UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

# PARALLEL SUBSPACE SUBCODES OF REED-SOLOMON CODES FOR MAGNETIC RECORDING CHANNELS

A DISSERTATION

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

Degree of

DOCTOR OF PHILOSOPHY

By

HAN WANG
Norman, Oklahoma
2010

Downloaded from SHAREOK. Persistent Handle URL: https://hdl.handle.net/11244/318825

PARALLEL SUBSPACE SUBCODES OF REED-SOLOMON CODES FOR MAGNETIC RECORDING CHANNELS

A DISSERTATION APPROVED FOR THE
SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

BY

Dr. Joao R. Cruz, Committee Chair

Dr. Marilyn Breen

Dr. Joseph Havlicek

Dr. Thordur Runolfsson

Dr. Tian-You Yu

© Copyright by HAN WANG 2010
All Rights Reserved.

*To the love, charity, and harmony of the world*

# ACKNOWLEDGEMENTS

First, I would like to express my greatest gratitude to my academic advisor Dr. Joao R. Cruz. Thank you most sincerely for the guidance, tolerance and advices during my Ph.D. study. I am always admiring your integrity, ingenuity and diligence. What you have taught me is far beyond the research itself, and I do believe it will benefit all my career life. Thank you for opening a door to a great treasure of coding theory.

Second, special great thanks to Dr. Havlicek for his kind helps to solve visa issues for my career.

I would like to thank Dr. Marilyn Breen, Dr. Joseph Havlicek, Dr. Thordur Runolfsson, and Dr. Tian-You Yu for not only serving as my graduate committee members, but also giving the great suggestions and knowledge for this work. Thank you all for your great courses. I give my special great appreciations to Dr. Marilyn Breen for elegant and inspiring discussions on the graph and convex based approach in finding the minimum convex of codewords over Galois field.

I would like to thank Dr. Runsheng He so much for recommending me here and some important advices for my career life. Best wishes to your business.

I would like to thank Dr. Haitao Xia for all kinds of helps these years. For years, you have kept offering good advices, which helped me a lot. Thank you!

I would like to thank Dr. Cheng Zhong for interesting discussions in coding theory and applications on JPWL. Your family is so nice to me, and gave me one of the best memories in Norman during the past years.

I would like to give specials thanks to all my fellow students at the Communications Signal Processing Laboratory for helpful discussions, especially to Wu Chang. I am so glad to be a member of this wonderful family.

I would like to thank Lynn hall for all the helps during these five years. Thank you sincerely.

Sincere thanks to my girl friend Chao for your encouragements and companies these years. We experienced and shared every great moment, and will build the splendid future. Thank you for your loves.

To a special friend of mine, thank for the memory during my first two years here. Wish you well and happy.

Great special thanks to Chang Cheng. You helped me to conquer the most difficult time during my study here. Thank you so so so much.

Great thanks to Bin Qin. You helped me a lot.

Thanks to people here who helped me and cared about me these years.

Finally, I want to dedicate this Ph.D. dissertation to my dear parents. You two are really cool. Thank you for what you have done to make me strong and wise in all aspects. Thank you for your loves.

# TABLE OF CONTENTS

ABSTRACT.....xix

**Chapter Introduction …….....1**

1.1 Motivation…….....2

1.2 Problem statement…….....4

1.3 Approach……..…...5

1.4 Contribution……..…….6

1.5 Outline……....6

**Chapter 2 Reed-Solomon codes: Encoding and decoding algorithms…..…….....8**

2.1 Introduction……..……9

2.2 Definition and encoding methods…….……10

2.2.1 Polynomial evaluation approach…….……..……11

2.2.2 Generator polynomial approach ……...……11

2.2.3 Discrete Fourier transform (DFT) over Galois Field……..……...12

2.3 Algebraic decoding algorithms……14

2.3.1 Hard-decision decoding algorithms……...13

2.3.2 Soft-decision decoding algorithms……20

2.4 Conversions and connections of different definitions…..……24

**Chapter 3 Chase Algorithms for Soft-Decision Decoding……..26**

3.1 Low complexity Chase (LCC) soft-decision decoding……27

3.1.1 Introduction……27

3.1.2 Low-complexity Chase-type decoding algorithm...……29

3.1.2.1 Complexity reduction……30

3.1.2.2 Polynomial interpolation and factorization……30

3.1.2.3 Algorithm analysis……..31

3.1.3 Simulation results and analysis……..33

3.1.4 Conclusions……....36

3.2 A Chase-GMD algorithm for soft-decision decoding……37

3.2.1 Introduction……....37

3.2.2 Interpolation-bases RS decoding algorithms and their recursive property……...38

3.2.3 Performance analysis of forward recursive algorithms……..…40

3.2.4 A Chase-GMD type algorithm using reliability information…..…...…44

3.2.5 Simulation results……48

3.2.6 Conclusion……53

**Chapter 4 Subcodes of Reed-Solomon codes with a parallel subcode structure……55**

4.1 Subspace subcodes of RS codes……..……56

4.2 Subcodes of RS codes with a parallel subcode structure…….……60

4.2.1 Class-I subcodes $\mathbb{C}_{\mathrm{I}}$……...60

4.2.2 Class-II subcodes $\mathbb{C}_{\mathrm{II}}$……...62

4.2.2.1 $\mathbb{C}_{\mathrm{II}}^{v}$ codes……..64

4.2.2.1.1 $\mathbb{C}_{\mathrm{II}}^{v=1}$ codes……...55

4.2.2.2 $\mathbb{C}_{\mathrm{II}}^{\langle v_1|v_2|\cdots|v_{max}\rangle}$codes……75

4.2.2.3 Multiple-stage $\mathbb{C}_{\mathrm{II}}^{\langle v_1 \supset v_{1.1} \supset v_{1.2} \supset \cdots | v_2 \supset v_{2.1} \supset v_{2.2} \supset \cdots | \cdots | v_{max} \supset v_{max\ .1} \supset v_{max\ .2} \supset \cdots \rangle}$ codes……..…78

4.3 Simulation results…..……80

4.4 Summary……...84

**Chapter 5 Iterative parallel local decoding of LDPC+RS concatenated codes...86**

5.1 Introduction……...87

5.2 Concatenated codes…….....88

5.3 Conventional LDPC+RS concatenation……...90

5.4 New LDPC+RS ($\mathbb{C}_{\mathrm{II}}^{v=1}$) concatenation system……...……..92

5.5 Iterative parallel local decoding of LDPC+RS concatenation system…..94

5.5.1 Avoiding trapping sets……...95

5.5.2 Iterative parallel local decoding algorithm ……...……99

5.6 Performance…….……..100

5.6.1 Sector length code……100

5.6.2 Big sector (4KB) sectors…….....105

5.7 Summary and recommendations..……..109

**Chapter 6 Conclusions……...112**

**Bibliography……115**

**APPENDIX A. PROOF OF THEOREM 4.2-5……127**

**APPENDIX B. EXAMPLE OF $\mathbb{C}_{II}^{v=1}$'S "LOCAL" BIT ERROR CORRECTION……..….133**

**APPENDIX C. EXAMPLE OF $\mathbb{C}_{\mathrm{II}}^{\langle v_1|v_2|\cdots|v_{max}\rangle} = \mathbb{C}_{\mathrm{II}}^{\langle v_1=1|v_2=2\rangle}$……..…..135**

**APPENDIX D. CONSTRUCTION METHOD FOR $\mathbb{C}_{\mathrm{II}}^{\langle v_1\supset\cdots|v_2\supset\cdots|\cdots|v_{max}\supset\cdots\rangle}$...141**

**APPENDIX E. ENCODING SCHEME FOR NEW LDPC+$\mathbb{C}_{\mathrm{II}}^{v=1}$ CONCATENATION SYSTEM……...……..143**

**APPENDIX F. ITERATIVE PARALLEL LOCAL DECODING WORKFLOW FOR $\mathbb{C}_{\mathrm{II}}^{\langle v_1|v_2|\cdots|v_{max}\rangle}$……...…….146**

# LIST OF TABLES

Table 3.1 Maximum number of multiplications required for decoding……....…51

Table 3.2 Average number of interpolation and factorizations……..53

# LIST OF FIGURES

Fig. 3.1. Performance of $RS(440,410)$ on a PMRC equalized to an optimal GPR4 target with 90% jitter noise. The target is [1 0.307 -0.039 -0.002] and user density 0.938……..……..…34

Fig. 3.2. Number of interpolations/factorizations as a function of $\eta$ for one sample execution of the LCC algorithm..……..……....35

Fig. 3.3. Performance of $RS(440,410)$ on a PMRC equalized to an optimal GPR4 target with 90% jitter noise. The target is [1 0.471 -0.068 -0.004] and user density 1.1257…….……36

Fig. 3.4. The reliability-sorted received vector and its interpolation order……....46

Fig. 3.5. Performance of RS (440, 410) on a PMRC equalized to an optimal GPR4 target with 90% jitter noise, and user density $D_u$=0.9381……..…49

Fig. 3.6. Comparison of the number of interpolated points between LCC and Chase-GMD algorithms for RS (440, 410) on a PMRC equalized to an optimal GPR4 target with 90% jitter noise and user density $D_u$ =0.9381…….……...…50

Fig. 3.7. Factorization success ratio vs. the number of GMD interpolation point pairs at various SNRs……..……...……52

Fig. 4.1. The binary 3-tuple of $RS(7,5,3)$codeword: (2,2,3,0,1,3,1) ……...…57

Fig. 4.2. The binary 2-tuple of $RS(7,5,3)$ codeword $(2,2,3,0,1,3,1)$……57

Fig. 4.3. The package structure of several parallel SSRS codes……60

Fig. 4.4. Class-I subcodes $\mathbb{C}_{\mathrm{I}}$ of RS codes……61

Fig. 4.5. Decoding structure for $\mathbb{C}_{\mathrm{I}}$ codes……62

Fig. 4.6. Class-II subcodes of RS codes……63

Fig. 4.7. $\mathbb{C}_{\mathrm{II}}^{v=1}$ subcodes of RS codes……66

Fig. 4.8. Vardy-Be'ery decomposition of the binary image of an RS generator matrix……67

Fig. 4.9. Wolf's star shape of trellis……68

Fig. 4.10. Removal of the glue generate matrix……68

Fig. 4.11. The multi-level decoding architecture for subcode $C(N,k,D,d)$ ……72

Fig. 4.15. An example of three-level decoding structure for subcode $C(N,k,D,d)$..73

Fig. 4.16. $\mathbb{C}_{\mathrm{II}}^{\langle v_1|v_2|\cdots|v_{max}\rangle}$ code……76

Fig. 4.14. Multiple parallel structures for $\mathbb{C}_{\mathrm{II}}^{\langle v_1|v_2|\cdots|v_{max}\rangle}$ with different $\langle v_1|v_2|\cdots|v_{max}\rangle$……78

Fig. 4.15. Multiple stage structures of $\mathbb{C}_{\mathrm{II}}^{\langle v_1\supset\cdots|v_2\supset\cdots|\cdots|v_{max}\supset\cdots\rangle}$……79

Fig. 4.16. Performance of $C(7,4,3,3)$ on a PMRC equalized to an optimal GPR4 target with 90% jitter noise. The user density is 1.5309. The code rate is 0.571.....81

Fig. 4.17. Performance of $C(460,410,11,11)$ over $GF(1024)$ on PMRC equalized to an optimal GPR4 target with 90% jitter noise. The user density is 0.9811. The cod rate is 0.891.....82

Fig. 4.18. Performance of $C(1023,973,25,9)$ on a PMRC channel equalized to an optimal GPR4 target with 90% jitter noise. The user density is 0.919. The code rate is 0.951.....84

Fig. 5.1. RS+CC Concatenated coding system.....89

Fig. 5.2. Conventional LDPC+RS coding system.....90

Fig. 5.3. LDPC+PSSRS.....92

Fig. 5.4. New LDPC+RS($\mathbb{C}_{\mathrm{II}}^{v=1}$) concatenation system.....93

Fig. 5.5. The decoding workflow of the new LDPC+RS ($\mathbb{C}_{\mathrm{II}}^{v=1}$) concatenation system.....94

Fig. 5.6. A (3,1) trapping set.....96

Fig. 5.7. "Virtual dynamic" check nodes added by $\mathbb{C}_{\mathrm{II}}^{v=1}$ codes.....97

Fig. 5.8. Optimization of number of turbo iterations and inner BP iterations.....101

Fig. 5.9. Optimization of the overall rate of the concatenation system.....102

Fig. 5.10. Comparison to the conventional concatenation system.……103

Fig. 5.11. Performance of iterative parallel local decoding algorithm for LDPC+RS system on a PMRC equalized to an optimal GPR4 target with 90% jitter noise. The user density is 1.0757……...105

Fig. 5.12. Performance of 4K-Byte sector length code of $LDPC + RS(3340, 2731)$ over $GF(4096)$ on a PMRC equalized to an optimal GPR4 target with 90% jitter noise. User density is 1.0757……...….107

Fig. 5.13. Performance of 4K-Byte sector length code of $LDPC + RS(3340, 2731)$ over $GF(4096)$ on a PMRC equalized to an optimal GPR4 target with 90% jitter noise. User density is 1.5073……108

Fig. B.1. The binary 3-tuple of $RS(7,5,3)$ codeword $(1,4,0,0,1,5,4)$…… 133

Fig. B.2. The binary 3-tuple of $RS(7,5,3)$ codeword: $(1,4,0,0,1,5,4)$…….……133

Fig. B.3. Example of $\mathbb{C}_{\mathrm{II}}^{v=1}$'s "local" bit error correction.…….……..134

Fig. C.1. $\mathbb{C}_{\mathrm{II}}^{\langle v_1=1 | v_2=2\rangle}$ structure I……..135

Fig. C.2. $\mathbb{C}_{\mathrm{II}}^{\langle v_1=1 | v_2=2\rangle}$ structure II…….135

Fig. C.3. $GF(16)$ element decomposition with basis $\mathcal{B}' = \{1, \alpha^2\}$ over $GF(4)$…139

Fig. E.1. Encoding scheme $I$…… 144

Fig. E.2. Encoding scheme $ES1.2.II$……..145

Fig. E.3. Encoding scheme using the $Q$-ary LDPC code……145

Fig. E.4. Binary $m$-tuple image of the scheme using a $Q$-ary LDPC codeword…..146

Fig. E.5. Encoding scheme using a multiple codeword interleaver……..146

Fig. F.1. Iterative parallel local decoding for LDPC+RS ($\mathbb{C}_{II}^{\langle v_1|v_2|\cdots|v_{max}\rangle}$) system..147

# ABSTRACT

Read channel architectures based on a single low-density parity-check (LDPC) code are being considered for the next generation of hard disk drives. However, LDPC-only solutions suffer from the error floor problem, which may compromise reliability, if not handled properly. Concatenated architectures using an LDPC code plus a Reed-Solomon (RS) code lower the error-floor at high signal-to-noise ratio (SNR) at the price of a reduced coding gain and a less sharp waterfall region at lower SNR. This architecture fails to deal with the error floor problem when the number of errors caused by multiple dominant trapping sets is beyond the error correction capability of the outer RS code. The ultimate goal of a sharper waterfall at the low SNR region and a lower error floor at high SNR can be approached by introducing a parallel subspace subcode RS (SSRS) code (PSSRS) to replace the conventional RS code. In this new LDPC+PSSRS system, the PSSRS code can help localize and partially destroy the most dominant trapping sets. With the proposed iterative parallel local decoding algorithm, the LDPC decoder can correct the remaining errors by itself. The contributions of this work are: 1) We propose a PSSRS code with parallel local SSRS structure and a three-level decoding architecture, which enables a trade off between performance and complexity; 2) We propose a new LDPC+PSSRS system with a new iterative parallel local decoding algorithm with a 0.5dB+ gain over the conventional two-level system. Its performance for 4K-byte sectors is close to the multiple LDPC-only architectures for perpendicular magnetic

recording channels; 3) We develop a new decoding concept that changes the major role of the RS code from error correcting to a “partial” trapping set destroyer.

# Chapter 1

# Introduction

## 1.1 Motivation

Error-correcting codes (ECCs) are currently used in most digital communication systems, including the read channels of hard disk drives (HDDs). The history of modern ECCs starts with the single error-correcting Hamming code [1], and progresses to multiple error-correcting codes, such as the Bose-Chaudhuri-Hocquenghem (BCH) code [2], which is not maximum distance separable (MDS). Later, Reed and Solomon [3] introduced an MDS code, the Reed-Solomon (RS) code, which has a unique and elegant algebraic structure. More recently, non-algebraic codes such as turbo codes [4], [5], and low-density parity-check (LDPC) codes [6], which utilize soft channel output reliability information for their iterative decoding algorithms, have become widely used.

There is always a tradeoff between complexity and coding gain. The soft-decision decoding (SDD) algorithms for RS codes perform very well for low-rate codes with error-correction capability beyond half the minimum distance. However, the coding gain for high-rate RS codes using SDD is much smaller. The computational complexity of SDD for RS codes becomes prohibitive with increasing code length and error correction capability. Modern ECCs must provide large coding gains with moderate complexity. The current trend is to replace current RS coded systems with LDPC-coded systems because of their large coding gain in random noise. LDPC codes are the best error correcting codes for approaching the capacity of the channel, thus they are the codes of choice for the next generation of many applications,

including magnetic recording systems. However, the performance of LDPC codes at high signal-to-noise ratio (SNR) is not accurately known, and they exhibit an error floor when decoded by the commonly used belief propagation (BP) algorithm. RS codes have a more elegant structure than LDPC codes, and their decoding performance is well determined at high SNRs. It is well known that the error floor of LDPC codes at high SNRs is caused by certain error patterns, which require a lot of effort to avoid. The error floor problem might preclude the usage of LDPC codes in systems, which require extremely high reliability.

One example of the difficulty in transitioning from RS to LDPC codes is the future generation of HDDs with 4K-Byte sectors [8]-[11]. For codes this long, the complexity of LDPC codes could be prohibitive. The "error-floor" problem could also be an issue at very high recording density, which is required to build high capacity storage systems with good "format efficiency" [8]-[11]. Efforts have been made to handle these problems. For example, it has been identified that there exist some error patterns which are the major cause of the "error-floor" [12]-[18]. To pre-detect and avoid these patterns, they are usually collected by computer and hardware simulations [13], to help lower the "error-floor". There is no evidence, however, that this method can theoretically solve the "error-floor" problem. Another approach is to use a concatenated system consisting of an RS code as an outer code and an LDPC code as an inner code (LDPC+RS system). This method can lower the "error-floor" at a cost of a reduced coding gain compared to an LDPC-only system. (The LDPC-

only system refers to a one-level coded system.) The conventional LDPC+RS system can lower the "error-floor" due to the outer RS code. In some cases, however, the LDPC code fails to decode for certain error patterns, creating more errors (error propagation) than the error-correction capability of the RS code. There is no easy way for the current conventional LDPC+RS system to avoid this situation by excluding these error patterns.

This motivates us to find ways to handle the error floor problem by designing new LDPC codes with local error correction capability that can handle certain error patterns, and by designing new LDPC+RS concatenation systems, in which the RS code can help dealing with certain error patterns.

## 1.2 Problem Statement

Read channel architectures based on a single LDPC code are being considered for the next generation of HDDs. However, LDPC-only solutions suffer from the error floor problem, which may compromise reliability if not handled properly. Concatenated architectures using an LDPC code plus an RS code lower the error-floor at high SNRs at a cost of a reduced coding gain and a less sharp waterfall region at lower SNRs. This architecture fails to deal with the error floor problem when the number of errors caused by multiple dominant error patterns is beyond the error correction capability of the outer RS code. Conventional LDPC+RS systems

for long codes may not be able to use SDD algorithms to improve the error correction capability of the RS code, because of their computational complexity.

This problem can be solved by changing the role of the RS code from that of error correction to removal of certain error patterns. An LDPC decoding failure is usually caused by some error patterns, and it is not necessary to correct all the errors in the pattern. If the troublesome error pattern is avoided, the LDPC decoder itself can correct the remaining errors instead of the RS decoder. The role of RS codes here is to help the LDPC decoder to avoid certain error patterns and let the LDPC decoder itself correct the remaining errors. Because the LDPC code is now able to deal with the remaining errors, it does not need a more powerful RS decoder with a high decoding computational complexity. The central idea is to use the RS decoder to remove problematic error patterns.

We propose a new code which can provide a local error correction capability in addition to its global error correction capability.

## 1.3 Approach

Conventional RS codes do not have local error correction capability. A new code is required to provide a local error correction capability as well as the global error correction capability. We propose new parallel subspace subcodes of RS (PSSRS) codes to replace the conventional RS codes. We utilize this PSSRS code in a new LDPC+PSSRS concatenated system. In this new concatenated system, the PSSRS

code can help locate and partially remove some problematic error patters. With the proposed iterative parallel local decoding algorithm, the LDPC decoder can correct the remaining errors by itself.

## 1.4 Contribution

The major contributions of this work are: 1) We propose a PSSRS code and a three-level decoding architecture, which enables a tradeoff between performance and complexity; 2) We propose a new LDPC+PSSRS system with a new iterative parallel local decoding algorithm which provides some gains over the conventional two-level system. Its performance for long (4K-Byte) sectors is close to the multiple LDPC-only architectures for perpendicular magnetic recording channels (PMRCs); 3) We develop a new decoding concept that changes the major role of the RS code from error correction to error pattern removal.

## 1.5 Outline

In Chapter 2, we present a brief review of RS codes, including encoding and decoding algorithms, and basic definitions.

In Chapter 3, we evaluate the performance of a low-complexity Chase (LCC) algorithm on PMRCs and propose a new interpolation-based Chase decoding algorithm based on the Chase and generalized minimum distance (Chase-GMD)

algorithms, which has the property of having decreasing complexity with increasing SNRs.

In Chapter 4, we provide an introduction to subspace subcodes of RS (SSRS) codes, which are a generalization of generalized BCH (GBCH) [2] and trace-shortened RS (TSRS) codes [19]. Then we propose new PSSRS codes, which have several parallel local SSRS structures. We divide PSSRS codes into two major types: Type I and Type II with unequal and equal error correction capabilities, respectively. We focus mostly on Type II codes, because they are easy to encode and decode.

In Chapter 5, we propose a new LDPC+PSSRS concatenated system. For this concatenated system, we provide a new iterative parallel local decoding algorithm to let the LDPC decoder replace the RS decoder in the elimination of the remaining errors from the initial burst of errors from the output of the LDPC decoder. We develop a new decoding framework that changes the major role of the RS code from error correction to error pattern removal. We evaluate the performance of the new LDPC+PSSRS system on PMRCs with standard and long sectors.

In Chapter 6, we provide a conclusion and discuss possible directions for future work.

# Chapter 2

# Reed-Solomon Codes: Encoding and Decoding Algorithms

## 2.1 Introduction

The emergence of the single-error correction Hamming code marked the beginning of modern coding theory. However, the success of ECCs was not widely demonstrated until the RS codes were introduced in 1960 as MDS polynomial codes [3], which have been the most successful and dominant codes for decades. RS codes are widely used in satellite communications, storage systems such as HDDs, compact discs (CDs), digital video discs (DVDs), blue-ray discs (BDs), solid state drive (SSD), JPEG-2000 for wireless (JPWL), etc.

The success of RS codes is due to their interesting properties, namely:

1) Cyclic linear MDS block codes.

2) Multiple error correction capability.

3) Non-binary codes capable to correct burst errors.

4) Easy to encode in a systematic way.

5) Flexible to decode with various decoding algorithms.

RS codes are a special subclass of the BCH codes, which were also proposed in 1960 as well as RS codes and at the same time as LDPC codes. Both RS and BCH codes are multiple-error correction codes. However, RS codes have a more elegant algebraic structure than BCH codes. The relationship between RS and BCH codes will be further discussed in this work. In this chapter we summarize the basic knowledge on RS codes and introduce the latest SDD algorithms. Unlike other works, we will focus on introducing the RS codes from a BCH-code point of view.

This chapter is organized as follows. In Section 2.2, we briefly introduce the definition and several encoding methods for RS codes. In Section 2.3, we introduce decoding algorithms with a special focus on the Welch-Berlekamp and SDD algorithms. In Section 2.4, we discuss a special relationship between RS codes and BCH codes.

## 2.2 Definition and encoding methods

RS codes are linear cyclic MDS block codes with length $n$, and dimension $k$ over a finite field $GF(q^m)$.

*Definition*: An RS code is the set of all codewords $c$ of length $n = q^m - 1$, with the alphabet of $GF(q^m)$ such that the codeword $c$ satisfies $Hc^T = 0$, where $H$ is the parity-check matrix of a primitive non-binary BCH code over $GF(q^m)$.

In this work, the default finite field is $GF(2^m)$ for practical purposes, unless we specify it otherwise.

*Notation*: Let $RS(n, k)$ denote an RS code with length $n$, and dimension $k$, over $GF(2^m)$. The transmitted messages $m = (m_0, m_1, \cdots \cdots, m_{k-1})$ are expressed in polynomial form as

$$m(x) = m_0 + m_1 x + m_2 x^2 + \cdots + m_{k-1} x^{k-1}. \tag{2.1}$$

There are several distinct encoding methods such as the polynomial evaluation, generator polynomial, and the discrete Fourier transform (DFT) on a Galois field [1], [2].

### 2.2.1 Polynomial evaluation approach

In [3], Reed and Solomon introduced a new non-binary cyclic block code and an original encoding method, which is known as the polynomial evaluation approach. The symbol values of the codewords are generated by evaluating the message polynomial in (2.1) with distinct non-zero elements of $GF(2^m)$.

*Polynomial evaluation*

Let $\alpha$ be a primitive element of $GF(2^m)$, and $\alpha^i$'s are distinct non-zero elements of $GF(2^m)$. The RS codewords $c$ are generated as:

$$c = (c_0, c_1, \cdots, c_{n-1}) = (m(\alpha^0), \cdots, m(\alpha^{n-1})). \tag{2.2}$$

### 2.2.2 Generator polynomial approach

An RS codeword can be generated in polynomial form as [1], [2],

$$c(x) = g(x)m(x), \tag{2.3}$$

where $g(x)$ is the generator polynomial, which was initially utilized in the generation the BCH codes, and is the product of the minimal polynomials of $2t$ consecutive elements of the form $\alpha^i, i = 1, \cdots, 2t$, where $t$ is the number of error corrections. Let

$M_{(i)}(x)$ be the minimal polynomial of $\alpha^i$ over $GF(q^m)$, then $g(x) = LCM(M_{(1)}(x)M_{(2)}(x)M_{(3)}(x)\cdots M_{(2t)}(x))$. We know that the minimal polynomial of $\alpha^i$ over $GF(q^m)$ is $(x-\alpha^i)$, so $g(x) = (x-\alpha^1)\cdots(x-\alpha^{2t})$. Codewords generated using (2.3) are non-systematic and the encoding method is inefficient.

RS codes can be efficiently encoded in a systematic way as follows,

$$c(x) = r(x) + x^{n-k}m(x), \tag{2.4}$$

where

$$r(x) = x^{n-k}m(x)\ mod\ g(x). \tag{2.5}$$

The systematic encoding of the generator polynomial approach is the most efficient encoding method for RS codes, which is widely used in modern applications. We can get the shortened RS codes easily by inserting a certain length of zeros into the message part to get the parity. It is not necessary to send these zeros, because they are known. Thus this approach can generate the shorten RS codes, which have a very flexible code length.

### 2.2.3 Discrete Fourier transform (DFT) over a Galois field

The DFT can be utilized to implement the encoding of RS codes [1], [2]. The definition of the DFT of a vector $v = (v_0, v_1, \cdots, v_{n-1})$ over a Galois field is

$$c \triangleq DFT(v) = (c_0, c_1, \cdots, c_{n-1}),$$

where $c_i = \sum_{j=0}^{n-1} v_j \alpha^{ij}, i = 0, \cdots, n-1$ and $\alpha$ is the primitive element of $GF(q)$. Then

$$v = IDFT(c)$$

$$v_i = \frac{1}{n} \sum_{j=0}^{n-1} c_j \alpha^{-ij} . \tag{2.6}$$

Here, the non-systematic encoding of RS codes using the DFT has two steps. The first step adds $n-k$ zeros to the message, $m = (m_0, m_1, \cdots, m_{k-1})$, and gets $v = (m_0, m_1, \cdots, m_{k-1}, 0, \cdots, 0)$. The second step performs the DFT on $v$ to get

$$c = DFT(v) = DFT(m_0, m_1, \cdots, m_{k-1}, 0, \cdots, 0). \tag{2.7}$$

## 2.3 Algebraic decoding algorithms

In 1960, Peterson introduced a decoding algorithm for the binary BCH code, which is extremely useful to correct a small number of errors [1], [2]. For a large number of errors, this algorithm becomes impractical because of the prohibitive complexity. In 1967, Berlekamp invented the most efficient decoding algorithm for both binary and non-binary BCH codes [1]. An equivalent method to Berlekamp's algorithm was proposed by Massey utilizing a linear feedback shift-register (LFSR) [1]. For non-binary BCH codes and RS codes, the error magnitudes can be solved by using Forney's algorithm [1].

Traditional algebraic algorithms can only correct up to $t$ errors. Since the 1990's, two new algebraic decoding algorithms using the channel reliability information have emerged, whose error correction capabilities go beyond the traditional correction bound $t$. The first is the Guruswami-Sudan (GS) algorithm [13] and the second is the Koetter-Vardy (KV) algorithm [14]. Both algorithms have very good performance for short low-rate codes. However the coding gain for long high-rate codes is much smaller, and the complexity to achieve a large gain is prohibitive, which seriously limits the practical application of these algorithms.

### 2.3.1 Hard-decision decoding algorithms

Hard-decision decoding algorithms depend only on the hard-decision vector of the channel outputs. There are two major categories. The algorithms in the first category such as Peterson's, Perterson-Gorensteian-Zierler's, Berlekamp-Massey's, and the Euclidean algorithm are all syndrome-based [1]. The second category is interpolation-based, such as the Welch-Berlekamp (WB) algorithm.

*Syndrome-based decoding general procedure*

1) *Compute the syndromes:* $S_j$.

Given the binary codeword polynomial $c(x) = c_0 + c_1 x + \cdots + c_{n-1} x^{n-1}$, the received polynomial $r(x) = r_0 + r_1 x + \cdots + r_{n-1} x^{n-1}$ is the sum of $c(x)$ and

$e(x)$, where $e(x) = e_{l_1}x^{l_1} + e_{l_2}x^{l_2} + \cdots + e_{l_v}x^{l_v}$ is the error polynomial, and $v$ is the number of errors.

$$S_j = r(\alpha^j) = c(\alpha^j) + e(\alpha^j) = e(\alpha^j) = \sum_{k=1}^{v} e_{l_k}(\alpha^j)^{l_k},$$

$$= \sum_{k=1}^{v} (\alpha^j)^{l_k} = \sum_{k=1}^{v} X_{l_k}^j, \qquad j = 1,2,\cdots,2t, \tag{2.8}$$

where $\{X_{l_k} = e_{l_k}{}^{-1}\}$ is the set of error locators.

2) *Determine the error-locator polynomial,*

$$\Lambda(x) = \prod_{k=1}^{v}(1 + X_{l_k}x) = \Lambda_0 + \Lambda_1 x + \cdots + \Lambda_v x^v. \tag{2.9}$$

3) *Find the roots of $\Lambda(x)$ using Chien's search [24].*

4) *Find the error values using Forney's algorithm [25].*

*Berlekamp-Massey (BM) algorithm*

The direct solution of the matrix form decoding algorithms such as Peterson's, and Perterson-Gorensteian-Zierler's is only suitable for codes with a small $t$. For a large $t$, the complexity of computing the inverse matrix becomes prohibitive. It needs a more efficient method, which was introduced by Berlekamp for decoding non-binary

BCH codes. Berlekamp's algorithm can obtain $\Lambda(x)$ by finding the polynomial satisfying the following key equation,

$$[1 + S(x)]\Lambda(x) \equiv \Omega(x) \bmod x^{2t+1}, \tag{2.10}$$

where $S(x)$ is the infinite-degree syndrome polynomial and $\Omega(x)$ is the error magnitude polynomial [1]. Berlekamp's algorithm is an iterative algorithm that solves the following equation [1]

$$[1 + S(x)]\Lambda^{(2k)}(x) \equiv 1 + \Omega_2 x^2 + \cdots + \Omega_{2k} x^{2k} \bmod x^{2k+1}, where\ k = 1\ to\ t.$$

Berlekamp's algorithm also works most efficiently for finding the $\Lambda(x)$ polynomial with the LFSR proposed by Massey [26]. The fundamental idea is based on

$$S_j \Lambda_0 + S_{j-1} \Lambda_1 + \cdots + S_{j-v} \Lambda_v = 0, \qquad \Lambda_0 = 1.$$

It means that the later syndrome can be described recursively in the sum-product form of the earlier coefficients of $\Lambda(x)$ and syndromes. The idea is to find the $\Lambda(x)$, which enables the same LFSR's outputs as the $2t$ syndromes. For details of the BM algorithm, please refer to [1], [2], [27].

*Welch-Berlekamp (WB) algorithm*

As we know, the syndrome-based algorithms need to calculate the syndromes for solving the key-equations for RS and BCH codes, whose syndrome sequences are

introduced by the received vector. In 1983, Welch and Berlekamp proposed a new decoding algorithm [28] for RS codes, which does not need to calculate the syndromes. The Welch-Berlekamp (WB) algorithm is the first interpolation-based algebraic hard-decision algorithm. This new decoding algorithm proposed a new key-equation for decoding RS codes [28], namely

$$p_i \alpha^i N(\alpha^i) = \dot{r}_i W(\alpha^i), \tag{2.11}$$

with

$$de\,g(N(x)) < de\,g(W(X)) = e \leq \frac{n-k}{2}. \tag{2.12}$$

Now what we need to do is to find out two polynomials, $N(x)$ and $W(x)$, such that the polynomial $xN(x) = yW(x)$ goes through all these data pairs: $\{(x,y)|x = \alpha^i, y = \frac{\dot{r}_i}{p_i}, i = 0,\cdots,n-k-1\}$, where $\dot{r}_i$ and $p_i$ are coefficients of polynomials $\dot{r}(x)$ and $p(x)$ defined in (2.13) and (2.14).

