A_j with $j = max\{k : k < i, k/p \notin \mathbb{Z}_{\geq 1}, A_k \neq \emptyset\}$.

We have modified the factorization algorithm for RS codes in order to get the factorization algorithm for subfield subcodes of RS codes as shown in Algorithm 9.

6.6 Evaluation based decoding of BCH codes

From theorem 5.4.6, it is clear that any BCH code can be generated by some of the elements in A. Let us consider the following family of codes:

$$\mathscr{C} = \{D =< Ev(S) >: S \in A\}$$

Our goal is to perform list decoding for this family of codes in $\mathscr{C}$. Notice that from the definition, we are interested in BCH codes or in a subcode of them. Let $D \in \mathscr{C}$ with parameters $[n, k', d']$ and $t = \lfloor \frac{d_{BCH}-1}{2} \rfloor$, then D is a subfield subcode of RS code with parameters $[n, k, d]$. Let us assume that we receive $r = c + e$, with $wt(e) = \tau$. From our interpretation of D in the previous chapter, there is a polynomial F with $deg(F) \leq k - 1$ such that $Ev(F) = c$. Since F_p^n is isomorphic to the polynomials of degree less than or equal to $n - 1$ with support in the cyclotomic cosets, there exist other polynomials, say E and R such that $Ev(E) = e$ and $Ev(R) = r$.

We can write $E(x) = E_0 + E_1x + \ldots + E_{n-1}x^{n-1}$, $F(x) = F_0 + F_1x + \ldots + F_{n-1}x^{n-1}$ and $R(x) = R_0 + R_1x + \ldots + R_{n-1}x^{n-1}$. We are dealing with the dual perspective of evaluation scheme here. The syndromes for the codeword S^c is again the evaluation of the codeword polynomial c in $\mathbb{F}_{p^m}^*$ which will actually give the coefficients of F in reverse order i.e. $S_1^c = F_{n-1}, \ldots, S_n^c = F_0$. Hence, it is clear that if we will be able to compute E, then F can be computed automatically as $F = R - E$ and therefore will be able to decode correctly.

Algorithm 9 Factorization algorithm for subfield subcodes of RS codes

Input: $Q(x,y)$ from interpolation, Degree(k)

Output: $\{f(x) : (y - f(x))|Q(x,y); degf(x) \leq k-1\}$

```
 1: Find a set of cyclotomic cosets, $\widehat{B} = \{I_{b_i} : I_{b_i} \subset \{1,...,k-1\}\}$
 2: Find the coset leaders of $\widehat{B}$ i.e. $\widehat{J} = \{b_i, b_j, ..., b_m\}$
 3: $\pi[0]$ = NIL; deg[0] = -1; $Q_0(x,y) = Q(x,y)$ ; $t = 1$; $u = 0$;
 4: DFS[$u$]
    /* DFS[$u$] : Depth First Search beginning at $u$ */
 5: if $deg(u) \leq b_m$ then
 6:   $R$ = RootList[$Q_u(0,y)$]
 7:   for $\alpha \in R$ do
 8:     if $u = 0$ && $\alpha \in \mathbb{F}_p$ then
 9:       $v = t$; $t = t + 1$
10:       $\pi[v]$= $u$; $deg[v] = deg[u] + 1$; Coeff[$v$] = $\alpha$;
11:       $Q_v(x,y) = \ll Q_u(x, xy + \alpha) \gg$
12:       DFS[$v$]
13:     else if $\alpha$! = Coeff[$u/p$]$^p$ then
14:       Break;
15:     else
16:       $v = t$; $t = t + 1$
17:       $\pi[v]$= $u$; $deg[v] = deg[u] + 1$; Coeff[$v$] = $\alpha$;
18:       if $u \in \widehat{J}$ then
19:         Coeff[$u$] = $P[r]$;
20:         deg[$u$] = $Q[r]$; /*$P$ and $Q$ are storing the coeffs and power final ev poly.*/
21:         $r := r + 1$;
22:       end if
23:       $Q_v(x,y) = \ll Q_u(x, xy + \alpha) \gg$
24:       DFS[$v$]
25:     end if
26:     $f(x)$ = Coeff[0] $+ \sum_{i \in Q[r]} f_{I_i, P[i]}$
27:     if $(y - f(x))|Q(x,y)$ then
28:       Output $f(x)$
29:     end if
30:   end for
```

The standard method to proceed (refer to [50]) is to assume that ν errors occurred in the positions $i_1, i_2, \ldots, i_\nu$. such that $\nu \leq t$. The error locator polynomial can be defined then as:

$$\Delta(x) = \prod_{j=1}^{\nu}(1 - x\alpha^{i_j}) = 1 + \Delta_1 x + \ldots + \Delta_\nu x^\nu$$

The coefficients of this polynomial are related to the error syndrome S^e with the following recurrent equation:

$$S^e_k = \sum_{j=1}^{\nu} \Delta_j S^e_{k-j} \qquad k = \nu + 1, \ldots, n-1 \tag{6.7}$$

One can actually understand this as an infinity recurrence sequence i.e., for $k = \nu + 1, \ldots, \infty$. By construction of code, $c(\alpha^j) = 0$ for $j = 1, \ldots, 2t$, or in frequency domain terms: the parity frequencies (S^c) have spectral components equal to zero. This implies $S^r_j = S^e_j$ for $j = 1, \ldots, 2t$ and so we have known S^e for the initial $2t$ positions out of n positions. Once all of S^e_j for $j = 2t+1, \ldots, n$ are computed using Equation 6.7 then $S^c = S^r - S^e$ and S^c will give the coefficients of F. An efficient manner to proceed is to compute $\Delta(x)$ using BM algorithm which finds the minimal polynomial (error locator) of a linearly recurrent sequence (syndromes) in an arbitrary field and will give the correct error locator polynomial if less than or equal to t errors occurred. The first $2t$ syndromes error syndromes ($S^e = S^r$) are used to compute $\Delta(x)$ then the error locator polynomial and the known S^e are used to compute the remaining error syndromes.

Remark 6.6.1. *The above defined algorithm for $\nu \leq t$ can be simplified using the properties from theorem 5.4.6. As already mentioned, E and F have their support in cyclotomic cosets. As discussed earlier, we have $S^r_j = S^e_j$ for $j = 1, 2, \ldots, 2t$ positions. Using the cyclotomic coset property we can determine some more S^e_j for free. Consider the cyclotomic cosets $I_{b_1}, I_{b_2}, \ldots, I_{b_z}$ such that $1 \leq b_1, b_2, \ldots, b_z \leq 2t$. Now, if we know $S^e_{b_i}$ with $i = 1, 2, ..., z$ that means we know S^e_j such that $j = b_i, pb_i, ..., p^{n_b - 1}b_i$ which is the whole cyclotomic coset I_{b_i}.*

$$S^e_{p^j b_i} = (S^e_{b_i})^{p^j} \qquad j = 1, \ldots, n_b - 1 \tag{6.8}$$

For determining the rest of the S^e, we can use Equation 6.7. $\Delta(x)$ can be computed using the BM algorithm. Once $\Delta(x)$ is known, the next step is to find the $S^e_{b_{z+1}}$, where

b_{z+1} is the index for the next unknown syndrome which has not been covered previously. The equation can be re-framed as,

$$S^e_{b_{z+1}} = -\sum_{j=1}^{t} A_j S^e_{b_{z+1}-j} \tag{6.9}$$

Since all the S^e_j with index $j < b_{z+1}$ are known, so $S^e_{b_{z+1}}$ can be determined using the above defined equation. Once we computed $S^e_{b_{z+1}}$, we can compute all S^e from the cyclotomic coset $I_{b_{z+1}}$. Now the same process can be repeated with b_{z+2} which is the index for the next unknown syndrome index and we can repeat it until we find all of S^e. The coefficients of the evaluation polynomial can then be calculated as $F(n-j) = S^r_j - S^e_j$.

6.7 List decoding via syndrome variety

Suppose $\nu > t$ errors occurred while transmitting the codeword c. The problem of decoding cyclic codes up to and beyond their true minimum distance can be solved by the use of Gröbner basis. The main concept is to convert the decoding problem in to an algebraic system of equations and use the algebraic tools to solve the problem. We propose a syndrome variety that can be used to decode BCH codes beyond the error correction capability. The advantage gained over the previously defined scheme is reduction in the number of variables using the cyclotomic coset structure.

Considering the fact that $F_j = 0$ for $j = n-1, \ldots, n-2t$ since the evaluation polynomial has degree $\leq k-1$, it follows that $E_j = R_j$ for some of the values of j. We can understand Equation 6.7 as a system of equations (not necessarily linear) where the variables are $(\Delta_1, \ldots, \Delta_\nu, S^e_{2t+1}, \ldots, S^e_{n-1})$. The total number of unknown variables are $\nu + n - 2t$. Let I be the ideal generated by the n equations in (6.7) and $V(I)$ is the syndrome variety associated to it. By construction it should contain the solution to our problem. If $\nu = t$, then it is well known that the syndrome variety will contain only one point which is the solution and it is then bounded distance decoding. This case has already been discussed in a more efficient manner in 6.6.1. For the rest of the section, we will discuss the case when there are more number of errors i.e. $\nu > t$.

Proposition 6.7.1. *If $\nu > t$ then the syndrome variety $V(I)$ provides all the codewords at a distance at most ν to the received word r.*

Proof. The syndrome polynomial can be written as a power series described in [109].

