TWO DIMENSIONAL IRREGULAR REPEAT-ACCUMULATE CODES

By

QINGWEI GE

A thesis submitted in partial fulfillment of
the requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

WASHINGTON STATE UNIVERSITY
School of Electrical Engineering and Computer Science

May 2009

To the Faculty of Washington State University:

The members of the Committee appointed to examine the thesis of QINGWEI GE find it satisfactory and recommend that it be accepted.

Benjamin Belzer, Ph.D., Chair

Krishnamoorthy Sivakumar, Ph.D., Co-chair

Thomas R. Fischer, Ph.D.

ACKNOWLEDGMENT

I would like to express my sincere thanks to my advisor Dr. Benjamin Belzer and co-advisor Dr. Krishnamoorthy Sivakumar. They have been providing me constant help regarding my study and research during the two years. They gave me a detailed guide to error control coding, and they spent many hours every week discussing research work with me. I appreciate their great efforts to explain things clearly and simply to me. They also gave me valuable ideas and advice on writing my thesis. Without them, this thesis would not be possible.

My special thanks should go to Dr. Thomas R. Fischer, member of my thesis committee. I gained an important foundation of information systems from his several inspiring courses. I also want to thank many other faculty members of the School of EECS who have given me valuable knowledge and advice.

The work in this thesis was supported partly by the National Science Foundation under grant CCF-0635390 and partly by the School of EECS, Washington State University. I appreciate their financial support.

Lastly, and most importantly, I wish to thank my parents Xiangrui Ge and Ruirong Li. They made exceptional efforts to support me all the way through my graduate school, both emotionally and financially. They had major influences on my character. They love me in every way they can. To them I dedicate this thesis.

TWO DIMENSIONAL IRREGULAR REPEAT-ACCUMULATE CODES

Abstract

by Qingwei Ge, M.S.
Washington State University
May 2009

Chair: Benjamin Belzer, Co-chair: Krishnamoorthy Sivakumar

Irregular Repeat-Accumulate (IRA) codes are a subclass of low-density parity-check (LDPC) codes with linear encoding complexity. They can be serially concatenated to form two dimensional block codes, called Serially Concatenated IRA (SC-IRA) codes. SC-IRA codes are constructed to address the error floor problem of LDPC codes. Previously a serial decoder structure with two component decoders was used to decode SC-IRA codes. In this thesis, the serial decoder structure is analyzed and extended. Then a novel one-graph decoding method is proposed.

Simulation results show that the one-graph decoding method is better than the serial decoding method in both low and high Signal-to-Noise Ratio (SNR) regions. There is about 0.7 dB gain in the waterfall region compared to the serial decoding method. The one graph decoding method also suggests a good way to construct very long LDPC-like codes with low error floors.

The performance of IRA codes and SC-IRA codes is analyzed based on a tight Maximum-Likelihood (ML) decoding upper bound. The ML bounds of several IRA codes and SC-IRA codes are computed for comparison.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 An Overview

The demand for information exchange has been ever increasing. A long way has been covered from the early simple data networks within institutions, to the complex nation-wide and global communication networks. These years new communication requirements arise: high throughput and high reliability data transmission. High quality multimedia networking and video conferencing are examples of high throughput data communication. The demand for high reliability data transmission comes from applications where an extremely low bit error rate ($10^{-12} - 10^{-15}$) is desired, for example, commercial or scientific satellite communications.

From the beginning of this century, people are experiencing an all-around digital world. Digital devices are working and communicating with each other anywhere and anytime. There is a whole literature addressing digital communications, and many communication systems have been standardized. For example, the 3rd generation cellular standards UTMS and CDMA2000, to name a few. These standards specify how the source coding, channel coding and modulation are done as well as the interconnections between these different standards, so that users can experience seamless