*WB algorithm decoding procedure:*

1) *Calculate the $p_i$ and $\dot{r}_i$ from the received vector $r(x)$.*

$$p(x) = \frac{g(x)}{x-1} = \prod_{i=1}^{n-k-1} (x - \alpha^i) = \sum_{i=0}^{n-k-1} p_i x^i \tag{2.13}$$

$$\dot{r}(x) = r(x)\ mod\ g(x). \tag{2.14}$$

2) *Use* $\{\alpha^i, p_i, \dot{r}_i\}$ *to calculate* $N(x)$ *and* $W(x)$,

$$p_i \alpha^i N(\alpha^i) = \dot{r}_i W(\alpha^i), i = 0, \cdots, n-k-1,$$

$$de\ g(N(x)) < de\ g(W(X)) \leq \frac{n-k}{2}.$$

3) *Use* $N(x)$ *and* $W(x)$ *to calculate the error locations and error values*.

Using Chien search [24] to find the roots of $W(x)$, which are the error locations.

The error values are $Y_i = f(Z_i)\frac{N(Z_i)}{W'(Z_i)}$, here $W'(x)$ is the formal derivative of $W(x)$ over $GF(q)$, and

$$f(Z) = Z^{-b} \sum_{i=0}^{n-k-1} \frac{p_i \alpha^{i(b+1)}}{\alpha^i - Z}\ , \tag{2.15}$$

$$W'(x) = \sum_{i=0}^{e} \alpha^i \prod_{k \neq i} (\alpha^k x - 1). \tag{2.16}$$

The WB algorithm can also be expressed in matrix form, which will not be addressed in this work.

In [29], one of the modification proposed by Gemmell and Sudan later leads to the GS algorithm [22]. From [29] and [30], the authors discovered that if $r_i = c_i$, then $r_i = m(\alpha^i)$. If $r_i \neq c_i$, then $W(\alpha^i) = 0$. Hence, we can get this equation,

$$W(\alpha^i) r_i = W(\alpha^i)\, m(\alpha^i), i = 0, \cdots, n-1. \tag{2.17}$$

Use $P(x) = W(x)\, m(x)$, and the key equation becomes

$$W(\alpha^i) r_i = P(\alpha^i), i = 0, \cdots, n-1. \tag{2.18}$$

With $P(x)$ and $W(x)$, the message polynomial is

$$m(x) = \frac{P(x)}{W(x)}. \tag{2.19}$$

We can solve this problem in a modified way. To find a bivariate polynomial $Q(x, y) = W(x)y - P(x)$ passing through all the data pairs $\{(x, y) | x_i = \alpha^i \; and \; y = r_i\}$. After factorization of $Q(x, y)$, we find a solution

$$y = m(x) = \frac{P(x)}{W(x)}, if \deg(y) < k. \tag{2.20}$$

This process is an interpolation-based process. It uses a bivariate polynomial containing $y$, whose degree is at most one. With the extension of this idea, Guruswami and Sudan later proposed a breakthrough decoding algorithm [22] with

$Q(x, y)$ passing through data pairs $m > 1$ times instead of one time. The degree of y is more than one, which will produce a list of possible message polynomials. With the channel output of reliabilities, so called "soft information", an extension of the GS decoding algorithm called the KV decoding algorithm adopted a strategic way of assigning different multiplicities to different data pairs. These two important decoding algorithms will be introduced in later sections.

### 2.3.2 Soft-decision decoding algorithms

SDD algorithms utilize channel output reliabilities to improve decoding performance. The early versions of soft-decision algorithms are generalized minimum distance (GMD) [31] and Chase [32], which were proposed decades ago playing an important roles in the decoding of RS codes. Soft information provides not only the information about the most reliable bits/symbols for hard-decision algorithms, but can also be the overall metric for list decoding. In addition, soft information can also be used to assign multiplicities to the number of interpolation on data pairs, as in the KV algorithm [23].

*Guruswami-Sudan algorithm*

Guruswami and Sudan proposed an interpolation-based decoding algorithm [22] as an extension of the key-equation of the WB algorithm [28]. The problem of finding two polynomials $N(x)$ and $W(x)$ has been converted into finding a bivariate

polynomial $Q(x,y) = W(x)y - P(x)$ passing through the all data pairs determined from the channel output information. The GS algorithm with interpolation multiplicity equaling to one can only generate at most one message polynomial, which does not make full use of the channel output information. When the multiplicities are larger than one, the degrees of $x$ and $y$ in $Q(x,y)$ are both more than one. After factorizations, there exist multiple candidate message polynomials, which lead to a list of candidate codewords. By evaluating the overall reliabilities based on the channel output information as a judging metric, the one with most overall reliability is declared the correct codeword.

The GS algorithm [13] is an interpolation-based list decoding algorithm, which finds out a bivariate polynomial with minimal $(1, k-1)$-weighted degree over the polynomial ring $GF(2^m)[x,y]$ passing through $N$ interpolation pairs with a given positive integer multiplicity, given by

$$Q(x,y) = \sum_{a=0}^{\infty}\sum_{b=0}^{\infty} q_{a,b}\, x^a\, y^b \in GF(2^m)[x,y]. \tag{2.21}$$

**Notation**: The $(w_x, w_y)$-weighted degree of a bivariate polynomial over $GF(2^m)$ is defined as

$$deg_{w_x,w_y}\big(Q(x,y)\big) = \max\{aw_x + bw_y | q_{a,b} \neq 0\}.$$

The $(u,v)$-th Hasse derivative of polynomial $Q(x,y)$ is defined as

$$D_H^{(u,v)}[Q(x,y)] = \sum_{i=u}^{deg_x(Q)} \sum_{j=v}^{deg_y(Q)} \binom{i}{u}\binom{j}{v} q_{i,j} x^{i-u} y^{j-v}, \tag{2.22}$$

and can be used to verify that the polynomial has a zero with a multiplicity $l$ at a certain pair $(x, y)$ if the condition $D_H^{(u,v)}[Q(x,y)] \equiv 0$ at $(u, v)$, for $0 \leq u + v < l$ is satisfied.

*Guruswami-Sudan algorithm decoding procedure*

1) *Initialize the selected data pairs based on the channel output information.*$\{(x_i, y_i)\}, i = 0, \cdots, n-1$. *Select the interpolation multiplicity* $l$.
2) *Interpolate to get* $Q(x, y)$ *with each* $point(x_i, y_j)$ $of$ *multiplicity* $l$.
3) *Factorization using Roth's factorization algorithm from [33].*
4) *Regenerate all the candidate codewords from the list of all the factors of the form* $(y - m_i(x))$ *with degree of x less than k [22].*

*Koetter-Vardy algorithm*

The GS algorithm has a much better performance than the conventional hard-decision decoding algorithms. However, the multiplicities are fixed all over the codeword, which ignores the fact that not all the data pairs are "reliable". When passing through the "non-reliable" data pairs with a large multiplicity, it may lead to a worse situation. The major limitation of this algorithm is that the multiplicities of each interpolation pair are the same, and when we increase the multiplicity, the overall complexity goes up by a factor of *N*.

It is a better strategy to assign multiplicities according to how reliable the data pair is. Koetter and Vardy proposed a new way to utilize the channel output information to decide the different multiplicities for data pairs. The KV algorithm [23] introduced a reliability matrix$\Pi[i,j]$ derived from the "soft" information. With a preselected total multiplicity $s$, the reliability matrix $\Pi[i,j]$ is converted to a multiplicity matrix $M[i,j]$. The total sum of each entry of $M[i,j]$ equals to $s$.

*The KV's algorithm for assigning multiplicities* [23].

1) Given the reliability matrix $\Pi[i,j]$, $i = \alpha^0, \alpha^1, \cdots, \alpha^{q-1}, j = 0, \alpha^0, \alpha^1, \cdots, \alpha^{q-1}$, and the total multiplicity $s$, let $\forall m_{i,j} = 0$ and $\Pi^*[i,j] = \Pi[i,j]$.
2) Find the $M[i,j]$ iteratively.

$If\ (s > 0\ )$ {Search in $\Pi[i,j]$ to find the largest $\pi_{i,j}$,

$$\pi^*_{i,j} = \frac{\pi_{i,j}}{m_{i,j}+1}, m_{i,j} = m_{i,j} + 1, s = s - 1\ \}$$

Else output $M[i,j]$.

*Koetter-Vardy algorithm decoding procedure*

1) *Initialize* $M[i,j]$.
2) *Initialize the selected data pairs based on the channel output information.*$\{(x_i, y_i)\}, i = 0, \cdots, n-1$.

3) *Interpolate to get* $Q(x,y)$ *with each point*$(x_i, y_j)$ *of multiplicity* $l_{i,j}$ *from* $M[i,j]$.
4) *Factorization using Roth's factorization algorithm from [33].*
5) *Regenerate all the candidate codewords from the list of all the factors of the form* $(y - m_i(x))$ *with degree of x less than k [22].*

## 2.4 Conversions and connections of different definitions

All definitions of RS and BCH codes are special cases of generalized RS codes (GRS) [2], [36], [37]. A codeword of the generalized RS codes is generated by the evaluation map encoding with a evaluation map set $x = [x_0, x_2, \cdots, x_{n-1}]$ and a weighted set $v = [v_0, v_2, \cdots, v_{n-1}]$. Here the all $x_i$'s are strictly distinct non-zero elements in $GF(q)$, and all $v_i$'s are also strictly non-zero but not necessarily distinct. We have $c = [v_0 m(x_0), v_1 m(x_1), \cdots, v_{n-1} m(x_{n-1})]$. Here $n \leq q - 1$.

The RS codes are a subclass of the non-binary BCH codes and the primitive $q$-ary BCH code must be a subfield subcode of the RS codes over $GF(q^d)$, where $d$ is an integer larger than one. The additional relations between the RS codes and BCH codes are as follows: 1) The trace mapping $F: GF(q^m) \rightarrow GF(q)$ can convert certain subcodes of RS codes into BCH codes over $GF(q)$. 2) The decomposition of certain RS polynomials over $GF(q^d)$ can introduce several parallel $q$-ary BCH polynomials

over $GF(q)$, here $q$ is a prime power. 3) The RS codes with error correction capability $t$ is the sum of a glue code and a linear combination of several BCH with the same error correction capability $t$. In Chapter 4, we will show a subcode of a PSSRS code with $m$ parallel primitive binary BCH codes in the structure, which enables a practical systematic scheme for the parallel local decoding of RS codes.

# Chapter 3

# Chase Algorithms for Soft-Decision Decoding

## 3.1 Low Complexity Chase (LCC) Soft-Decision Decoding

### 3.1.1 Introduction

Algebraic soft-decision RS decoding algorithms [22], [23], [38], [39] with improved error-correcting capability and comparable complexity to standard algebraic hard-decision algorithms are very attractive candidates for possible implementation in the next generation of read channels. In this chapter, we investigate the performance of a low-complexity Chase (LCC)-type soft-decision RS decoding algorithm [38], recently proposed by Bellorado and Kavčić, on PMRCs at various user densities for sector-long RS codes of practical interest. Previous results for additive white Gaussian noise channels have shown that for a moderately long high-rate code the LCC algorithm can achieve a coding gain comparable to the KV algorithm with much lower complexity. We present a set of numerical results that show that this algorithm provides small coding gains, on the order of 0.2 dB, with similar complexity to the hard-decision algorithms currently used, and that larger coding gains can be obtained if we use more test patterns, which significantly increases its computational complexity.

RS codes and algebraic hard-decision decoding algorithms are the current standard for magnetic recording systems. The development of algebraic soft-decision decoding algorithms for RS codes, such as the GS [22] and the KV algorithms [23], opened up the possibility of using soft-decision decoding in future generations of read channels to obtain improved performance without having to change the codes.

Although they provide significant coding gains over hard-decision decoding algorithms, the main drawback is that the cost in terms of computational complexity is prohibitively high. A large body of work has been quickly assembled on ways to further improve their coding gain and on reducing the overall decoding complexity [38], [40], [41]. A recent example of such work is a LCC soft-decision decoding algorithm, proposed by Bellorado and Kavčić, which utilizes a re-encoding technique [39] and a simplified factorization method to reduce complexity while using a Chase algorithm to enhance performance.

The LCC algorithm [38] is a symbol-level interpolation-based soft-decision list decoding algorithm implemented in a computationally efficient manner. By sorting the received symbols by their reliability, the LCC algorithm divides the received vector into two disjoint parts, namely a set of common interpolation points and a set of uncommon elements or test patterns used in the Chase algorithm. These test patterns are generated using the channel soft information to identify the least reliable positions, and generate a set of bivariate polynomials, which produce a corresponding list of candidate message polynomials. After calculating the product of the reliabilities of the symbols in each candidate codeword in the list, the one with the largest value is chosen as the correct codeword.

### 3.1.2 Low-complexity Chase-type decoding algorithm

The LCC algorithm for decoding RS codes was proposed in [38] and its performance on additive white Gaussian channels was shown to be similar to the KV algorithm with a complexity comparable to classical hard-decision decoding algorithms.

The algorithm can be informally described as follows. Let an $RS(n,k)$ codeword $\boldsymbol{c}$ be transmitted through a channel and consider a channel detector which computes a $2^q \times n$ reliability matrix $\prod$ whose entries $\pi_{i,j}$ are the symbol's soft information. Let us denote $\prod(\alpha, j)$ as the entry in the $j^{\text{th}}$ column of $\prod$ indexed by $\alpha \in GF(2^q)$.

From the reliability information, hard-decision vectors

$$y^{HD} = (y_0^{HD}, \cdots, y_{n-1}^{HD}) \text{ and } y^{2HD} = (y_0^{2HD}, \cdots, y_{n-1}^{2HD})$$

can be generated as follows

$$y_i^{HD} = \arg max_{\alpha \in GF(2^q)} \prod(\alpha, i)\,, \tag{3.1}$$

$$y_i^{2HD} = \arg max_{\alpha \in GF(2^q), \alpha \neq y_i^{HD}} \prod(\alpha, i). \tag{3.2}$$

Using $y^{HD}$ , $y^{2HD}$ and $\prod$, we calculate the figure-of-merit

$$\gamma_i = \frac{\max_{\alpha \in GF(2^q), \alpha \neq y_i^{HD}} \prod(\alpha, i)}{\max_{\alpha \in GF(2^q)} \prod(\alpha, i)} \leq 1, \tag{3.3}$$

which is a measure of the confidence in the hard-decision for that particular symbol. For $\gamma_i \approx 1$, it is very likely that an error may have occurred. The LCC algorithm sorts all the $\gamma_i$'s, and segments the coordinate positions into two parts by selecting $\eta \ll n$ coordinates with the largest $\gamma_i$'s as the uncommon part $I = \{i_1, \cdots, i_\eta\}$, and $\overline{I} = \{0, \cdots, n-1\} \setminus I$ as the common part composed by the remaining more reliable positions. The LCC algorithm forms test vectors equivalent in all coordinate positions except the least reliable ones, where there are two hard-decision choices at each position. Thus the algorithm will form a test set of cardinality $2^\eta = card(2^{|I|})$.

#### 3.1.2.1. Complexity reduction

The LCC algorithm utilizes a re-encoding procedure to reduce complexity by "*zeroing-out* $k$ entries" [38]. Let $y = c + e$ be any test vector, where $e$ is an error vector, and let $J$ contain the $k$ most reliable positions. We can use erasure decoding to find a codeword $\psi = (\psi_0, \psi_1, \cdots, \psi_{n-1})$ with $\psi_i = y_i$ for $\forall i \in J$, and add it to $\boldsymbol{y}$ to get a new codeword with these $k$ most reliable positions equal to zero.

In our work, since we use the evaluation-map encoding method, the erasure decoding is implemented by Lagrange interpolation. With $J \subset I$ being "zeroed-out", the complexity of the interpolation of the common part is significantly reduced.

#### 3.1.2.2. Polynomial interpolation and factorization

The interpolation step generates polynomials $Q(x, y)$ with $deg_y(Q) \leq 1$, which can be expressed as [38]

$$Q(x,y) = q_{\bar{z}}(x)v(x) + y \cdot q_z(x), \tag{3.4}$$

where $v(x) = \prod_{i \in J}(x - x_i)$ thus the polynomial $Q(x,y)$ passes through the $(x_i, 0)$'s and $Q(x_i, 0) = 0$. If the root of $Q(x,y) = 0$ is $m(x)$, then $Q\big(x, m(x)\big) = 0$, and

$$m(x) = \frac{q_{\bar{z}}(x)v(x)}{q_z(x)}. \tag{3.5}$$

**3.1.2.3. Algorithm analysis**

The multiplicity matrix $L$ used by the LCC algorithm is a symbol-level matrix with only one entry $l_{i,j} = 1$ per column and all the other entries are zero. The cost of decoding with such a multiplicity matrix is

$$C = \frac{1}{2}\sum_{i=0}^{2^q-1}\sum_{j=0}^{n-1} l_{i,j}(l_{i,j} + 1) = n. \tag{3.6}$$

For a long high-rate code, $3 < lim_{k \to n}\sqrt{1 + \frac{8n}{k-1}} < 4$, and $deg_y(Q)$ is given by [16]

$$d_y = \left\lfloor \frac{1+\sqrt{1+\frac{8C}{k-1}}}{2} \right\rfloor - 1 = \left\lfloor \frac{1+\sqrt{1+\frac{8n}{k-1}}}{2} \right\rfloor - 1 = 1. \tag{3.7}$$

The interpolation step for the KV algorithm can be described as follows.

$G \leftarrow \left\{g_0 = 1, g_1 = y, \cdots\cdots, g_{d_y} = y^{d_y}\right\}$

$for\ each\ point(x_i, y_j) with\ multiplicity\ l_{i,j} > 0\ do$

$for\ u \leftarrow 0\ to\ l_{i,j} - 1\ do$

$for\ v \leftarrow 0\ to\ l_{i,j} - 1 - u\ do$

$f \leftarrow min_{deg\ (1,k-1)}\{g \in G\ such\ that\ D_H^{(u,v)}[g(x_i, y_j)] \neq 0\}$

$for\ g \in G\ such\ that\ g \neq f\ do$

$g \leftarrow g \cdot D_H^{(u,v)}[f(x_i, y_j)] - f \cdot D_H^{(u,v)}[g(x_i, y_j)]$

$end\ for$

$f \leftarrow (x - x_i) \cdot f$

$end\ for$

$end\ for$

$end\ for.$

Given $l_{i,j} = 1$, only the $(0,0)$-th Hasse derivative needs to be calculated and $D_H^{(0,0)}[f(x,y)] = f(x,y)$, therefore, the interpolation update step simplifies to

$$g \leftarrow g \cdot f(x_i, y_j) - f \cdot g(x_i, y_j),$$

$$f \leftarrow (x - x_i) \cdot f,$$

which is exactly the same as the LCC interpolation update step in Algorithm 1 of [38].

In summary, the LCC algorithm is a special case of the GS algorithm with multiplicity one or a special case of the KV algorithm with total multiplicity $s = n$, and a multiplicity assignment of a single entry equal to one for each coordinate position. This is the basis for the complexity reduction for both the interpolation update and the polynomial factorization steps.

### 3.1.3 Simulation results and analysis

We consider a shortened $RS(440,410)$ code over $GF(2^{10})$on an equalized PMRC with 90% jitter noise, and a Bahl-Cocke-Jelinek-Raviv (BCJR) algorithm as the soft-decision channel detector. Fig. 3.1 shows that the LCC algorithm outperforms traditional hard-decision decoding algorithms. At a frame-error rate (FER) of $10^{-4}$, the LCC algorithm has coding gains of about 0.2 dB and 0.4 dB over hard-decision decoding for $\eta = 4$ and 8, respectively. As we increase $\eta$, we get better performance, but the number of test patterns increases exponentially. In Fig. 3.2 we show the number of interpolations and factorizations for the LCC algorithm with full factorization.

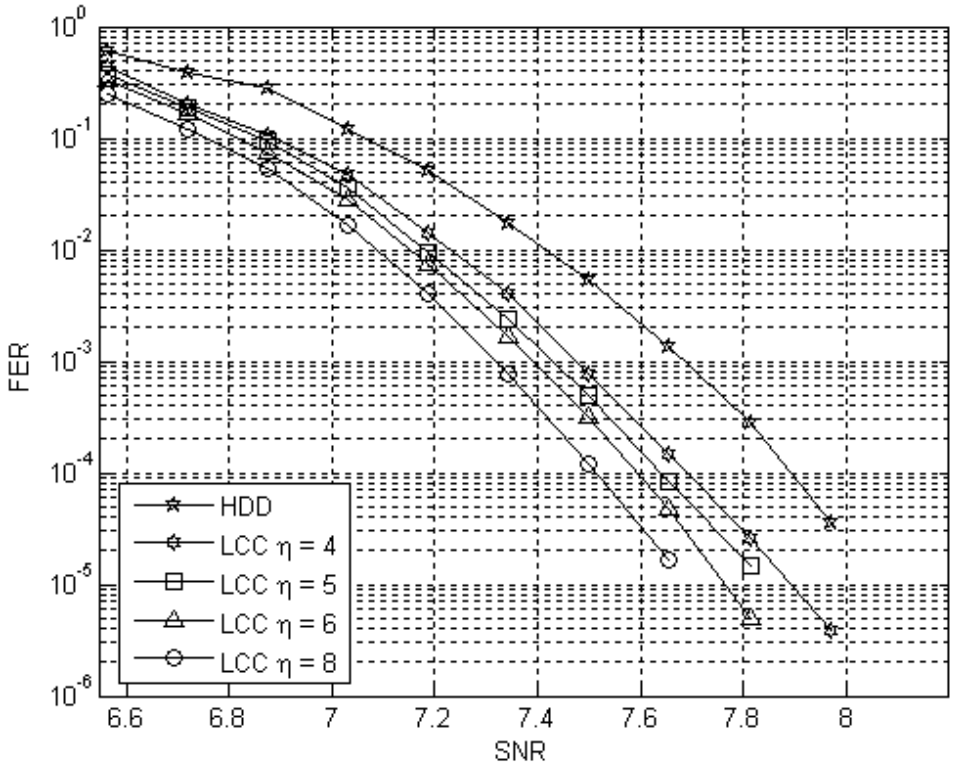


Fig. 3.1. Performance of $RS(440,410)$ on a PMRC equalized to an optimal GPR4 target with 90% jitter noise. The target is [1 0.307 -0.039 -0.002] and user density 0.938.

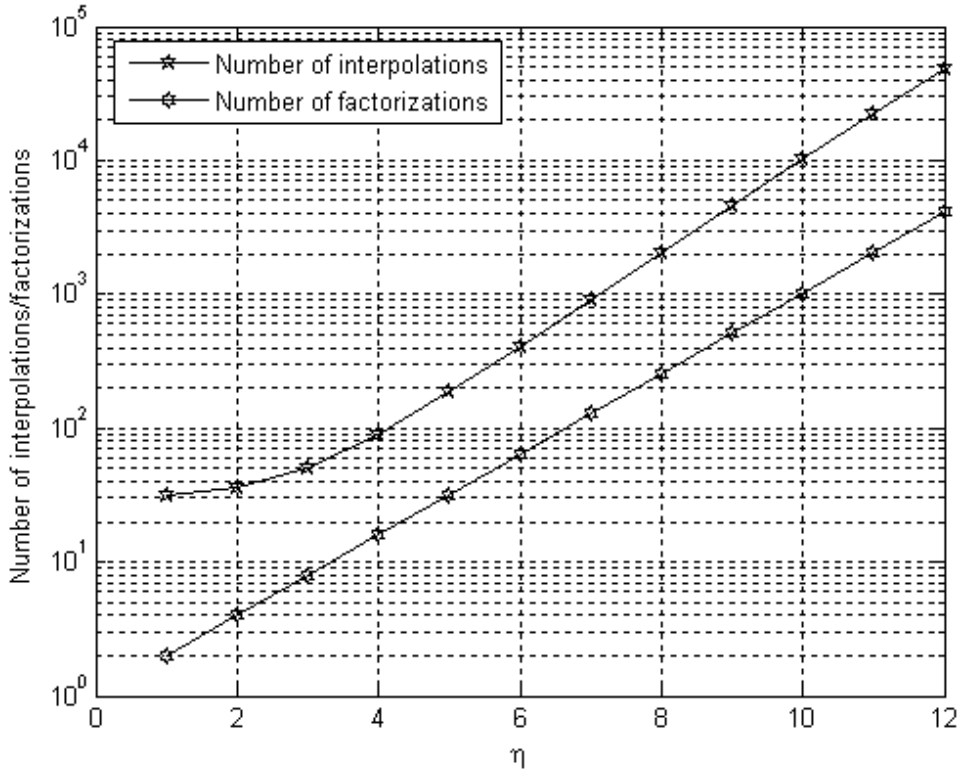


Fig. 3.2. Number of interpolations/factorizations as a function of $\eta$ for one sample execution of the LCC algorithm.

Simulations at higher user densities produced similar results as shown in Fig.3.3 for user density 1.1257, and for user density 1.313 (not shown).

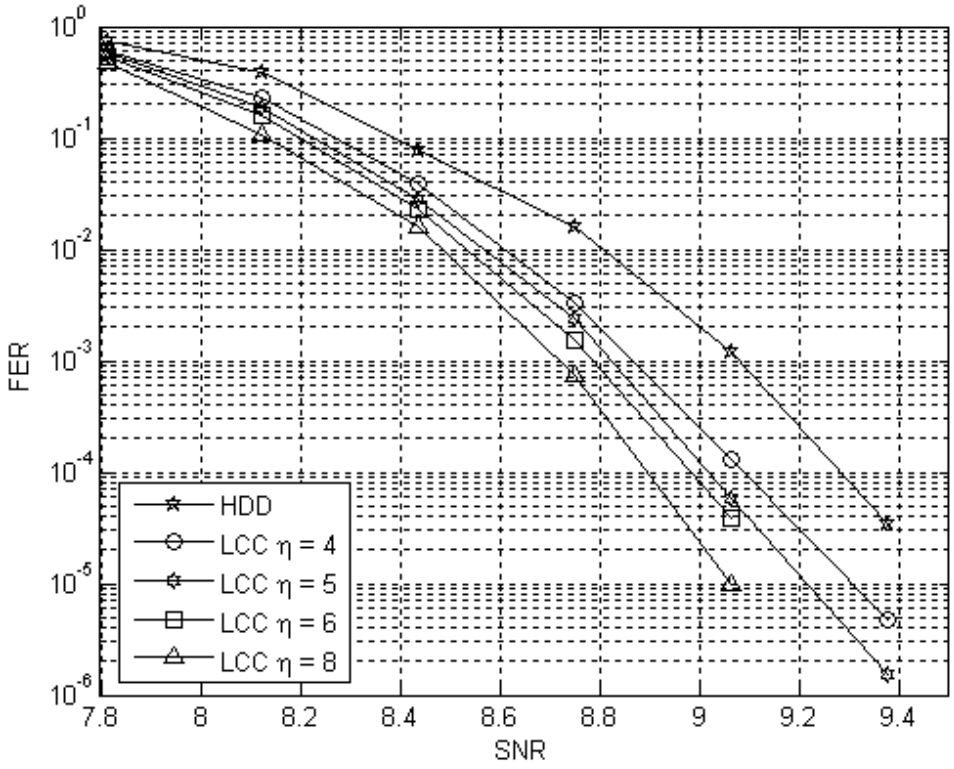


Fig. 3.3. Performance of $RS(440,410)$ on a PMRC equalized to an optimal GPR4 target with 90% jitter noise. The target is [1 0.471 -0.068 -0.004] and user density 1.1257.

### 3.1.4 Conclusions

The LCC algorithm provides small gains with comparable complexity to standard hard-decision algorithms for high-rate RS codes on PMRCs. In order to deliver large performance gains, however, it requires a very large number of test vectors and the

overall computational complexity of the algorithm, which is exponential in $\eta$, becomes impractical.

## 3.2 A Chase-GMD algorithm for soft-decision decoding

### 3.2.1 Introduction

RS codes are widely used on both transmission and storage channels to support high data rates in packet oriented applications, and are usually decoded using efficient hard-decision algebraic algorithms. Guruswami and Sudan [22] and Koetter and Vardy [23] proposed soft-decision algebraic RS decoding algorithms with improved performance but high decoding complexity. These soft-decision algebraic algorithms consist of three main steps: multiplicity computation, polynomial interpolation and factorization. The polynomial interpolation is the most computational complex step, and the common feature of these two algorithms and of a host of related ones which we will refer to as interpolation-based algorithms. The focus of most of the recent work on interpolation-based algorithms has been on improving their performance and reducing their computational complexity [38]-[41]. For example, in [40], a Chase-type algorithm with improved performance was introduced, and in [41], Gross *et al.* used a re-encoding technique to reduce the complexity of the interpolation step. Bellorado and Kavcic [38] presented a low complexity algorithm utilizing both the re-encoding technique and a Chase algorithm

and using a constant multiplicity of one. Forward generalized minimum distance (GMD)-type and recursive Chase algorithms were proposed in [42].

In this chapter, we further investigate the recursive property of the interpolation-based RS decoding algorithms originally presented in [42], and propose an efficient recursive algorithm for algebraic soft-decision decoding of RS codes combining the Chase [32] and GMD [31] concepts to achieve short latency as well as improved decoding performance by further exploiting the channel reliability information. The paper is organized as follows: In Section 3.2.2, we present a brief summary of interpolation-based algorithms and discuss their recursive property, and in Section 3.2.3 we investigate the decoding performance of the reliability-based recursive algorithms. In Section 3.2.4, we propose a Chase-GMD algorithm based on the recursive interpolation property and the utilization of symbol reliabilities. In Section 3.2.5, we discuss the simulation results and summarize our conclusions in Section 3.2.6.

### 3.2.2 Interpolation-based RS decoding algorithms and their recursive property

An RS code of length $n$ and dimension $k$ can be generated by evaluating the information polynomial $f(x)$ over the nonzero distinct elements of $GF(q)$, where $q = n + 1$. For a received vector $\boldsymbol{y} = [y_0 \quad y_1 \quad \cdots \quad y_{n-1}]$ with channel reliability $\Pi$ we can generate a set of symbol pairs $\{x_i, y_i\}$ and the associated probabilities $Pr\{y_i | x_i, \Pi\}$, which can be used to obtain an integer constraint matrix

$M$. An interpolation algorithm is then used to find a bivariate polynomial $Q(x,y)$ passing through symbol pairs $\{x_i, y_i\}$ and satisfying the constraints defined by $M$. Finally, a factorization algorithm is used to find factors of the form $y - f(x)$, and generate a list of possible decoding answers. The decoding answer is chosen to be the most likely one on the list. Further details on interpolation-based algorithms can be found in [22], [23], and references therein.

Interpolation-based algorithms can be viewed as finding a set of surfaces that pass through a number of given points $\{x_i, y_i\}$ subject to certain constraints, and the intersection of the lowest degree surface with the plane $Q(x,y) = 0$ is the list of decoding candidates $y = f(x)$. The polynomial interpolation procedure is the key step and can be formulated recursively [42] as:

$$Q^{(N)} = W^{(N-1)} W^{(N-2)} \dots W^{(0)} Q^{(0)} = \prod_{i=0}^{N-1} W^{(i)} Q^{(0)}, \qquad (3.8)$$

where $Q^{(0)} = [y^0 y^1 \dots y^{L-1}]^T$, each $W^{(i)}$ is an $LxL$ matrix related to an interpolation point pair with a given multiplicity $m_i$ and $L-1$ is the highest degree of $y$ needed for the polynomial interpolation. The final output bivariate polynomial $Q(x,y)$ is selected from the $L$ polynomials in vector $Q^{(N)}$ as the one with the smallest weighted degree. It is worth noting that the order of multiplication of the vectors $W^{(i)}$ can be changed arbitrarily, and that reliability information can be used to select a specific order to improve the performance of the algorithm [42]. In addition, the

interpolation of a given point can be reversed recursively since $Q^{(i-1)} = W^{(i)^{-1}} Q^{(i)} = W^{(i)^{-1}} W^{(i)} Q^{(i-1)}$.

### 3.2.3 Performance analysis of forward recursive algorithms

Forward recursive interpolation-based algorithms are erasure-and-trial algorithms, which can be viewed as using a set of multiplicity matrices $M_i$ at each interpolation step, with $M_N = M$ [42]. For Chase-type algorithms, the set of multiplicity matrices is further expanded to include $M_i^{(j)}$, where the superscript denotes the $j$-th test pattern. Let $B_i^{(j)} = M^{(j)} - M_i^{(j)}$ be the erasure matrix whose nonzero entries correspond to the number of polynomial constraints not yet satisfied for the $j$-th test pattern. At each step of the forward recursive algorithm, the correct codeword is found, if the condition in Theorem 3 in [23] is satisfied, namely

$$S_{M_i^{(j)}}(c) > \deg_{1,k-1}\left(Q_{M_i^{(j)}}(x,y)\right), \tag{3.9}$$

where $S_{M_i^{(j)}}(c)$ is the score of a codeword $\boldsymbol{c}$and the function $\deg_{1,k-1}(\cdot)$is the weighted degree of a bivariate polynomial. We can express the score as

$$S_{M_i^{(j)}}(c) = \sum_{t=0}^{N-1} m_t - \sum_{j=0}^{e-1} e_j - \sum_{l=0}^{f-1} b_l = m_{total} - \sum_{j=0}^{e-1} e_j - \sum_{l=0}^{f-1} b_l, \tag{3.10}$$

where the summation of the nonzero entries $\{m_t\}_0^{N-1}$ in $M$ is $m_{total}$ ; $\sum_{l=0}^{f-1} b_l$ is the summation of all nonzero entries in $B_i^{(j)}$ , which can be denoted as $\{b_l\}_0^{f-1}$; and $\sum_{j=0}^{e-1} e_j$ is the summation of the nonzero erroneous entries $\{e_j\}_0^{e-1}$ in $M_i^{(j)}$. In addition, let us define another matrix $R_i^{(j)}$ containing all the "correct" entries in $M_i^{(j)}$ , which are denoted as $\{r_i\}_0^{g-1}$. The right-hand side of (3.14) is given by

$$\deg_{1,k-1}\left(Q_{M_i^{(j)}}(x,y)\right) = \Delta(C-F)$$

$$= \Delta\left(\frac{1}{2}\sum_{t=0}^{N-1} m_t(m_t+1) - \frac{1}{2}\sum_{l=0}^{f-1} b_l(b_l+1)\right)$$

$$= \Delta\left(\frac{1}{2}\sum_{i=0}^{g-1} r_i(r_i+1) + \frac{1}{2}\sum_{j=0}^{e-1} e_j(e_j+1)\right).$$

(3.11)

Equation (3.11) is bounded by $\Delta(C-F) < \sqrt{2(k-1)(C-F)}$, where $C$ is the cost of the whole multiplicity matrix $M_i^{(j)}$, and $F$ the cost of the erasure matrix $B_i^{(j)}$, that is $\frac{1}{2}\sum_{l=0}^{f-1} b_l(b_l+1)$. To get the correct codeword, the score given in (3.10), must be larger than the weighted degree given in (3.11), which can be expressed as

$$\sum_{j=0}^{e-1} e_j\left(e_j+1\right) < \frac{1}{k-1}\left(\sum_{i=0}^{g-1} r_i\right)^2 - \sum_{i=0}^{g-1} r_i(r_i+1). \qquad (3.12)$$

Given a multiplicity matrix $M_i^{(j)}$, as long as the "correct" entries $r_i$ and "erroneous" entries $e_i$satisfy (3.12) the correct codeword will be found.