The $m-th$ syndrome is then $S_m^e = \sum_{i=1}^{n-1} e_i\alpha_i^m$ and the syndrome polynomial is defined as $S(x) = 1/x \sum_{m=1}^{\infty} S_m^e x^{-m} = \sum_{i=1}^{\tau} \frac{e_i}{x-\alpha_i}$. In particular, condition 6.7 implies that $S(x)\Delta(x)$ is a polynomial. The latter is true iff $\Delta(x)\sum \frac{e_i}{x-\alpha_i} \in \mathbb{F}_{p^m}[x]$ iff $(x-\alpha^i) \mid \Delta(x)$ for every i with $e_i \neq 0$. Therefore for any codeword at a distance at most τ to the received word there is a Δ satisfying condition 6.7 , which clearly implies it must be a solution in $V(I)$. □

In general, it is difficult to find the solution of the variety in $\nu + n - 2t$ variables as one needs to check $p^{m(\nu+n-2t)}$ points which is the brute force or exhaustive mechanism of doing this. However this is an inefficient way. Like in [101], we can use Gröbner basis, although it is much more efficient than brute force but with a large number of variables it is quite complex. Hence, we will discuss in details a nice concept that can reduce the number of variables thus reducing the complexity of the algorithm as compared to previously discussed algorithms. Still, we are required to compute the Gröbner basis but the advantage is with a reduced number of variables.

In the case of BCH codes, we can apply some nice algebraic properties associated with the evaluation polynomial of the code in order to reduce the number of variables associated with the system of equations. As it has already been discussed, E and F both have support in cyclotomic cosets, i.e. if the coefficient $E_i \neq 0$ the $E_{p^i \ mod \ p^m-1} = E_i^p \ mod \ p^m - 1$. Since, we know E_j for $j = n-1, \dots, n-1-2t$ we also know all the elements involved in their cyclotomic cosets i.e. $E_{p^k j \ mod \ p^m-1}, k = 1, \dots, m-1$. With the unknown variables, we apply the same rule. Let S_i^e is an unknown variable presented by x_i then $S^e_{p^k i \ mod \ p^m-1}$ can be represented in terms of x_i and is equal to $x_i^{p^k}, k = 1, \dots, m-1$. Using the property of cyclotomic cosets the number of variables can be reduced significantly. So, finally we have $\nu + \eta$ variables where η is the number of cyclotomic cosets which are unknowns. Under this setup, the computation of $V(I)$ is feasible. The decoding methodology of a BCH code defined over $\mathbb{F}_p$, beyond the error correction capability is given by the Algorithm 10. The algorithm follows [101] except the part where cyclotomic structure is used to reduce the number of variables.

Algorithm 10 List decoding for BCH codes over $\mathbb{F}_p$

Input: received codeword, $r(x) = c(x) + e(x)$ as a polynomial in $\mathbb{F}_p[x]/x^n - 1$, BCH decoding capability t, our list decoding capacity τ and $GF(p, m)$.

Output: A list containing all the codewords at a distance τ to the received word

From now onwards, R and E are $n-$tuples representing the coefficients of the polynomial we should evaluate in order to get r and e respectively.

```
    BEGIN
1:  if $\tau \leq t$ then
2:      Apply BM algorithm with the necessary modifications mentioned in remark 6.6.1
3:  else
4:      Compute the cyclotomic cosets:$CC := [I_b : b \in B]$.
5:      Compute the syndromes of received word: $S^r := [Evaluate(r, \alpha^i) : i \in [1, \dots, n]]$
6:      Compute the known error syndromes: $S^e[i] = S^r[i] : i \in [1, \dots, 2t]$.
7:      Compute more known error syndromes: $S^e[p^k i \ mod \ p^m - 1] = S^e[i]^{p^k}$, for $i = 1, \dots, 2t$ and $k = 1, \dots, |I_i - 1|$.
8:      Repeat some error syndromes: $S^e[p^m - 1 + i] = S^e[i] : i \in [1, \dots, \tau]$.
9:      Compute the unknown cyclotomic cosets $U = B \setminus (B \cap [1 \dots 2t])$.
10:     Add variables: $S^e[p^k i \ mod \ p^m - 1] = S_i^{p^k}$ for $i \in U$ and $k = 0, \dots, |I_i - 1|$.
11:     Coefficients of error locator polynomial: $\Delta = [\Delta_\tau, \dots, \Delta_1, 1]$.
12:     Compute the ideal: $I = [I \cup \sum_{j=1}^{\tau+1} \Delta_j S^e[i - \tau - 1 + j] : i \in [\tau + 1 \dots p^m - 1 + \tau]$.
13:     Compute the variety:$V = V(I)$ in the variables $\{\Delta_\tau, \dots, \Delta_1\} \cup \{S_u : u \in U\}$.
14:     Compute the list: $L = Ev(R - E) : E \in V$, $E$ is a polynomial with coefficient as the reverse of the error syndrome provided by the each solution in $V$.
15:     return $(L)$.
16: end if
    END
```

6.8 Example and results

Consider C to be the RS code with parameters $[63, 49, 15]$ and the subfield subcode D which is a BCH code with parameters $[63, 24, 15]$. The error correction capability is 7. We choose a word c over $\mathbb{F}_2$ as:

$c =$ [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 1 1 0 0 0 1 0 0 1 0 1 0 1 0 0 1 0 0 0 0 0 0 0 0 1 0 1 0 0 1 1 1 0 0 0 1 1 0 1 1];

The maximum number of errors that can be corrected is 7 (unique decoding). With our method, let us assume that we can decode maximum of $\tau = 10$ errors. So, the error vector e is introduced with non-zero errors in the first ten positions, hence,

$r =$ [0 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 0 0 0 1 0 0 1 0 1 0 1 0 0 1 0 0 0 0 0 0 0 0 1 0 1 0 0 1 1 1 0 0 0 1 1 0 1 1].

In this case, we have: $\Delta = [\Delta_{10}, \ldots, \Delta_1, 1]$. Thus,

$S^e = [\alpha^4, \alpha^8, \alpha^{55}, \alpha^{16}, \alpha^{23}, \alpha^{47}, 1, \alpha^{32}, \alpha^{45}, \alpha^{46}, \alpha^{38}, \alpha^{31}, \alpha^{32}, 1, S_{15}, \alpha, \alpha^{53}, \alpha^{27}, \alpha^8, \alpha^{29}, S_{21}$
$, \alpha^{13}, S_{23}, \alpha^{62}, \alpha^{52}, \alpha, S_{27}, 1, S_{23}^4, S_{15}^2, S_{31}, \alpha^2, \alpha^{59}, \alpha^{43}, 1, \alpha^{54}, \alpha^{19}, \alpha^{16}, S_{15}^{32}, \alpha^{58}, \alpha^4, S_{21}^2, S_{23}^{32},$
$\alpha^{26}, S_{27}^4, S_{23}^2, S_{31}^{32}, \alpha^{61}, 1, \alpha^{41}, S_{15}^{16}, \alpha^2, S_{23}^{16}, S_{27}^2, S_{31}^{16}, 1, S_{15}^8, S_{23}^8, S_{31}^8, S_{15}^4, S_{31}^4, S_{31}^2, S_0, \alpha^4, \alpha^8,$
$\alpha^{55}, \alpha^{16}, \alpha^{23}, \alpha^{47}, 1, \alpha^{32}, \alpha^{45}, \alpha^{46}]$

I has 63 equations and 16 variables, while if we do not use the cyclotomic structure it has 59 variables. One can check that $V(I)$ has just one point which is $[\alpha^{45}, \alpha^{40}, \alpha^{10}, \alpha^{60}, \alpha^{27}, \alpha^{18}, \alpha^{18}, \alpha^{42}, \alpha^{46}, \alpha^4, \alpha^7, 1, \alpha^{13}, \alpha^{45}, \alpha^{29}, 0]$. Thus, $S_{15}^e = \alpha^7, S_{21}^e = 1, S_{23}^e = \alpha^{13}, S_{27}^e = \alpha^{45}, S_{31}^e = \alpha^{29}, S_0^e = 0$.

Therefore R and E can be computed and the polynomial evaluating to c is $R-E$. In this example, we are reducing by 73% the total number of variables which are previously required. Also, for this example we obtained just one codeword which implies unique decoding for this case.

In table 6.2, we have shown the reduction in the number of variables as compared to the previous scheme for various BCH codes with different parameters. For all the codes given in the table, the algorithm corrects beyond the error correction capability of the BCH code which is defined as subfield subcode of given RS codes. With the huge reduction in the number of variables, the proposed algorithm is much less complex than the previous algorithm discussed in [101].

Table 6.2: Decoding BCH codes $[n, k', d']$ beyond error correction capability

$[n, k', d']$ BCH codes	$[n, k, d]$ RS codes	ν	no. of vars (previous)	no. of vars (new)	percentage reduction
$[63, 24, 15]$	$[63, 49, 15]$	10	59	16	73%
$[63, 36, 11]$	$[63, 53, 11]$	7	60	15	75%
$[127, 50, 27]$	$[127, 101, 27]$	15	116	23	80%
$[127, 78, 15]$	$[127, 113, 15]$	9	122	21	83%

6.9 Conclusion

In this chapter, we first discussed how list decoder of RS codes can be used efficiently for decoding BCH codes up to half of the actual minimum distance. We proved that BCH codes are uniquely decoded up to a bound of $min\{t_{GS}, \frac{d'-1}{2}\}$ by using list decoder of RS codes. Also, we computed the probability of having more than one BCH codeword in the list which is found to be very low for codewords in subfield which demonstrated the use of list decoder of RS codes for BCH codes. Also, we discussed about how the algebraic structure of polynomials evaluating to $\mathbb{F}_p$ can be used in order to simplify the list decoding algorithm for BCH codes via syndrome variety. By applying the algebraic structure, we reduced the number of variables in variety by huge amount which in turn facilitate the easier computation of Gröbner basis as compared to previous algorithms.