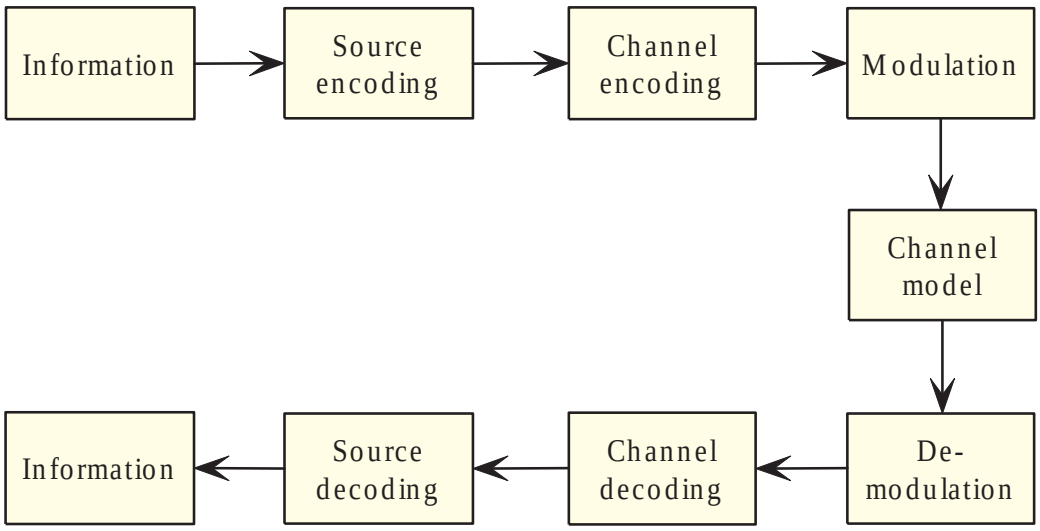


Figure 1.1: A block diagram of a typical digital communication system.

services.

Figure 1.1 is a block diagram of a typical digital communication system. Before transmission on the channel, digitization, source encoding, channel encoding and modulation are performed on the information source. At the receiver, demodulation, channel decoding, source decoding and signal reconstruction are performed to extract the original information. Each of these blocks is a full fledged subject, and many researchers are still making contributions to them. The Irregular Repeat-Accumulate (IRA) codes we will discuss in this thesis belong to the channel encoding and decoding blocks, which employ Forward Error Control (FEC) codes.

Forward Error Control codes were invented to transmit digital information reliably over noisy channels. In FEC, redundant information is added in a controlled manner to the source information, so that if the source information is corrupted by channel noise, it can be recovered with the assistance of the redundant information. The redundancy protects the information against channel noise. FEC deals with how to

arrange the redundancy effectively, or in other words, how to use less redundancy to protect more information, while ensuring the quality of transmission.

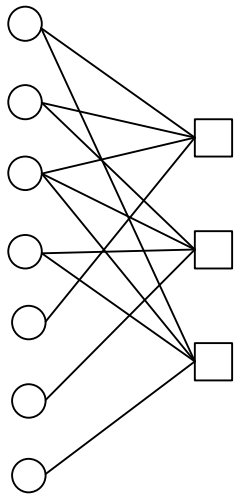
The channels over which the information is transmitted vary. There are several common models to describe the channel characteristics. The Additive White Gaussian Noise (AWGN) channel and the Fading channel are the two most commonly used models in designing good FEC codes. In this thesis, we only consider the AWGN channel.

Linear block codes and convolutional codes are the two main categories of FEC codes. Low-density Parity-check (LDPC) codes and Turbo codes are well known examples of these two categories, respectively. With the invention of soft decision iterative decoding algorithms, these codes are able to approach the Shannon limit within less than 1 dB.

1.2 Introduction to LDPC and IRA Codes

LDPC codes, which were discovered by Gallager [1] in the early 1960s and re-discovered by Tanner in 1981 [2], are the most extensively studied codes today. LDPC codes can approach the Shannon limit within less than 0.1 dB [3]. However, LDPC codes did not receive attention until the concepts of Tanner graphs and iterative decoding were applied to their decoding algorithms.

LDPC codes can be represented by Tanner graphs. Tanner graphs are bipartite graphs that are characterized by two types of nodes: the variable nodes and the check nodes. The edges connecting the variable nodes and the check nodes in the graph correspond to the 1s in the parity check matrix. Typically the Sum Product Algorithm (SPA) [4] is used to iteratively decode the LDPC codes on the Tanner graphs.



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

Figure 1.2: The Tanner graph and the parity check matrix of a (7,4) Hamming code.