Consider the error-correction capability of the Guruswami-Sudan (GS) algorithm with constant multiplicity $m_t = m$ and error-only decoding; using (3.11) and (3.12), we can derive a bound on the number of correctable errors $e$ as

$$e < n - \sqrt{(k-1)n(m+1)/m}.$$

As $m \to \infty$ and if we use an error-and-erasure decoding algorithm, the bound becomes

$$e < (n-f) - \sqrt{(k-1)(n-f)}.$$

As originally discussed in [23], finding a good multiplicity matrix for the interpolation is one of the key issues. The performance improvement obtained by soft-decision RS decoding algorithms is a direct consequence of using the channel output information to find the best multiplicity matrix, i.e., finding the *best multiplicity computation method*. In reality, we only have the received reliability matrix $\Pi$, with entries $\pi_{ij}$, and an intuitive way to compute the multiplicity matrix is to use a large enough scalar $\lambda$ to multiply the reliability matrix as in [43], which leads to$\lfloor \lambda \Pi \rfloor = M$. Assume that $\pi_{e_j}$ corresponds to the reliability of an "erroneous point" in

the reliability matrix, and $\pi_{r_i}$ corresponds to the reliability of a "correct point". Then (3.12) becomes

$$\sum_{j=0}^{e-1} \lfloor \lambda \pi_{e_j} \rfloor \left( \lfloor \lambda \pi_{e_j} \rfloor + 1 \right) < \frac{1}{k-1} \left( \sum_{i=0}^{g-1} \lfloor \lambda \pi_{r_i} \rfloor \right)^2 - \sum_{i=0}^{g-1} \lfloor \lambda \pi_{r_i} \rfloor \left( \lfloor \lambda \pi_{r_i} \rfloor + 1 \right), \tag{3.13}$$

and for $\lambda \to \infty$, we have

$$\sum_{j=0}^{e-1} \left( \pi_{e_j} \right)^2 < \frac{1}{k-1} \left( \sum_{i=0}^{g-1} \pi_{r_i} \right)^2 - \sum_{i=0}^{g-1} \left( \pi_{r_i} \right)^2. \tag{3.14}$$

With $e = nq - n$ and $g = n$, (3.14) corresponds to (28) in [14], and describes the asymptotic performance when the total multiplicity $m_{total}$ goes to infinity, which requires an infinite decoding complexity. A significant amount of work, such as [43] [44], has been done on finding the *best multiplicity computation method* given a fixed moderate value for $m_{total}$, to minimize the decoding failure probability $Pr\{S_M > \Delta(M) | \Pi, m_{total}\}$.

Another way to improve decoding performance would be to feed the soft-decision decoder output back to fine tune the computation of the multiplicity matrix. The ultimate goal of iterative soft-decision RS decoding is to find a way to generate soft information from the decoder just like in the decoding of low-density parity-check

codes [45]. However, the search for such method is an open problem. Some alternate methods use hard information to generate an input to feed back to the channel detector/decoder, for example the Chase and GMD-type algorithms used in [38], [42], and in [46]-[48] and references therein. For any given channel reliability information $\Pi$ and reliability ordered received vector, we can modify the multiplicity matrix accordingly if one decoding trial fails. For example, in the Chase-type interpolation-based recursive algorithm, we try to minimize

$$Pr\{S_{\Pi} < \Delta(\Pi)\} = 1 - \bigcup_{i,j} Pr\left\{S_{M_i^{(j)}} > \Delta\left(M_i^{(j)}\right) | \Pi, m_{total}\right\}, \qquad (3.15)$$

for a given test pattern.

### 3.2.4 A Chase-GMD type algorithm using reliability information

In order to utilize the recursive property of interpolation-based algorithms and the reliability information, we propose a forward recursive Chase-GMD algorithm with reduced average decoding complexity and improved decoding performance. In contrast to the algorithms proposed in [42], we have moved the Chase flipping patterns to the early interpolation stages, and process the remaining point pairs using a GMD-type algorithm. Simulation results show an early convergence to the correct codeword. The detailed algorithm is as follows.

From the received channel output information, we can generate a set of symbol pairs$\{x_i, y_i\}$ with corresponding multiplicities $m_i$ .

Step 1: Sort the sequence of symbol pairs in decreasing order of their reliability, and identify the $\eta$ least reliable symbol pairs, or the symbol pairs that contain the $\eta$ least reliable bits. (Note that in the latter case the number of symbol pairs may not be $\eta$.)

Step 2: Divide the sorted symbol pairs into two groups: the first $k$ pairs form the reliable group and the remaining $n$-$k$ the unreliable group. The unreliable group can be further divided into two groups: a Chase group consisting of the $\eta$ least reliable symbol pairs and the GMD group consisting of the remaining $n$-$k$-$\eta$ pairs.

Step 3: Generate an initial bivariate polynomial $Q(x, y)$ passing through all the symbol pairs in the reliable group. (If one uses the re-encoding algorithm [43], this step becomes trivial.)

Step 4: Factorize the intermediate bivariate polynomial $Q(x, y)$ and obtain a decoding answer. If the stopping criterion is met go to End.

Step 5: For the given Chase test pattern, let $Q(x, y)$ pass through the $\eta$ additional symbol pairs.

Step 6: Factorize the intermediate bivariate polynomial $Q(x, y)$ and obtain a decoding answer. If the stopping criterion is met go to End.

Step 7: Recursively let $Q(x, y)$ pass through one more symbol pair in the GMD group. If all symbol pairs in the GMD group have been used, generate the next Chase test pattern and go to Step 5.

Step 8: Factorize the intermediate bivariate polynomial $Q(x, y)$ and obtain a decoding answer. If the stopping criterion is met go to End, otherwise go to Step 7.

End.

The stopping criterion can be implemented by checking the Euclidean distance between the decoded codeword and the received vector, and if the distance is smaller than a programmable threshold, the decoding is considered successful. The number of symbols in the Chase group is used to trade-off complexity for performance. The value for $\eta$ is usually chosen to be small to keep the complexity low. The order of Steps 5 and 7 can be interchanged, giving rise to an interpolation-based GMD-Chase algorithm. A variation of the proposed algorithm terminates Step 7 after a fixed number of points have been interpolated but before all points have been used.

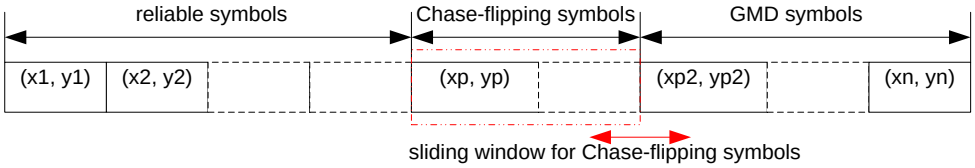


Fig. 3.4. The reliability-sorted received vector and its interpolation order.

The reliability-based forward recursive algorithm described above performs multiple factorization trials in contrast to the GS or KV algorithms, which perform a single factorization step. However, simulation results show that only a small number of factorization trials are needed at high SNRs. Given the benefits of the shorter

decoding latency, due to the fact that most of the time we do not need to interpolate through all the point pairs before we can find the correct answer, and the potential performance improvement [42], the slight increase in the number of factorization trials is a good trade-off.

A way to further reduce the number of factorization trials is as follows. After sorting the received vector, we select the first $k+1$ pairs from the reliable group instead of $k$ pairs in our original algorithm. Then we save the most reliable pair and use the remaining $k$ pairs from the reliable group to generate our initial bivariate polynomial. In some cases, we may even sacrifice the code rate and use some extra redundancy to protect one pair to make it very likely to be error free. If the generated initial bivariate polynomial passes through this highly reliable pair, we will execute the factorization step. Otherwise, we will continue our interpolation using additional pairs in the unreliable group. Prior to any factorization, we check the latest intermediate polynomial to make sure that it passes through this pair; otherwise we skip the factorization step. This interpolation check and factorization flow reduces the total number of factorizations and the complexity of the decoding algorithm.

The basic concept of the proposed algorithm can be used to modify the original interpolation-based algorithms described in [22], [23] as well as for some reduced complexity algorithms using the re-encoding technique, such as the low-complexity Chase (LCC) algorithm proposed by Bellorado and Kavcic [38], which uses re-

encoding and a bivariate polynomial with $y$-degree of one. In their implementation, the factorization step is very simple, which will also ease the complexity of the proposed modification.

### 3.2.5 Simulation results

As an illustrating example, we use the proposed interpolation-based recursive algorithm on an equalized PMRC. A symbol-based BCJR algorithm is used as the channel decoder and the results are compared with the LCC algorithm presented in [38]. The multiplicity for each non-zero entry in $M$ is set to only one, and the complexity comparison is based on the number of points interpolated.

In Fig.3.5, we compare the frame-error rate (FER) performance of the proposed Chase-GMD algorithm to hard-decision RS decoding, and the LCC algorithm on a channel equalized to an optimal GPR4 target with 90% jitter noise for a shortened $RS(440,410)$ code over $GF(1024)$, with code rate $R$=0.9318 at user density $D_u$=0.9381. The user density is defined as $D_u = T_{50}/T_u$, where $T_{50}$ is the parameter in the hyperbolic-tangent model [49], and $T_u$ is the user bit interval. Note that $T_u = T_c * R$, where $T_c$ is the channel bit interval.

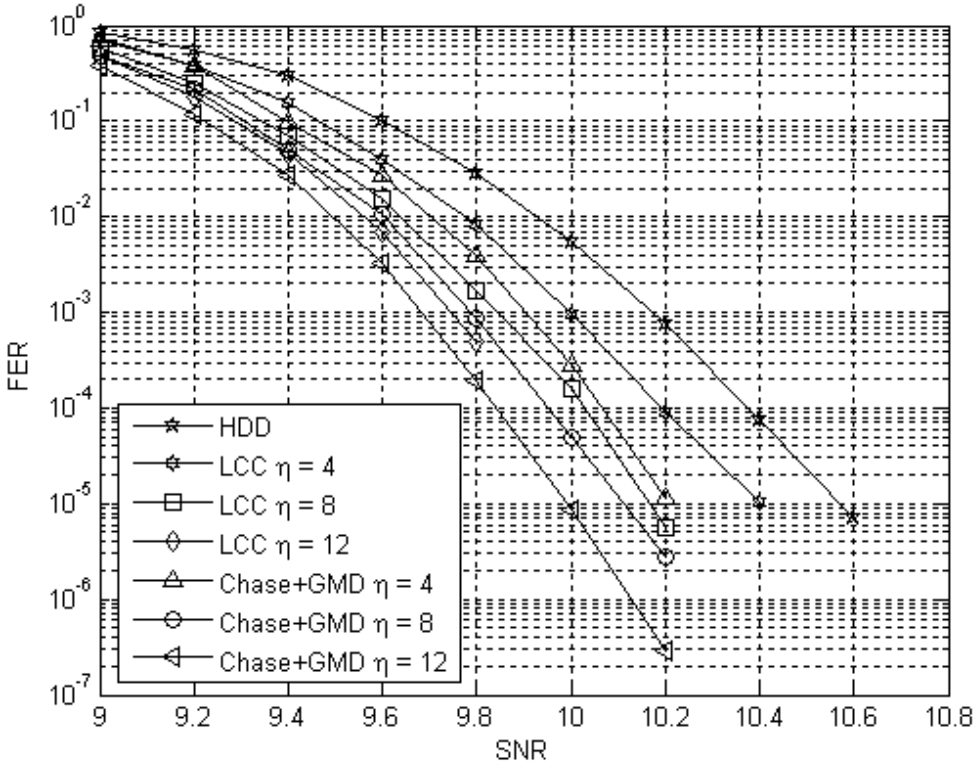


Fig. 3.5. Performance of RS (440, 410) on a PMRC equalized to an optimal GPR4 target with 90% jitter noise, and user density $D_u$=0.9381.

Note that this is not the total number of interpolations for one codeword, since we have $2^{\eta}$ patterns per codeword. At high SNR, the proposed Chase-GMD algorithm only interpolates a few point pairs in the GMD group, before it finds the correct answer, i.e., it only interpolates fewer than *n-k* points.

The simulation results show that the Chase-GMD algorithm slightly outperforms the LCC algorithm for the same number of test patterns. However, when we compare the complexity of the two algorithms in terms of the number of

interpolated points required to execute them, as shown in Fig. 3.6, a different picture emerges. The proposed Chase-GMD algorithm with $\eta$=12 has a 0.4-dB gain over the LCC algorithm with $\eta$=4 at a FER of $10^{-5}$ with almost the similar complexity in terms of the number of the multiplications.

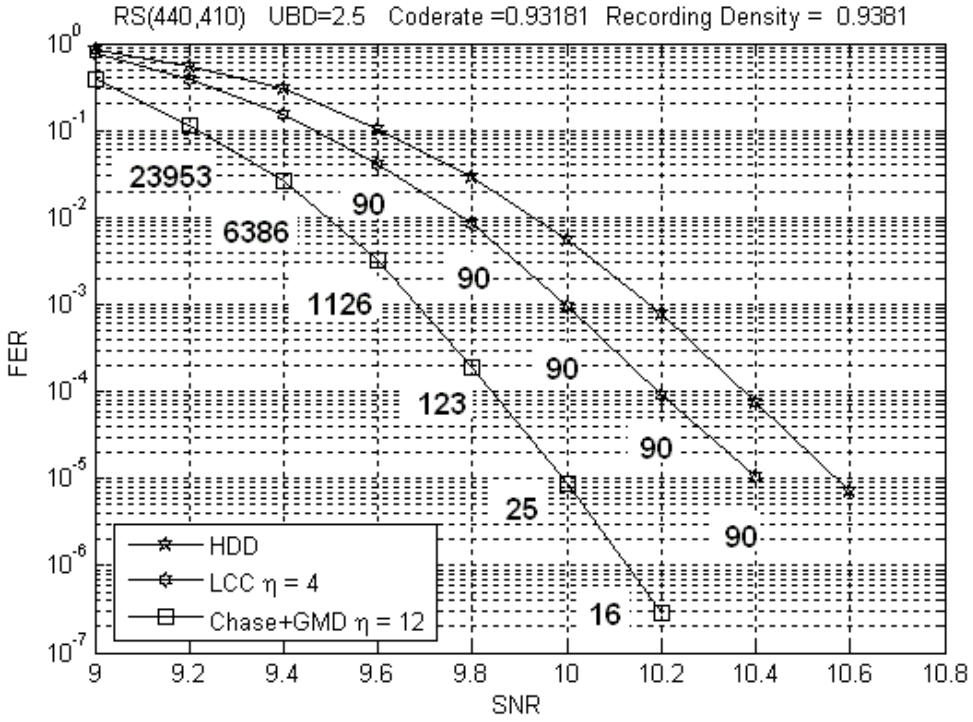


Fig. 3.6. Comparison of the number of interpolated points between LCC and Chase-GMD algorithms for RS (440, 410) on a PMRC equalized to an optimal GPR4 target with 90% jitter noise and user density $D_u$ =0.9381.

Note that the LCC algorithm has a constant number of total interpolations and it always performs $2^{\eta}$ factorizations. From Figs.3.6 and 3.7, we can see that the average number of interpolations and factorizations for the Chase-GMD algorithm decreases with increasing SNR, and that it has a very low average number of factorizations. In Table I we compare the complexity of the proposed (C-GMD) algorithm to LCC decoding. At high SNRs, our algorithm has a very low average number of total interpolations (AI) and factorizations (AF), and provides a significant coding gain with a total complexity in terms of the number of multiplications approximately twice as the Berlekamp-Massey algorithm for HDD.

TABLE 3.1

MAXIMUM NUMBER OF MULTIPLICATIONS REQUIRED FOR DECODING

| Algorithm | RS(440,410) |
|---|---|
| HDD | 26085 |
| LCC $\eta = 4$ AF = 16 | 201449 |
| LCC $\eta = 8$ AF = 256 | 3189797 |
| LCC $\eta = 12$ AF = 4096 | 51009009 |
| C-GMD $\eta = 4$ AF = 4 | 49672 |
| C-GMD $\eta = 8$ AF = 4 | 50148 |
| C-GMD $\eta = 12$ AF = 4 | 50784 |
| C-GMD $\eta = 4$ AF = 8 | 99348 |
| C-GMD $\eta = 8$ AF = 8 | 99984 |
| C-GMD $\eta = 12$ AF = 8 | 100780 |
| C-GMD $\eta = 4$ AF = 12 | 149184 |
| C-GMD $\eta = 8$ AF = 12 | 149980 |
| C-GMD $\eta = 12$ AF = 12 | 150936 |

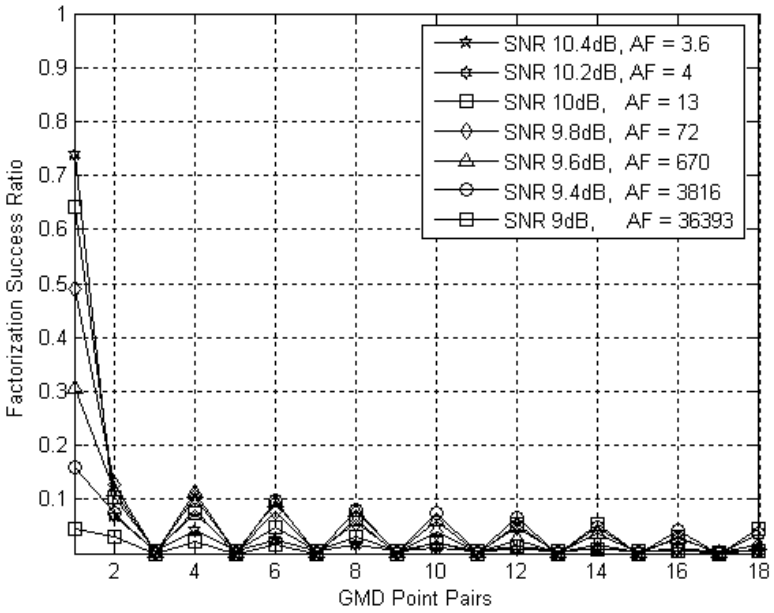


Fig. 3.7. Factorization success ratio vs. the number of GMD interpolation point pairs at various SNRs.

In Fig. 3.8, we present the factorization success ratio distribution for point pairs in the GMD group, which show that the first four pairs can lead to a cumulative probability > 90% to obtain the correct codeword at a given SNR. The figure shows that there is a very large probability to get the correct codeword with a relatively small number of total factorizations, and therefore low complexity. The figure displays factorization success ratio curves at various SNRs and the average number of factorizations.

TABLE 3.2

AVERAGE NUMBER OF INTERPOLATIONS AND FACTORIZATIONS

| Algorithm RS(440,410) | AF / AI at 9dB | AF / AI at 9.6dB | AF / AI at 10.2dB |
|---|---|---|---|
| LCC $\eta = 4$ | 16/90 | 16/90 | 16/90 |
| LCC $\eta = 8$ | 256/2070 | 256/2070 | 256/2070 |
| LCC $\eta = 12$ | 4096/49170 | 4096/49170 | 4096/49170 |
| C-GMD $\eta = 4$ | 314/366 | 26/32 | 4/8 |
| C-GMD $\eta = 8$ | 3190/4352 | 109/149 | 4/12 |
| C-GMD $\eta = 12$ | 36393/60657 | 670/1126 | 4/16 |

In Table II, we present the average number of interpolations and factorizations executed in the processing of point pairs in the GMD group before we find the correct codeword. The same observations on the low complexity of the Chase-GMD algorithm, especially at high SNRs, can be made from this set of results. All complexity results presented refer to the original algorithm. Further reductions in complexity can be realized by using the reduced factorization implementation.

### 3.2.6 Conclusion

The Chase-GMD type algorithm for soft-decision RS decoding presented has an appreciable gain over standard hard-decision decoding algorithms on PMRCs, which compares favorably with recently proposed implementations of Chase-type algorithms. More importantly, for similar complexity at high SNR, our algorithm exhibits significant coding gains over other low complexity implementations. These

algorithms are particularly attractive for magnetic recording systems which operate at relatively high SNR.

# Chapter 4

# Subcodes of Reed-Solomon Codes With a Parallel Subcode Structure

## 4.1 Subspace subcodes of RS codes

In 1992, G. Solomon proposed a non-linear non-binary code [50], which is a new cyclic code of length $2^m - 1$ with a smaller alphabet size of $v$-bit symbols over $GF(2^m)$. This code was initially proposed to have a code rate as close to the RS code rate as possible [50] and a higher $v$-bit symbol dimension than BCH codes of the same length. In [51], G. Solomon did not give a formula for the binary dimension for this new code, although he did provide a method to find out the number of codewords for some special cases, which are highly dependent on the choices of the primitive roots for the $GF(2^m)$ and the subspace for the $v$-bit symbols [20], [21]. Later, in [52], a new trace-shortened RS code (TSRS) is proposed to generalize this non-linear non-binary code by extending the choices of the subspace to some larger classes instead of just some special cases [20]. Hattori later proposed the SSRS codes to further extend the restrictions on the choices of the subspace to arbitrary subspaces, and provide a formula for the bit dimension [20], [21]. From this formula in [20], one can find out the subspace which leads to a code with the maximum bit dimension, when $m$ is small.

Given an RS code over $GF(2^m)$, an SSRS code [20], [21] is a subcode whose binary $m$-tuple projections consist of all zeros for the $\mu$-dimensional subspace of $GF(2^m)$, where $\mu$ is a positive integer and $0 < \mu < m$ [20]. Let $v = m - \mu$, and consider a simple example of an SSRS code.

Given an $RS(7,5,3)$ code, consider the codeword $(2,2,3,0,1,3,1)$, since $m = 3$, the corresponding binary 3-tuple is shown in Fig. 4.1.

| 2 | 2 | 3 | 0 | 1 | 3 | 1 |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 1 | 1 | 1 |

.

Fig. 4.1. The binary 3-tuple of $RS(7,5,3)$ codeword: $(2,2,3,0,1,3,1)$.

Given that the top row of the binary 3-tuple consists of all zeros, we can only consider the binary 2-tuple ($v = 2$) given in Fig. 4.2.

| 2 | 2 | 3 | 0 | 1 | 3 | 1 |
|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 1 | 1 | 1 |

.

Fig. 4.2. The binary 2-tuple of $RS(7,5,3)$ codeword $(2,2,3,0,1,3,1)$.

The codeword $(2,2,3,0,1,3,1)$ is an SSRS codeword with $v = 2$ over $GF(2^3)$. The set of codewords with this property is an SSRS code of the parent $RS(7,5,3)$ code. This SSRS code is based on a 2-dimensional subspace of $GF(2^3)$ [20].

In general, given $GF(2^m)$, the SSRS code based on a $v$-dimensional subspace of $GF(2^m)$ is not linear over $GF(2^v)$, when $v \neq 1$ [20]. However, it is always linear on $GF(2)$. When $v$ is a factor of $m$ (notated as $v|m$), a $v$-dimensional subspace of $GF(2^m)$ is a subfield of $GF(2^m)$with a certain basis [20], [21], and $GF(2^m)$ is the

extension field of $GF(2^v)$ [20]. A codeword of an SSRS code is also a codeword of the parent RS code, so the SSRS code is linear over $GF(2^m)$. However, the linear combination of SSRS codewords over $GF(2^m)$ in most cases is not an SSRS codeword.

The dimension of an SSRS code is strictly dependent on the exact number of SSRS codewords [20]. SSRS codes based on a $v$-dimensional subspace of $GF(2^m)$ could have different dimensions, which depends on the choice of the basis for this $v$-dimensional subspace of $GF(2^m)$ [20]. When $m$ is small, it is possible to get all of the different dimensions by trying all possible bases. It is not computationally feasible to find out all possible dimensions when $m$ is large [20]. The alternative way is to find a larger class or an even larger category, whose elements are equivalent, meaning that every two elements have the same dimension [20]. Unfortunately, the number of these classes or categories is still too large, when $m$ is large. A dimension formula is introduced in [20], which is strongly dependent on the choice of the subspace and the parent RS code. A lower bound formula is also introduced in [20]. Most SSRS codes with higher dimensions do not have an integer "symbol" dimension [20], because the binary dimension is neither a multiple of $v$ nor $m$, which seriously limits the application of an SSRS code with a higher dimension [20]. Interested readers can refer to [20] for more details on the formulas.

There are two encoding methods for SSRS codes in general: the first one is frequency domain encoding, and the second one is systematic encoding [20]. Frequency domain encoding needs to calculate two sets of information bits: the independent coefficients and the dependent ones. After that, all other coefficients are set to zeros, and then the DFT over a Galois field is used to get the time domain codeword [20]. It is complicated to find out these two coefficient sets, when $m$ is large. The encoding of SSRS codes described in [20] is not perfect and needs further development, because it does not always encode systematically. When $d_{min} \gg m - 1$, the mapping $\Phi$ method described in [20] will not always exist. The size of the mapping $\Phi$ will grow exponentially with $m$, which is not suitable for implementation. This encoding problem is solved by selecting a suitable subspace [53] or by a "partial" systematical encoding in [54]. However, the dimension of the SSRS codes generated by the method in [53] is equal to the lower bound introduced in [20]. We can find that the $a_R$'s in [54] are not totally free, thus the encoding is only "partially" systematic.

It is easy to find out that SSRS codes have a smaller alphabet size compared to the parent RS codes but with the same length from its definition. How can we make full use of this structural property? Can we have a code, which has several SSRS codes in parallel as a package like the one shown in Fig. 4.3? What code $\mathbb{C}$ will that be? How can we select suitable SSRS codes to make this new code linear, cyclic, and easy to encode and decode?

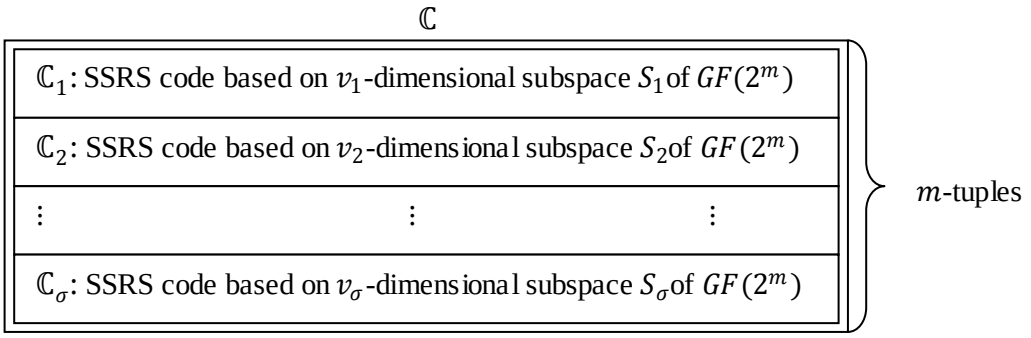


Fig. 4.3. The package structure of several parallel SSRS codes.

## 4.2 Subcodes of RS codes with a parallel subcode structure

First, we only use narrow-sense RS codes unless we specify it to be a generalized RS code. We propose a new subcode of RS codes with a parallel SSRS structure. We call this new subcode the parallel subspace subcode of RS codes (PSSRS). The PSSRS code has two major classes, which are the Class-I subcode $\mathbb{C}_{\mathrm{I}}$ and the Class-II subcode $\mathbb{C}_{\mathrm{II}}$.

*Theorem 4.1*: For narrow sense RS codes, given $RS(n, k_1, d_1)$ and $RS(n, k_2, d_2)$ with $d_1 < d_2$, then $RS(n, k_2, d_2)$ is a subcode of $RS(n, k_1, d_1)$ [23].

*Theorem 4.2*: A codeword of $\mathbb{C}$ in Fig. 4.3 is still an RS codeword.

For proof, please refer to Appendix A.

### 4.2.1 Class-I subcodes $\mathbb{C}_{\mathrm{I}}$

We now define $\mathbb{C}$ in Fig. 4.4 as the Class-I subcodes $\mathbb{C}_{\mathrm{I}}$ of RS codes with a parallel SSRS structure over $GF(2^m)$.

*Definition*: Class-I subcodes $\mathbb{C}_{\mathrm{I}}$ are the subcodes of RS codes with a parallel SSRS structure over $GF(2^m)$, in which there exists at least a pair $\{(i,j)|i \neq j\}$ satisfying $v_i \neq v_j$. $\mathbb{C}_i$'s are SSRS codes based on different parent RS codes with different $v_i$-dimensional subspaces of $GF(2^m)$ and different error-correction capabilities $t_i$.

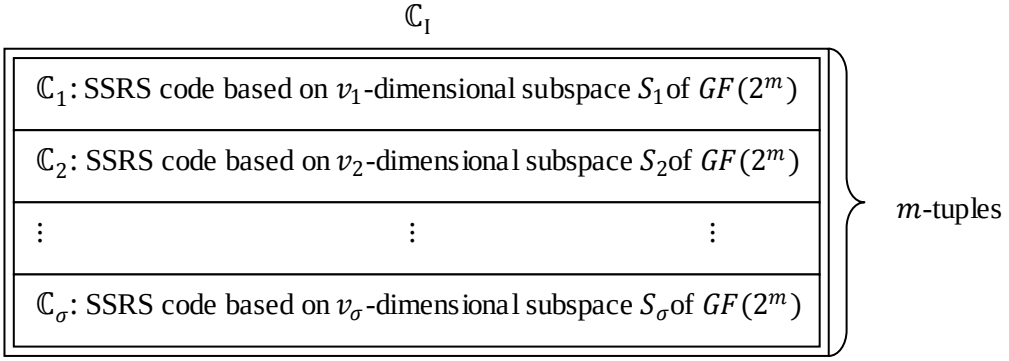


Fig. 4.4. Class-I subcodes $\mathbb{C}_{\mathrm{I}}$ of RS codes.

$\mathbb{C}_{\mathrm{I}}$ is a linear combination of $\{\mathbb{C}_1, \mathbb{C}_2, \cdots, \mathbb{C}_\sigma\}$

$$\mathbb{C}_{\mathrm{I}} = \sum_{i=1}^{\sigma} \alpha^{\left(\sum_{j=1}^{i} v_j\right) - v_i} \mathbb{C}_i. \tag{4.1}$$

The binary dimension of $\mathbb{C}_{\mathrm{I}}$ is denoted as $K_2(\mathbb{C}_{\mathrm{I}})$, and is given by

$$K_2(\mathbb{C}_{\mathrm{I}}) = \sum_{i=1}^{\sigma} K_2(\mathbb{C}_i). \tag{4.2}$$

Because $\mathbb{C}_{\mathrm{I}}$ is a linear combination of $\{\mathbb{C}_1, \mathbb{C}_2, \cdots, \mathbb{C}_\sigma\}$, the "global" symbol-based error-correction capability of $\mathbb{C}_{\mathrm{I}}$ is denoted as $t_G = \mathrm{MIN}(t_1, t_2, \cdots, t_\sigma)$.

The real error-correction capability $t_{\mathbb{C}_{\mathrm{I}}}$ of $\mathbb{C}_{\mathrm{I}}$ has a lower bound and an upper bound

$$\text{MIN}(t_1, t_2, \cdots, t_\sigma) \le t_{\mathbb{C}_\text{I}} \le \sum_{i=1}^{\sigma} t_i . \tag{4.3}$$

Class-I codes can be denoted as $\mathbb{C}_\text{I}(n, k, T, V)$, where $T = (t_1, t_2, \cdots, t_\sigma)$ and $V = (v_1, v_2, \cdots, v_\sigma)$.

The possible decoding structure is shown in Fig. 4.5.

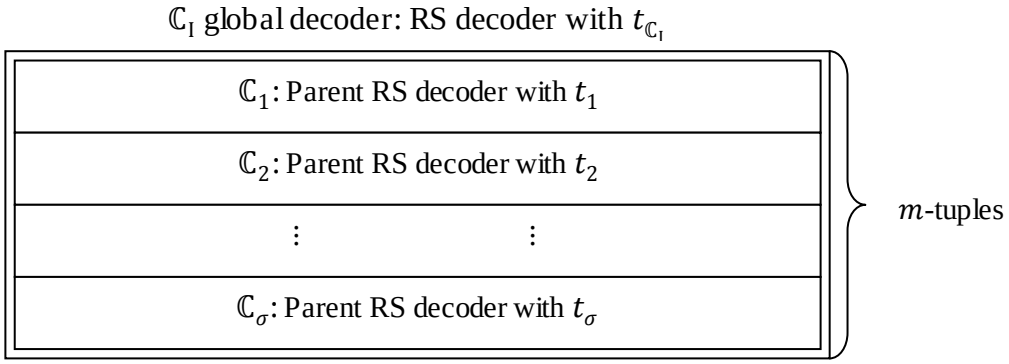


Fig. 4.5. Decoding structure for $\mathbb{C}_\text{I}$ codes.