Chapter 7

Analysis of a Set of Error Correcting Schemes in Multihop Wireless Sensor Networks

7.1 Introduction

The technology of WSN combines sensing, computation as well as communication into a tiny device called sensor nodes. A WSN consists of spatially distributed sensor nodes (or detectors) that monitor a physical or environmental condition such as temperature, pressure, sound etc. and convert it into a signal that can be read by an observer (base station) as shown in Figure 7.1. The idea is to pass the data cooperatively through the network of nodes to the base station. Each wireless network is built of several hundreds or thousands of nodes where each node is either connected to one or many sensor nodes. Each sensor node typically consists of processing capability (microcontrollers), a radio transceiver with an internal antenna or a connection to external antenna, may contain various types of memories (like program data or flash memories), an energy source (like battery or solar cells) and an electronic circuit for interfacing with the sensors or energy source. This new technology of sensor networks has unlimited potential for numerous applications for example environmental, medical, military, transportation, entertainment, crisis management, homeland defense, and smart spaces. Although reliability is the primary requirement of any communication, energy consumption is a major constrain in any WSN. The amount of energy consumed at the node level determines the lifetime of a sensor, which in turn determines the lifetime of a sensor network. Battery capacity of each sensor node is limited and usually it is inconvenient to replace them. Any operation (like computation or transmission) performed on a sensor consumes energy which results in discharging of battery. Hence, efficient use of resources has now become an important issue in order to improve the life span of WSN. Among various resource management methods power control in WSN is significant to overcome unnecessary interference and to save the battery life of the sensors.

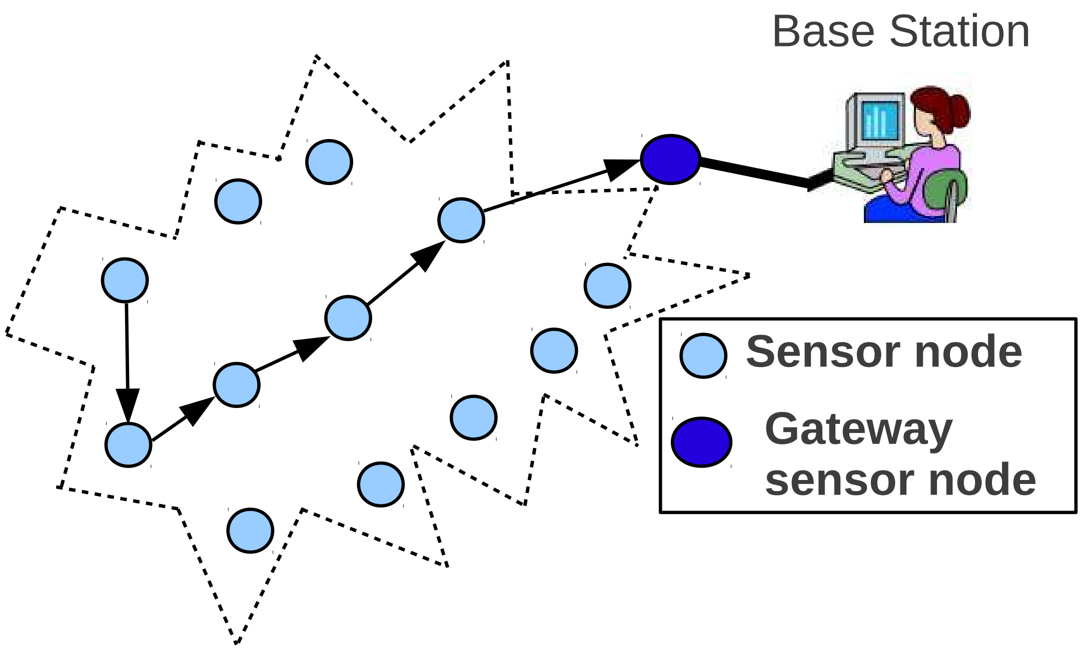

Figure 7.1: Multi-hop WSN architecture

The final goal of a WSN is to collect the data at a sink node where the data is analyzed. In a multi-hop scenario, information passes through various nodes before reaching the final destination. Passing information through such a network may result in additional errors introduced at every node. Data reliability and security are important issues for any communication system. For recovering the erroneous packets, three basic schemes are FEC, ARQ and Hybrid ARQ. ARQ is very simple to use but the disadvantage of using it is the additional retransmission energy cost and area overhead [110]. HARQ that combines ARQ and FEC is even worse [111] since it consumes a lot of energy and is limited to some specific applications. The limited battery capacity of each sensor node makes the minimization of the power consumption at each sensor node one of the primary concerns in WSN, in order to increase their lifetime. the energy constrained transmission issue of WSN makes FEC a popular technique to be used in such networks rather than ARQ and HARQ. With the use of FEC, there are no delays in message flows, though the packet might get lost if the error correction scheme is not strong enough.

To the best of our knowledge, all the schemes using FEC either do complete or do partial encoding/decoding at every node. For most of the codes used in WSN, encoding is simple and energy consumption is low [112]. However, the decoding process is usually complex and it consumes a significant amount of energy. Decoding being done at every node, results in higher energy consumption which reduces the network life in turn.

Two important questions that need to be answered while applying any of these schemes for data transmission via WSN are:

- What is the energy consumed while transmitting the data from source node to sink node given the amount of data that has to be transferred? This will be answered in section 7.3 for our transmission scheme.
- What is the probability of reproducing the original data overcoming all the errors introduced by noisy channels while transmission? This will be discussed later in section 7.4 for the proposed transmission method.

These two questions are important because if the coding gain is smaller than the amount of energy consumed for transmitting extra bits then it is not worth using FEC. It is better to transmit the uncoded data and adapt ARQ. Also, the probability of obtaining a transmitted word in error at the receiver's end should be low enough. This chapter compares several powerful codes for transmission over WSN (in relation to energy consumption at the node level and the probability of reproducing the original data) having the encoder's complexity similar to RS encoder complexity. The decoders are of high complexity with good error correction capabilities at the receiver end. The assumption made here is that decoding takes place only at the base station which is not energy constrained. This makes the scheme independent of decoding energy and complexity as the main motive is to reduce the power consumption on the sensor and relay nodes. It reduces the power consumed at each of the sensor nodes because now rather than encoding and decoding (processing) data at each of the sensor nodes it is just relayed, and only encoded once (at first node) and decoded once at sink. The energy and probability model discussed in later sections tries to answer the two questions mentioned above.

7.2 System model

Due to lack of infrastructure, the topology of a WSN is random and dynamic. For simplicity, a linear model is assumed with equispaced nodes as shown in Figure 7.2. It illustrates the linear model of packet forwarding from source node to sink node. It is considered to be an optimum model for a multi-hop WSN for transmitting a single packet of data from source node to sink node [113].

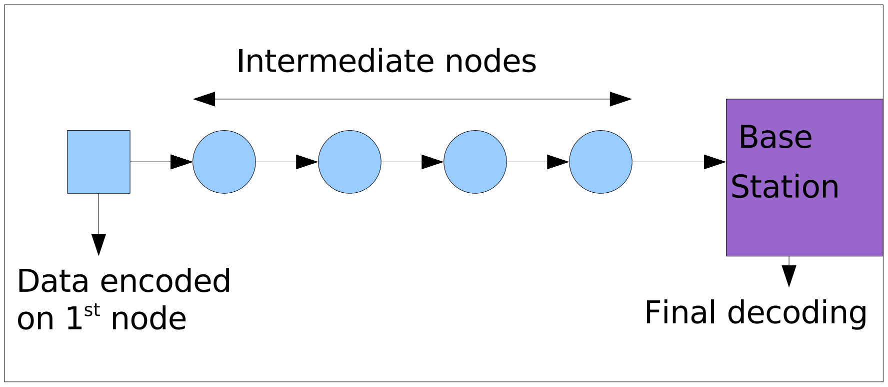

Figure 7.2: System model

7.3 Energy model

As already mentioned, in order to compare the performance of various codes, we need to know the amount of energy consumed at node level while transmitting the data. The architecture with annotated energy consumption for sending an n-bit data packet over a single hop wireless link of distance d is shown in Figure 7.3.

For this model the final decoding energy has also been ignored since the base station is not power constrained and only the encoding energy for the first node is considered. The communication energy consumed at send a data packet can be computed as follows:

$$E_{total} = E_{enc} + E_{TX} + E_{RX} \tag{7.1}$$

where

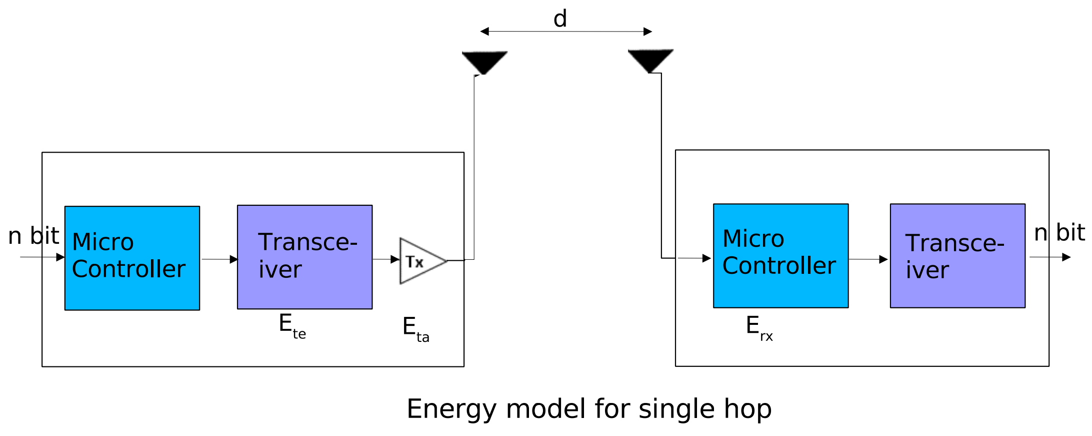

Figure 7.3: Energy model

E_{total} : Total energy consumed in the network

E_{enc} : Energy consumed by the encoder at the first node

E_{TX} : Energy consumed in transmission by all nodes

E_{RX} : Energy consumed in receiving the data by all nodes except first one

The above equation can be written in more details:

$$E_{total} = E_{enc} + \sum_{i=1}^{m} N_b E_{tx/b} + \sum_{i=1}^{m-1} N_b E_{rx/b} \tag{7.2}$$

where,

m : Number of hops

N_b : Total number of bits transmitted

$E_{tx/b}$: Energy consumed in transmitting a single bit from a node

$E_{rx/b}$: Energy consumed in receiving a single bit at a node

The term $E_{tx/b}$ can be represented as [114]:

$$E_{tx/b} = E_{te} + E_{ta} d^{\alpha} \tag{7.3}$$

where,

E_{te} is the power consumption at transmitter electronics

α is the path loss component usually varies between $2-4$ with $\alpha = 3$ being a typical value when scattering is considered [115].