Figure 1.2 is an example of the Tanner graph and the corresponding parity check matrix. It is the Tanner graph of a $(7, 4)$ Hamming code. The circles are variable nodes, and the squares are check nodes. Each variable node in the Tanner graph corresponds to a column in the H matrix; each check node corresponds to a row in the H matrix. If the jth element in the ith column in the H matrix is a 1, then there is an edge connecting the jth check node and the ith variable node. An edge must connect exactly one variable node and one check node. The number of edges a node has is called the degree of the node. For example, in Figure 1.2 the 1st, 2nd and 4th variable nodes have two edges, and thus their degree is 2. All the check nodes in Figure 1.2 have a degree of 4. The Tanner graph of an LDPC code is constructed the same way as that of the Hamming code, except that the parity check matrix of an LDPC code is sparse (low density), i.e., the number of 1s is small compared to the number of 0s.

The encoding of LDPC codes involves a $K \times N$ generator matrix multiplica-

tion, where K is the information sequence size and N is the codeword size. The maximum-likelihood (ML) decoding algorithm is the optimal decoding algorithm for LDPC codes. Unfortunately, exact ML decoding of general linear codes is an NP-hard problem [15]. Although it is shown that exact ML decoding of expander codes, a special case of LDPC codes, is possible with an expected polynomial complexity [16], at this time there is no practical ML decoding algorithm available for general linear codes. Instead, sub-optimal decoding algorithms are used to decode LDPC codes, for example, bit-flipping decoding [1], majority-logic decoding [5] and iterative SPA decoding [4]. Among these decoding algorithms, SPA decoding gives the best performance with low computational complexity. But SPA also has an error floor problem, which will be discussed later.

Irregular Repeat-Accumulate (IRA) codes are an ensemble of linear block codes that are similar to LDPC codes [6]. IRA codes have linear encoding complexity and the same decoding complexity as LDPC codes. IRA codes avoid the generator matrix multiplication in the encoding procedure by introducing an accumulator into the code structure. IRA codes are actually an extension of Repeat-Accumulate (RA) codes. RA codes were studied by Divsalar et. al. for theoretical purposes in the 1990s. In RA codes, all the variable nodes have the same repeat index. In other words, the degrees of the variable nodes are the same. In IRA codes, however, the variable nodes can have different degrees. With a carefully chosen variable node degree sequence, IRA codes can achieve a much smaller decoding threshold than RA codes. The threshold of a degree sequence is defined as the smallest SNR point at which the received channel values begin to converge to their true values with an infinite codeword size.

Theoretically, if the SNR is high enough, the bit-error-rate (BER) can be arbitrarily low. In other words, the BER will decrease consistently as SNR goes up. But

for both the Turbo codes and LDPC codes, researchers find that when the SNR is above a certain point, the decreasing rate of the BER will slow down and may even not decrease any more. The SNR-BER curve drops rapidly in the low SNR region (also known as "waterfall" region), and it becomes flat in the high SNR region (the bottom of "waterfall"), which looks like a "floor". Thus this phenomenon is named the "error floor" problem.

The error floor problem is unacceptable when high accuracy transmission is desired. Because of the error floor problem, transmitting the information at a very high power will not guarantee a very low BER. A lot of work has been done to address this problem, for example, structured LDPC and structured IRA codes [7] [8], generalized and doubly generalized LDPC codes [9], etc. In this thesis, we will study the IRA codes and how they can be used to construct serially concatenated codes with lower error floors.

The serial or parallel concatenation of short codes is a standard approach to construct longer codes with larger minimum codeword weight. These codes often have special advantages such as scalable parallel encoding and decoding algorithms or better BER performance. For example, in the *Digital Video Broadcasting - Satellite Second Generation* (DVB-S2) standard, a concatenation of BCH (Bose-Chaudhuri-Hocquengham) codes with LDPC codes is used as the inner and outer code respectively to form a serial concatenation, which gives a lower BER than using only either one of them [10]. The serial concatenation scheme provides the receiver powerful error correcting ability to ensure high quality video broadcasting even in harsh weather conditions.