Class-I codes are very flexible in structure but of limited practical interest, because there exists at least one $v_j > 1$, which leads to a very complicated systematic encoder with a possible non-$v_j$-multiple binary dimension [20], [21]. Class-I codes need several different encoders due to different $v_j$'s, which will increase the hardware implementation cost and complexity.

### 4.2.2 Class-II subcodes $\mathbb{C}_\text{II}$

For practical purposes, we would like to introduce the following simplified version.

*Definition*: Class-II subcodes $\mathbb{C}_{\mathrm{II}}$ are the subcodes of RS codes with a parallel SSRS structure over $GF(2^m)$, in which each arbitrary pair$\{(i,j)|i \neq j\}$ satisfies $v_i = v_j = v = m/\sigma$, where $\sigma|m$. $\mathbb{C}_i$'s are SSRS codes based on the same parent RS code with same $v$-dimensional subspace of $GF(2^m)$ and the same error-correction capability $t$.

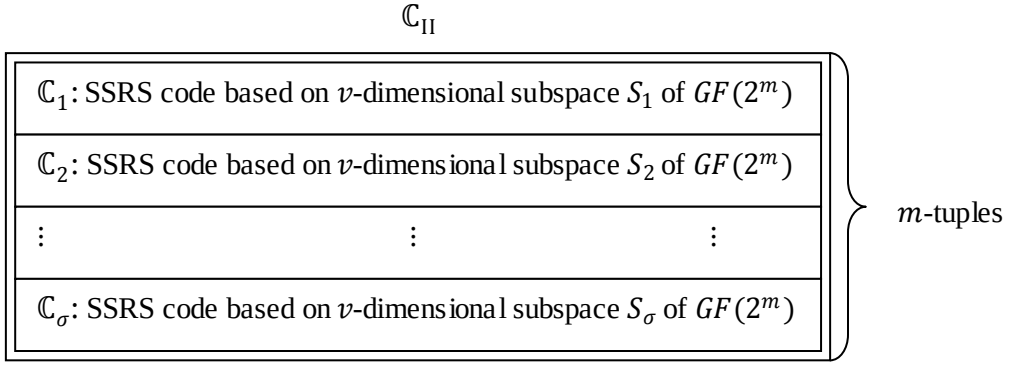


Fig. 4.6. Class-II subcodes of RS codes.

It is shown that $\mathbb{C}_{\mathrm{II}}$ is also a linear combination of $\{\mathbb{C}_1, \mathbb{C}_2, \cdots, \mathbb{C}_\sigma\}$

$$\mathbb{C}_{\mathrm{II}} = \sum_{i=1}^{\sigma} \alpha^{\left(\sum_{j=1}^{i} v_j\right) - v_i} \mathbb{C}_i = \sum_{i=1}^{\sigma} \alpha^{(i-1)v} \mathbb{C}_i. \tag{4.4}$$

The binary dimension of $\mathbb{C}_{\mathrm{II}}$ is

$$K_2(\mathbb{C}_{\mathrm{II}}) = \sum_{i=1}^{\sigma} K_2(\mathbb{C}_i). \tag{4.5}$$

The error-correction capability $t_{\mathbb{C}_{\mathrm{II}}}$ of $\mathbb{C}_{\mathrm{II}}$ also has a lower bound and an upper bound

$$t \le t_{\mathbb{C}_{\mathrm{II}}} \le \sigma t = \frac{mt}{v}. \tag{4.6}$$

It is easy to see that when $v = 1$, we can reach a maximum upper bound $mt$. This maximum upper bound is far beyond the conventional error-correction capability of the parent RS code with the error-correction capability $t$. However, this results in further dimension degradation [20]. The stochastic error-correction capability deserves further study. The simulation result shows that the performance of a Class-II code is marginal compared to an RS code with the same code rate and length over the same field. We propose three subclasses of the Class-II code $\mathbb{C}_{\mathrm{II}}$. They are the $\mathbb{C}_{\mathrm{II}}^{v}$ code, the $\mathbb{C}_{\mathrm{II}}^{\langle v_1|v_2|\cdots|v_{max}\rangle}$ code and the $\mathbb{C}_{\mathrm{II}}^{\langle v_1 \supset \cdots|v_2 \supset \cdots|\cdots|v_{max} \supset \cdots\rangle}$ code, respectively.

#### 4.2.2.1 $\mathbb{C}_{\mathrm{II}}^{v}$ codes

The $\mathbb{C}_{\mathrm{II}}^{v}$ Code has two categories. The SSRS codes of the first one are all on a 1-dimension subspace of $GF(2^m)$. For the second code, the SSRS codes are all on a $d$-dimension subspace of $GF(2^m)$, where $d$ is a factor of $m$ and $d \neq 1$. We denote it as $d|m$.

**Notation**: $\mathbb{C}_{\mathrm{II}}^{v}$ codes with $v = 1$ are denoted as $\mathbb{C}_{\mathrm{II}}^{v=1}$.

**Notation**: $\mathbb{C}_{\mathrm{II}}^{v}$ codes with $v = d, d|m, d \neq 1$ are denoted as $\mathbb{C}_{\mathrm{II}}^{v=d}$.

### $\mathbb{C}_{II}^{v=1}$ *codes*

First, let us focus on $\mathbb{C}_{\mathrm{II}}^{v=1}$. The SSRS code with a 1-dimension subspace of $GF(2^m)$ is actually equivalent to the primitive binary BCH code [20]. We know that if $v = 1$ or $v = m - 1$, SSRS code is always "ordinary", which is defined in [20] as an SSRS code whose binary dimension reaches the lower bound. In [20], the author calls the code with a higher binary dimension than the lower bound as "exceptional" and concludes that the SSRS code with $v = 1$ has a unique dimension which reaches the lower bound of the SSRS code.

There are several advantages for $\mathbb{C}_{\mathrm{II}}^{v=1}$.

1) We do not need to find the "exceptional subspace", which is very difficult to find, when $m$ is large [20].
2) The SSRS code based on a 1-dimension subspace is linear over $GF(2^d)$, when $d|m$ [20], [22].
3) The SSRS code based on a 1-dimension subspace is actually equivalent to the primitive binary BCH code, which can be encoded systematically [1], [54].
4) The primitive binary BCH code is the subfield subcode of the parent RS code [1], [2]. It can be decoded by all existing available RS decoders [54] with less complexity, because its alphabet is over $GF(2)$.

5) It is easy to implement in hardware, because the symbol level RS decoder can be repeatedly used on both symbol level decoding and local bit level binary BCH decoding, where the binary BCH codeword is also a codeword of the global RS code.

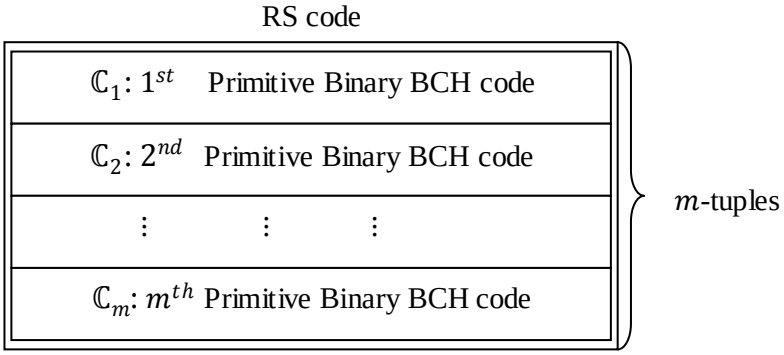


Fig. 4.7. $\mathbb{C}_{\mathrm{II}}^{v=1}$ subcodes of RS codes.

*$\mathbb{C}_{II}^{v=1}$ codes from the modification of the Vardy-Be'ery decomposition*

We developed a new scheme to generate $\mathbb{C}_{\mathrm{II}}^{v=1}$ from a modification of Vardy-Be'ery's decomposition [55]. In [55], Vardy and Be'ery demonstrated that the binary image of the generator matrix of RS codes is a "concatenation" of $m$ binary BCH code generator matrices along the diagonal of the matrix with a glue vector code generator matrix on the bottom. Vardy and Be'ery used a trellis decoding [55] and later Fedorenko [56] came up with a star trellis decoding for RS codes. In [56], the gain for small codes can achieve 2-3dB over hard-decision decoding. In [55], Vardy and Be'ery introduced an RS decoder utilizing the soft bit-level information. To use the same notation as in [55], consider the code $RS\ (N, K, D)$ over $GF(2^m)$ with

roots $(\alpha^{s+1}, \alpha^{s+2}, \cdots, \alpha^{s+N-K-1})$. A linear mapping $\phi$: $GF(2^m)^N \rightarrow GF(2^{mN})$ will convert a symbol level codeword over $GF(2^m)$ to a sequence of bits of length $mN$ with the basis $(\gamma_1, \gamma_2, \cdots, \gamma_m)$ of $GF(2^m)$ over $GF(2)$. Here, $\alpha$ is the primitive element of $GF(2^m)$. The binary BCH code $\mathcal{B}(N, k, d)$ is generated by the roots $(\alpha^{s+1}, \alpha^{s+2}, \cdots, \alpha^{s+N-K-1})$ of the RS code and their cyclotomic conjugates over $GF(2)$ [1], [55]. As described in [55], we can generate $\mathcal{B}$ with the combination of $k$ independent codewords in $\mathcal{B}$, namely, $\mathcal{B}_1, \mathcal{B}_2, \cdots, \mathcal{B}_k$.

Thus, we have the following illustration for the permutated version of $G_{RS}$ in [55].

$$\begin{bmatrix} G_{BCH} & & & \\ & G_{BCH} & & \\ & & \ddots & \\ & & & G_{BCH} \\ & G_{glue} & & \end{bmatrix}$$

Fig. 4.8. Vardy-Be'ery decomposition of the binary image of an RS code generator matrix [55].

In [55], the RS codewords are the sum of the concatenation of the $m$ binary BCH codewords and the glue codeword generated by the sub-generator matrix of glue vector in the binary image of $G_{RS}$. For long, high-rate RS codes, the binary image length will be even longer, which will make it prohibitive to calculate the glue vector part. A star-shape factor graph structure is given in [57] with the Wolf trellis

in [58] based on the Vardy-Be'ery decomposition, where the glue node acts as the constraint node.

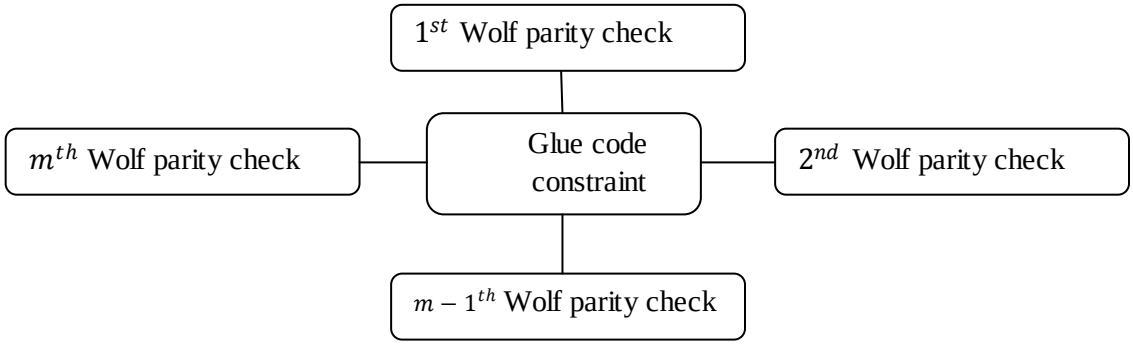


Fig. 4.9. Wolf's star shape of trellis [57].

Without correctly decoding the glue vector in [55], it will not be possible to decode the full RS codeword correctly.

$$\begin{bmatrix} G_{BCH} & & & \\ & G_{BCH} & & \\ & & \ddots & \\ & & & G_{BCH} \\ & G_{glue} & & \end{bmatrix} \rightarrow \begin{bmatrix} G_{BCH} & & \\ & G_{BCH} & \\ & & \ddots \\ & & G_{BCH} \end{bmatrix}$$

Fig. 4.10. Removal of the glue generate matrix.

Here we propose a way to remove the glue vector shown in Fig. 4.10. The new "permutated" form of the subcodes of the original RS codes with concatenation of the $m$ binary BCH codes [55] is actually a $\mathbb{C}_{\mathrm{II}}^{v=1}$ code after de-permutation. This is a

new way to generate a $\mathbb{C}_{\mathrm{II}}^{v=1}$ code with the global symbol error correction capability equaling to the local bit error correction capability. It is not the best encoding method.

Further, $m$ parallel binary BCH code can correct some 'local' error distributions. The subcode will enable to eliminate some symbol level errors by correcting some 'local' bit-level errors.

*A special subcode of $\mathbb{C}_{II}^{v=1}$ : Thangaraj and Raj's subcodes with non-trivial trace*

In [59], Thangaraj and Raj introduced a subcode of RS codes with a non-trivial trace, which is a special subcode of $\mathbb{C}_{\mathrm{II}}^{v=1}$. In this section, we use the notation of Thangaraj and Raj and refer to their code as: $SRS(t, t')$ [59]. The method using the modification of Vardy-Be'ery decomposition can generate $\mathbb{C}_{\mathrm{II}}^{v=1}$ codes with "global" $t_G$ equaling to "local" $t_L$, which are special cases of SRS codes, when $t = t'$. How about the situation when $t_G \neq t_L$ for $\mathbb{C}_{\mathrm{II}}^{v=1}$? How can we generate this special code? Thangaraj and Raj's SRS codes provide the answer. Before showing that, we need a new notation for this special subcode of $\mathbb{C}_{\mathrm{II}}^{v=1}$ codes.

**Notation**: Denote $C(n, k, D, d)$ as the notation for the special $\mathbb{C}_{\mathrm{II}}^{v=1}$ codes, Thangaraj and Raj's SRS codes. Here $D = 2t_G + 1 = 2t + 1$ and $d = 2t_L + 1 = 2t' + 1$.

Now, let us show how to generate this code with the construction method in [59].

The parent narrow sense $RS(n, k, D = 2t_G + 1)$ has generator polynomial

$$g(x) = \prod_{i=1}^{2t_G} (x + \alpha^i).$$

Let $z = \{1,2,\cdots,2t_G\}$, and $z_{\mathbb{C}_{\mathrm{II}}^{v=1}} = z \cup \mathsf{C}$, here $\mathsf{C}$ is the union of cyclotomic cosets of $\{1,2,\cdots,2t_L\}\ modulo\ n$ with respect to $GF(2)$.

The generator polynomial

$$g_{\mathbb{C}_{\mathrm{II}}^{v=1}}(x) = \prod_{i \in z_{\mathbb{C}_{\mathrm{II}}^{v=1}}} (x + \alpha^i)$$

defines a $\mathbb{C}_{\mathrm{II}}^{v=1}$ code.

In [59], the authors did not give a proof that this $\mathbb{C}_{\mathrm{II}}^{v=1}$ code is a $C(n,k,D,d)$ code with $D = 2t_G + 1$ and $d = 2t_L + 1$. We show our proof in Appendix A.

*Theorem 4.3:* The $\mathbb{C}_{\mathrm{II}}^{v=1}$ code generated by $g_{\mathbb{C}_{\mathrm{II}}^{v=1}}(x)$ is a $C(n,k,D,d)$ with $D = 2t_G + 1$ and $d = 2t_L + 1$.

*Theorem 4.4*: $t_L \leq t_G$ with $D = 2t_G + 1$ and $d = 2t_L + 1$.

The binary BCH code is the subfield subcode of an RS code [20], [26]. Thus, these $m$ parallel binary BCH codewords can be decoded by either an RS hard-decision decoder or a soft-decision decoder over $GF(2^m)$ instead of a binary BCH decoder over $GF(2)$. Using an RS symbol level soft-decision decoding algorithm, the $C(n,k,D,d)$ code can correct more 'local' bit-level errors, which are treated as the symbol-level errors.

*A multi-level decoding architecture*

In [59], Thangaraj and Raj proposed a scheme of two-stage decoding. In Section VI of [59], the first stage using BCH decoders produces $2^m$ test patterns for the second stage RS decoder, which is a list decoding. Section VII of [59] introduced the "soft-guided" decoder for trace in the first stage with a soft RS decoder for the second stage. Parts VII-B and VII-C utilized the bitwise-MAP decoder of [60] in the first stage but a hard-decision decoder and a KV in the second stage, respectively.

Based on the above, we modified the scheme by adding an extra stage in first place which leads to a three level decoding structure. Meanwhile, the decoders of each level are all RS decoders, because the binary BCH code can be decoded by RS decoders [2], [21]. This is a flexible multi-level decoding architecture. It is a slightly generalized version of Thangaraj and Raj's architecture in [59]. It has the flexible combination of three-level RS decoding algorithms. Different combinations can lead to different gains with different complexity. A powerful RS decoding algorithm is highly recommended in the second level in order to get extra bit-level error corrections, which can cause a significant error deduction in the number of the global symbol-level errors.

The three-level decoding architecture is shown in Fig. 4.11. The level 1 (L1) RS decoder can handle a small number of errors, which may be enough at high SNR's.

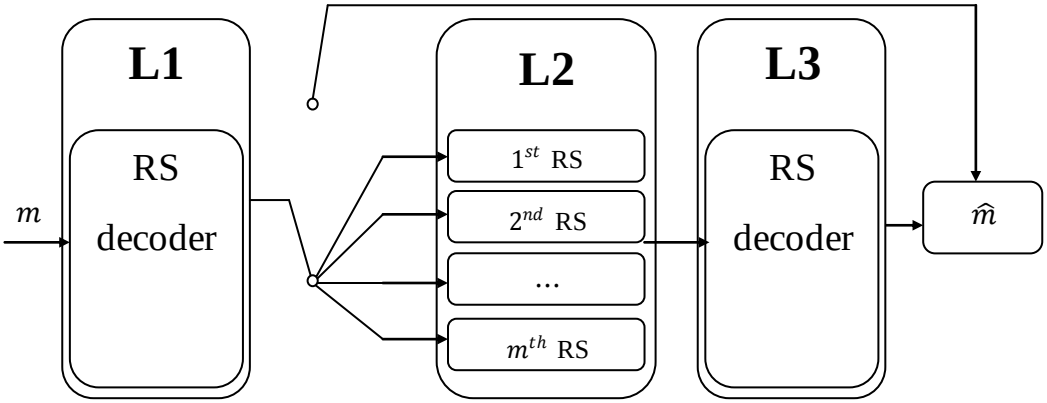


Fig. 4.11. The multi-level decoding architecture for subcodes $C(N, k, D, d)$.

The performance of the L2 decoder plays the most important role in the overall decoding performance. An iterative decoder for L2 is not recommended if low complexity is required. The level 3 (L3) is the final level for error correction. If a higher coding gain is to be achieved, a high complexity decoding algorithm is required.

Next we present an example of possible decoder combinations. Let us select the BM algorithm for the L1 decoder. The L2 decoders can be selected from any of the following algorithms GS, Chase, GMD, Bit Generalized Minimum Distance (BGMD) [61], and KV. The L3 decoder could use the LCC algorithm. The choice combination of algorithms is very flexible and therefore can accommodate computational complexity and hardware implementation requirements.

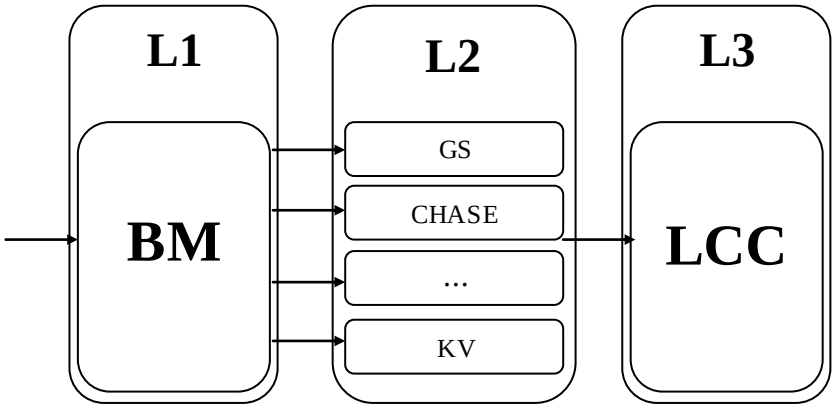


Fig. 4.12. An example of three-level decoding structure for subcodes $C(N, k, D, d)$.

*Rotation structure of* $C(n, k, D, d)$

*Definition*: $s$-shift

Given a codeword of a cyclic code $c = (c_0, c_1, c_2, \cdots, c_{n-1})$, a shift is defined as a $s$-shift, when $c = (c_0, c_1, c_2, \cdots, c_{n-1}) \rightarrow c' = (c_s, c_{s+1}, \cdots, c_{n-1}, c_0, \cdots, c_{s-1})$.

*Definition:* Rotation $\Re(s_1, s_2, \cdots, s_{m-1})$

$\Re(s_1, s_2, \cdots, s_{m-1})$ is defined, when each tuple of the $m$-tuple of a codeword performs a corresponding $s_i$-shift, $i = 1,2, \cdots, m-1$.

*Definition:* Rotation structure of $C(n, k, D, d)$

$C(n, k, D, d)$ is said to have a rotation structure, if after an arbitrary rotation $\Re(s_1, s_2, \cdots, s_{m-1})$, a codeword of $C(n, k, D, d)$ is still a codeword of $C(n, k, D, d)$.

*Theorem 4.5*: A $C(n, k, D, d)$ has a rotation structure, if $D = d$.

A $C(n, k, D, d)$ code with a rotation structure can make error redistributions, and be used to get some new codes with more elegant structure. However, we will not address this topic in this work.

*$\mathbb{C}_{II}^{v}$ codes, $v|m$ and $v \neq 1$*

Unlike $\mathbb{C}_{\mathrm{II}}^{v=1}$, when $v \neq 1$, the $v$-dimensional subspaces are not always equivalent, because some of them are "exceptional" and others are "ordinary" [20]. From Appendix A in [59, pp. 157], we can find that given $m = 4$ and $v = 2$, when the basis is $\{1, \alpha\}$, the 2-dimensional subspace is "ordinary"; however the one with basis $\{1, \alpha^5\}$ is "exceptional" [20]. Here, define $\alpha$ to be the primitive element of $GF(2^4)$. The above SSRS code with basis {1, $\alpha^5$} is actually a subfield subcode of the parent RS code [1], [20]. The maximum binary dimension is dependent on the parent RS code, the subspace dimension $v$, and also the basis of the subspace [20], which can be found out from the examples in Appendix C of [20]. For example, given $m = 6$ and $v = 2$, when the parent RS code has a symbol dimension of 35, the binary dimensions of the SSRS codes with basis $\{1, \alpha\}$, $\{1, \alpha^9\}$, and $\{1, \alpha^{21}\}$ are 24, 27, and 26, respectively [20, Appendix C, p. 192]. The two SSRS codes with binary dimensions of 27 and 26 above are "exceptional", but the one with the basis $\{1, \alpha^{21}\}$ is the subfield subcode of the parent RS code [1], which tells that the subfield subcode of parent RS code could be "exceptional" but may not have the

maximum dimension [20]. From [20, Appendix C, p. 192], it can be seen that given parent RS codes with symbol dimensions from 54 to 62, the SSRS codes based on all $v = 2$ dimensional subspaces are all "ordinary".

Based on the facts above, we make the following observations: 1. It is difficult to get a maximum binary dimension SSRS code from the parent RS code. 2. The maximum binary dimension may not be a multiple of $v$, which will create a non-integer "$2^v$"-symbol dimension. 3. It is hard to encode systematically.

To limit implementation complexity, in this work, we use the special case of this kind of SSRS codes: the $2^v$-ary BCH codes over $GF(2^m)$ for these parallel SSRS codes in Fig. 4.6.

**Notation:** $C(n, k, D, d_{(v)})$

$C(n, k, D, d_{(v)})$ denotes $\mathbb{C}_{\mathrm{II}}^{v}$ codes with $D = 2t_G + 1$ over $GF(2^m)$ and $d_v = 2t_L + 1$ over $GF(2^v)$. Here $v \neq 1$.

The construction methods are the same as those in Section 4.2.2.1.1.3.

### 4.2.2.2 $\mathbb{C}_{\mathrm{II}}^{\langle v_1|v_2|\cdots|v_{max}\rangle}$ codes

Actually, by "embedding" certain zeros [1], [59], the parent RS code over $GF(2^m)$ can be modified to a more generalized subcode of RS code of Class-II. This code can have several parallel $2^v$-ary BCH codes structures simultaneously shown in Fig. 4.13.

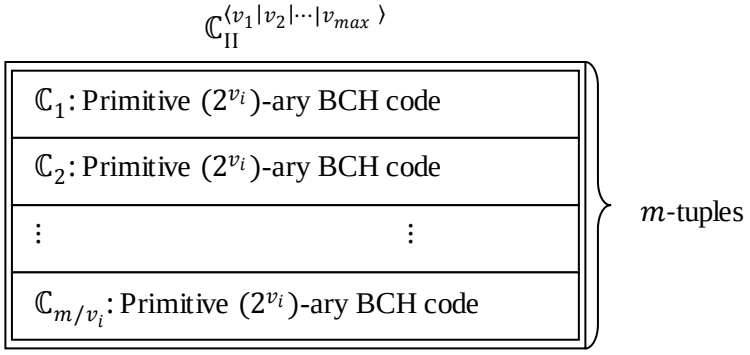


Fig. 4.13. $\mathbb{C}_{\mathrm{II}}^{\langle v_1|v_2|\cdots|v_{max}\rangle}$ code.

Here $v_i$ is a divider of $m$. The parent RS code is over $GF(2^m)$. From the construction method in [59] and the "extraneous zeros" concept in [1], we derived this Zeros-Embedding Algorithm to get the $\mathbb{C}_{\mathrm{II}}^{\langle v_1|v_2|\cdots|v_{max}\rangle}$ code, which is a generalized more powerful version of $\mathbb{C}_{\mathrm{II}}$ with multiple parallel structures simultaneously with an arbitrary $\{t_{L_1}, t_{L_2}, \cdots, t_{L_{max}}\}$.

*Zeros-Embedding Algorithm*

1) Given $GF(2^m)$, $m$ is a positive integer, there exist a series of positive dividers

$$\{v_1, v_2, \cdots, v_{max} \mid \ \forall v_i | m, v_1 = 1 < v_2 < \cdots < v_{max} \ < m\}.$$

2) Given a parent narrow sense $RS(n, k, D = 2t_G + 1)$ with generator polynomial

$$g(x) = \prod_{i=1}^{2t_G} (x + \alpha^i),$$

$\mathbb{C}_{\mathrm{II}}^{\langle v_1|v_2|\cdots|v_{max}\rangle}$ has $d = |\{v_1, v_2, \cdots, v_{max}\}|$ distinct parallel structures, which contain parallel primitive $(2^{v_i})$-ary BCH codes respectively with corresponding error correction capabilities of $\{t_{L_1}, t_{L_2}, \cdots, t_{L_{max}}\}$.

3) Compute the corresponding cyclotomic cosets

$$\mathsf{C}_{t_{L_i}} \text{ of } \{1,2,\cdots,2t_{L_1}\}, \cdots, \{1,2,\cdots,2t_{L_{max}}\}$$

module $n$ with respect to $GF(2^{v_1}), \cdots, GF(2^{v_{max}})$.

4) $\mathsf{C}$ is the union of the $\mathsf{C}_{t_{L_i}}$: $\mathsf{C} = \mathsf{C}_{t_{L_1}} \cup \mathsf{C}_{t_{L_2}} \cup \cdots \cup \mathsf{C}_{t_{L_{max}}}$ .

Let $z = \{1,2,\cdots,2t_G\}$, and $z_{\mathbb{C}_{\mathrm{II}}^{\langle v_1|v_2|\cdots|v_{max}\rangle}} = z \cup \mathsf{C}$.

$$g_{\mathbb{C}_{\mathrm{II}}^{\langle v_1|v_2|\cdots|v_{max}\rangle}}(x) = \prod_{i \in z_{\mathbb{C}_{\mathrm{II}}^{\langle v_1|v_2|\cdots|v_{max}\rangle}}} (x + \alpha^i)$$

5) Compute the codeword of $\mathbb{C}_{\mathrm{II}}^{\langle v_1|v_2|\cdots|v_{max}\rangle}$.

$$c_{\mathbb{C}_{\mathrm{II}}^{\langle v_1|v_2|\cdots|v_{max}\rangle}} = g_{\mathbb{C}_{\mathrm{II}}^{\langle v_1|v_2|\cdots|v_{max}\rangle}}(x) m(x).$$

The code $\mathbb{C}_{\mathrm{II}}^{\langle v_1|v_2|\cdots|v_{max}\rangle}$ has multiple distinct parallel structures, which can make error redistributions possible with a proper basis selection. Multiple parallel structures for $\mathbb{C}_{\mathrm{II}}^{\langle v_1|v_2|\cdots|v_{max}\rangle}$ are shown in Fig. 4.14.

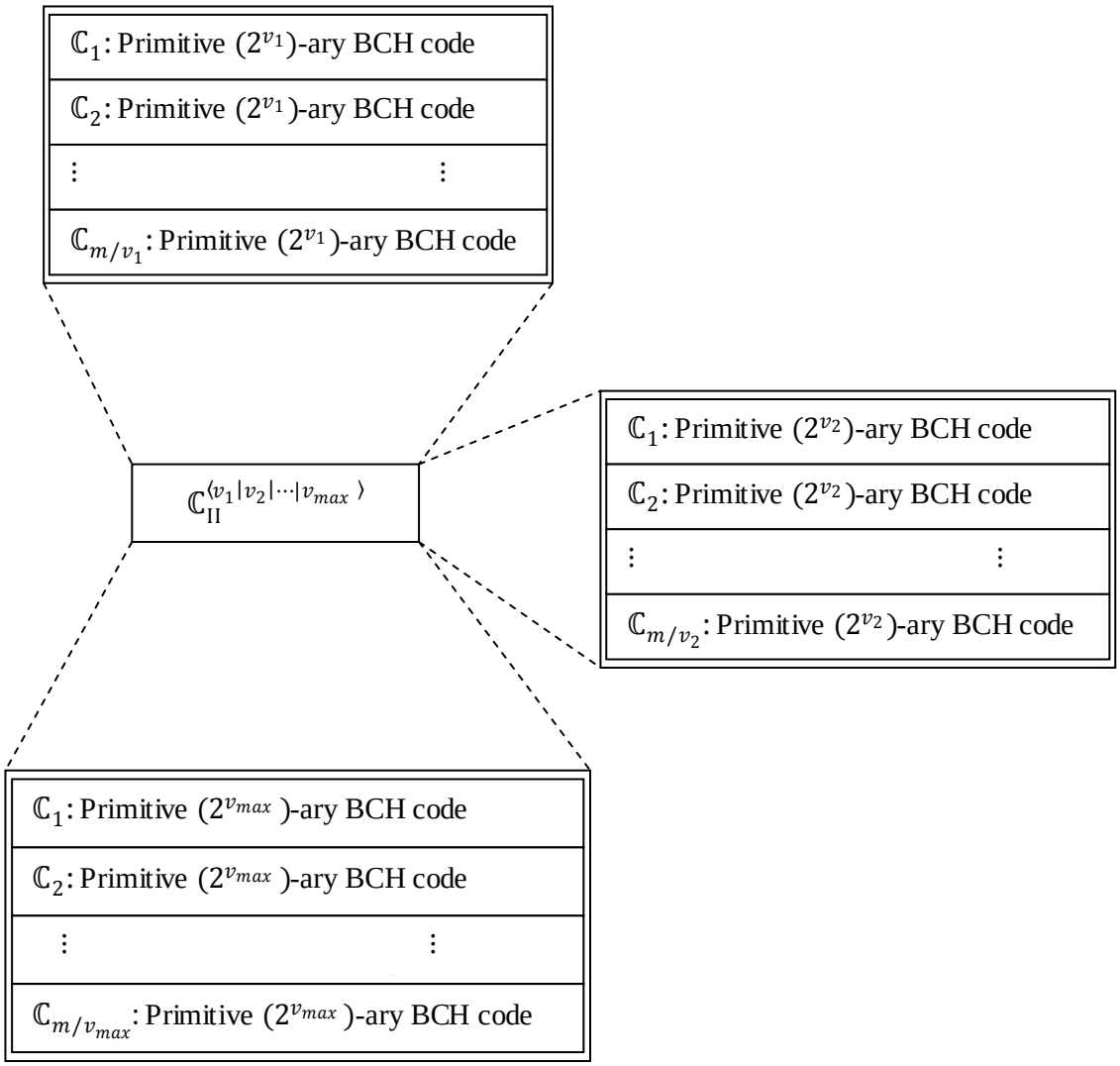


Fig. 4.14. Multiple parallel structures for $\mathbb{C}_{\mathrm{II}}^{\langle v_1|v_2|\cdots|v_{max}\rangle}$ with different $\langle v_1|v_2|\cdots|v_{max}\rangle$.

### 4.2.2.3 Multiple-stage $\mathbb{C}_{\mathrm{II}}^{\langle v_1\supset v_{1.1}\supset v_{1.2}\supset\cdots|v_2\supset v_{2.1}\supset v_{2.2}\supset\cdots|\cdots|v_{max}\supset v_{max.1}\supset v_{max.2}\supset\cdots\rangle}$ codes

Let us simplify the notation to $\mathbb{C}_{\mathrm{II}}^{\langle v_1\supset\cdots|v_2\supset\cdots|\cdots|v_{max}\supset\cdots\rangle}$. From [55, Fig. 2], we know that $\mathbb{C}_{\mathrm{II}}^{v}$ with $v \neq 1$ should have a recursive structure as shown in Fig. 4.15.