E_{ta} is the power consumption of transmitter amplifier can be given as:

$$E_{ta} = \frac{(\frac{S}{N})_{r(i)}(NF_{RX})(N_0)(BW)(\frac{4\pi}{\lambda})^{\alpha}}{(G_{ant})(\eta_{amp})(R_{bit})} \tag{7.4}$$

where, $(S/N)_{r(i)}$ is the desired SNR at the i^{th} receiver, NF_{RX} is noise figure at receiver, N_0 in the thermal noise, BW is the bandwidth of channel noise, λ is wavelength, G_{ant} is antenna gain, η_{amp} is the transmitter efficiency and R_{bit} is the raw channel rate in bps.

If the channels have low noise proposed transmission scheme may completely recover the original data with the additional advantage of low energy consumption.

7.4 Model for error probability

Also, another important parameter while comparing the performance of two codes is the probability of receiving the correct word at the base station. In this section, we will discuss about the computation of probability of receiving correct word or probability of receiving wrong word at the base station. The probability of error that gets corrected at the base station depends on several factors like channel model, number of hops and the error decoding capability of the code.

7.4.1 Wireless channel model

For a typical multihop WSN, Rayleigh slow fading channel attenuation model is assumed as defined in [112]. The modulation technique used here is a non-coherent (envelope or square-law) detector with binary orthogonal FSK signals. For this particular case the probability of bit error [116]:

$$P_{FSK} = \frac{1}{2 + \bar{\gamma_b}} \tag{7.5}$$

where, $\bar{\gamma_b}$ is the average received bit signal to noise ratio which depends on the receiver characteristics and the distance between the receiver and transmitter.

7.4.2 Probability of error for our model

Only linear block codes with encoding similar to RS codes are discussed here and it is assumed that the channel is uniform over which the code is sent, which results in constant symbol error probability (P_s). The model is shown in Figure 7.4.

With the above constrains and assuming FSK-modulation the probability of n sym-

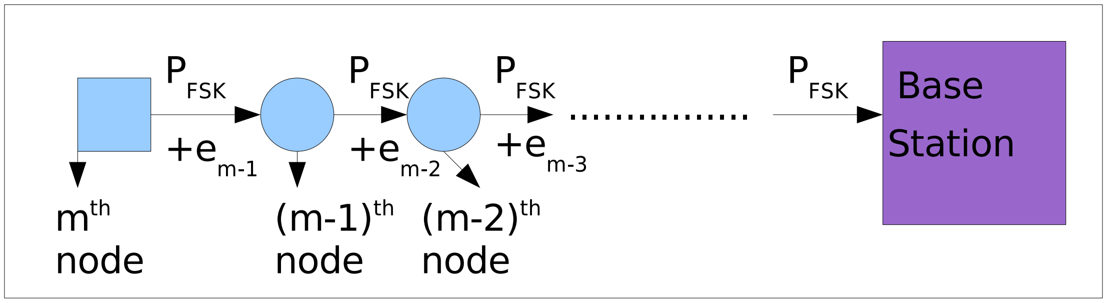

Figure 7.4: Error probability model

bol code word being in error assuming we can correct up to t errors is given by [112]:

$$P_e(\epsilon) = \sum_{i=t+1}^{n} \binom{n}{i} \epsilon^i (1-\epsilon)^{n-i} \tag{7.6}$$

where, ϵ denotes the bit error probability and $\binom{n}{i}$ is the binomial co-efficient. The probability that the packet is received with no errors at sink node is:

$$P_c = \prod_{i=1}^{m} (1 - P(i)) \tag{7.7}$$

where $P(i)$ is the probability of receiving a packet with errors from $(m-i+1)^{th}$ node to $(m-i)^{th}$ node. For our case, the probability of having no error for each link is fixed i.e. $P(i)$ and is given by $(1 - P_{FSK})$. Hence, P_c is given by $(1 - P_{FSK})^m$. Therefore, the bit error probability is:

$$\epsilon = 1 - (1 - P_{FSK})^m \tag{7.8}$$

In our assumptions, the decoding is done only once at the sink node. All the other nodes just keep on transmitting the code word and adding error to it. Thus the packet error probability at sink node can be calculated as:

$$P_e = \sum_{i=t+1}^{n} \binom{n}{i} (1 - (1 - P_{FSK})^m)^i (1 - P_{FSK})^{m(n-i)} \tag{7.9}$$

which results in,

$$P_e = \sum_{i=t+1}^{n} \binom{n}{i} (1 - (1 - \frac{1}{2 + \bar{\gamma_b}})^m)^i (1 - \frac{1}{2 + \bar{\gamma_b}})^{m(n-i)} \tag{7.10}$$

The probability of successful transmission is $1 - P_e$ and the expected number of transmissions for successful end to end packet transmission is $\frac{1}{1-P_e}$.

7.5 Energy consumption and error correction capability of different decoding schemes

In previous sections, we have computed the energy consumption at node level and probability of error at base station for any code used while data transmission for our system model. In this section, we will go through the code parameters for various powerful codes in order to compute the two performance parameters for them. As the main focus of this thesis is RS type codes so we will compare various powerful RS type codes. Also, RS code is considered to be the best choice for WSN having maximum energy efficiency in proper channel conditions or when relay nodes are sufficient in numbers i.e. greater than 5 [117]. This section presents several codes that have RS encoding complexity.

Considering,

k : Number of information symbols.

n : Length of the code word.

T : Maximum number of the errors that a code can correct.

$\mathbb{F}_{q^2}$: Field over which codes are defined.

E_{total} : Total energy consumed evaluated from Equation 7.1.

7.5.1 Uncoded data

For later comparisons uncoded data is transmitted first through the network. The number of bits transmitted is equal $k\ log_2(q^2)$.

$$E_{total} = k\ log_2(q^2)[mE_{tx/b} + (m-1)E_{rx/b}] \tag{7.11}$$

$$T_{uncoded} = 0 \tag{7.12}$$

7.5.2 Reed-Solomon codes

For a single RS encoder, the energy consumed in encoding is considered as E_{RS}. The length of the code is $q^2 - 1$ for RS codes which is equivalent to $(q^2-1)log_2(q^2)$ bits transmitted.

$$E_{total} = E_{RS} + (q^2-1)log_2(q^2)[mE_{tx/b} + (m-1)E_{rx/b}] \tag{7.13}$$

$$T_{RS} \leq \frac{q^2 - k - 1}{2} \tag{7.14}$$

7.5.3 List-decoder for RS codes

For the list decoding scheme presented by McEliece in [76], the total energy consumed on nodes is equal to RS codes because the encoding process is the same but the error correction capability is much higher than that of the RS decoder.

$$T_{List-Decoded} \leq n(1 - \sqrt{R}) \tag{7.15}$$

where, R is the rate of the code that is $\frac{k}{n}$. If the size of the list is sufficiently small, then it is considered to be reasonable amount of recovery from error. Also, there is a possibility of complete recovery of codeword with the list decoding algorithm.

7.5.4 Hermitian codes

Hermitian codes (H) obtained from Hermitian curves can be considered as generalized RS codes defined over the field $\mathbb{F}_{q^2}$. The length of these codes are q^3, which in much larger than RS codes defined over the same alphabet. The systematic encoding of Hermitian codes [118] can be done by using q RS like encoders. Therefore, the encoding energy used at first node is presented by qE_{RS}.

$$E_{total} = qE_{RS} + q^3 \ log_2(q^2)[mE_{tx/b} + (m-1)E_{rx/b}] \tag{7.16}$$

$$T_H \leq \frac{2q^3 - q^2 + q - 2k + 2}{2} \quad [21] \tag{7.17}$$

7.5.5 Multivariate interpolation decoded RS codes (MIDRS)

The idea is that M-RS codes transmitted together and are decoded using $M+1$ variate interpolation [119]. The encoding is of complexity order of M RS encoders. It sends M RS codes which is $M(q^2-1)log_2 q^2$ bits in total.

$$E_{total} = ME_{RS} + M(q^2-1)log_2(q^2)[mE_{tx/b} + (m-1)E_{rx/b}] \tag{7.18}$$

$$T_{MIDRS} \leq Mn(1 - R^{\frac{M}{M+1}}) \tag{7.19}$$

The decoder can correct up to T_{MIDRS} errors in M blocks. The error correction capability is much higher than Guruswami-Sudan list decoding algorithm mentioned above for certain conditions/error patterns.

Table 7.1: Code parameters

code	k	n	T
RS code	k	q^2-1	$(\frac{q^2-1-k}{2})$
List-Decoder	k	q^2-1	$n(1-\sqrt{\frac{k}{n}})$
Hermitian	k	q^3	$(\frac{2q^3-q^2+q-2k+2}{2})$
MIDRS	Mk	$M(q^2-1)$	$Mn(1-{\frac{k}{n}}^{\frac{M-1}{M}})$

7.5.6 Table

The parameters of the various codes mentioned above are summarized in the Table 7.1 below: The packet error probability can be evaluated from Equation 7.6 and 7.9 using different value for the error correction capability (T) for various codes.

7.6 Results and discussion

Based on the computation of transmission energy and probability of error for several codes, we will compare their performances in this section. The energy and probability of error values are computed for Atmel Atmega 128L 8-bit microcontroller and RFM-TR100 transceiver. For Figure 7.5 & Figure 7.6, the codes are of almost equal length. Hermitian code transmission have the lowest energy and uncoded transmission has the highest energy consumption for low error probability. This is because, the smallest number of bits are transmitted for Hermitian code as when the codes are of equal length, then the field over which Hermitian codes are defined is smaller. So, the Hermitian code is the best to use if codes are of same length. For Figure 7.7, the codes are defined over the same field i.e. $\mathbb{F}_{64}$.