In this thesis, we study the serial concatenation of two component IRA codes in a two dimensional block. It is similar to product codes [11] [12] [13], but the interleaver

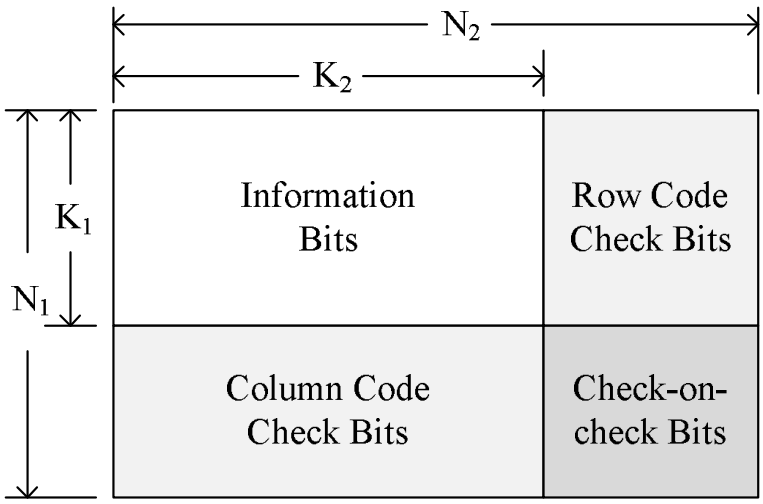


Figure 1.3: The codeword block of an SC-IRA code.

between the two component codes makes it different from regular product codes. Figure 1.3 shows a codeword of the serially concatenated IRA (SC-IRA) codes. It has four blocks: the information block, the row check block, the column check block and the check-on-checks block. It is observed that SC-IRA codes have lower error floors than their equivalent IRA codes. Various aspects of SC-IRA codes are studied in the following chapters.

1.3 Thesis Outline

Chapter 2 discusses IRA codes and SC-IRA codes. Various aspects of IRA and SC-IRA code design are explored, including general design methods, degree distributions, loop spectrum properties and interleaver design and optimization. Chapter 3 discusses two possible decoding schemes for SC-IRA codes. First the serial decoding scheme is presented, and the EXIT chart technique is used to optimize the serial decoder.

Next the one-graph decoding scheme is proposed and analyzed. Simulation results of these two decoding schemes are also provided and compared in this chapter. Chapter 4 provides performance analysis of the IRA codes and SC-IRA codes. A step-by-step derivation is outlined on how to obtain the Input and Output Weight Enumerators (IOWEs) of these codes. A tight maximum-likelihood decoding upper bound is used to predict the code's performance; relevant results are given. Chapter 5 concludes the thesis and outlines possible future work on this topic.

Chapter 2

IRA and SC-IRA Codes

2.1 Introduction to IRA Codes

Irregular Repeat-Accumulate (IRA) codes [6] are an ensemble of Turbo-like LDPC codes. They are like Turbo codes because they have an accumulator that accumulates the check nodes. The accumulator also gives IRA codes the advantage of linear encoding complexity. They belong to the category of LDPC codes because they can be represented by Tanner graphs and be decoded in linear time using the SPA. IRA codes are also a generalization of RA codes, by allowing the variable nodes to have different degrees. IRA codes can be analyzed using the techniques developed for RA codes with some adaptation.

Thus IRA codes are a class of simple yet good linear block codes that keep the advantages of Turbo codes, LDPC codes and RA codes. They have linear encoding complexity and a small decoding threshold, and yet are simple to construct and analyze. These advantages of IRA codes make them good candidates for serial concatenation to study the error floor problem.

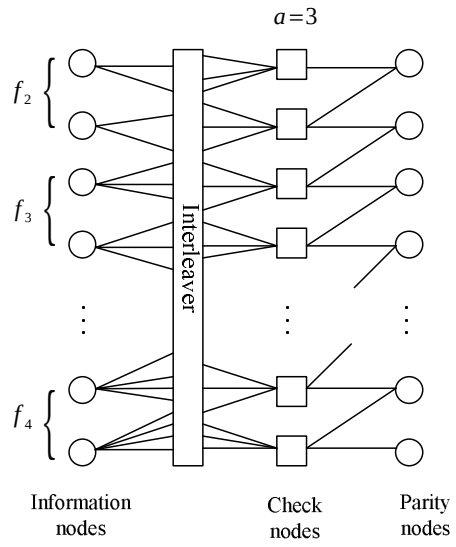


Figure 2.1: The Tanner graph of an example IRA code (from [6]). Three groups of variable nodes, f_2, f_3 and f_4, are shown. The check nodes have left degree $a = 3$.