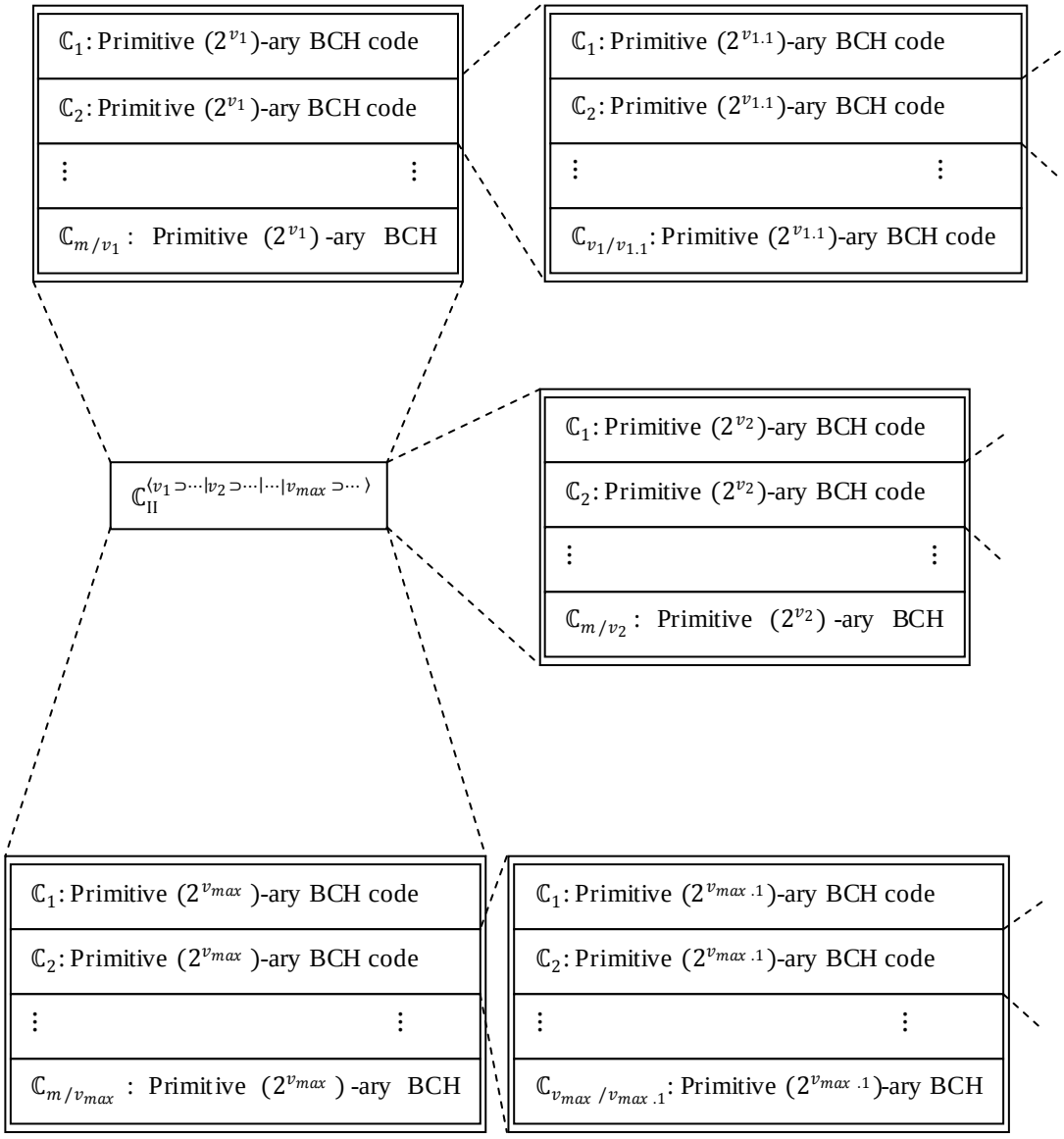


Fig. 4.15. Multiple stage structures of $\mathbb{C}_{\mathrm{II}}^{\langle v_1 \supset \cdots | v_2 \supset \cdots | \cdots | v_{max} \supset \cdots \rangle}$.

Here we have $v_{i.d_i} = 1|\cdots|v_{i.2}|v_{i.1}|v_i$. We can arbitrarily select a depth $d_i$ which satisfies $1 \leq v_{i.d_i}|\cdots|v_{i.2}|v_{i.1}|v_i$. For the construction method of $\mathbb{C}_{\mathrm{II}}^{\langle v_1 \supset \cdots | v_2 \supset \cdots | \cdots | v_{max} \supset \cdots \rangle}$, please refer to Appendix D for details.

## 4.3 Simulation results

The performance of the PSSRS is investigated on a PMRC equalized to an optimal GPR4 target with 90% jitter noise. The channel detector uses the BCJR algorithm to provide the soft-decision information to soft-decision decoder. We investigated several different codes from the short, low-rate code, and the shortened sector length code to the long, high-rate code. We chose the multi-level decoding architecture for PSSRS codes, and used the conventional RS hard-decision decoding as the baseline.

Firstly, we tested a $C(7,4,3,3)$ code over $GF(2^3)$. The $C(7,4,3,3)$ code has a global symbol level error correction capability of one, and a local bit level error correction capability of one, respectively. The curve marked with the left-pointing triangle stands for the performance of the conventional RS hard-decision decoding. In Fig. 4.16, the curve marked with the circle stands for the performance of the $C(7,4,3,3)$ code with the multi-level decoding architecture. The RS decoding algorithm combination consists of hard-decision decoding RS algorithms on all three levels. The one marked with squares also uses a multi-level architecture with a hard-decision decoding RS algorithm for the L1, and an LCC algorithm for both L2 and L3. On L2 and L3, the LCC algorithms both use $\eta = 1$. It is shown in Fig. 4.16 that the $C(7,4,3,3)$ algorithm outperforms the conventional hard-decision decoding

algorithm at an FER of $10^{-5}$. The coding gains are about 1dB, and 2dB over RS hard-decision decoding only at an FER of $10^{-5}$, respectively.

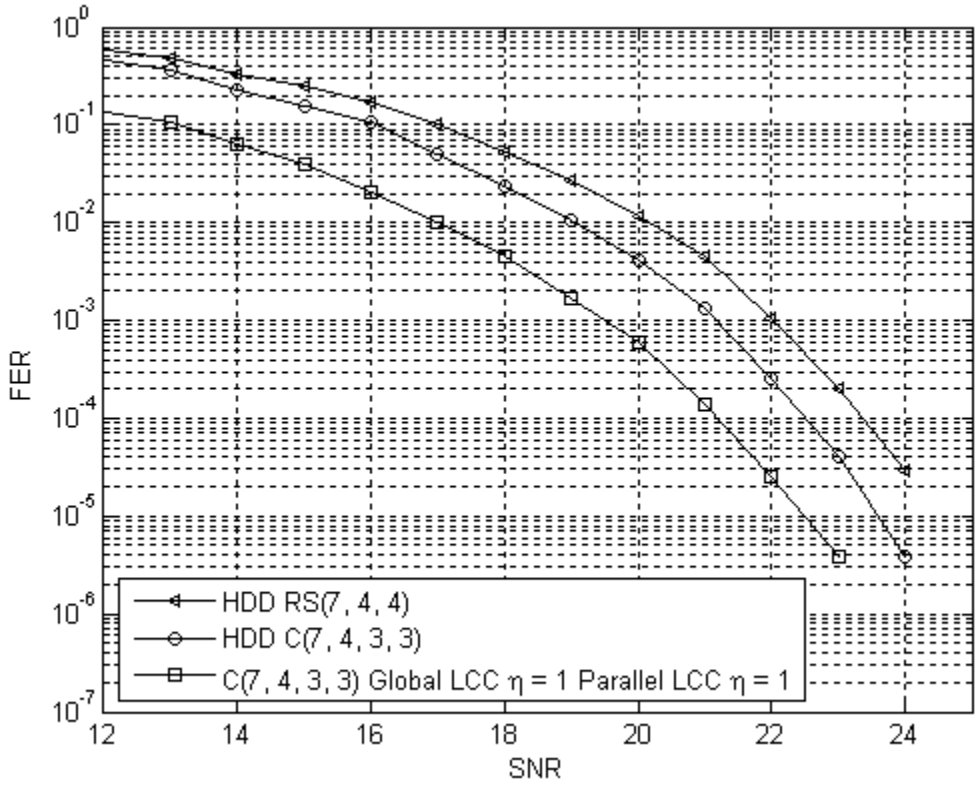


Fig. 4.16. Performance of $C(7,4,3,3)$ over $GF(2^3)$ on a PMRC equalized to an optimal GPR4 target with 90% jitter noise. The user density is 1.5309. The cod rate is 0.571.

Second, we tested a shortened $C(460,410,11,11)$ code over $GF(2^{10})$ under the same conditions. Fig. 4.17 shows its performance using a three–level decoding architecture. We use a hard-decision decoding algorithm for L1 and parallel LCC

algorithms with $\eta = 7$ for L2 and an LCC algorithm with $\eta = 1$ for L3. We observe a coding gain of about 0.4 dB over hard-decision decoding at an FER of $10^{-4}$. The code rate is 0.891.

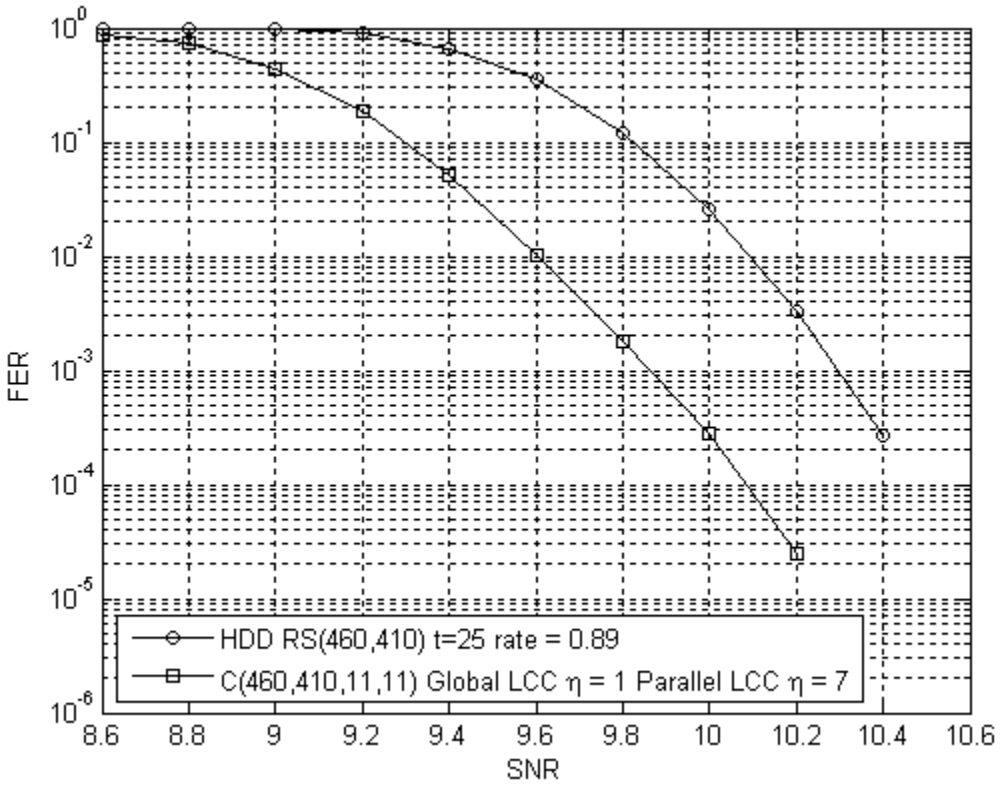


Fig. 4.17. Performance of $C(460,410,11,11)$ over $GF(2^{10})$ on a PMRC equalized to an optimal GPR4 target with 90% jitter noise. The user density is 0.9811. The cod rate is 0.891.

We finally tested a $C(1023,973,25,9)$ code over $GF(2^{10})$ under the same condition. This is a long, high-rate code, where the code rate $R = 0.951$. Fig. 4.17 shows that the three-level decoding algorithm has a coding gain of about 0.5 dB over

hard-decision decoding at an FER of $10^{-4}$. To evaluate the stochastic error-correction capability, we tested the performance of the pseudo RS hard-decision decoding with $t = 40$ and $t = 50$. The performance of the curve with $t = 40$ over the one of the $C(1023,973,25,9)$ code with three-level decoding architecture is marginal. The coding gain of the pseudo RS hard-decision decoding with $t = 50$ over the $C(1023,973,25,9)$ code is about 0.5 dB at an FER of $10^{-4}$. The stochastic error-correction capability is very close to the theoretic upper bound of the $C(1023,973,25,9)$ code. It shows that the stochastic error-correction capability can reach or even go beyond the upper bound by choosing some more powerful RS decoders for both L2 and L3.

The simulation results show that the short, low-rate PSSRS codes can get larger coding gains compared to the long, high-rate codes. The RS decoder at L2 is the most important overall. A moderate coding gain can be achieved with low-complexity RS decoding algorithms on both L2 and L3. For ultra long, high-rate codes, the PSSRS using multi-level decoding architecture can achieve more gain over other soft-decision decoding algorithms with the same computational complexity, because the conventional ultra long, high-rate RS codes have a relatively much higher error-correction capability $t$ than both the global symbol level error-correction capability and the local bit level error-correction capability of PSSRS codes.

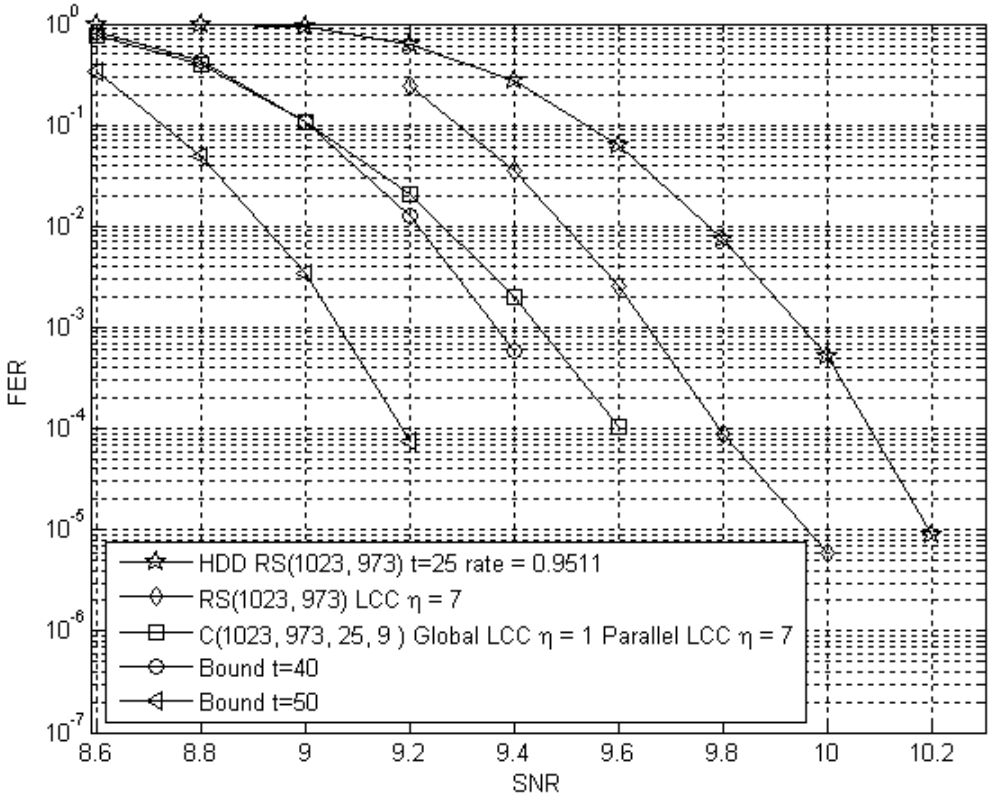


Fig. 4.18. Performance of $C(1023,973,25,9)$ over $GF(2^{10})$ on a PMRC equalized to an optimal GPR4 target with 90% jitter noise. The user density is 0.919. The cod rate is 0. 951.

## 4.4 Summary

In this chapter, we propose new PSSRS codes with a parallel SSRS structure. We also propose a flexible multi-level decoding architecture, which enables a flexible combination of choices of RS decoding algorithms. For practical purposes, we propose the systematic encoding algorithms for $\mathbb{C}_{\mathrm{II}}^{v=d}$ codes, $\mathbb{C}_{\mathrm{II}}^{\langle v_1|v_2|\cdots|v_{max}\rangle}$ codes,

and $\mathbb{C}_{\mathrm{II}}^{\langle v_1 \supset \cdots | v_2 \supset \cdots | \cdots | v_{max} \supset \cdots \rangle}$ codes, whose SSRS codes are "ordinary". The PSSRS code can achieve a better performance over the conventional RS code of the same length and the same rate with the same computational complexity if L2 decoding is in a real parallel fashion. The simulation results show that the PSSRS code is very promising for the very long, high-rate codes because of a lower decoding computational complexity, which is caused by the fact that PSSRS codes have a relatively small global symbol level error-correction capability and even smaller parallel local bit level error-correction capabilities without sacrificing the overall stochastic error-correction capability. The PSSRS code also offers some local robustness, which cannot be achieved by the conventional RS codes. The PSSRS can get a better performance, if a faster "exceptional" SSRS systematic encoding method can be found.

# Chapter 5

# Iterative Parallel Local Decoding of LDPC+RS Concatenated Codes

## 5.1 Introduction

The performance of LDPC codes has been shown to have large gains over RS hard-decision decoding. Unlike RS codes, LDPC codes always have an "error-floor" problem at high SNRs [13], [16], [62], [63], which seriously limits its practical application in systems requiring extremely high reliability, such as storage systems. Lots of research has been carried out to study long codes for the next generation of 4KB sectors [8]-[11]. As we know, for this form of very long codes, the decoding complexity of LDPC codes could be prohibitive for practical implementations. The "error-floor" problem will be extremely severe at very high densities which are required to build high capacity storage systems with good "format efficiency" [8]-[11]. Efforts have been carried out to handle these problems. There exist some dominant error patterns called "trapping sets" [54]-[60], which are one of the major issues causing the "error-floor". Pre-detecting and avoiding these "trapping sets", which are usually collected by computer and hardware simulations [13], helps to somehow lower the "error-floor". There is no evidence that this method can theoretically solve the "error-floor" problem. Another effort is to use a concatenation system where an RS code works as an outer code and an LDPC code as an inner code. This method can lower the "error-floor" with a degradation of the coding gain compared to the LDPC only system. The conventional LDPC+RS system can lower the "error-floor" due to the outer RS code. But in some cases, the LDPC code has a failure in decoding certain error patterns, and causes error propagation, which create

more errors than the error-correction capability of the RS code. The current conventional LDPC+RS system cannot avoid this situation by skipping certain "trapping sets" easily.

We propose some new schemes for LDPC+RS concatenation systems, some of which have demonstrated the ability to achieve:

1) Parallel decoding.
2) Changing the error-pattern by "locally" eliminating some error bits.
3) Redistributing the error bits by means of a multiple codeword interleaver.
4) $0.5dB^{+}$ gain over conventional LDPC+RS concatenation systems on PMRCs with an iterative parallel local decoding algorithm.
5) Lowering the complexity of very long LDPC codes.

## 5.2 Concatenated codes

A concatenated code is a class of codes, which contains an inner code and an outer code [1]. The most famous example is the concatenated code for NASA's deep space communication in the Voyager program, which uses an $RS(255,223,33)$ code and a rate $1/2$ convolutional code (CC) with constraint length seven [1]. Although RS codes have a better performance at high SNRs, the CC code outperforms RS codes at low channel SNRs [1]. The RS+RS concatenation codes are very suitable for high reliability systems [1]. However, to achieve a moderately reliable performance for a

power-poor channel like the satellite downlink system [1], the RS+CC concatenation is a better choice [1], [64].

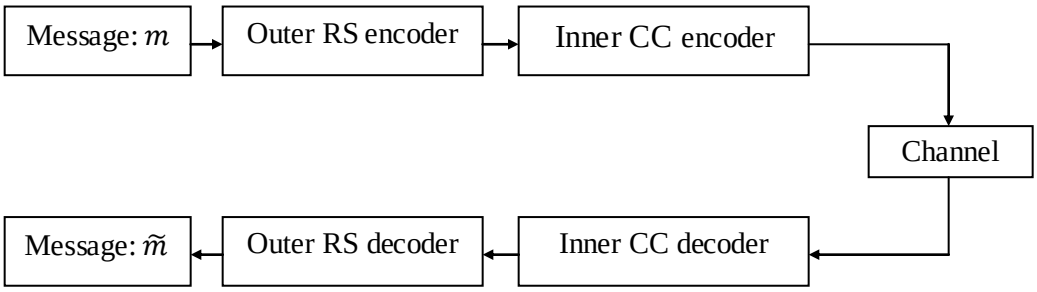


Fig. 5.1. RS+CC Concatenated coding system [1], [64].

This concatenation system still has applications nowadays. However, with the rediscovery of LDPC codes, people replace the convolutional code with a more powerful LDPC code as the inner code, because LDPC codes can get a much better performance at low SNRs. The outer RS code can remove the remaining errors from the output of the LDPC decoder, which inherently has an "error-floor" problem. An LDPC+BCH concatenation scheme was proposed for the DVB-C2 standard [65], which is reported to approach the theoretic Shannon limit. For storage systems, one of the architectures of the current generation of HDD systems uses a concatenation of an inner LDPC code with an RS outer code [66], which will be introduced in the next section.

## 5.3 Conventional LDPC+RS concatenation

LDPC is reported to approach the Shannon capacity limit over several channels [67], [68]. Using the performance of the RS hard-decision decoding as the baseline, the LDPC codes can achieve a large gain of more than 2dB in certain channels. However, the LDPC code is not as efficient as the RS code for dealing with burst errors [66], which are common in magnetic storage channels. The media noise, thermal asperity and media defects are the major sources contributing to the burst errors [1]. For decades, RS codes were utilized as the conventional method to dealing with these burst errors due to their burst error correction nature [1], especially when the alphabet size is large. There are two choices for the next generation reading channels in magnetic storage systems: the first is the LDPC only (one level) system. The second is the conventional LDPC+RS concatenation system. The LDPC+RS concatenation system has a smaller gain at low SNRs and a lower error floor than the LDPC only system. The inner LDPC code is more powerful for handling random noises, and can correct most random errors.

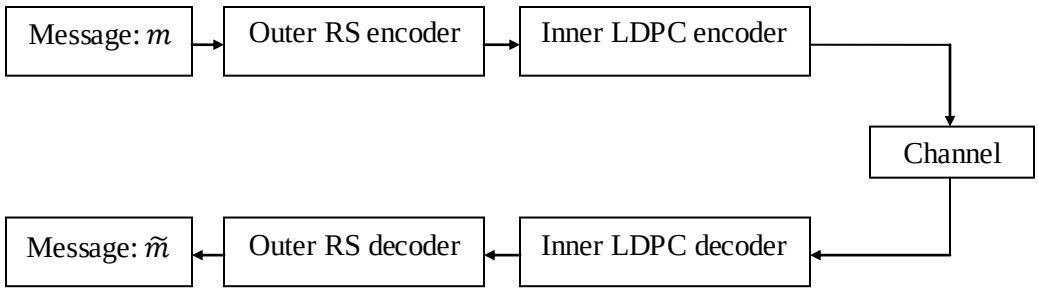


Fig. 5.2. Conventional LDPC+RS coding system.

The outer RS code is designed to correct burst errors from the output of the inner decoder or certain residual errors. The performance of the LDPC+RS concatenation system on magnetic recording channels from different points of view is investigated in [66], [69]-[71]. The authors concluded that the outer RS code cannot simply improve the overall performance, which needs an optimal selection of parameters for both LDPC and RS codes at a particular bit density [69], [71]. In the low SNR region, the LDPC only system is better than the LDPC+RS concatenation system [69], because of two reasons: the LDPC nature and the density penalty of the LDPC+RS concatenation system. The worst situation in this LDPC+RS concatenation system happens, when the number of the burst errors is beyond the error correction capability of the outer RS code [71]. Thus RS codes cannot correct the burst errors from the output of inner LDPC codes, and the concatenation structure reduces the overall rate causing a severe density penalty [69], [71]. Some of the errors at the output of the LDPC decoder are caused by dominant "trapping sets" [13], [72]. The outer RS codes with the error correction capability $t$ can only deal with the situation that it has no more than $t$ symbol errors. To decode successfully, the conventional LDPC+RS system requires the error correction capability of RS codes beyond the number of errors from the LDPC output. In this work, we proposed a new decoding framework where it is not necessary for RS codes to correct all the errors all at once, but just partially destroy the "trapping set", then the remaining errors can be corrected by the LDPC decoder in an iterative scheme. This idea cannot

work in the conventional LDPC+RS system, because the conventional RS codes cannot partially correct errors. However the PSSRS codes can perform partial correction. In this work, we will focus on the $\mathbb{C}_{\mathrm{II}}$ codes, especially the $\mathbb{C}_{\mathrm{II}}^{v=1}$ codes, because the $\mathbb{C}_{\mathrm{II}}^{v=1}$ codes have a higher dimension, which suffers a smaller density penalty.

## 5.4 New LDPC+RS ($\mathbb{C}_{\mathrm{II}}^{v=1}$) concatenation system

We can simply replace RS codes with $\mathbb{C}_{\mathrm{II}}^{v=1}$ codes as the outer code in the conventional concatenation system.

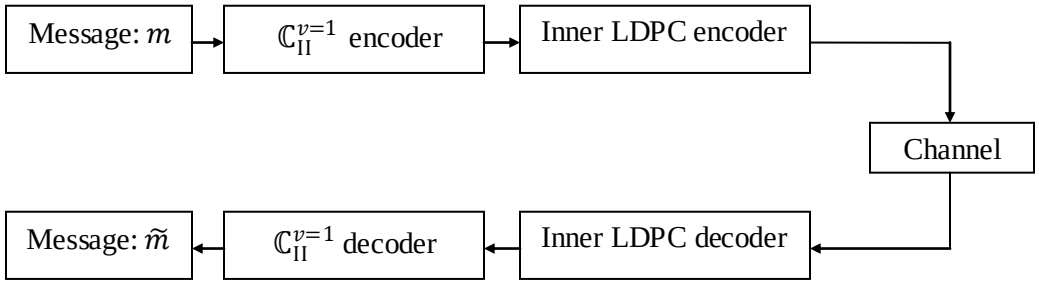


Fig. 5.3. LDPC+PSSRS in a conventional format.

In this concatenation system, $\mathbb{C}_{\mathrm{II}}^{v=1}$ codes try to correct all the burst errors from the output of the LDPC decoder output. The encoding procedure is the same as the conventional concatenation system. The multiple-level decoding architecture provides a flexible choice of RS decoders based on gain and complexity. In Chapter

4, we have shown that the $\mathbb{C}_{\mathrm{II}}^{v=1}$ with RS hard-decision decoding algorithms on all three levels only provides a marginal gain over the conventional hard-decision decoding. There is no advantage by simply using the $\mathbb{C}_{\mathrm{II}}^{v=1}$ codes to replace the RS codes with the conventional decoding framework. As said in Section 5.3, $\mathbb{C}_{\mathrm{II}}^{v=1}$ codes can partially correct the error pattern, and LDPC codes can correct the rest of the errors. In the concatenation system in Fig. 5.3, $\mathbb{C}_{\mathrm{II}}^{v=1}$ codes cannot correct any errors from the output of the LDPC decoders, when the error patterns are distributed in the parity part of the LDPC codewords. Thus, we proposed another concatenation structure.

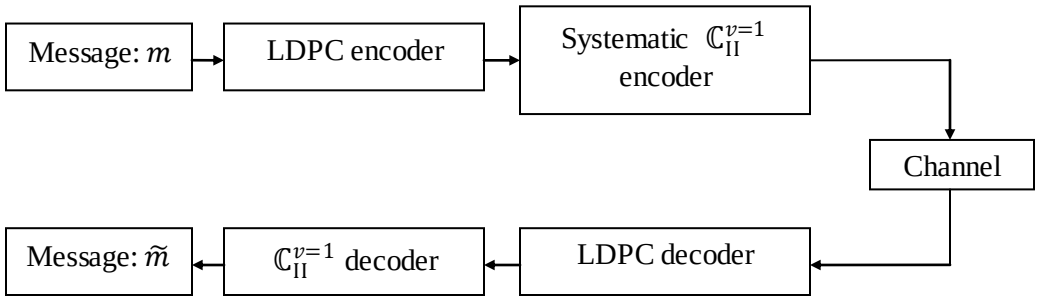


Fig. 5.4. New LDPC+RS ($\mathbb{C}_{\mathrm{II}}^{v=1}$) concatenation system.

In this new system, we change the order of the $\mathbb{C}_{\mathrm{II}}^{v=1}$ encoder and the LDPC encoder. This will not change the position of the messages, because both the $\mathbb{C}_{\mathrm{II}}^{v=1}$ and LDPC encoder are systematic. At the channel output, we use the LDPC decoder to correct most errors in the LDPC codewords. The remaining errors in the LDPC codewords can be partially corrected by some of the $m$ parallel local structure of

$\mathbb{C}_{\mathrm{II}}^{v=1}$ codes. The error patterns from the output of the LDPC decoder are distributed into the $m$ parallel local structure of $\mathbb{C}_{\mathrm{II}}^{v=1}$ codes. If at least one of these $m$ parallel local structures can correct some part of the errors, the $\mathbb{C}_{\mathrm{II}}^{v=1}$ codes then change the error patterns by partially correcting some errors. Then the new error pattern may be corrected by the LDPC decoders. The decoding workflow is shown in Fig. 5.5.

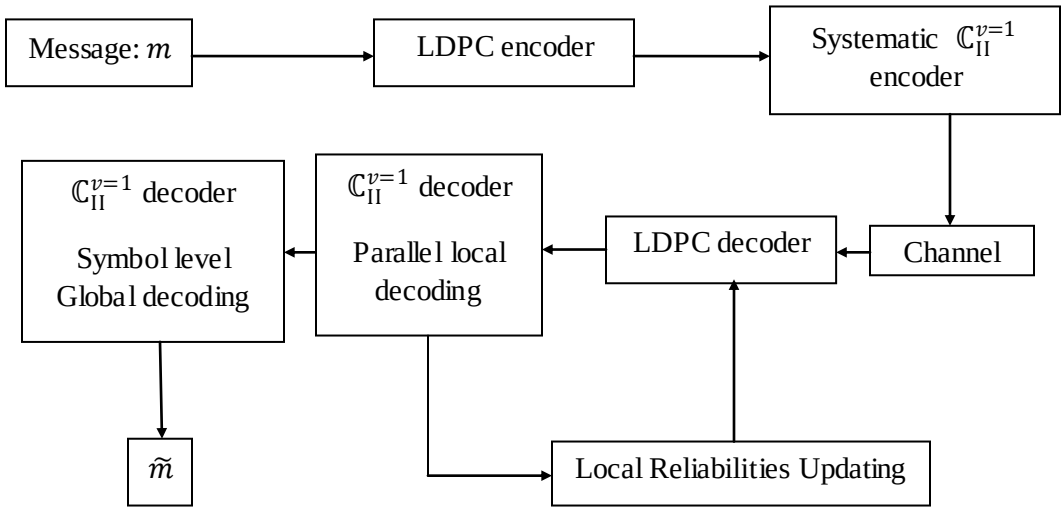


Fig. 5.5. The decoding workflow of the new LDPC+RS ($\mathbb{C}_{\mathrm{II}}^{v=1}$) concatenation system.

## 5.5 Iterative parallel local decoding of LDPC+RS concatenation system

Compared to the conventional LDPC+RS concatenation system, our method uses the parallel structure of $\mathbb{C}_{\mathrm{II}}$ codes to correct a small number of partial errors of

stable “trapping sets”. There has been a lot of work about the error floor of the iterative decoding algorithms based on graphical model with cycles [13], [14], [16], [72]-[75], which show that the major cause of the error floor are trapping sets [13] and absorbing sets [73], [74]. The $(a, b)$ trapping sets are usually referred to $a$ variable nodes in error and $b$ check nodes of odd degrees associated with these variable nodes in the subgroup of these wrong variable nodes [13]. It is very difficult to find the complete list of trapping sets [13], but it is possible to search for dominant larger classes, of which it is enough to find a representative. Absorbing sets are essentially special cases of trapping sets [13], [73], so in this work, in general, we just refer to trapping sets as the structure causing the error floor problem in the high SNR region.

### 5.5.1 Avoiding trapping sets

The idea of avoiding trapping sets is to destroy the trapping sets iteratively in two possible ways. The first one is to correct part of the variable nodes in error. The second one is to add additional “virtual dynamic” check nodes connecting to these variable nodes. Currently, these two are equivalent due to the fact that the graphical model of the parity check matrix makes no changes to the BP algorithm, although we add some additional “virtual dynamic” check nodes connecting to these variable nodes. The “virtual dynamic” check nodes do not get involved into the graphical model for BP decoding, thus make no changes to the neighborhood relationship of

the original graph. Later we will design an LDPC code with the local structure, which can decode locally and in parallel. In that decoding algorithm, some real dynamic check nodes will be added to the Tanner graph of the local parity check matrix, or added between two or more local Tanner graphs as a "bridge". The local Tanner graph has a larger girth or is totally cycle free. The purpose is to avoid trapping sets as much as we can. Furthermore the local structure can locate the trapping sets locally, and the current trapping sets will be destroyed in most cases by adding some other local Tanner graphs. To do this process iteratively based on what trapping sets we hunt, we will select the dynamic check nodes to add.