Hermitian and MIDRS codes have almost the same performance here while uncoded transmission is again the worst. This proves that Hermitian codes are one of the strongest codes among the codes, we have used for transmission over WSN. Hermitian codes have higher correction capability and larger length over the smaller field as compared to RS codes, or even list decoded RS codes have higher error correction capability. Since, our transmission scheme is independent of the decoder's complexity or in other words the amount of energy consumed by the decoder, the most powerful

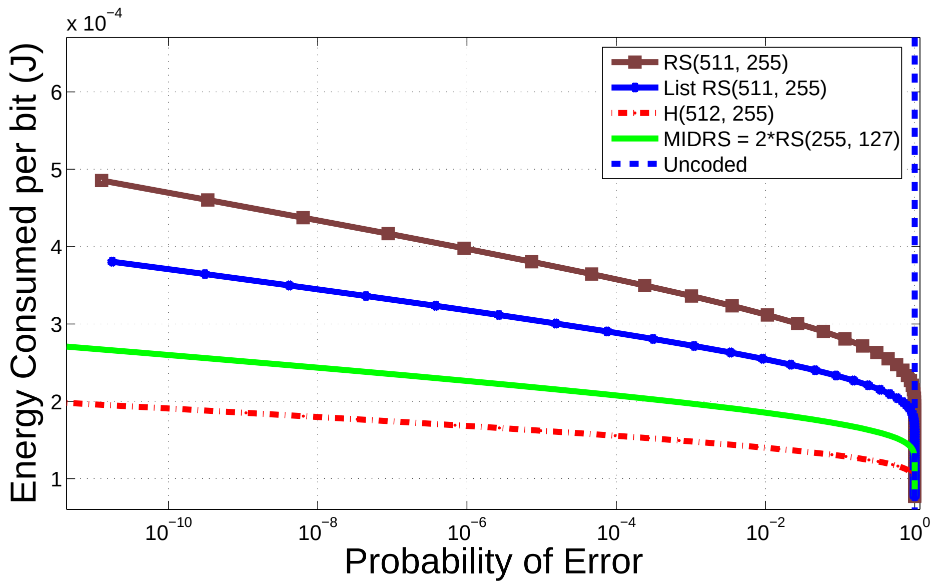

Figure 7.5: Energy per bit vs probability of error for 4 hops

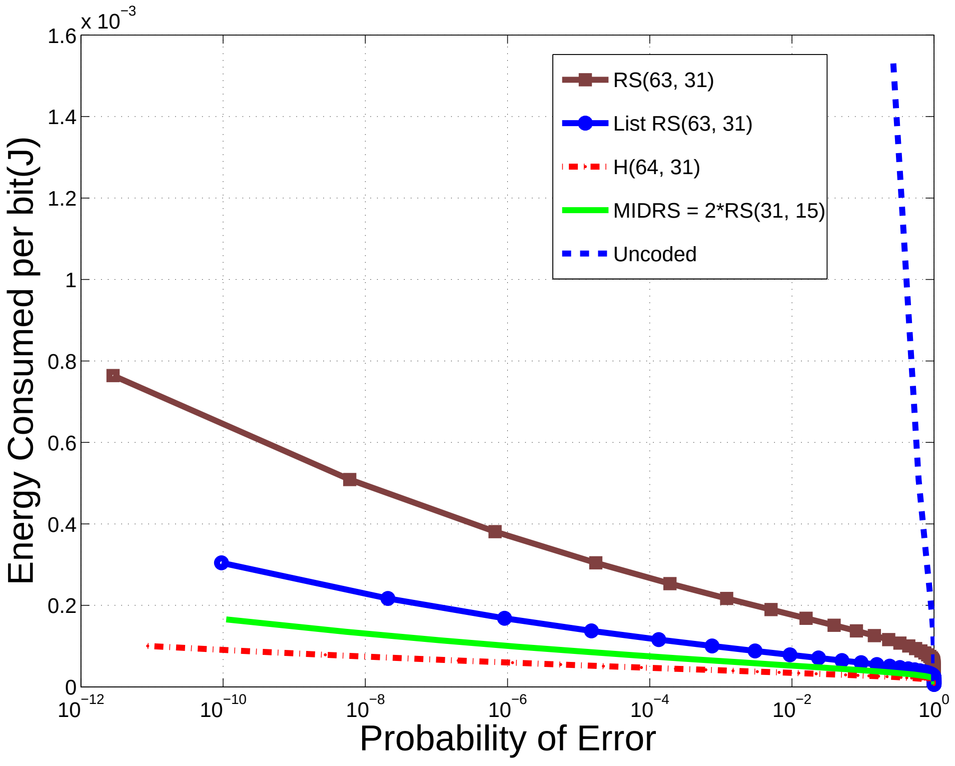

Figure 7.6: Energy per bit vs probability of error for 10 hops

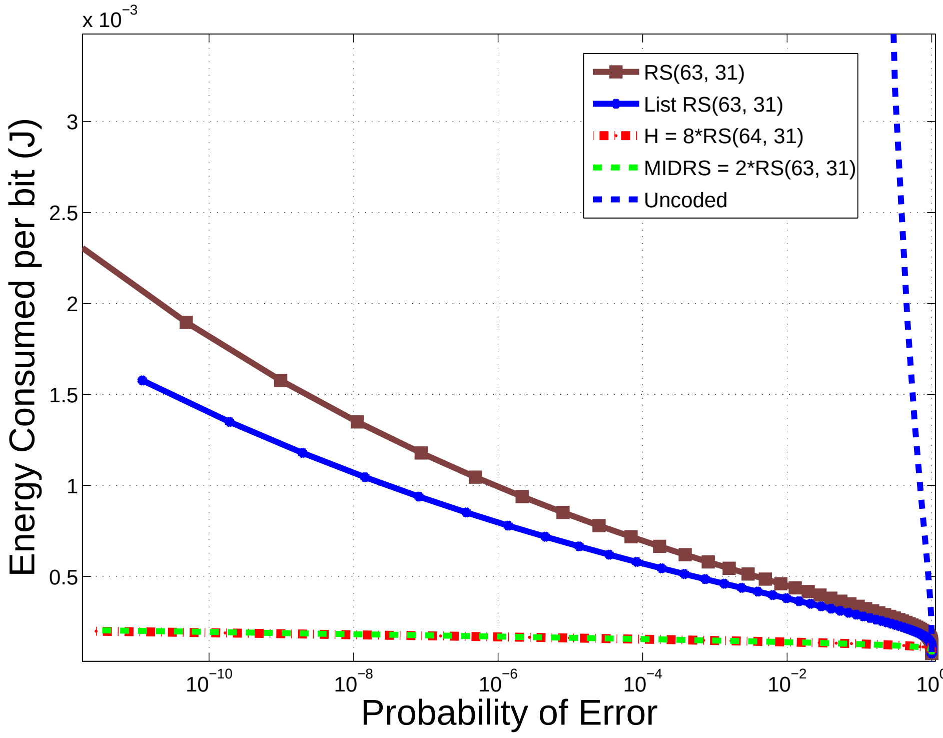

Figure 7.7: Energy per bit vs probability of error for 4 hops

code will consume minimum energy to give lower error probability. This is one of the applications where we can clearly see Hermitian codes usefulness over RS codes as the decoder's complexity is not involved in computing energy at the node level.

7.7 Conclusion

Various coding schemes have been examined for reliable communication of data over multi-hop WSNs. A new scheme for data transmission has been designed which is different from the previous works of encoding/decoding at each and every node. This new scheme encodes the data only at the source node and the final decoding takes place on a base station which is not power constrained. Thus, it saves a lot of power at node level which will increase the network life and is good for energy constrained networks. Our method is to use efficient and powerful codes which are easy to encode. According to the graphs, if the lengths of the codes are the same Hermitian codes are the best and if they are defined over same field, Hermitian and MIDRS codes have almost the same performance. Also, it is shown that less energy is consumed at nodes while maintaining a low error probability. Future work will include different channel models, and networking schemes including non-linear model for distance between the nodes.

Chapter 8

Conclusion and Future work

A few interesting problems in the field of coding theory are to construct the optimal codes having the maximum possible minimum distance for a specific code length and dimension and to develop a simple decoding algorithm for them. However, most of the algebraic codes involve simple encoding techniques but still decoding techniques complexity is a huge issue under investigation by researchers. The use of error correcting codes are always bounded by constraints like area (memory/total gate count) and decoding time, which in turn is responsible for hardware resources and energy consumption respectively. The thesis addressed the problem of optimal code construction by computing the subfield subcodes of Generalized Toric codes. Addressing the need of developing efficient decoding algorithms, which optimize the decoding time as well as the circuits area simultaneously, the thesis presented efficient decoding algorithms for RS, Hermitian and BCH codes. Also, a novel transmission scheme to use powerful codes for WSN is discussed.

8.1 Concluding Remarks

In this thesis, we attacked the important issues of coding theory both from a mathematical and engineering perspective. In the following subsections, we will discuss the important contributions of the thesis.

8.1.1 Efficient decoding techniques

The thesis focused on enhancing the efficiency of unique and list decoding algorithms for several RS type linear block codes like BCH, RS and Hermitian codes. The classical approach for the decoding algorithm has two sub-problems, involving the computation of error locator polynomial firstly and then the computation of error locations and error values. However, in this thesis the main focus is on the direct decoding algorithms. It does not involve the computation of error locations and error values separately and the encoding algorithms are based on an evaluation approach. The decoding algorithm skipped the Chein search and Forney's formula which reduced the complexity of implementation as it does not require any extra circuitry for them. We discussed the direct decoding algorithm for RS and Hermitian codes.