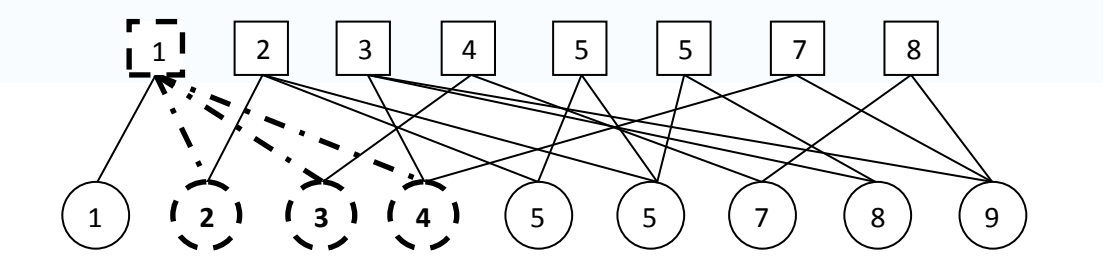


Fig. 5.6. A (3,1) trapping set.

In Fig. 5.6, we have a (3,1) trapping set, whose variable nodes 2, 3, 4 are in error. This will cause an LDPC decoding failure. In Fig. 5.7, the $\mathbb{C}_{\mathrm{II}}^{v=1}$code helps to add some "virtual dynamic" check nodes as the $\mathbb{C}_{\mathrm{II}}^{v=1}$code global check nodes and the $\mathbb{C}_{\mathrm{II}}^{v=1}$code local check nodes. Currently these "dashed virtual dynamic" connections are not involved in the BP decoding, but just correct some wrong variables (variable

4, say) to destroy the (3,1) trapping set. We have to repeat this very important idea that in order to correct the errors coming from the trapping sets, it is not necessary to expect and require the outer RS code to correct all the errors due to the fact that multiple trapping sets will happen simultaneously in one LDPC codeword, of which the number of total errors will go beyond the error correction capability of the outer codes.

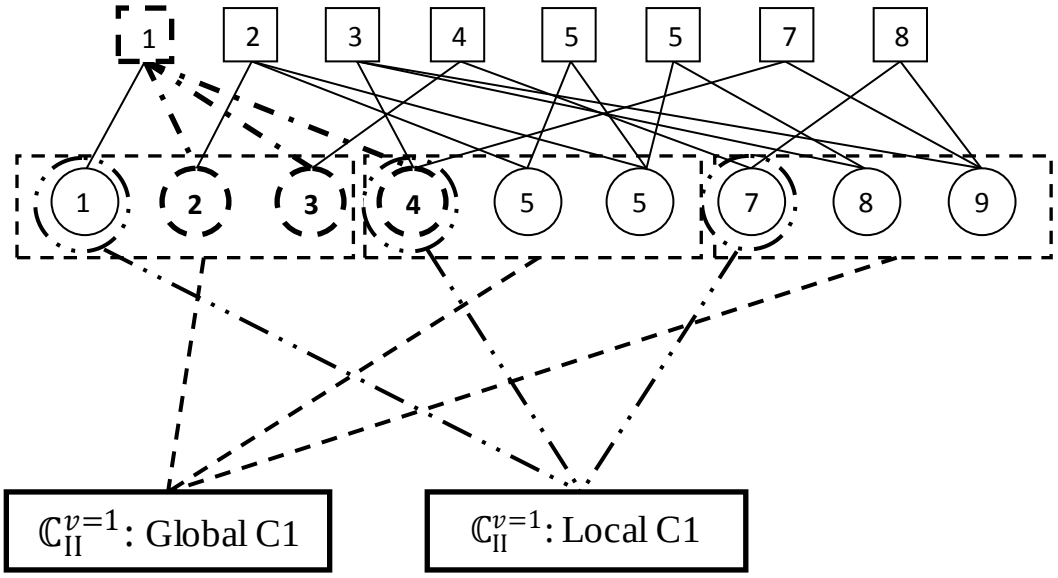


Fig. 5.7. "Virtual dynamic" check nodes added by $\mathbb{C}_{\mathrm{II}}^{v=1}$ codes.

The more efficient way is to destroy the trapping set by either correcting some partial errors or changing the trapping set structure. This idea is partially motivated, strengthened and tested by the examples in [72], [73], [76]. In [72], the author proposed a bi-mode erasure decoding in order to change the structure of some typical trapping sets. In [73], the bit-pinning technique is to fix some variable bits, which

have very high possibilities to be trapped in a trapping set, to a maximum reliability value. In [76], a post-processing method is adopted to weaken the message passing among the nodes in the strapping set, however, it strengthens the one from outside the trapping set.

In this LDPC+$\mathbb{C}_{\mathrm{II}}^{v=1}$ concatenation system, the $m$ parallel local structure can be utilized to correct local errors in order to partially change the error patterns. The LDPC decoder may correct the changed error patterns or get "trapped" again. Then the $\mathbb{C}_{\mathrm{II}}^{v=1}$ codes have to correct some local errors again. This iterative process continues until the inner and outer codewords are both verified to be valid or a pre-setup maximum iteration has been reached. In some cases, the $\mathbb{C}_{\mathrm{II}}^{v=1}$ codes have too many errors in all $m$ parallel local structures, none of which can correct some partial errors. Thus the LDPC+RS ($\mathbb{C}_{\mathrm{II}}^{v=1}$) concatenation system is recommended, using the Chase type algorithm to decode the $\mathbb{C}_{\mathrm{II}}^{v=1}$ codeword or flip the bit to destroy the trapping sets directly. The LDPC+ $\mathbb{C}_{\mathrm{II}}^{\langle \mathrm{v}_1 | \mathrm{v}_2 | \cdots | \mathrm{v}_{\max} \rangle}$ concatenation system is recommended to the channel without a severe density penalty because of a lower dimension. However, $\mathbb{C}_{\mathrm{II}}^{\langle \mathrm{v}_1 | \mathrm{v}_2 | \cdots | \mathrm{v}_{\max} \rangle}$ will bring various "virtual dynamic" check nodes of different combinations in order to destroy most trapping sets. The simulation performance of this novel LDPC+RS ($\mathbb{C}_{\mathrm{II}}^{v=1}$) concatenation system shows a great improvement in destroying the trapping sets and leads to a $0.5dB^{+}$ coding

gain and a sharp waterfall region comparing to the LDPC only system which has demonstrated a strong tendency to error floor at high user density in the PRMC.

### 5.5.2 Iterative parallel local decoding algorithm

We propose an iterative parallel local decoding algorithm for LDPC+PSSRS systems. In the following algorithm, we specify the PSSRS code to be a $\mathbb{C}_{\mathrm{II}}^{v=1}$ code.

*Iterative parallel local decoding algorithm*

1) Encode the message information using systematic LDPC encoder.
2) Interleave the LDPC codeword(s).
3) Use the interleaved LDPC codeword(s) as the message of the systematic $\mathbb{C}_{\mathrm{II}}^{v=1}$ encoder.
4) At the channel output, de-interleave the information part of $\mathbb{C}_{\mathrm{II}}^{v=1}$ codeword, which is the LDPC codeword.
5) Perform the LDPC decoding, and get a candidate LDPC codeword.
6) Interleave the LDPC codeword, and replace the information part of $\mathbb{C}_{\mathrm{II}}^{v=1}$ codeword.
7) Perform $m$-tuple parallel local decoding of the $\mathbb{C}_{\mathrm{II}}^{v=1}$ code.
8) If decoding successfully, update the information part reliabilities; else, remain the information part of $\mathbb{C}_{\mathrm{II}}^{v=1}$ codeword with no changes.
9) If the iterative maximum number has been reached, go to 10); else go to 5).

10) Perform a $\mathbb{C}_{\mathrm{II}}^{v=1}$ global decoding.

11) Output the message $\tilde{m}$.

## 5.6 Performance

We evaluate the performance of the new LDPC+PSSRS concatenation system for both sector length and 4KB sector formats. For each evaluation, the conventional LDPC+RS concatenation system is optimized and used as the baseline curve. For a fair comparison, two systems must have same overall lengths and code rates. We fix the code rates of all LDPC and RS codes to be very close to 0.9. The new LDPC+PSSRS concatenation system is optimized with the same setup as the conventional LDPC+RS concatenation system. The global and local error correction capabilities of the PSSRS code are also optimized.

### 5.6.1 Sector length code

To get an optimal iterative scheme for the sector length system, the turbo equalization has been adopted, which is inspired by the work in [77]. Selecting the optimal parameters of global (turbo) iterations and inner (BP) iterations is based on the simulation results. The optimization process is based on the conventional LDPC+RS concatenation system with the designed LDPC code encoded with the progressive edge-growth (PEG) algorithm in [78]. The simulation is evaluated on a

PMRC with a noise mixture of 90% jitter noise and 10% electric noise. From the left part of Fig. 5.8, it can be found that under various turbo iterations, the performance begins to be stable for eight inner iterations. From the right part of Fig. 5.8, the performance stabilizes for six turbo iterations. So we choose the optimal iteration scheme for the conventional LDPC+RS concatenation system as: six turbo iterations and eight inner iterations.

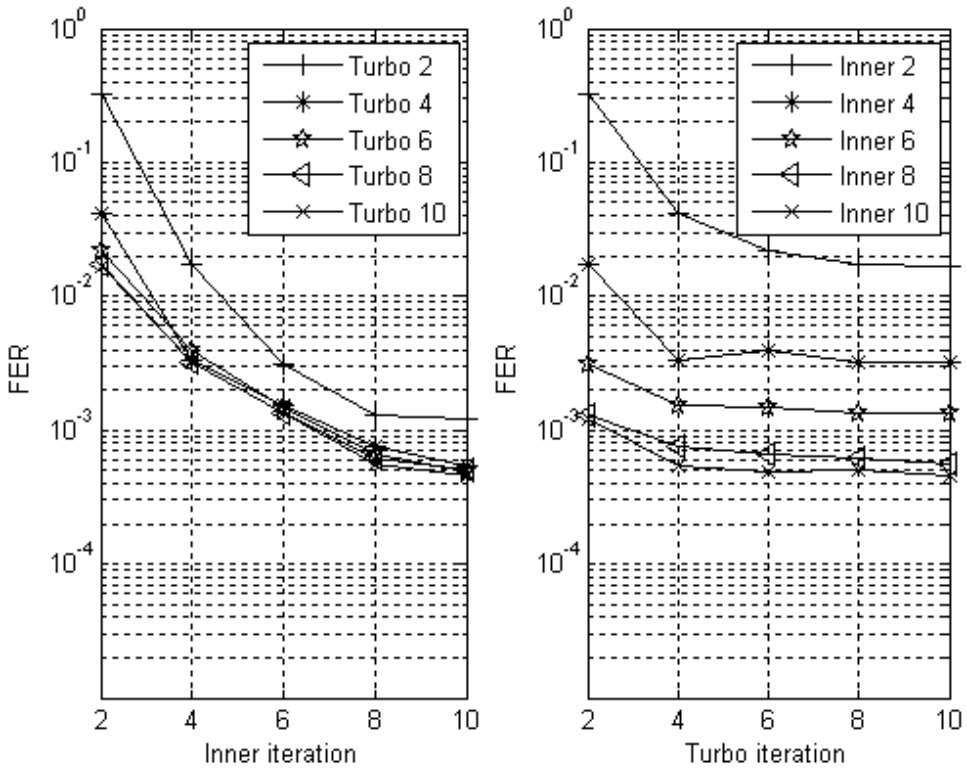


Fig. 5.8. Optimization of number of turbo iterations and inner BP iterations.

We use an optimal conventional LDPC+RS concatenation system as the baseline with overall rate around 0.81 with the LDPC code rate about 0.9 and the RS code rate about 0.9. We finalize the conventional LDPC+RS concatenation system, whose overall rate is $410/505 \approx 0.818$ with an $RS(456,410)$ code and a binary $LDPC(5050,4560)$ code. With simulations, we tested several different overall rates around 0.81 as shown in Fig. 5.9 for the new LDPC+PSSRS concatenation system. It shows that a binary $LDPC(4550,4100) + \mathbb{C}_{\mathrm{II}}^{\nu=1}(505,455)$ is the best. Here the shortened $\mathbb{C}_{\mathrm{II}}^{\nu=1}(505,455)$ code is over $GF(2^{10})$.

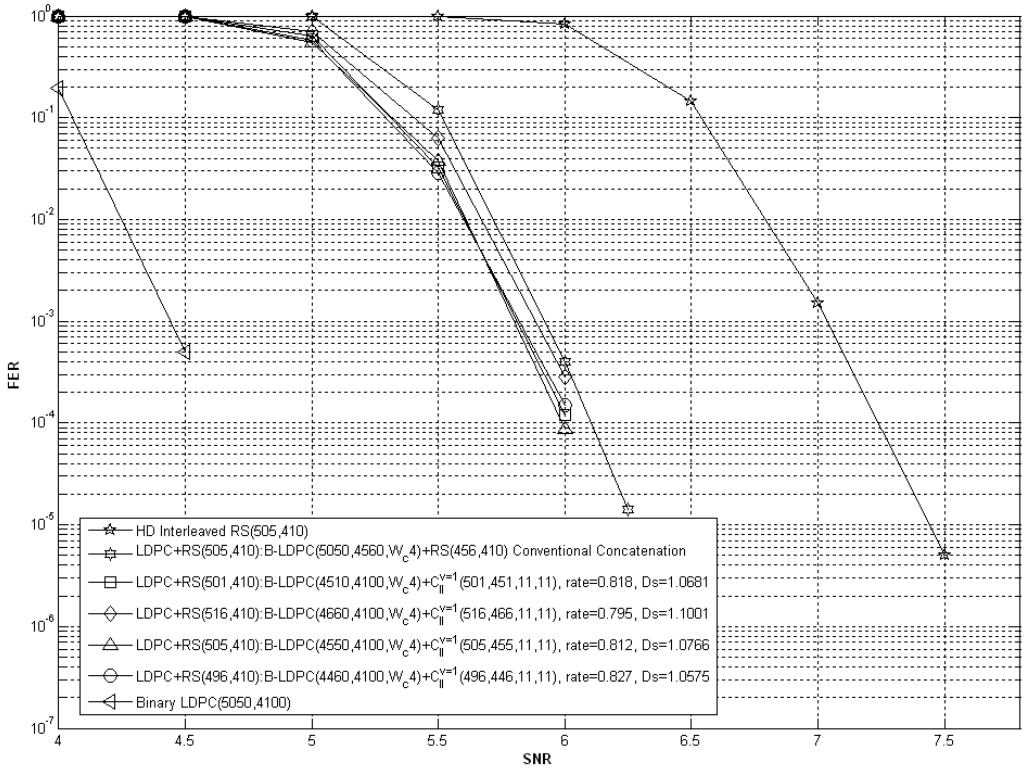


Fig. 5.9. Optimization of the overall rate of the concatenation system.

Further, the simulations in Fig. 5.9 show that simply replacing the RS code with a $\mathbb{C}_{\mathrm{II}}^{\nu=1}$ code can only get a marginal gain over the conventional LDPC+RS concatenation system, due to the fact that the local error correction capability is small, which cannot correct the errors from output of the inner LDPC code.

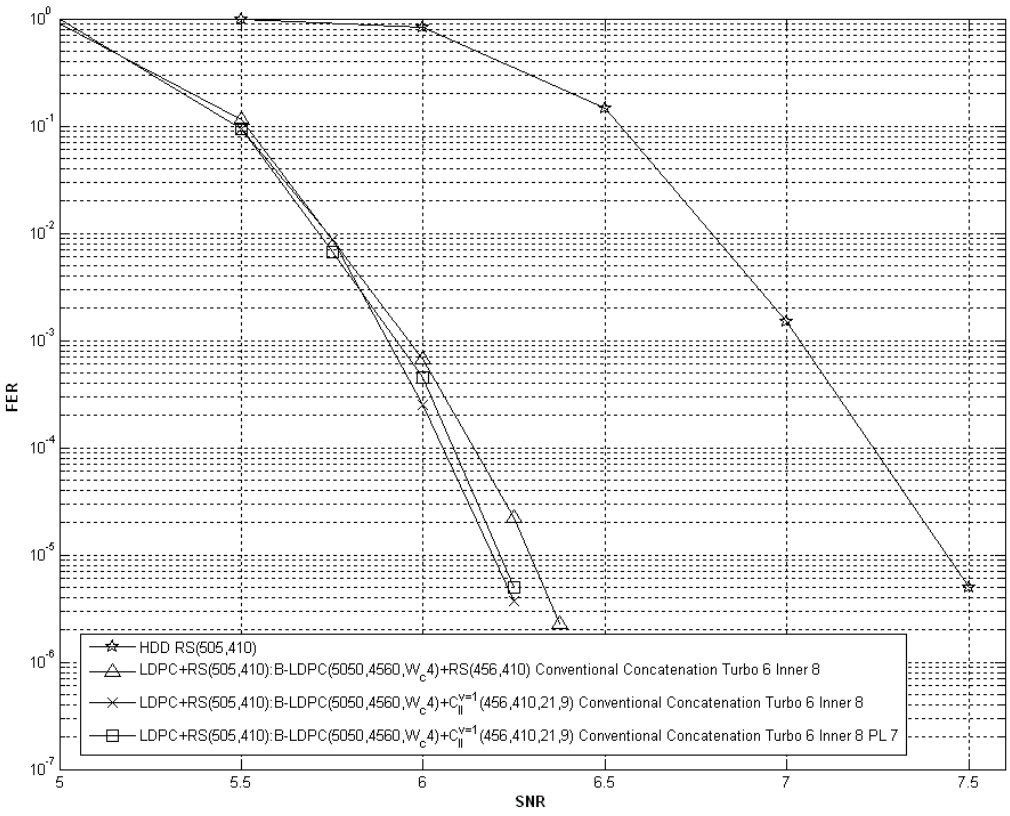


Fig. 5.10. Comparison to the conventional concatenation system.

The local error correction capability is very important for the iterative parallel local decoding, because a small error correction capability has a smaller possibility

to destroy the trapping sets. Meanwhile, it might create some additional errors, which makes iterative parallel local decoding ineffective.

In Fig. 5.10, even after the iterative parallel local decoding has been adopted to decode the LDPC codeword, the performance does not improve. There is no advantage to simply replace the RS codes with the $\mathbb{C}_{\mathrm{II}}^{v=1}$ code in the conventional concatenation system. How about the new LDPC+$\mathbb{C}_{\mathrm{II}}^{v=1}$ concatenation system shown in Fig. 5.4? In the new concatenation system, we implement the $\mathbb{C}_{\mathrm{II}}^{v=1}$ code as the pseudo-inner code from the channel point of view. But from a decoding point of view, we actually decode the LDPC codeword first, since the $\mathbb{C}_{\mathrm{II}}^{v=1}$ encoder is systematic.

We use a binary $LDPC(4550,4100)$ and a $\mathbb{C}_{\mathrm{II}}^{v=1}(505,455)$ code for this new concatenation system. The number of parallel local iterations is seven. The $\mathbb{C}_{\mathrm{II}}^{v=1}(505,455)$ is a $C(505,455,11,11)$ code, whose global and local error correction capabilities are both five. Because it is over $GF(2^{10})$, there are ten parallel local structures. For each parallel local iteration, the $\mathbb{C}_{\mathrm{II}}^{v=1}(505,455)$ code can correct some partial errors for the LDPC code, if at least one local structure has less than or equal to five local errors occurring. The performance shows that implementing the $\mathbb{C}_{\mathrm{II}}^{v=1}$ code as the pseudo-inner code has a better performance. At the FER of $10^{-5}$, the gain over the conventional concatenation system is about 0.45 dB. It is obvious that a higher local error correction capability plays a very important role in

destroying the trapping sets, which turns out to have a sharper waterfall. A soft-decision RS decoding algorithm can make the gain even larger with a price of increased complexity.

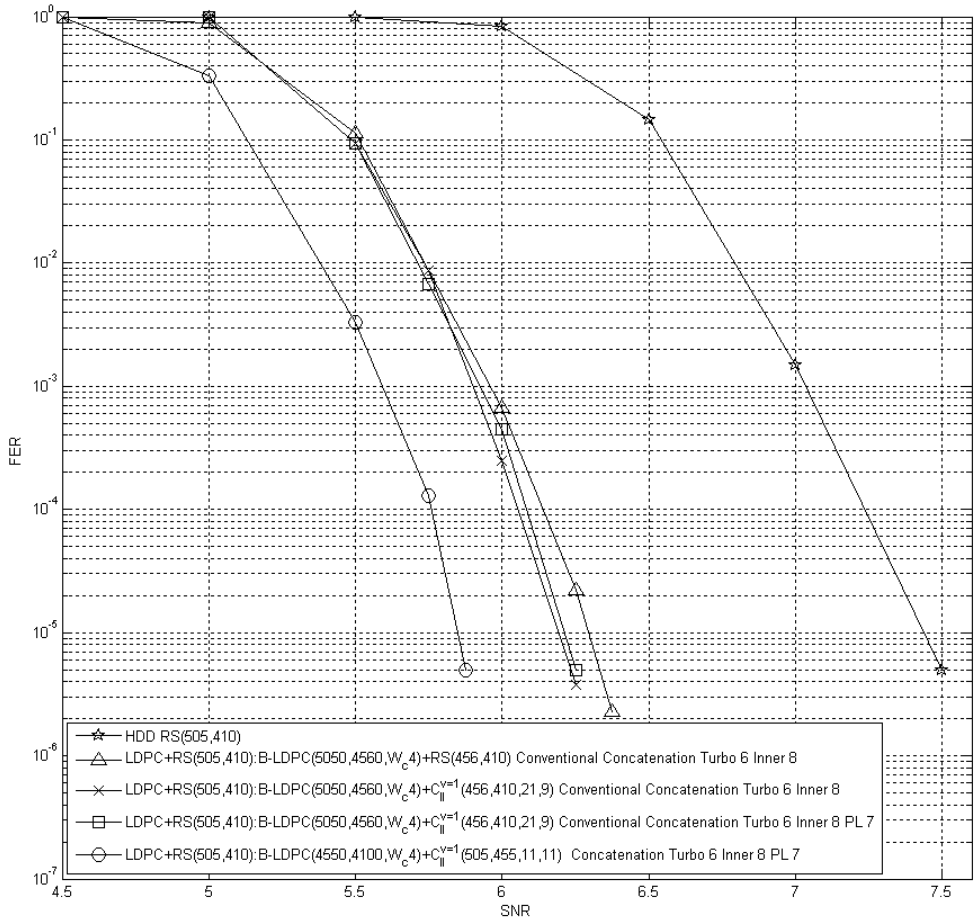


Fig. 5.11. Performance of iterative parallel local decoding algorithm for LDPC+RS system on a PMRC equalized to an optimal GPR4 target with 90% jitter noise. The user density is 1.0757.

### 5.6.2 Long (4KB) sectors

The computational complexity for 4KB sectors and a code rate 0.81 binary LDPC code (B-LDPC) is beyond our computational capability. Interested reader can refer to [79] for a non-binary LDPC and B-LDPC of rate 0.9 for the 4KB sector. In this simulation, we encode this 4KB information messages into twelve parallel $B-LDPC(3034,2731)$, which have a total of $2731*12/8=4096.5$ bytes. Then encoded these twelve parallel binary $LDPC(3034,2731)$ with a systematic $\mathbb{C}_{\mathrm{II}}^{v=1}$ encoder: $C(3340,3034,57,57)$ over $GF(2^{12})$ . Thus we have a $LDPC+\mathbb{C}_{II}^{v=1}(3340,2731)$ code with an overall rate $2731/3340=0.818$. For a relatively fair comparison, we compared the performance to the one of the eight consecutive $LDPC(5010,4096)$ codes. A frame error is defined as any decoding failure in these eight consecutive $LDPC(5010,4096)$ codewords. An interleaver is implemented here for these twelve parallel $LDPC(3034,2731)$ codes. The major function of this interleaver is to distribute the errors relatively uniformly at low SNRs, and to partition every trapping set into $m$-parallel local structures at high SNRs. Because at high SNRs, if the system does not have an interleaver, it might cause the situation where most local structure do not have "enough" errors, and one might have some dominant trapping sets. It will fail to decode this local structure and cause a global decoding failure as the conventional concatenation system, whose output of the LDPC decoder has errors beyond the error correction capability of the outer RS

decoder. With the interleaver, it can distribute these errors from these dominant trapping sets to $m$-parallel local structures, which actually break these trapping sets into several pieces. So it makes full use of the error correction capabilities of every local structure and is likely to destroy the trapping set.

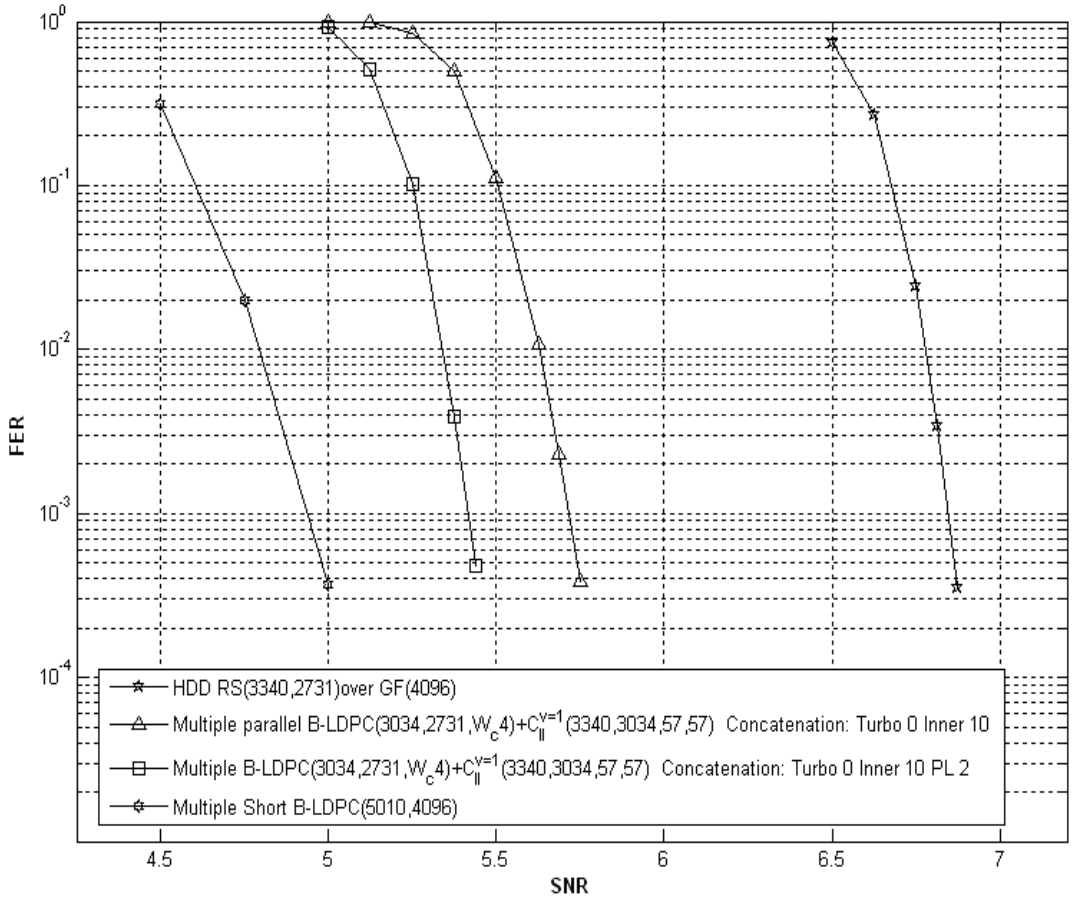


Fig. 5.12. Performance of 4K-Byte sector length code of $LDPC + RS(3340, 2731)$ over $GF(2^{12})$ on a PMRC equalized to an optimal GPR4 target with 90% jitter noise. User density is 1.0757.

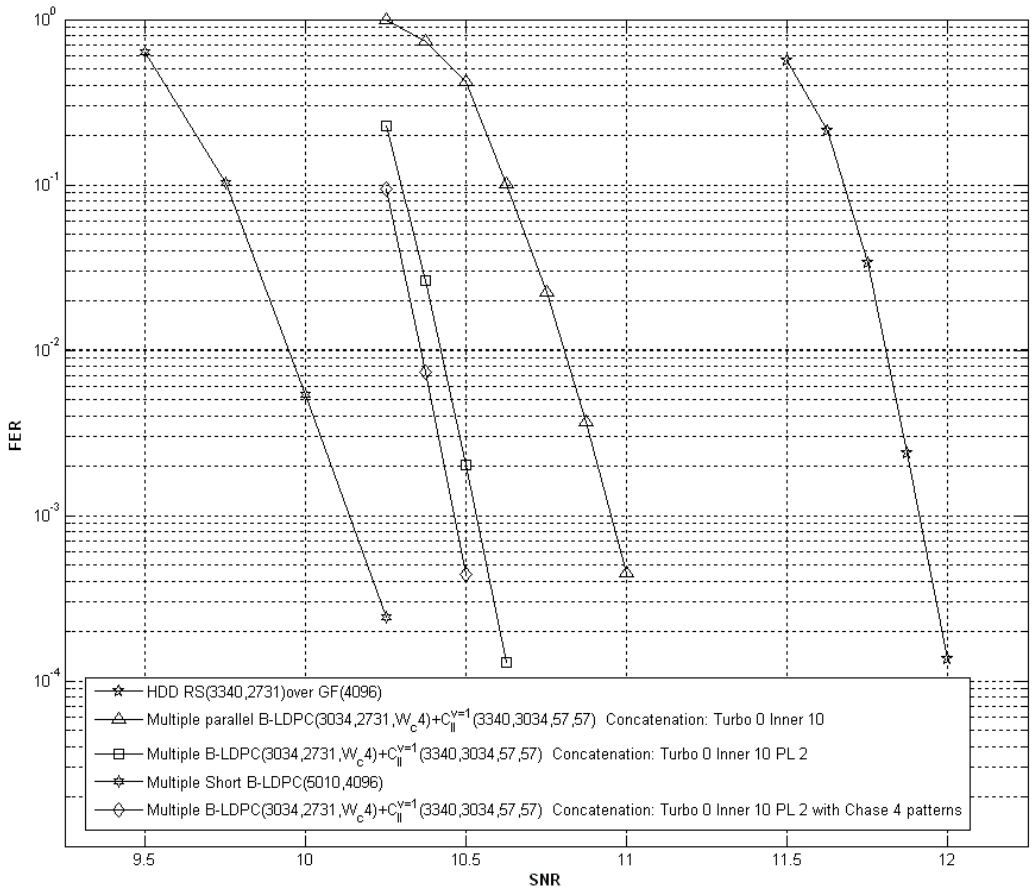


Fig. 5.13. Performance of 4KB sector length code of $LDPC + RS(3340, 2731)$ over $GF(2^{12})$ on a PMRC equalized to an optimal GPR4 target with 90% jitter noise. User density is 1.5073.

Due to the computational complexity, we did not use the turbo iteration scheme and we used only two parallel local iterations. We simulated with two different densities for PMRCs with 90% jitter noise. The performance results show that with the increase of the density, the LDPC only curve is getting closer to the iterative

parallel decoding curve. At user density 1.5073, the gain is less than 0.25dB at an FER of $10^{-4}$. Meanwhile, the LDPC only curve has a slight tendency to have an error floor. This provides us an alternative solution that the LDPC only system might be outperformed by this iterative parallel local decoding algorithm in a PMRC at a very high density, which has a sharper waterfall region and much lower error floor. The parallel structure can enable extremely long codes with length longer than 4KB system in both theory and practice.

## 5.7 Summary and recommendations

In this chapter, we propose a new LDPC+PSSRS concatenation system and compare it with the conventional LDPC+RS system. An iterative parallel local decoding algorithm for this new concatenation system is proposed to get a sharper waterfall region with the idea of partially destroying the most dominant "trapping sets". The simulation of codes of sector length of 4KB sectors have shown about 0.5 dB gain over the conventional LDPC+RS system. The PSSRS can provide the ability to help the LDPC code itself to eliminate the remaining errors instead of the RS code by providing a parallel local structure, which can partially destroy some parts of the most dominant "trapping sets", even when the number of the overall global system errors goes beyond the error-correction capability of the RS code of the same length and rate. This new concatenation system is specially suited for extremely long codes

for the next generation of drives, because the complexity for decoding the RS code of the same length and same rate as the PSSRS code is prohibitive when using a soft-decision decoding algorithm, but the PSSRS code has much smaller global and local error-correction capability, which introduces a moderate computational complexity with similar performance. Next we list several cases that need further study.

1) The reliability updating between symbol level and parallel local bit-level needs further research.
2) A better strategy to select $\mathbb{C}_{\mathrm{II}}$ or $\mathbb{C}_{\mathrm{I}}$ codes: current concatenation system is simply designed to have equal error protection no matter what the structure of the codeword. Actually, the structure of codeword will play an important role in decoding. $\mathbb{C}_{\mathrm{I}}$ codes need to be studied further, because they can provide unequal error protections.
3) For the LDPC+RS($\mathbb{C}_{\mathrm{II}}$) concatenation system, the current LDPC code structure design is independent of the RS($\mathbb{C}_{\mathrm{II}}$). In the future, the LDPC decoding algorithm can be modified to get RS ($\mathbb{C}_{\mathrm{II}}$) decoding involved dynamically. A better dynamic LDPC decoding strategy in BP decoding is required.
4) $\mathbb{C}_{\mathrm{II}}^{\langle v_1 \supset \cdots | v_2 \supset \cdots | \cdots | v_{max} \supset \cdots \rangle}$ can bring more dynamic local structures. How should it be used to help redistribute the errors and destroy most trapping sets efficiently?

5) To evaluate how the most dominant trapping sets can be handled with these new concatenation structure and how much lower the error floor will be?
6) Current LDPC code structure cannot be changed dynamically, which is unlike some $\mathbb{C}_{II}$ codes. How do we construct an LDPC code having a local structure or logically local structure from the decoding point of view, which can be used to cooperate with $\mathbb{C}_{II}$ codes?