An efficient encoding and decoding scheme based on evaluation has been proposed for RS codes in chapter 3. The proposed decoding algorithm directly outputs the message symbols without going through the steps of Chien search and Forney's formula. This makes the hardware circuitry much simpler for the proposed decoder. The proposed decoding algorithm requires only the syndrome calculation and BM algorithm blocks. Also, the extra clock cycles required to run the Chien search and Forney's formula have been saved. The hardware implementations on a commercially available $65nm$ ASIC process showed that the proposed decoder is area efficient as well as faster in comparison to the classical one. For a $\text{RS}(255, 239)$ code, the proposed decoder resulted in a reduction of 57.6% in hardware area and 53.07% reduction in decoding time as compared to the classical decoder [58]. One more important observation that has been made is that the area for the classical decoder increases significantly even for small variation in error correction capability, t or with the dimension k of the code as compared to the proposed decoder due to the Chien search and Forney's formula complexity being t dependent.

The evaluation based novel encoding and decoding algorithm has been extended for shortened RS codes [59]. The proposed decoder resulted in significant memory savings for software implementations at the expense of latency. RS $(32, 24)$ for resource constrained WSN application on a ATMEL Atmega 128 microcontroller showed that the new decoder and encoder is 52% and 72% of the traditional method respectively in memory footprint. The algorithm is well suited for resource constrained embedded systems such as WSN requiring enhanced reliability.

Though RS codes are one of the most popular codes for industries their restriction is that the size is limited to the size of the chosen alphabet that is the finite field over which they are defined. As compared to RS codes, Hermitian codes are much longer than the size of the codeword alphabet and have higher error correction capability at high code rates. Although Hermitian codes offer desirable properties over RS codes, the Hermitian decoder has higher complexity and a complicated architecture which is the biggest hurdle to make it useful for industrial applications. From an architectural point of view the most efficient decoding algorithm is Kötter's algorithm. The space complexity is $O(q^4)$ and time complexity is $O(q^6)$ for Kötter's decoder. The thesis also focused on reducing the complexity of the Hermitian decoder further by use of the recently proposed Lee-O'Sullivan algorithm. We proposed an efficient architecture for the implementation of Lee-O'Sullivan decoder. The algorithm iteratively computes the sent message through a majority voting procedure using the Gröbner bases of interpolation modules. Similar to Kötter's algorithm, Lee-O'Sullivan algorithm also has a regular structure which makes it simple and suitable for VLSI implementation. The decoding algorithm directly gives the message word without going through the separate steps like Chien search and Forney's formula as required after Kötter's algorithm. In terms of hardware requirements, for the widely used high rate Hermitian codes, the Lee-O'Sullivan algorithm is q times faster than Kötter's algorithm with the same space complexity of $O(q^4)$. Further speed improvements are achieved by combining the main idea of Guruswami list decoding with the Lee-O'Sullivan algorithm. In terms of hardware, the addition of this concept, will further reduce the running time of the algorithm and make the circuitry 2 times faster than the original Lee-O'Sullivan algorithm.

Another interesting sub-class of cyclic codes is BCH codes. BCH codes are subfield subcodes of RS codes and we studied them as evaluation codes. The evaluation polynomial for BCH codes has some special algebraic properties based on cyclotomic coset structure. By exploiting those properties, we reduced the complexity of list decoding algorithm via syndrome variety in chapter 6. From simulation, it is shown that the number of variables are significantly reduced by $70 - 80\%$ as compared to the algorithm discussed in [101] which resulted in efficient computation of Gröbner basis. Also, the cyclotomic coset based properties resulted in reducing the complexity of the RR factorization algorithm for BCH codes.

8.1.2 Construction of optimal codes

Addressing another important problem in the field of coding theory, we came to the construction of optimal codes. A new procedure to construct the optimal codes is proposed by constructing the subfield subcodes of Generalized Toric Codes [87] in chapter 5. The subfield subcodes of Generalized Toric codes are the multidimensional analogues of BCH codes, which may be seen as subfield-subcodes of generalized RS codes. The evaluation polynomial for subfield-subcodes of Generalized Toric Codes has been identified which in turn allowed us to determine the dimensions and obtain the bounds for minimum distance. Several examples of binary and ternary subfield-subcodes of Generalized Toric Codes showed that these are the best known codes for a given length and dimension which in turn prove that this can be used as a method of constructing some of the best known codes.

8.1.3 Applications of complex coding schemes

A novel transmission scheme is presented in chapter 7 which uses efficient and powerful codes which are easy to encode but have complex decoding schemes for reliable communication. We examined the performance of various powerful codes for data transmission over multihop WSN [48]. We proposed a model for energy consumption at the node level and the probability of receiving a correct word for multihop WSN at base station using a simple transmission scheme that involves encoding only at the first node and decoding only at the base station. The powerful codes allow the correction of a larger number of errors. Several powerful codes like RS, list decoded RS codes, multivariate interpolation decoded RS codes (MIDRS) and Hermitian codes, having simple encoding, are investigated for this transmission scheme. According to the results, if lengths of the codes are the same, Hermitian codes have the best performance. However, if the codes are defined over the same field, Hermitian and MIDRS codes have almost the same performance. Also, the results proved that less energy is consumed at nodes while maintaining a low error probability.

8.2 Future work

Despite many advances made in coding theory, researchers are still looking for new, better codes and for the simplified decoding and encoding schemes. For many applications like WSN, where memory and battery life of sensor nodes are huge constrains, it is important to find efficient encoding and decoding algorithms. We believe that significant advances can be made in the field of coding theory by addressing the following issues:

- **Efficient Implementation of Hermitian codes:** An efficient architecture for Hermitian decoder based on the lee-O'Sullivan algorithm is presented in the thesis. The circuitry is much simpler than the one proposed in Kötter's algorithm as the decoding algorithm directly gives the message word at the end of the decoding algorithm without separately using the steps like Chien search and Forney's formula. An interesting future work would be to implement it in hardware and confirm the results as we would expect the reduction in decoding time from the theoretical results. We proposed the architecture for decoder is already presented in [120]. Hermitian codes can be made industrially more applicable by having computationally simpler and area efficient methods. The proposed decoder may satisfy the resource and throughput constraints imposed by the applications depending upon the implementation results.

- **Subfield-Subcodes of Hermitian codes:** Subfield subcodes of Hermitian codes are of particular interest to researchers as they have the simplicity of the smaller field and the nice rich algebraic structure of the bigger code. Recently, a complete characterization of the subfield-subcodes has already been done in [121] and it would be a very interesting problem to compare the performances of BCH with these codes as it is always interesting comparing Hermitian to RS codes.

- **Wireless Sensor Networks:** We developed a novel transmission scheme for WSN and evaluated the performance of several powerful codes for the same. We used a particular channel model and linear system model with equi-spaced nodes. It will be interesting to check the same for different channel models and networking schemes including non-linear model for distance between the nodes.

Bibliography

[1] Lin, S.; Costello, D.J. and Miller, M.J.; "Authomatic-repeat-request error-control schemes", IEEE Communication, vol. 22, no. 12, 1984, pp. 5-7.

[2] Treciokas, R.; "Application of forward error correction to a Rayleigh fading h.f. communication channel", In proceeding of Electrical Engineers, 1978, pp. 173-178.

[3] Wicker, S.B.; "Adaptive rate error control through the use of diversity combining and majority-logic decoding in a hybrid-ARQ protocol", IEEE Transactions on Communications, vol. 38, no. 3, 1990, pp. 263-266.

[4] Peek, H,; Bergmans, J.; Haaren, J.V.; Toolenar, F. and Stan, S.; "Origins and successors of the compact disc", Philips research book, Springer, vol . 11, 2009.

[5] Shannon, C. E.; "A mathematical theory of communication", Bell System Technical Journal, vol 27, 1948, pp. 623-656.

[6] Hamming, Richard W.; "Error detecting and error correcting codes", Bell System Technical Journal, Vol. 26, no. 2, 1950, pp. 147-160.

[7] Schwartz, M.; Bennett, W. R. and Stein, S., "Communication systems and techniques", New York: IEEE Press, 1996.

[8] Hocquenghem, A.; "Codes correcteurs d'erreurs", Chiffres (Paris), vol.2, 1959, pp. 147-156.

[9] Bose, R. C. and Ray-Chaudhuri, D. K.; "On a class of error correcting binary group codes", Information and Control, ISSN 0890-5401, vol. 3, no. 1, 1960, pp. 68-79.

[10] Reed, I. S. and Solomon, G.;"Polynomial codes over certain finite fields," SIAM Journal of Applied Mathematics, vol. 8, 1960, pp. 300-304.

[11] Stichtenoth, H.; “Algebraic function fields and codes”, Graduate Texts in Mathematics 254, Springer Verlag, 2009.

[12] Thompson, T. M.; “From error-correcting codes through sphere packings to simple groups”, Math. Assoc. Amer., Washington DC, 1984.

[13] Vermani, L. and Jindal, S.; “A note on maximum distance separable (optimal) codes”, IEEE Transaction on Information Theory, 1983, vol. 29, pp. 136-137.

[14] Elias, P., “List decoding for noisy channel”, Res. Lab Electron, MIT, Cambridge, MA, Tech Rep, 1957.

[15] Wozencraft, J.M., “List decoding, quarterly progress report”, Res. Lab. Electronics, MIT, Cambridge, MA, 1958,vol. 48, 90-95.

[16] Hill, J.; Szewczyk, R.; Woo, A.; Hollar, S.; Culler, D. and Pister, K.; “System architecture directions for networked sensors”, In proceeding of International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), 2000, pp. 93-104.

[17] Sharma, A.; Shinghal, K.; Srivastava, N. and Singh, R.; “Energy management for wireless sensor network nodes”, International Journal of Advances in Engineering and Technology (IJAET), vol. 1, 2011, pp. 7-13.

[18] Stichtenoth, H.; “A note on Hermitian codes over GF(q2)”, IEEE Transactions on Information Theory, vol. 34, 1988, pp. 1345-1348.