# Chapter 6

# Conclusions

In this work, we proposed a new class of SSRS codes with a parallel structure, which we named PSSRS code. The PSSRS code introduced the possibility of local error correction capabilities and global error correction, which provides robustness to both the random errors and the burst errors. Compared to conventional RS codes of the same length and dimension, the PSSRS codes have some major advantages: 1. They provide local robustness. 2. They have a parallel SSRS structure, which allows for parallel decoding. 3. Their error correction capabilities are beyond the half minimum distance of conventional RS codes. 4. The global symbol error correction capability and local error correction capability are both much smaller than the one for the conventional RS codes, thus the PSSRS code can provide a larger gain with the same decoding computational complexity, which is especially suitable for the soft SDD algorithms. 5. For long codes, the PSSRS code affords a practical complexity to achieve a moderate gain. 6. The rotation structure of some class-II codes can introduce error redistributions.

The PSSRS code may offer a possible alternative solution for solving the problem in conventional LDPC+RS concatenated systems, when the number of errors in certain error patterns causing the error floor, is beyond the error correction capability of the conventional RS code. We proposed a new LDPC+PSSRS concatenated system with a new iterative parallel local decoding algorithm. We also developed a new decoding framework that views the major role of the RS code as removing troublesome error patterns. In this new LDPC+PSSRS system, the PSSRS

code can help locate and eliminate some trapping sets. With the proposed iterative parallel local decoding algorithm, the LDPC decoder can correct the remaining errors by itself. We evaluated this LDPC+PSSRS system with this iterative parallel local decoding algorithm with a sector long code for the next generation 4KB sectors on the PMRC under various densities. The performance showed that it can achieve about 0.5 $dB^{+}$ gain over the optimal conventional LDPC+RS system with RS hard-decision decoding algorithms, making its performance close to the multi-LDPC only system. It is highly recommended to use a low-complexity powerful SDD algorithm to get even better performance.

We also proposed a Chase-GMD type algorithm, which has about 0.6 dB gain over standard hard-decision decoding algorithms on PMRCs, and compares favorably with recently proposed implementations of Chase-type algorithms. For similar complexity at high SNR, our algorithm exhibits a 0.4 $dB^{+}$ gain over other low complexity implementations.

This new LDPC+PSSRS concatenation system using the iterative parallel local decoding algorithm with our Chase-GMD algorithm on both L2 and L3 is particularly attractive for PMRCs which operate at relatively high SNRs.

The future research directions are 1) PSSRS based LDPC codes. 2) Parallel bit and symbol level partial local decoding using the adaptive BP decoding algorithms based on $\mathbb{C}_{\mathrm{II}}^{\langle v_1|v_2|\cdots|v_{max}\rangle}$ and $\mathbb{C}_{\mathrm{II}}^{\langle v_1\supset\cdots|v_2\supset\cdots|\cdots|v_{max}\supset\cdots\rangle}$ codes.

# BIBLIOGRAPHY

[1] S. B. Wicker, *Error Control Systems for Digital Communication and Storage*. Upper Saddle River, NJ: Prentice Hall, 1995.

[2] F. J. McWilliams and N. J. A. Sloane, *The Theory of Error Correcting Codes*. 2nd ed. New York, NY: North-Holland, 1978.

[3] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *J. Soc. Indust. and Appl. Math.*, vol. 8, pp. 300-304, 1950.

[4] C. Berrrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo codes," in *Proc. IEEE Int. Conf. Commun.*, Geneva, Switzerland, 1993, pp. 1064-1070.

[5] C. Berrrou and A. Glavieux, "Near optimum error correcting coding and decoding: turbo-codes," *IEEE Trans. Commun.*, vol. 44, no. 10, pp. 1261-1271, Oct. 1996.

[6] R. G. Gallager, *Low-Density Parity-Check Codes*, Cambridge, MA: MIT Press, 1963.

[7] R. G. Gallager, "Low density parity-check codes," *IRE Trans. Inf. Theory*, vol. 8, no. 1, pp. 21-28, Jan. 1962.

[8] M. Hassner and E. Grochowski. (2009, Nov. 3). *4K Byte-Sector HDD-Data Format Standard*. [Online]. Available: http://www.idema.org/.

[9] P. Chicoine, M. Hassner, E. Grochowski, S. Jenness, M. Noblitt, G. Silvus, C. Stevens, and B. Weber. (2007, Apr. 20). *Hard Disk Drive Long Data Sector White Paper*. [Online]. Available: http://www.idema.org/.

[10] K. Fung. *Large Blocks for Reliability*. [Online]. Available: http://www.idema.org/.

[11] M. Hassner. *4K-Block Format Efficiency*. [Online]. Available: http://www.idema.org/.

[12] C. Cole, S.G. Wilson, E.K. Hall, T.R. Giallorenzi, "A general method for finding low error rates of LDPC codes," eprint arXiv:cs/0505051, 05/2005.

[13] T. Richardson, "Error floors of LDPC codes," in *Proc. 41st Allerton Conf. Commun., Control, and Computing*, Monticello, IL, 2003, pp.1426-1435.

[14] X. Hu, B.V.K. Vijaya Kumar, Z. Li, and R. Barndt, "Error floor estimation of long LDPC codes on partial response channels," in *Proc. IEEE Global Commun. Conf.*, Washington, DC, 2007, pp. 259-264.

[15] Z. Zhang, L. Dolecek, M. Wainwright, V. Anantharam, B. Nikolic, "Quantization effects in low-density parity-check decoders," in *Proc. IEEE Int. Conf. Commun.*, Glasgow, UK, Jun. 2007, pp. 6231-6237.

[16] D. MacKay and M. Postol, "Weaknesses of Margulis and Ramanujan-Margulis low-density parity check codes," *Electron. Notes in Theoretical Comput. Sci.*, vol. 74, 2003.

[17] S. Laendner and O. Milenkovic, "Algorithmic and combinatorial analysis of trapping sets in structured LDPC codes," in *Proc. IEEE Int. Conf. Wireless Commun. and Mobile Comput.*, Maui, HI, 2005, pp. 630-635.

[18] X. Hu, M.P.C. Fossorier, and E. Eleftheriou, "On the computation of the minimum distance of low-density parity-check codes," in *Proc. IEEE Int. Conf. Commun.*, Paris, France, 2004, pp. 757-771.

[19] R. J. McEliece and G. Solomon, " Trace-Shortened Reed-Solomon Codes." JPL NASA, Pasadena, CA, REP. 42-117, 1994.

[20] M. Hattori, "Subspace subcodes of Reed-Solomon codes," Ph.D. dissertation, Dept. Elect. Eng., Calif. Inst. Technol., Pasadena, 1995.

[21] M. Hattori, R. J. McEliece, and G. Solomon, "Subspace subcodes of Reed-Solomon codes," *IEEE Trans. Inf. Theory*, vol. 44, no. 5, pp. 1851-1880, Sept. 1998.

[22] V. Guruswami and M. Sudan, "Improved decoding of Reed-Solomon and algebraic-geometry codes," *IEEE Trans. Inf. Theory*, vol. 45, no. 6, pp. 1757-1767, Sept. 1999.

[23] R. Koetter and A. Vardy, "Algebraic soft-decision decoding of Reed-Solomon codes," *IEEE Trans. Inf. Theory*, vol. 49, no. 11, pp. 2809-2825, Nov. 2003.

[24] R. T. Chien, "Cyclic decoding procedures for Bose-Chaudhuri-Hocquenghem codes," *IEEE Trans. Inf. Theory*, vol. 10, no. 4, pp. 357-363, Oct. 1964.

[25] G. D. Forney, "On decoding BCH codes," *IEEE Trans. Inf. Theory*, vol. 11, no. 4, pp. 549-557, Oct. 1965.

[26] J. L. Massey, "Shift-register synthesis and BCH decoding," *IEEE Trans. Inf. Theory*, vol. 15, no. 1, pp. 122-127, Oct. 1969.

[27] R. E. Blahu, *Theory and Practice of Error Control Codes*, Reading, MA: Addison-Wesley, 1983.

[28] L. Welch and E. R. Berlekamp, "Error Correction for Algebraic Block Codes," U.S. Patent 4 533 470, Sep. 27, 1983.

[29] P. Gemmell and M. Sudan, "Highly resilient correctors for polynomials," *Inf. Process. Lett.*, vol. 43, pp. 169-174, 1992.

[30] S. V. Fedorenko, "A simple algorithm for decoding Reed-Solomon codes and its relation to the Welch-Berlekamp algorithm," *IEEE Trans. Inf. Theory*, vol. 51, no. 3, pp. 1196-1198, Mar. 2005.

[31] G. D. Forney, Jr., "Generalized minimum distance decoding," *IEEE Trans. Inf. Theory*, vol. 12, no. 2, pp. 125-131, Apr. 1966.

[32] D. Chase, "A Class of algorithms for decoding block codes with channel measurement information," *IEEE Trans. Inf. Theory*, vol. 18, no. 1, pp. 170-182, Jan. 1972.

[33] R. M. Roth and G. Ruckenstein, "Efficient decoding of Reed-Solomon codes beyond half the minimum distance," *IEEE Trans. Inf. Theory*, vol. 46, no, 1, pp. 246-257, Jan. 2000.

[34] D. J. C. Mackay and R. M. Neal, "Near Shannon limit performance of low density parity check codes," *IEEE Electron. Lett.*, vol. 32, pp. 1545, Aug. 1995.

[35] J. Jiang and K. R. Narayanan, "Iterative soft-input soft-output decoding of Reed-Solomon codes by adapting the parity-check matrix," *IEEE Trans. Inf. Theory*, vol. 52, no. 8, pp. 3746-3756, Aug. 2006.

[36] C. Zhong, "Effcient soft-decision decoding of Reed-Solomon codes," Ph.D. dissertation, Dept. Elect. Eng., Univ. of Oklahoma, Norman, 2008.

[37] S. Lee and B. V. K. V. Kumar, "Application of Soft-Decision Decoders to Non Narrow-Sense Reed-Solomon Codes," in *Proc. IEEE Int. Conf. Commun.*, Glasgow, Scotland, 2007, pp. 5225-5230.

[38] J. Bellorado and A. Kavcic, "A low-complexity method for Chase-type decoding of RS codes," in *Proc. IEEE Int. Symp. Inf. Theory*, Seattle, WA, 2006, pp. 2037-2041.

[39] R. R. Nielsen and T. Hoholdt, "Decoding RS codes beyond half the minimum distance," in Coding Theory, Cryptography and Related Areas, J. Buchmann, T. Hoholdt, T. Stichtecnoth, and H. Tapia-Recillas, Eds., 2000, pp. 221-235.

[40] H. Xia, C. Zhong, and J. R. Cruz, "Chase-type algorithm for soft-decision RS decoding on Rayleigh fading channels," in *Proc. IEEE Global Telecommun. Conf.*, San Francisco, CA, 2003, pp. 1751-1755.

[41] W. J. Gross, F. R. Kschichang, R. Koetter, and P. G. Gulak, "Towards a VLSI architecture for interpolation-based soft-decision RS decoders," *J. VSLI Signal Process.*, vol. 39, pp. 93-111, Jan. 2005.

[42] H. Xia and J. R. Cruz, "Reliability-based forward recursive algorithms for algebraic soft-decision decoding of RS codes," *IEEE Trans. Commun.*, vol. 55, no. 7, pp. 1273-1278, July 2007.

[43] W. J. Gross, F. R. Kschischang, R. Koetter, and P. G. Gulak, "Simulation results for algebraic soft-decision decoding of RS codes," in *Proc. 21st Biennial Symp. Commun.*, Kingston, Ontario, Canada, 2002, pp. 356-360.

[44] F. Parvaresh, and A. Vardy, "Mulitplicity assignments for algebraic soft-decoding of RS codes," in *Proc. IEEE Int. Symp. Inf. Theory*, Yokohama, Japan, 2003, pp. 205.

[45] D. J. C. MacKay and R. M. Neal, "Near Shannon limit performance of low density parity check codes," *IEEE Electron. Lett.*, vol. 32, pp. 1645, Aug. 1996.

[46] T.-H. Hu and S. Lin, "An efficient hybrid decoding algorithm for RS codes based on bit reliability," *IEEE Trans. Commun.*, vol. 51, no. 7, pp. 1073-1081, July 2003.

[47] W. Jin and M. Fossorier, "Efficient box and match algorithm for reliability-based soft-decision decoding of linear block codes," in *Proc. Inf. Theory Appl. Workshop*, La Jolla, CA, 2007, pp. 160-169.

[48] W. Jin and M. Fossorier, "Probabilistic sufficient conditions on optimality for reliability based decoding of linear block codes," in *Proc. IEEE Int. Symp. Inf. Theory*, Seattle, WA, 2006, pp. 2235-2239.

[49] H. Sawaguchi, Y. Nishida, H. Takano, and H. Aoi, "Performance analysis of modified PRML channels for perpendicular recording systems," *J. Magnetism Magn. Materials*, vol. 235, pp. 265-272, Oct. 2001.

[50] G. Solomon, "Non-linear, non-binary cyclic group codes." *JPL TDA Progress Report*, vol. 43, pp. 84-95, Feb. 1992.

[51] G. Solomon, "Non-linear, non-binary cyclic group codes." in *Proc. IEEE Int. Symp. Inf. Theory*, San Antonio, Texas, 1993, pp. 192.

[52] R. J. McEliece and G. Solomon, " Trace-shortened Reed-Solomon codes." *JPL TDA Progress Report*, vol. 42, pp. 119-128, Feb. 1994.

[53] M. Van Dijk, L. Tolhuizen, "Efficient encoding for a class of subspace subclodes," *IEEE Trans. Inf. Theory*, vol. 45, no. 6, pp. 2142-2145, Sept. 1999.

[54] C. Le Dantec and P. Piret, "An encoder to match Reed-Solomon codes over GF(q) to subalphabet of GF(q)," *IEEE Trans. Inf. Theory*, vol. 45, no. 5, pp. 1697-1701, July. 1999.

[55] A. Vardy and Y. Be'ery, "Bit level soft-decision decoding of Reed-Solomon codes," *IEEE Trans. Commun.*, vol. 39, no. 3, pp. 440-444, Mar. 1991

[56] Sergei V. Fedorenko, "The Star trellis decoding of Reed-Solomon codes," eprint arXiv:0707.1025v1, 07/2007.

[57] T. R. Halford, V. Ponnampalam, A. J. Grant, and K. M. Chugg, "Soft-In Soft-Out decoding of Reed-Solomon codes based on Vardy and Be'ery decompostion," *IEEE Trans. Inf. Theory*, vol. 51, no. 12, pp. 4353-4358, Dec. 2005.

[58] J. Wolf, "Efficient maximum likelihood decoding of linear block codes using a trellis," *IEEE Trans. Inf. Theory*, vol. 24, no. 1, pp. 75-80, Jan. 1978.

[59] A. Thangaraj and S. J Raj, “Reed-Solomon Subcodes with Nontrivial Traces: Distance Properties and Soft-Decision Decoding,” eprint arXiv:0810.0567v1, 10/2008.

[60] A. Ashikhmin and S. Litsyn, “Simple MAP decoding of first-order Reed-Muller and Hamming codes,” *IEEE Trans. Inf. Theory*, vol. 50, no. 8, pp. 1812-1818, Aug. 2004.

[61] J. Jiang and K. R. Narayanan, “Algebraic soft-decision decoding of Reed-Solomon codes using bit-level soft information,” *IEEE Trans. Inf. Theory*, vol. 54, no. 9, pp. 3907-3928, Sept. 2008.

[62] P. Lee *et al.*, “Error floors in LDPC codes: Fast simulation, bounds and hardware emulation,” in *Proc. IEEE Int. Symp. Inf. Theory*, Toronto, Ontario, Canada, 2008, pp. 444-448.

[63] C. Cole, S.G. Wilson, E.K. Hall, T.R. Giallorenzi, “A general method for finding low error rates of LDPC codes,” eprint arXiv:cs/0505051, 05/2005.

[64] L. J. Deutsch, "The effects of Reed-Solomon code shortening on the performance of coded telemetry systems." *JPL TDA Progress Report 42-75*, July-Sept. 1983.

[65] DVB-C2 Bluebook A138, pp. 16. [Online]. http://www.dvb.org.

[66] S. Sankaranyanan, A. Kuznetsov, and D. Sridhara, "On the concatenation of LDPC and RS codes in magnetic recording systems," in *Proc. IEEE Global Commun. Conf. Workshops*, Washington, DC, 2007, pp. 1-1.

[67] S.-Y. Chung, G. D. Forney Jr. T. J. Richardson, and R. Urbanke, "On the design of low-density parity-check codes within 0.0045dB of the Shannon limit," *IEEE Commun. Lett.*, vol. 5, pp. 58-60, Feb. 2001.

[68] T. J. Richardon, M. A. Shokrollahi, and R. L. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 619-637, Feb. 2001.

[69] E. M. Kurtas, A. V. Kuznetsov, and I. Djurdjevic, " System perspectives for the application of structured LDPC codes to data storage devices," *IEEE Trans. Magn.*, vol. 42, no. 2, pp. 200-207, Feb. 2006.

[70] W. Tan, "Design of inner LDPC codes for magnetic recording channel," *IEEE Trans. Magn.*, vol. 44, no. 1, pp. 217-222, Jan. 2008.

[71] S. Jeon, X. Hu, and B. V. K. V. Kumar, "Evalution of the concatenation of LDPC and RS codes in magnetic recording channel using field programmable gate arrays," in *Proc. IEEE Global Telecommun. Conf.*, New Orleans, LA, 2008, pp. 1-5.

[72] Y. Han, W. E. Ryan, "LDPC decoder strategies for achieving low error floors," in *Proc. Inf. Theory and Applica. Workshop*, San Diego, CA, 2008, pp. 277-286.

[73] Z. Zhang, L. Dolecek, B. Nikolic, V. Ananatharam, and M. J. Wainwright, "Lowering LDPC error floors by postprocessing," in *Proc. IEEE Global Telecommun. Conf.*, New Orleans, LA, 2008, pp. 1-6.

[74] L. Dolecek, Z. Zhang, V. Anantharam, M. Wainwright, and B. Nikolic, "Analysis of absorbing sets for array-based LDPC codes," in *Proc. IEEE Int. Conf. on Commun.*, Glasgow, Scotland, 2007, pp. 6261-6268.

[75] B. J. Frey, R. Kotter, and A. Vardy, "Skewness and pseudocodewords in iterative decoding," in *Proc. IEEE Int. Symp. Inf. Theory*, Cambridge, MA, 1998, pp. 148.

[76] Y. Zhang, and W. Ryan, "Toward low LDPC-code floors: a case study," *IEEE Trans. Commun.*, vol. 57, no. 6, pp. 1566-1573, Jun. 2009.

[77] W. Chang, and J. R. Cruz, "RS plus LDPC codes for perpendicular magnetic recording," submitted for publication.

[78] X.-Y. Hu, E. Eleftheriou, and D. M. Arnold, "Regular and irregular progressive edge-growth Tanner graphs," *IEEE Trans, Inf. Theory*, vol. 51, no. 1, pp. 386-398, Jan. 2005.

[79] W. Chang and J. R. Cruz, "Nonbinary LDPC codes for 4-kB sectors," *IEEE Trans. Magn.*, vol. 44, no. 11, pp. 3781-3784, Nov. 2008.

# APPENDIX A

## Proof of *Theorem 4.2-5*.

*Theorem 4.2*: A codeword of $\mathbb{C}$ in Fig. 4.3 is still an RS codeword.

Proof: Each $\mathbb{C}_{\mathrm{i}}, i = 1, \cdots, \sigma$ is an SSRS code of a different parent RS code over $GF(2^m)$. Thus, every $\mathbb{C}_{\mathrm{i}}$ has the same error correction capability as its parent RS code. We denote the corresponding error correction capability as $t_{\mathrm{i}}, i = 1, \cdots, \sigma$. Sort $t_{\mathrm{i}}$, and get $t_{\mathrm{i}_1} \leq t_{\mathrm{i}_2} \leq \cdots \leq t_{\mathrm{i}_\sigma}$. From theorem 4.1, we know that every $\mathbb{C}_{\mathrm{i}}$ is the subcode of $\mathbb{C}_{\mathrm{i}_1}$. Every codeword of $\mathbb{C}_{\mathrm{i}}$ is a codeword of a RS code with error correction capability $t_{\mathrm{i}_1}$. Thus the linear combination of arbitrary codewords $c_{\mathrm{i}}$ over $GF(2^m)$ is still a codeword $c$ of the RS code with error correction capability $t_{\mathrm{i}_1}$.

$$c = \sum_{i=1}^{\sigma} \alpha^{\left(\sum_{j=1}^{i} v_j\right) - v_i} c_i = \sum_{i=1}^{\sigma} \beta_i c_i \,, \beta_i \in GF(2^m)$$

So $c$ is a codeword of the RS code with error correction capability $t_{\mathrm{i}_1}$. Thus, $\mathbb{C}$ is a subcode of the RS code with error correction capability $t_{\mathrm{i}_1}$. ■

*Theorem 4.3:* The $\mathbb{C}_{\mathrm{II}}^{v=1}$ code generated by $g_{\mathbb{C}_{\mathrm{II}}^{v=1}}(x)$ is a $C(n, k, D, d)$ with $D = 2t_G + 1$ and $d = 2t_L + 1$.

Proof: Given the message polynomial $m(x) = m_0 + m_1 x + m_2 x^2 + \cdots + m_{k-1} x^{k-1}$, with $k = n - |\, z_{\mathbb{C}_{II}^{v=1}}|$. The codeword polynomial is

$$c(x) = m(x) g_{\mathbb{C}_{\mathrm{II}}^{v=1}}(x).$$

Denote $C_i^q$ as the cyclotomic coset of $i$ module $n$ with respect to $GF(q)$. Here $q = 2$.

$$J = \{Minimum\ element\ of\ each\ C_i^q\ without\ repetition, i = 1,2,\cdots,2t_L\},$$

$$c(x) = g_{\mathbb{C}_{\mathrm{II}}^{v=1}}(x) m(x) = \left[\prod_{j\in J}\prod_{i\in C_j^q}(x+\alpha^i)\right]\left[\prod_{l\notin \mathbb{C}\ and\ l\in z_{\mathbb{C}_{II}^{v=1}}}(x+\alpha^l)\right] m(x).$$

From [23, pp. 56, theorem 3-4], we know that if $p(x)$ is the minimal polynomial of $\alpha \in GF(q^m)$ over $GF(q)[x]$, $p(x)$ has roots which are exactly the conjugates of $\alpha$ with respect to $GF(q)$.

$$\left[\prod_{j\in J}\prod_{i\in C_j^q}(x+\alpha^i)\right] = \mathrm{LCM}(\text{minimum polynomials of } J \text{ over } GF(q)\ ).$$

So it is the generator polynomial $g_{t_L}^{binBCH}(x)$ of the primitive binary BCH code over $GF(q)$ with error-correction capability $t_L$.

$$g_{t_L}^{binBCH}(x) = \prod_{j\in J}\prod_{i\in C_j^q}(x+\alpha^i),$$

$$c(x) = g_{\mathbb{C}_{\mathrm{II}}^{v=1}}(x)m(x) = g_{t_L}^{binBCH}(x)\left[\prod_{l\notin \mathbb{C}\ and\ \ l\in z\ \mathbb{C}_{II}^{v=1}} (x+\alpha^l)\right]m(x)$$

$$= g_{t_L}^{binBCH}(x)M(x),$$

$$M(x) = \left[\prod_{l\notin \mathbb{C}\ and\ \ l\in z\ \mathbb{C}_{II}^{v=1}} (x+\alpha^l)\right]m(x).$$

Let $\kappa = degree(M(x))$,

$$M(x) = M_0 + M_1 x + \cdots + M_{\kappa-1}x^{\kappa-1},$$

$$c(x) = g_{t_L}^{binBCH}(x)M(x) = g_{t_L}^{binBCH}(x)\cdot(M_0 + M_1 x + \cdots + M_{\kappa-1}x^{\kappa-1}),$$

it assumes the basis for $GF(q^m)$ is $\mathcal{B} = \{\beta_0, \beta_1, \cdots, \beta_{m-1}\} = \{1, \alpha, \cdots, \alpha^{m-1}\}$, then

$$M_i = M_i^{(0)}\beta_0 + M_i^{(1)}\beta_1 + \cdots + M_i^{(m-1)}\beta_{m-1},$$

$$c(x) = g_{t_L}^{binBCH}(x)\left\{\sum_{i=0}^{\kappa-1}\left[M_i^{(0)}x^i\right]\beta_0 + \sum_{i=0}^{\kappa-1}\left[M_i^{(1)}x^i\right]\beta_1 + \cdots + \sum_{i=0}^{\kappa-1}\left[M_i^{(m-1)}x^i\right]\beta_{m-1}\right\}.$$

The $j^{th}$ tuple of the binary $m$-tuple of $c(x)$ is

$$g_{t_L}^{binBCH}(x)\sum_{i=0}^{\kappa-1}\left[M_i^{(j)}x^i\right].$$

It is a binary BCH codeword polynomial with error-correction capability $t_L$.

$$g(x) = \prod_{i=1}^{2t_G} (x + \alpha^i)$$

is a factor of

$$g_{\mathbb{C}_{\mathrm{II}}^{v=1}}(x) = \prod_{i \in z_{\mathbb{C}_{\mathrm{II}}^{v=1}}} (x + \alpha^i).$$

Thus $c(x)$ generated by $g_{\mathbb{C}_{\mathrm{II}}^{v=1}}(x)$ is also a codeword which can be generated by $g(x)$ with error-correction capability $t_G$ by theorem 4.1.

Concluding all above, the code generated by $g_{\mathbb{C}_{\mathrm{II}}^{v=1}}(x)$ is a $C(n, k, D, d)$ with $D = 2t_G + 1$ and $d = 2t_L + 1$. ■

*Theorem 4.4*: $t_L \leq t_G$ with $D = 2t_G + 1$ and $d = 2t_L + 1$.

Proof: The symbol level RS codeword with $t_G$ is the linear combination of the $m$-parallel primitive binary BCH codewords with $t_L$. Since the primitive binary BCH code is not maximum distance separable (MDS), the total binary dimension of $m$-parallel primitive binary BCH codewords is strictly less than $m(n - d + 1)$ [23, pp. 176, BCH bound]. Thus the maximum symbol dimension of symbol level RS code is less than $n - d + 1$, which comes from the fact that $n - d + 1 = [m(n - d + 1)]/m$. Because an RS code is a MDS code, the dimension is $n - D + 1$. That is $n - D + 1 \leq n - d + 1$, which leads to $t_L \leq t_G$. ■

*Theorem 4.5*: A $C(n, k, D, d)$ has a rotation structure, if $D = d$.

Proof: Given $D = d$, it means $t_G = t_L$, and

$$c(x) = g_{\mathbb{C}_{\mathrm{II}}^{v=1}}(x)m(x) = g_{t_L}^{binBCH}(x)\left[\prod_{l\notin \mathbb{C}\ and\ \ l\in z_{\mathbb{C}_{II}^{v=1}}} (x+\alpha^l)\right]m(x)$$

$$= g_{t_L}^{binBCH}(x)M(x).$$

If $t_G = t_L$, then

$$g_{\mathbb{C}_{\mathrm{II}}^{v=1}}(x) = g_{t_L}^{binBCH}(x),$$

$$\left[\prod_{l\notin \mathbb{C}\ and\ \ l\in z_{\mathbb{C}_{II}^{v=1}}} (x+\alpha^l)\right] = 1,$$

$$M(x) = \left[\prod_{l\notin \mathbb{C}\ and\ \ l\in z_{\mathbb{C}_{II}^{v=1}}} (x+\alpha^l)\right]m(x) = m(x).$$

Then $\kappa = degree\big(M(x)\big) = degree\big(m(x)\big) = k$, and

$$M(x) = M_0 + M_1x + \cdots + M_{\kappa-1}x^{\kappa-1} = m_0 + m_1x + \cdots + m_{\kappa-1}x^{k-1},$$

$$c(x) = g_{t_L}^{binBCH}(x)M(x) = g_{t_L}^{binBCH}(x)\cdot(m_0 + m_1x + \cdots + m_{\kappa-1}x^{k-1}).$$

Given that the basis for $GF(q^m)$ is $\mathcal{B} = \{\beta_0, \beta_1, \cdots, \beta_{m-1}\} = \{1, \alpha, \cdots, \alpha^{m-1}\}$, i.e.

$$M_i = M_i^{(0)}\beta_0 + M_i^{(1)}\beta_1 + \cdots + M_i^{(m-1)}\beta_{m-1}$$

$$= m_i^{(0)}\beta_0 + m_i^{(1)}\beta_1 + \cdots + m_i^{(m-1)}\beta_{m-1},$$

$$c(x) = g_{t_L}^{binBCH}\ (x)\left\{\sum_{i=0}^{\kappa-1}\left[m_i^{(0)}x^i\right]\beta_0 + \sum_{i=0}^{\kappa-1}\left[m_i^{(1)}x^i\right]\beta_1 + \cdots + \sum_{i=0}^{\kappa-1}\left[m_i^{(m-1)}x^i\right]\beta_{m-1}\right\}.$$

After an arbitrary rotation $\Re(s_0, s_2, \cdots, s_{m-1})$,

$$c_{\Re}(x) = g_{t_L}^{binBCH}\ (x)\left\{\sum_{i=0}^{\kappa-1}\left[\alpha^{s_0}m_i^{(0)}x^i\right]\beta_0 + \sum_{i=0}^{\kappa-1}\left[\alpha^{s_1}m_i^{(1)}x^i\right]\beta_1 + \cdots\right.$$
$$\left. + \sum_{i=0}^{\kappa-1}\left[\alpha^{s_{m-1}}m_i^{(m-1)}x^i\right]\beta_{m-1}\right\}.$$

The $j^{th}$ tuple of the binary $m$-tuple of $c(x)$ is:

$$g_{t_L}^{binBCH}\ (x)\sum_{i=0}^{\kappa-1}\left[\alpha^{s_j}m_i^{(j)}x^i\right].$$

It is a binary BCH codeword polynomial with error-correction capability $t_L$.

Given

$$g(x) = \prod_{i=1}^{2t_G}(x+\alpha^i)\ is\ a\ factor\ of\ g_{\mathbb{C}_{\mathrm{II}}^{v=1}}(x) = \prod_{i\in z\ \mathbb{C}_{\mathrm{II}}^{v=1}}(x+\alpha^i),$$

i.e. $g_{\mathbb{C}_{\mathrm{II}}^{v=1}}(x) = g_{t_L}^{binBCH}\ (x)$, and $g(x)$ is a factor of $g_{t_L}^{binBCH}\ (x)$.

$$c_{\Re}(x) = g(x)m_{\Re}(x).$$

Conclude above, $c_{\Re}(x)$ is a codeword of $C(n,k,D,d)$ by theorem 4.1. ■

# APPENDIX B

## Example of $\mathbb{C}_{\mathrm{II}}^{\nu=1}$'s "local" bit error correction.

Let $(1,4,0,0,1,5,4)$ be a codeword of $RS(7,5,3)$. $(1,4,0,0,1,5,4) \in RS(7,5,3)$ and $(1,4,0,0,1,5,4) \in \mathbb{C}_{\mathrm{II}}^{\nu=1}$. The binary image of $(1,4,0,0,1,5,4)$ is shown in Fig. B.1.

| 1 | 4 | 0 | 0 | 1 | 5 | 4 |
|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 1 | 0 |

Fig. B.1. The binary 3-tuple of $RS(7,5,3)$ codeword $(1,4,0,0,1,5,4)$.

$(0,1,0,0,0,1,1)$, $(0,0,0,0,0,0,0)$, and $(1,0,0,0,1,1,0) \in BCH(7,4,3)$. Suppose that three symbol errors existed in the received vector, which is actually caused by the bit errors underlined in Fig. B.2.

| 0 | 6 | 4 | 0 | 1 | 5 | 4 |
|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 1 | 0 |

Fig. B.2. The binary 3-tuple of $RS(7,5,3)$ codeword: $(1,4,0,0,1,5,4)$.

The received vector (0,6,4,0,1,5,4) cannot be corrected by $RS(7,5,3)$. However, the binary image of (0,6,4,0,1,5,4) shows us that it can be corrected by the parallel sub-structure of $\mathbb{C}_{\mathrm{II}}^{v=1}$. Namely, (0,1,1,0, 0,1,1) can be decoded by $BCH(7,4,3)$, (0,1, 0,0,0,0,0) can be decoded by $BCH(7,4,3)$, and (0,0,0,0,1,1,0) can be decoded by $BCH(7,4,3)$.

The "local" bit error correction process is shown in Fig. B.3.

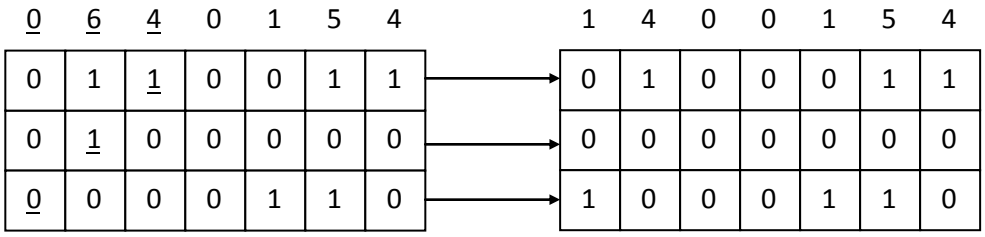


Fig. B.3. Example of $\mathbb{C}_{\mathrm{II}}^{v=1}$'s "local" bit error correction.

From this example, we can see that $RS(7,5,3)$ can only correct one error, so that it cannot decode (0,6,4,0,1,5,4) . However, using the parallel sub-structure, we can decode (1,4,0,0,1,5,4) from (0,6,4,0,1,5,4) which has three errors.

# APPENDIX C

**Example of** $\mathbb{C}_{\mathrm{II}}^{\langle v_1|v_2|\cdots|v_{max}\rangle} = \mathbb{C}_{\mathrm{II}}^{\langle v_1=1|v_2=2\rangle}$.

For example, with $m = 4$, $\mathbb{C}_{\mathrm{II}}^{\langle v_1|v_2|\cdots|v_{max}\rangle} = \mathbb{C}_{\mathrm{II}}^{\langle v_1=1|v_2=2\rangle}$. As above defined, $\mathbb{C}_{\mathrm{II}}^{\langle v_1=1|v_2=2\rangle}$ has two different parallel structures, because $m$ has two distinct integer dividers:$v_1 = 1$, $v_2 = 2$. Structure I and II are shown in Fig. C.1, and Fig. C.2, respectively.

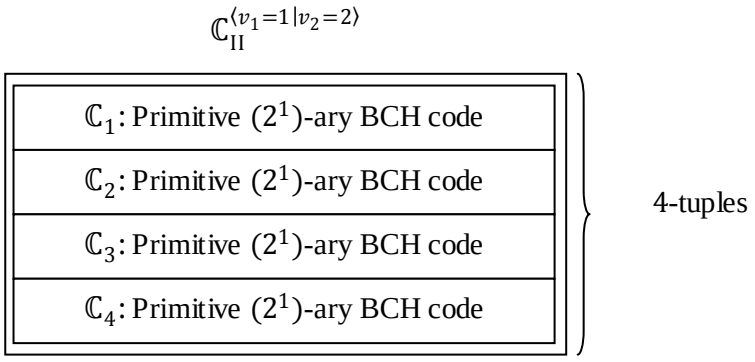


Fig. C.1. $\mathbb{C}_{\mathrm{II}}^{\langle v_1=1|v_2=2\rangle}$ structure I.

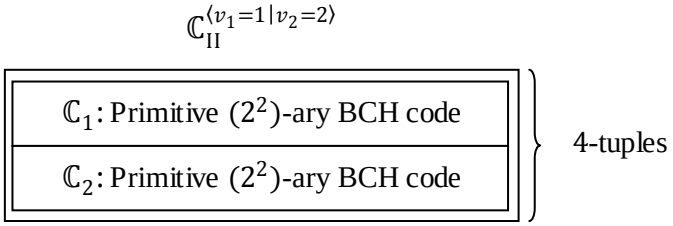


Fig. C.2. $\mathbb{C}_{\mathrm{II}}^{\langle v_1=1|v_2=2\rangle}$ structure II.

As an example, given a parent narrow sense $RS(15,9,7)$ code over $GF(2^m)$ with $m = 4$, the zeros of the generator polynomial are $z = \{1,2,3,4,5,6\}$, and

$$g(x) = \prod_{i=1}^{6}(x + \alpha^i).$$

The subcode of the parent RS code has two distinct parallel structures with $\{t_{L_1} = 2, t_{L_2} = 3\}$ and $\{v_1 = 1, v_2 = 2\}$.

The corresponding cyclotomic cosets $\mathsf{C}_{t_{L_i}}$ are:

$\mathsf{C}_{t_{L_1}}$: cyclotomic coset of $\{1,2,3, 2t_{L_1} = 4\}$ $module\ n = 15$ with respect to $GF(2^{v_1}) = GF(2)$, $\{1,2,4, 8\}$ and $\{3,6,12, 9\}$.

$\mathsf{C}_{t_{L_2}}$: cyclotomic coset of $\{1,2,3, 4,5,2t_{L_2} = 6\}$ $module\ n = 15$ with respect to $GF(2^{v_2}) = GF(4)$ is: $\{1,4\}$, $\{2,8\}$, $\{3,12\}$, $\{5\}$, $\{6,9\}$.

$\mathsf{C}$ is the union of the $\mathsf{C}_{t_{L_i}}$,

$$\mathsf{C} = \mathsf{C}_{t_{L_1}} \cup \mathsf{C}_{t_{L_2}} = \{1,2,3,4,5,6,8,9,12\},$$

$$z_{\mathbb{C}_{\mathrm{II}}^{\langle v_1 | v_2 | \cdots | v_{max} \rangle}} = z \cup \mathsf{C} = \{1,2,3,4,5,6,8,9,12\}.$$

The zeros we embedded are $\{8,9,12\}$, and

$$g_{\mathbb{C}_{\mathrm{II}}^{\langle v_1|v_2|\cdots|v_{max}\rangle}}(x) = \prod_{i\in z_{\mathbb{C}_{\mathrm{II}}^{\langle v_1|v_2|\cdots|v_{max}\rangle}}} (x+\alpha^i).$$

We can compute the codeword of $\mathbb{C}_{\mathrm{II}}^{\langle v_1|v_2|\cdots|v_{max}\rangle}$ as

$$c_{\mathbb{C}_{\mathrm{II}}^{\langle v_1|v_2|\cdots|v_{max}\rangle}} = g_{\mathbb{C}_{\mathrm{II}}^{\langle v_1|v_2|\cdots|v_{max}\rangle}}(x)m(x).$$

For structure I:

$$\begin{aligned} c_{\mathbb{C}_{\mathrm{II}}^{\langle v_1|v_2|\cdots|v_{max}\rangle}} &= g_{\mathbb{C}_{\mathrm{II}}^{\langle v_1|v_2|\cdots|v_{max}\rangle}}(x)m(x) \\ &= (x+\alpha^1)(x+\alpha^2)(x+\alpha^4)(x+\alpha^8) \\ &\cdot (x+\alpha^3)(x+\alpha^6)(x+\alpha^{12})(x+\alpha^9)\cdot(x+\alpha^5)m(x) \\ &= (x^4+x+1)\cdot(x^4+x^3+x^2+x+1)\cdot(x+\alpha^5)m(x) \\ &= g_{t_{L_1}=2}^{binBCH}(x)\cdot(x+\alpha^5)m(x) = g_{t_{L_1}=2}^{binBCH}(x)\cdot M(x). \end{aligned}$$

Let $\kappa = degree(M(x))$.

$$M(x) = M_0 + M_1 x + \cdots + M_{\kappa-1}x^{\kappa-1},$$

$$c(x) = g_{t_{L_1}=2}^{binBCH}(x)M(x) = g_{t_{L_1}=2}^{binBCH}(x)\cdot(M_0 + M_1 x + \cdots + M_{\kappa-1}x^{\kappa-1}).$$

Assume that the basis for $GF(2^4)$ is $\mathcal{B} = \{\beta_0, \beta_1, \cdots, \beta_3\} = \{1, \alpha, \cdots, \alpha^3\}$. Then

$$M_i = M_i^{(0)}\beta_0 + M_i^{(1)}\beta_1 + \cdots + M_i^{(3)}\beta_3,$$

$$c(x) = g_{t_{L_1}=2}^{binBCH}(x)\left\{\sum_{i=0}^{\kappa-1}\left[M_i^{(0)}x^i\right]\beta_0 + \sum_{i=0}^{\kappa-1}\left[M_i^{(1)}x^i\right]\beta_1 + \cdots + \sum_{i=0}^{\kappa-1}\left[M_i^{(3)}x^i\right]\beta_3\right\}.$$

The $j^{th}$ tuple of the binary $m$-tuple of $c(x)$ is:

$$g_{t_{L_1}=2}^{binBCH}(x)\sum_{i=0}^{\kappa-1}\left[M_i^{(j)}x^i\right].$$

It is a binary BCH codeword polynomial with error-correction capability $t_{L_1}=2$.

$g(x)=\sum_{i=1}^{6}(x+\alpha^i)$ is a factor of $g_{\mathbb{C}_{\mathrm{II}}^{\langle v_1|v_2|\cdots|v_{max}\rangle}}(x)=(x+\alpha^1)(x+\alpha^2)(x+\alpha^4)(x+\alpha^8)\cdot(x+\alpha^3)(x+\alpha^6)(x+\alpha^{12})(x+\alpha^9)\cdot(x+\alpha^5)$.

That is: $c(x)=g(x)\cdot(x+\alpha^8)(x+\alpha^9)(x+\alpha^{12})\cdot m(x)$. This is a codeword of the parent RS code with the generator polynomial $g(x)$.

For structure II:

$$\begin{aligned}c_{\mathbb{C}_{\mathrm{II}}^{\langle v_1|v_2|\cdots|v_{max}\rangle}} &= g_{\mathbb{C}_{\mathrm{II}}^{\langle v_1|v_2|\cdots|v_{max}\rangle}}(x)m(x)\\ &=(x+\alpha^1)(x+\alpha^4)\cdot(x+\alpha^2)(x+\alpha^8)\cdot(x+\alpha^3)(x+\alpha^{12})\\ &\cdot(x+\alpha^5)\cdot(x+\alpha^6)(x+\alpha^9)m(x)\\ &=(x^2+x+\beta)\cdot(x^2+x+\beta^2)\cdot(x^2+\beta^2x+1)\cdot(x+\beta)\\ &\cdot(x^2+\beta x+1)m(x)=g_{t_{L_2}=3}^{4-ary\ BCH}(x)\cdot m(x).\end{aligned}$$

Here $\beta=\alpha^5,\beta^2=\alpha^{10}\in GF(4)=\{0,1,\beta,\beta^2\}$ with $\beta^3=1$, and

$$c_{\mathbb{C}_{\mathrm{II}}^{\langle v_1|v_2|\cdots|v_{max}\rangle}}=g_{t_{L_2}=3}^{4-ary\ BCH}(x)\cdot m(x).$$

Choose $\mathcal{B}'=\{\beta_0',\beta_1'\}=\{1,\alpha^2\}$, every $\tau\in GF(2^4)$ can be represented as a form:

$\tau = \tau_0 \cdot \beta_0^{'} + \tau_1 \cdot \beta_1^{'}$, with $\tau_0, \tau_1 \in GF(4) = \{0,1,\beta,\beta^2\}$.

| + | 0 | 1 | $\beta$ | $\beta^2$ |
|---|---|---|---|---|
| 0 | 0 | 1 | $\alpha^5$ | $\alpha^{10}$ |
| $\beta_1^{'}$ | $\alpha^2$ | $\alpha^8$ | $\alpha^1$ | $\alpha^4$ |
| $\beta\beta_1^{'}$ | $\alpha^7$ | $\alpha^9$ | $\alpha^{13}$ | $\alpha^6$ |
| $\beta^2\beta_1^{'}$ | $\alpha^{12}$ | $\alpha^{11}$ | $\alpha^{14}$ | $\alpha^3$ |

Fig. C.3. $GF(16)$ element decomposition with basis $\mathcal{B}^{'} = \{1, \alpha^2\}$ over $GF(4)$.

Then,

$$c_{\mathbb{C}_{\text{II}}^{\langle v_1|v_2|\cdots|v_{max}\rangle}} = g_{t_{L_2}=3}^{4-ary\ BCH}(x) \cdot m(x)$$

$$= g_{t_{L_2}=3}^{4-ary\ BCH}(x) \cdot \left\{ \sum_{i=0}^{k-1} \left[ m_i^{(0)} x^i \right] \beta_0^{'} + \sum_{i=0}^{k-1} \left[ m_i^{(1)} x^i \right] \beta_1^{'} \right\}.$$

The $j^{th}$ tuple of the 4-ary 2-tuple of $c(x)$ is:

$$g_{t_{L_2}=3}^{4-ary\ BCH}(x) \sum_{i=0}^{k-1} \left[ m_i^{(j)} x^i \right].$$

It is a 4-ary BCH codeword polynomial with error-correction capability $t_{L_2} = 3$.

$g(x) = \sum_{i=1}^{6}(x + \alpha^i)$ is a factor of $g_{\mathbb{C}_{\text{II}}^{\langle v_1|v_2|\cdots|v_{max}\rangle}}(x) = (x+\alpha^1)(x+\alpha^2)(x+\alpha^4)(x+\alpha^8)\cdot(x+\alpha^3)(x+\alpha^6)(x+\alpha^{12})(x+\alpha^9)\cdot(x+\alpha^5)$. That is: $c(x) = g(x)\cdot(x+\alpha^8)(x+\alpha^9)(x+\alpha^{12})\cdot m(x)$. This is a codeword of the parent RS code with generator polynomial $g(x)$. Previously we used $C(n,k,D,d)$ as the

notation for the generalized $\mathbb{C}_{\mathrm{II}}^{v=1}$codes with $D = 2t_G + 1$ and $d = 2t_L + 1$. For the generalized $\mathbb{C}_{\mathrm{II}}^{\langle v_1|v_2|\cdots|v_{max}\rangle}$ codes, we use $C(n, k, D, d_{(v_1)}, \cdots, d_{(v_{max})})$ with $D = 2t_G + 1, d_{(v_1)} = 2t_{L_1} + 1, \cdots, d_{(v_{max})} = 2t_{L_{max}} + 1$. As an example, given the parent RS(15,9,7) code, the $\mathbb{C}_{\mathrm{II}}^{v=1}$ with $t_{L_1} = 2$ is $C(15,6,7,5)$; the $\mathbb{C}_{\mathrm{II}}^{\langle v_1|v_2|\cdots|v_{max}\rangle} = \mathbb{C}_{\mathrm{II}}^{\langle v_1=1|v_2=2\rangle}$ with $D = 5, t_{L_1} = 2, t_{L_2} = 3$ is $C(15,5,7,5_{(1)},7_{(2)})$, which has two distinct parallel structures.

# APPENDIX D

## Construction method for $\mathbb{C}_{\mathrm{II}}^{\langle v_1 \supset \cdots | v_2 \supset \cdots | \cdots | v_{max} \supset \cdots \rangle}$.

1) Given $GF(2^m)$, $m$ is positive integer, there exist series of positive dividers

$$\{v_1, v_2, \cdots, v_{max} \mid \ \forall v_i | m, v_1 = 1 < v_2 < \cdots < v_{max} \ < m\}.$$

2) Given a parent narrow sense $RS(n, k, D = 2t_G + 1)$ with generator polynomial

$$g(x) = \prod_{i=1}^{2t_G} (x + \alpha^i).$$

The desinged $\mathbb{C}_{\mathrm{II}}^{\langle v_1 \supset v_{1.1} \supset \cdots \supset v_{1.d_1} | v_2 \supset v_{2.1} \supset \cdots \supset v_{2.d_2} | \cdots | v_{max} \supset v_{max\ .1} \supset \cdots \supset v_{max\ .d_{max}} \rangle}$ code has $|\{v_1, v_2, \cdots, v_{max}\}|$ distinct parallel structures in each of which there are $d_i$ stages of parallel primitive $(2^{v_{i.j}})$-ary BCH codes structures respectively with corresponding error correction capability of $t_{L_{i.j}}$, satisfying that $t_{L_i} \geq t_{L_{i.1}} \geq t_{L_{i.2}} \geq \cdots \geq t_{L_{i.d_i}}$.

3) *For i = 1 to max*

Start from each cyclotomic coset $\mathrm{C}_{t_{L_i}}^{(0)}$ of $\left\{1,2,\cdots,2t_{L_{i.d_i}}\right\}$ module $n$ with respect to $GF(2^{v_{i.d_i}})$.

*For* $k = 1$ *to* $d_i - 1$

$\mathrm{C}_{t_{L_i}}^{(k)}$ is the cyclotomic coset of the union of $\mathrm{C}_{t_{L_i}}^{(k-1)} \cup \left\{1,2,\cdots,2t_{L_{i.(d_i-k)}}\right\}$ module $n$ with respect to $GF(2^{v_{i.(d_i-k)}})$.

*End*

$C_{t_{L_i}}$ is the cyclotomic coset of the union of $C_{t_{L_i}}^{(d_i-1)} \cup \{1,2,\cdots,2t_{L_i}\}$ module $n$ with respect to $GF(2^{v_i})$.

*End*

4) $C = C_{t_{L_1}} \cup C_{t_{L_2}} \cup C_{t_{L_3}} \cup \cdots \cup C_{t_{L_{max}}}$ .

5) Let $z = \{1,2,\cdots,2t_G\}$, and use the simplified notation $\mathbb{C}_{\mathrm{II}}^{\langle v_1 \supset \cdots | v_2 \supset \cdots | \cdots | v_{max} \supset \cdots \rangle}$.
$z_{\mathbb{C}_{\mathrm{II}}^{\langle v_1 \supset \cdots | v_2 \supset \cdots | \cdots | v_{max} \supset \cdots \rangle}} = z \cup C$.

$$g_{\mathbb{C}_{\mathrm{II}}^{\langle v_1 \supset \cdots | v_2 \supset \cdots | \cdots | v_{max} \supset \cdots \rangle}}(x) = \prod_{i \in z_{\mathbb{C}_{\mathrm{II}}^{\langle v_1 \supset \cdots | v_2 \supset \cdots | \cdots | v_{max} \supset \cdots \rangle}}} (x + \alpha^i).$$

6) Compute the codeword of $\mathbb{C}_{\mathrm{II}}^{\langle v_1 \supset \cdots | v_2 \supset \cdots | \cdots | v_{max} \supset \cdots \rangle}$:

$$c_{\mathbb{C}_{\mathrm{II}}^{\langle v_1 \supset \cdots | v_2 \supset \cdots | \cdots | v_{max} \supset \cdots \rangle}} = g_{\mathbb{C}_{\mathrm{II}}^{\langle v_1 \supset \cdots | v_2 \supset \cdots | \cdots | v_{max} \supset \cdots \rangle}}(x) m(x).$$

The notation used for the generalized $\mathbb{C}_{\mathrm{II}}^{\langle v_1 | v_2 | \cdots | v_{max} \rangle}$ codes is

$$C\left(n, k, D, \left[d_{(v_1)}, \cdots, d_{(v_{1.d_1})}\right], \cdots\right)$$

wtih

$$D = 2t_G + 1, \quad d_{(v_1)} = 2t_{L_1} + 1, \ \cdots, \ d_{(v_{max})} = 2t_{L_{max}} + 1, d_{\left(v_{i.d_i}\right)} = 2t_{L_{i.d_i}} + 1.$$

# APPENDIX E

## Encoding schemes for new LDPC+$\mathbb{C}_{\mathrm{II}}^{v=1}$ concatenation sytem.

The message is first encoded into a long LDPC code $\mathcal{L}$ or $m$ parallel short LDPC codewords $\mathcal{L}_{\mathcal{p}_1}, \mathcal{L}_{\mathcal{p}_1}, \cdots, \mathcal{L}_{\mathcal{p}_m}$. Then the LDPC codeword(s) is (are) then encoded by the systematic $\mathbb{C}_{\mathrm{II}}^{v=1}$ encoder as the information message of the $\mathbb{C}_{\mathrm{II}}^{v=1}$ codeword. Here the $\mathbb{C}_{\mathrm{II}}^{v=1}$ code is acting as the inner code. However, at the channel output, because the $\mathbb{C}_{\mathrm{II}}^{v=1}$ encoding is systematic, the message part of $\mathbb{C}_{\mathrm{II}}^{v=1}$ codeword is the LDPC codeword. It is well known that LDPC is a more powerful code to deal with more errors, so we use the LDPC decoding to eliminate most errors in the message part of the $\mathbb{C}_{\mathrm{II}}^{v=1}$ codeword. After the LDPC decoding, we then use $\mathbb{C}_{\mathrm{II}}^{v=1}$ decoding to remove remaining errors from the output of the LDPC decoding as well as those from the parity part of $\mathbb{C}_{\mathrm{II}}^{v=1}$ codeword.

*Encoding scheme I*

Given a long binary LDPC codeword $\mathcal{L}$, we partition it into $m$ parts: denote as $\mathcal{L}_1, \mathcal{L}_2, \cdots, \mathcal{L}_m$ in the binary form. Given $GF(2^m)$, each symbol has a binary $m$- tuple image. $\mathcal{L}_1, \mathcal{L}_2, \cdots, \mathcal{L}_m$ have the same length, say $n$, so we take the same index bit $b_{i,j}$ from each $\mathcal{L}_i$, where $i = 1, \cdots, m$ and $j = 1, \cdots, n$. We take every $m$ $b_{i,j}$ $(i = 1, \cdots, m)$ of the same index $j$ as the binary $m$- tuples of a symbol over $GF(2^m)$. In the way, we

can convert $\mathcal{L}_1, \mathcal{L}_2, \cdots, \mathcal{L}_m$ into $n$ symbols over $GF(2^m)$, which are the messages of the systematic $\mathbb{C}_{\mathrm{II}}^{v=1}$ encoder. The work flow is shown in Fig. E.1.

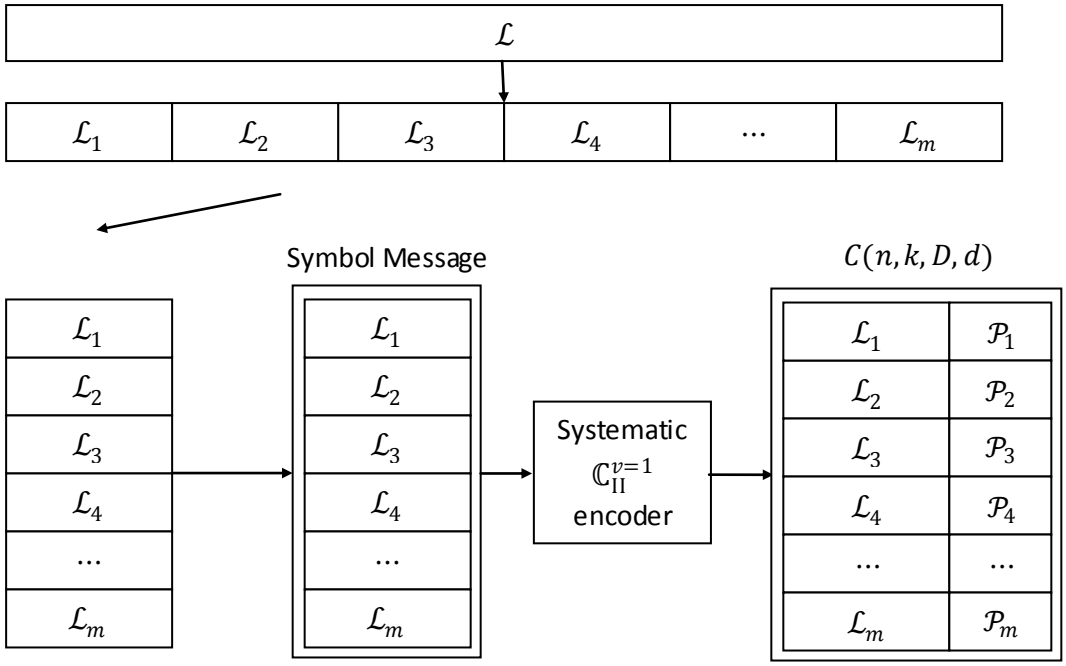


Fig. E.1. Encoding scheme *I*.

As noted in Chapter 4, $C(n,k,D,d)$ is the symbol level $\mathbb{C}_{\mathrm{II}}^{v=1}$ codeword, whose binary $m$-tuples are $m$ parallel binary BCH codewords. So in encoding scheme *I*, each $\mathcal{L}_i$ and each $\mathcal{P}_i$ are the corresponding message part and the parity part of the binary BCH codeword, respectively.

*Encoding scheme II*

In this scheme, we have $m$ parallel short binary LDPC codewords, $\mathcal{L}_{\mathcal{P}_1}, \mathcal{L}_{\mathcal{P}_1}, \cdots, \mathcal{L}_{\mathcal{P}_m}$ instead of a long binary LDPC codeword in the scheme *I*.

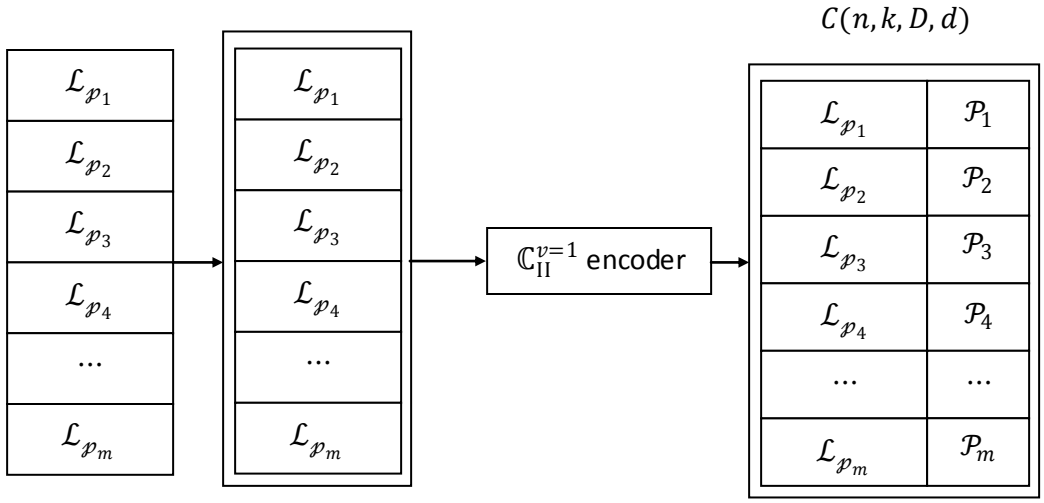


Fig. E.2. Encoding scheme *II*.

*Encoding scheme III*

In the scheme *III*, we use the $Q$-ary LDPC codeword over $GF(2^m)$ directly as the message part of the $\mathbb{C}_{\mathrm{II}}^{v=1}$ codeword

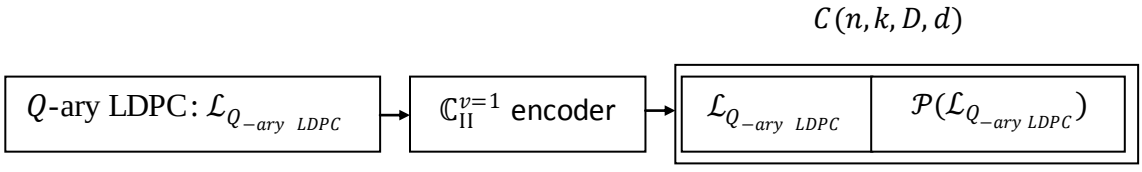


Fig. E.3. Encoding scheme using the $Q$-ary LDPC code.

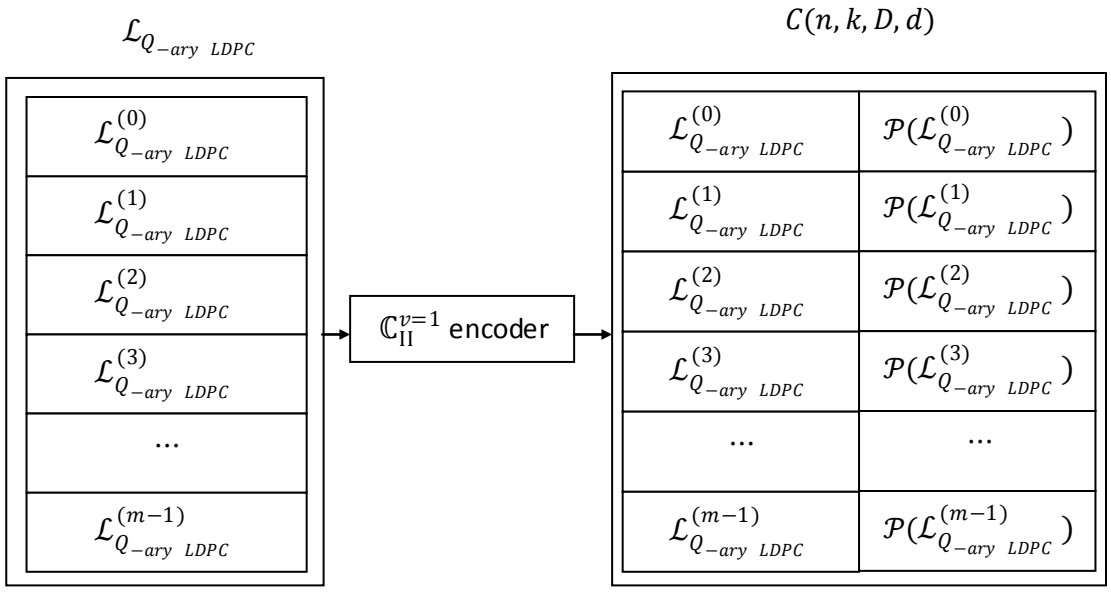


Fig. E.4. Binary $m$-tuple image of the scheme using a $Q$-ary LDPC codeword.

*Encoding scheme IV*

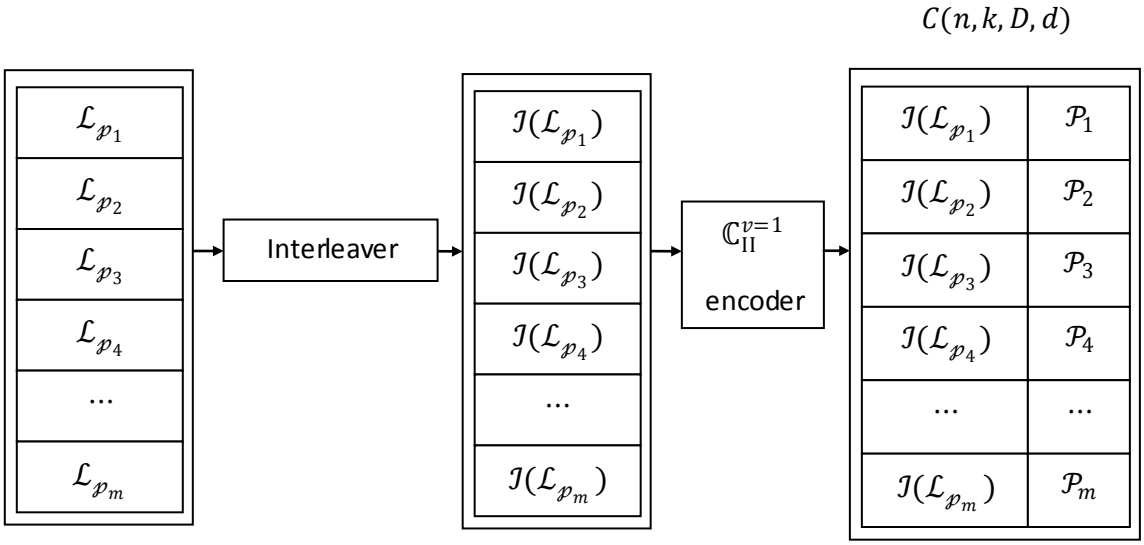


Fig. E.5. Encoding scheme using a multiple codeword interleaver.

# APPENDIX F

**Iterative parallel local decoding workflow for $\mathbb{C}_{\mathrm{II}}^{\langle v_1|v_2|\cdots|v_{\max}\rangle}$.**

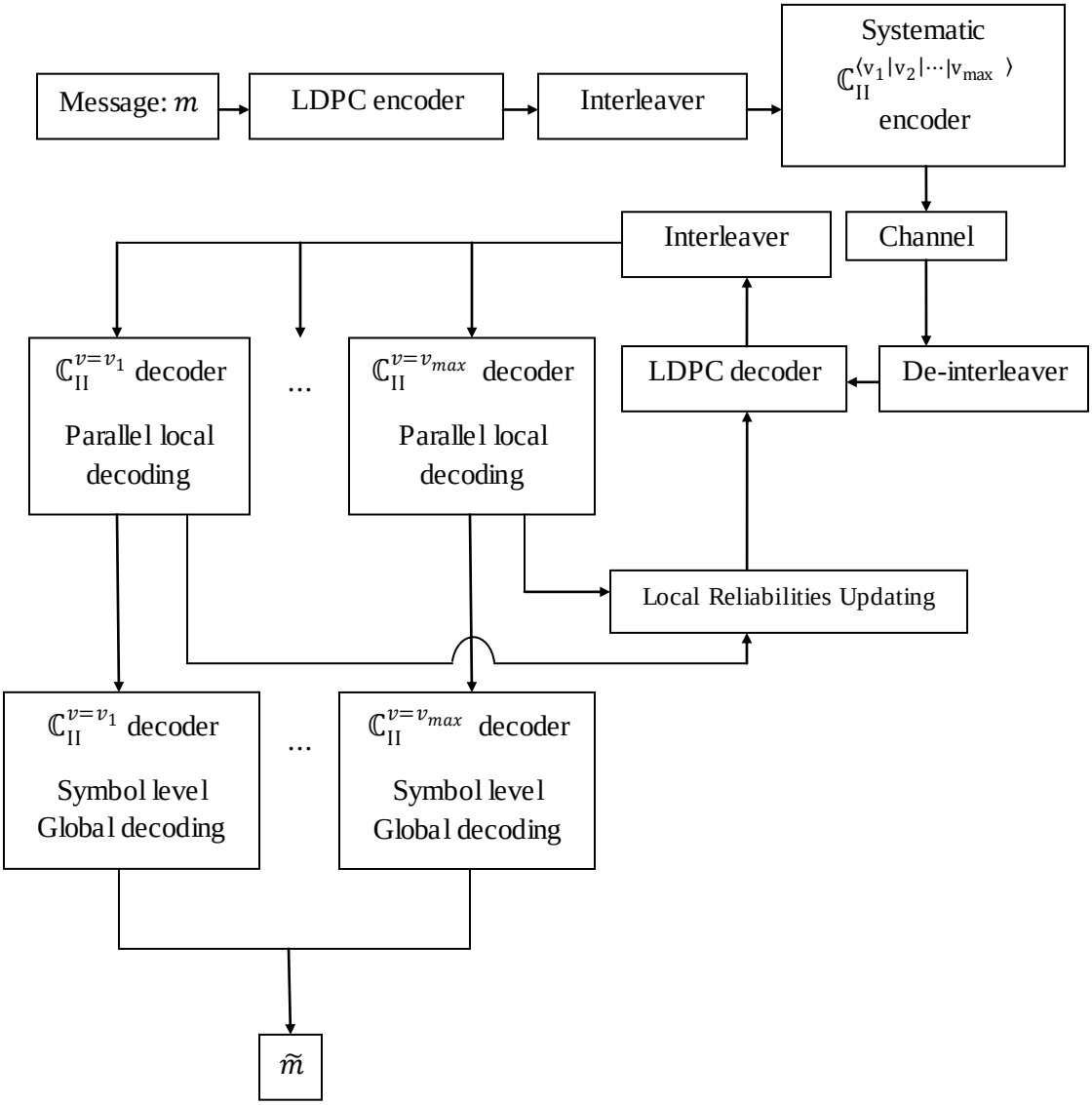


Fig. F.1. Iterative parallel local decoding for LDPC+RS ($\mathbb{C}_{\mathrm{II}}^{\langle v_1|v_2|\cdots|v_{\max}\rangle}$) system.