[19] Hansen, J.; “Toric surfaces and error-correcting codes”, Coding theory, cryptography and related areas, Springer, 2000, pp. 132-142,.

[20] Yaghoobian, T. and Blake, I.F.; “Hermitian codes as generalized Reed-Solomon codes”, Design, Codes and Cryptography, vol. 2, no. 1, 1992, pp. 5-17.

[21] Popovici, E.; “Algorithms and architectures for Decoding Reed-Solomon and Hermitian Codes”, PhD Thesis, University College Cork, Ireland, 2002.

[22] Massey, J. L.; “Shift-register synthesis and BCH decoding”, IEEE Transactions on Information Theory, vol. 15, 1969, pp. 122-127.

[23] Berlekamp, E.R.; “Algebraic coding theory”, McGraw-Hill, New York, 1968.

[24] Sarwate,D. V. and Shanbhag, N. R.; “High-speed architectures for Reed-Solomon decoders,” IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 9, 2001 , pp. 641-655.

[25] Blahut, R.E.; “Algebraic codes for data transmission”, Cambridge University Press, 2003.

[26] Lin, S. and Costello, D.; “Error control Coding: fundamentals and applications”, New Jersey, USA: Prentice Hall, 1983.

[27] Guruswami, V.; “List decoding of error-correcting codes”, Lecture Notes in Computer Science, Springer, 2004.

[28] Singleton, R.C.; “Maximum distance q-ary codes”, IEEE Transactions on Information Theory, vol. 10, no. 2, 1964, pp. 116-118.

[29] Gilbert, E.N.; “ A comparison of signaling alphabets”, Bell Sys. Tech. J., vol. 31, 1952, pp. 504-522.

[30] Varshamov, R.R.; “ Estimate of the number of signals in error correcting codes”, Dokl. Acad. Nauk, vol. 117, 1957, pp. 739-741.

[31] Tuechler, M. and Hagenauer, J.; “Channel coding”, lecture script, Munich University of Technology, 2003, pp. 111-162.

[32] Prange, E.;“Cyclic error correcting codes in two symbols”, AFCRCTN, Air Force Cambridge Research Center, Cambridge, 1957, pp. 57-103.

[33] Raghavendra, C.S.; Sivalingam, K.M. and Znati, T.; “Wireless sensor networks”, Kluwer Academic Publishers, Boston, 2006.

[34] Wicker, S.B. and Bhargava, V.K.; “Reed-Solomon codes and their applications”, IEEE Press, 1999.

[35] Sklar, B.; “Digital communications: fundamentals and applications”, Second Edition, 2001.

[36] Diaz, V.; Guevara, C. and Vath, M.; "Codes from n-Dimensional polyhedra and n-Dimensional cyclic codes", In proceeding of Summer Institute in Mathematics for Undergraduates (SIMU), 2001.

[37] Hansen, J.; " Toric varieties Hirzebruch surfaces and error-correcting codes", Applied Algebra in Engineering Communication and Computing, vol. 13, 2002, pp. 289-300.

[38] Joyner, D.; "Toric codes over finite fields", Applied Algebra in Engineering Communication and Computing, vol. 15, 2004, pp. 63-79.

[39] Ruano, D.; "On the structure of generalized Toric codes", Journal of Symbolic Computation, vol. 44, no. 5, 2009, pp. 499-506.

[40] Little J. and Schwarz R.; "On $m-$dimensional Toric codes", available at http://arxiv.org/abs/cs/0506102.

[41] Lint, J.H.V and Springer, T.; "Generalized Reed-Solomon codes from algebraic geometry", IEEE Transactions on Information Theory, vol. 46, no. 2, 1987, pp. 305-308.

[42] Goppa, V.D.; "Geometry and codes", Kluver academic publishers, 1988.

[43] Ţsfasman, M.A.; Vládut, S.G. and Zink, T.; "Modular curves, Shimura curves and Goppa codes, better than Varshamov-Gilbert bound", Mathematische Nachrichten, vol. 104, 1982, pp. 13-28.

[44] Fuhrmann, R.; Garcia, A. and Torres, F.; "On maximal curves", Journal of Number Theory, vol. 67, no. 1, 1997, pp. 29-51.

[45] Popovici, E.M.; O'Sullivan, M.E.; Fitzpatrick, P. and Kötter, R.; "A fast parallel implementation of a Hermitian decoder", In proceeding of IEEE International Symposium on Information Theory (ISIT), 2001, pp. 311.

[46] Blake, I.; Heegard, C.; Hohold, T and Wei, V.; "Algebraic geometric codes", IEEE Transactions on Information Theory, vol. 44, no. 6, 1998, pp. 2596-2618.

[47] Stichtenoth,H.; "A note on Hermitian curves over $GF(q^2)$", IEEE Transactions on Information Theory, vol. 34, Nov. 1988, pp. 1345-1348.

[48] Srivastava, S.; Spagnol, C. and Popovici, E.; "Analysis of a set of error correcting codes in Multihop WSN", In proceeding of IEEE PhD Research In Microelectronics Engineering (PRIME), 2009, pp. 1-4.

[49] Lee, H.; "An area-efcient Euclidean algorithm block for Reed-Solomon decoder," IEEE Computer Society Annual Symposium on VLSI, 2003, pp. 209-210.

[50] Blahut, R. E.; "Theory and practice of error-control codes", Reading, Massachusetts : Addison-Wesley Publishing Company, 1983.

[51] Gao, S.; "A new algorithm for decoding Reed-Solomon codes", Communications, Information and Network Security, Kluwer Academic Publishers, 2003, pp. 55-68.

[52] Kampf, S and Bossert, M.; "The Euclidean Algorithm for Generalized Minimum Distance Decoding of Reed-Solomon Codes", IEEE International Symposium on Information Theory (ISIT), 2010, pp. 1090-1094.

[53] Popovici, E. and Fitzpatrick, P.; "Reed-Solomon codecs for optical communications", In proceeding of IEEE International Conference on Microelectronics (MIEL), May 2002, Vo1. 2, pp. 613-616.

[54] Wicker, S.B., "Error Control Systems for digital communication and storage", Prentice Hall, 1995.

[55] Chien, R. T.; "Cyclic decoding procedures for BCH codes", IEEE Transactions on Information Theory, vol. 27, 1981, pp. 254-256.

[56] Forney, G. D.; "On decoding of BCH codes", IEEE Transactions on Information Theory, vol. 11, 1965, pp. 393-403.

[57] Muthiah, D. and Raj, A.; "Implementation of high-speed LFSR design with parallel architectures", In proceeding of IEEE International Conference on Computing, Communications and Applications (ICCCA), 2012, pp. 1-6.

[58] Srivastava, S.; McSweeney, R.; Spagnol; C. and Popovici, E.; "Efficient Berlekamp-Massey based recursive decoder for Reed-Solomon codes", In proceeding of IEEE International Conference on Microelectronics (MIEL), 2012, pp. 379-382.

[59] Srivastava, S.; McSweeney, R.; Spagnol, C. and Popovici, E.; "Area efficient evaluation based coding of shortened Reed-Solomon codes for memory constrained applications", presented in IET Irish Signals and Systems, 2012.

[60] McSweeney, R.; "Energy efficient link-layer coding and compression for Wireless Body Area Networks", PhD Thesis, University College Cork, Ireland, 2011.

[61] Macdonald, T.G. and Pursley, M.B.; "Hermitian codes for frequency-hop spread-spectrum packet radio networks", IEEE Transactions on Wireless Communications, vol. 2 ,no. 3, 2003, pp. 529-536.

[62] O'Sullivan, M.E., "Decoding of codes defined by a single point on a curve, IEEE Transactions on Information Theory, Vol. 41 , 1992, pp. 1709-1719.

[63] Sakata, S., "Fast erasure-and-error decoding of any one-point AG codes up to the Feng-Rao bound", IEEE International Symposium on Information Theory, 1995, pp. 96.

[64] Justesen, J.; Larsen, K. J.; Jensen, H. E. and Hholt, T.; "Fast decoding of codes from algebraic plane curves", IEEE Transactions on Information Theory, vol. 38, Jan. 1992, pp. 111-119.

[65] Sakata, S,; Justesen, J.; Madelung, Y.; Jensen, H. E. and Hholt, T.; "Fast decoding of AG-codes up to the designed minimum distance", IEEE Transactions on Information Theory, vol. 41 , Nov. 1992, pp. 1672-1677.

[66] Jensen,C. D.; "Fast decoding of codes from algebraic geometry", IEEE Transactions on Information Theory, vol. 40, Jan. 1994, pp. 223-230.

[67] Leonard, D.; "Error-locator ideals for algebraic geometric codes", IEEE Transactions on Information Theory, vol. 41, May 1995, pp. 819-825.

[68] Feng, G.L.; Wei, V.; Rao, T. R. N. and Tzeng, K. K.; "Simplified understanding and efficient decoding of a class of algebraic-geometric codes", IEEE Transactions on Information Theory, vol. 40, July 1994, pp. 981-1002.

[69] OSullivan, M. E.; "Decoding of codes defined by a single point on a curve", IEEE Transactions on Information Theory, vol. 41, Nov. 1992, pp. 1709-1719.

[70] Sakata, S.; Jensen,H. E. and Hholt, T.; " Generalized Berlekamp-Massey decoding of algebraic geometric codes up to half the FengRao bound", IEEE Transactions on Information Theory, vol. 41, pp. 1762-1768, Nov. 1995.

[71] Lee. K and Sullivan, M.E.; " List decoding of Hermitian codes using Gröbner bases", IEEE Transactions on Information Theory, vol. 44, 2009, pp. 1662-1675.

[72] Lee, Kwankyu; Bras-Amorós, Maria; O'Sullivan, Michael; "Unique Decoding of Plane AG Codes via Interpolation", IEEE Transactions on Information theory, vol. 58, no. 6, 2012, pp. 3941-3950.

[73] Kötter, R.; "A fast parallel implemetation of a Berlekamp-Massey algorithm for Algebraic Geometric codes", IEEE Transactions on Information Theory, vol. 44, No. 4, July 1998, pp. 1353-1368.

[74] Buchberger, B. and Vinkler, F.; "Gröbner bases and applications", Cambridge University Press, 1997.

[75] Sudan, M.; "Decoding of Reed-Solomon codes beyond the error-correction bound", Journal of complexity, 1997, vol.13, No. 1, pp. 180-193.

[76] McEliece, R.J., "The Guruswami-Sudan decoding algorithm for Reed-Solomon codes", IPN Progress report, 2003, pp. 42-153.

[77] Ren, J.; "On the structure of Hermitian codes and decoding for burst errors", IEEE Transactions on Information Theory, vol. 50, no. 11, Nov. 2004, pp. 2850-2854.

[78] Lee, K.;"Fast interpolation decoding of Hermitian codes", submitted to IEEE Transactions on Information Theory, 2013.

[79] Helgert, H.J.;"Alternant codes(linear error correcting codes)", Journal of Information and Control, 1974, vol. 26, pp. 369-380.

[80] Jie, C. and Junying, P.; "Subspace subcodes of generalized Reed-Solomon codes" Acta Mathematica Sinica English Series, vol. 17, no. 4, 2001, pp. 503-508.

[81] Delsarte, P.; "On subfield subcodes of modified Reed-Solomon codes", IEEE Trans. Information Theory, vol.21, no. 5, 1975, pp. 575-576.

[82] Hattori, M; McEliece, R. J. and Solomon, G.; "Subspace subcodes of Reed-Solomon codes", IEEE Trans. Inform. Theory, vol. 44, no. 5, 1998, pp. 1861-1880.

[83] Shibuya, T.; Matsumoto, R. and Sakaniwa, K.; " An improved bound for the dimension of subfield subcodes" IEICE Transactions on Fundamentals of Electronics, Communications and Computer Science, vol. E80 A, no. 5, 1997, pp. 876-880.

[84] Stichtenoth, H;" On the dimension of subfield subcodes", IEEE Transactions on Information Theory, vol.36, no. 1, 1990, pp. 90-93.

[85] Hattori,M.; McEliece, R.J. and Solomon, G.; "Subspace subcodes of Reed-Solomon codes", IEEE Transactions on Information Theory, vol.44, no. 5, 1998, pp. 1861-1880.

[86] Cui, J. and Ying, J., "Subspace subcodes of Generalized Reed-Solomon codes", Acta Mathematicae Applicatae Sinica, vol. 17, no. 4, 2001, pp. 503-508.

[87] Hernando, F.; O'Sullivan; M.; Popovici, E. and Srivastava, S.; "Subfield Subcodes of Generalized Toric codes", IEEE International Symposium on Information Theory, 2010, pp. 1025-1029.

[88] Little, J and Schenck, H.; "Toric surface codes and Minkowski sums", SIAM J. Discrete Math, vol. 20, no. 4, 2006, pp. 999-1014.

[89] Ruano, D.; " On the parameters of r-dimensional Toric codes", Finite Fields Applications, vol. 13, no. 4, 2007, pp. 962-976.

[90] Bras-Amorós, M. and O'Sullivan, M. E.; "Duality for some families of correction capability optimized evaluation codes", Adv. Math. Commun., vol. 2, no. 1, 2008, pp. 15-33.

[91] Sloane, N. J. A.; "A survey of constructive coding theory and a table of binary codes of highest known rate", Discrete Mathematics, vol.3, 1972, pp. 265-294.

[92] Rifa , J.;"Decoding a bit more than the BCH bound", Lecture Notes in Computer Science, no. 781, Springer, ISSN: 3-540-54522-0, 1993, pp. 287-304.

[93] Feng, G.L. and Tzeng, K.; " A generalization of the Berlekamp-Massey algorithm for multisequence shift-register synthesis with applications to decoding cyclic codes" IEEE Transactions on Information Theory, Vol. 37, no. 5, 1991, pp. 1274-1287.

[94] Bours, P.; Janssen, C. M. J.; van Asperdt, M.; van Tilborg, H. C. A.; "Algebraic decoding beyond e_{BCH} of some binary cyclic codes, when $e > e_{\mathrm{BCH}}$" IEEE Transactions on Information Theory, vol. 36, no. 1, 1990, pp. 214-222

[95] Duursma, I. M. and Kötter, R.; "Error-locating pairs for cyclic codes" IEEE Transactions on Information Theory, vol. 40, no. 4, 1994, pp. 1108-1121.

[96] Nilsson, J. E. M; "An algebraic procedure for decoding beyond e_{BCH}", IEEE Transactions on Information Theory, Vol. 42, no. 2, 1996, pp. 649-652.

[97] Pellikaan, R. and Wu, X.; "List decoding of $q-$ary Reed-Muller codes", IEEE Transactions on Information Theory, vol. 50, no. 4, 2004, pp. 679-682.

[98] Roth, R. and Ruckenstein, G.; "Efficient decoding of Reed-Solomon codes beyond half the minimum distance", IEEE Transactions on Information Theory, vol. 46, no. 1, 2000, pp. 246-257.

[99] Sudan, M.; "Decoding of Reed Solomon Codes beyond the Error-correction bound", Journal of Complexity, vol. 13, 1997, pp. 180-193.

[100] Guruswami, V. and Sudan, M.; "Improved decoding of Reed Solomon and algebraic geometric codes", IREE Transactions on Information Theory, vo1. 45, no. 6, 1998, pp. 1757-1767.

[101] Augot, D.; Bardget M. and Faugére, J.C.; "Efficient decoding of (binary) cyclic codes beyond the error correction capability using Gröbner basis", INRIA report, no. 4652, 2002.

[102] Orsini, E. and Sala, M.; " Correcting errors and erasures via syndrome variety", Journal of pure and applied algebra, 2005, pp. 191-226.

[103] Augot, D. and Sendrier, N.; "Idempotents and the BCH bound", IEEE Transactions on information Theory, vol. 40, no. 1, 1994, pp. 204-207.

[104] Tadao, K. and Nobuki, T.; "Some remarks on BCH bounds and minimum weights of binary primitive BCH codes" IEEE Transactions on Information Theory, vol. 15, no. 3, 1969, pp. 408-413.

[105] Wu, Y.; "New list decoding algorithms for Reed-Solomon and BCH codes", IEEE Transactions on Information Theory, vol. 54 , no. 8, 2008, pp. 3611-3631.

[106] Lee, K., O'Sullivan, M. E., "List decoding of Reed-Solomon codes from a Gröbner basis perspective", Journal of Symbolic Computation, vol. 43, no. 9, 2008, 645-658.

[107] beelen, P. and Brander, K.; "Key equations for list decoding of Reed-Solomon codes and how to solve them", Journal of Symbolic Computation, vol. 45, no. 7, 2010, pp. 773-786.

[108] Nielsen, R. and Hoholdt T.; "Decoding Reed-Solomon codes beyond half the minmimum distance", Coding Theory, Cryptography and Related Areas, Buchmann, J.; Hoholdt, T; Stichtenoth T.; Tapia-Recillas, H; Eds. Berlin, Germany:Springer-Verlag (2000), 221-236.

[109] O'Sullivan, M.E.; "The key equation for one-point codes and efficient error evaluation", Journal of Pure and Applied Algebra, vol. 169, no. 2-3, 2002, pp. 295-320.

[110] Akyildiz, I. F.; Su, W.; Sankarasubrama, Y. and Cayciri, E. ; "Wireless Sensor networks: A Survey", Computer Networks Journal, Elsevier, 2002, pp. 393-422.

[111] Agarwal, R. ; Popovici, E. M. and O'Flynn, B. ; "Adaptive Wireless Sensor Networks: A System Design Perspective to Adaptive Reliability", In proceeding of Wireless Communications and Sensor Networks, 2006, pp. 216-225.

[112] Karvonen, H.; Shelby, Z. and Pomalaza-Ráez, C.; "Coding for Energy Efficient Wireless Embedded Networks", International Workshop on Wireless Ad-hoc Networks, 2004, pp. 300-304.

[113] Karvonen, H. and Pomalaza-Ráez, C. "Coding for Energy Efficient Multihop Wireless Sensor Networks", In Proceeding of Nordic Radio Symposium, 2004, pp. 1-5.

[114] Chen, P.; O'Dea, B. and Callaway, E.; "Energy Efficient System Design with Optimum Transmission Range for Wireless Ad Hoc Networks", In proceeding of IEEE International Conference on Communications (ICC) , 2002, pp. 945-952.

[115] Howard, S.L; Schlegel, C. and Iniewski, K. "Error control coding in low-power wireless sensor networks: When Is ECC Energy Efficient?", EURASIP Journal on Wireless Communications and Networking, 2006, pp. 1-14.

[116] Proakis, J.G.; "Digital communications", McGraw-Hill, New York, Fourth Edition, 2001.

[117] Kashani, Z.H. and Shiva, M.; "Channel Coding in Multi-hop Wireless Sensor Network", In proceeding of International Conference on ITS Telecommunications (ITST), 2006, pp. 21-23.

[118] Agarwal, R.; Koetter, R. and Popovici, E.M.; "A Low Complexity Algorithm and Architecture for Systematic Encoding of Hermitian Codes", In proceeding of IEEE International Symposium on Information Theory (ISIT), 2007, pp. 1336-1340.

[119] Parvaresh, F. and Vardy, A.; "Multivariate interpolation decoding beyond the Guruswami-Sudan radius," In proceeding of Annual Allerton Conference on Communications, Control, and Computing, 2004.

[120] Srivastava, S.; Lee, K and Popovici, E.; "Fast parallel implementation of Hermitian decoder based on interpolation", submitted to IET Communications, 2013.

[121] Pinero, F. and Janwa, H. ; "On the subfield-subcodes of Hermitian Codes", Designs, Codes and Cryptography, 2012, accepted.