2.2 Structure of IRA Codes

An RA code has two components: a repeater followed by an accumulator. The connections between the repeater and the accumulator are specified by an interleaver. Because of its simple structure, RA codes are very simple to analyze.

IRA codes are improved over RA codes by allowing the variable nodes to have different degrees (repeat indices). Figure 2.1 from [6] is the Tanner graph of an example IRA code. The first column is the information node sequence, the second is the check node sequence and the third is the parity node sequence. For systematic IRA codes, the information nodes and parity nodes are to be transmitted over the channel. The interleaver specifies the connections between the information nodes and the check nodes.

In the parity node sequence, the ith parity node is connected to the ith check

node and the $(i-1)$th parity node. The parity nodes accumulate the check nodes up to the last parity node. Thus the parity node sequence forms an accumulator.

The following is a summary of IRA code encoding and degree distribution design procedure from [6]. We are going to propose the design principle of SC-IRA codes based on the methodology from [6]. The encoding complexity of IRA codes grows linearly with code block size N. Suppose it is an (N, K) systematic IRA code, and there are N_c check nodes and N_c parity nodes. The encoding procedure is as follows:

$$x_i = x_{i-1} + \sum_{j=1}^{a} v_{(j-1)a+i}, \tag{2.1}$$

for $i = 1, 2, ..., N_c$, where x_i is the ith parity node and $v_{(j-1)a+i}$ is the jth information node connected to the ith check node. The summation is performed at each check node, which sums up all the information nodes that are connected to that check node.

IRA codes are defined with parameters $(f_2, f_3, ..., f_{d_v}; a)$, where f_i is the fraction of the information nodes that have a degree of i, and a is the left degree of the check nodes (see Figure 2.1). The degree of a node means the number of variable nodes that it is connected to. All the IRA codes in this thesis have a constant check degree $a = 8$.

The number of edges emanating from information nodes must equal the number of left-side edges emanating from the check nodes. Kf_i is the number of nodes of degree i, so Kif_i is the number of edges from information nodes of degree i. Therefore we have

$$\sum_{i=0}^{d_v} Kif_i = aN_c. \tag{2.2}$$

Then using Equation (2.2), the rate of an IRA code is

$$\begin{aligned} R &= \frac{K}{K + N_c} \\ &= \frac{a}{a + \sum if_i}. \end{aligned} \tag{2.3}$$

Another representation of the code parameters is the degree sequence pair (λ, ρ), where $\lambda_i \in \lambda$ is the fraction of edges that are connected to information nodes of degree i, and $\rho_i \in \rho$ is the fraction of edges that are connected to check nodes of degree i. The degree generating sequences are defined as

$$\lambda(x) = \sum_i \lambda_i x^{i-1}, \tag{2.4}$$

and

$$\rho(x) = \sum_i \rho_i x^{i-1}. \tag{2.5}$$

(λ, ρ) is called the degree distribution. The relationship between f_i and λ_i is

$$f_i = \frac{\lambda_i / i}{\sum_j \lambda_j / j}. \tag{2.6}$$

2.3 IRA Code Design

IRA codes are channel capacity approaching codes. IRA codes with infinite code sizes can approach the Shannon limit arbitrarily close on the binary erasure channel, and perform within a few tenths of a dB away from the channel capacity on the AWGN channel. Good finite length IRA codes should have a small threshold and a low BER in the higher SNR region (i.e., lower error floor). A low threshold means the channel values begin to converge to the true codewords at low SNR. The threshold is determined by the code's degree distribution. It can be analyzed on the AWGN channel using linear programming.

The problem of finding a good degree sequence for IRA codes can be formulated in this way [6]: maximize the code rate R under the condition that the mean of the log-likelihood ratio (LLR) messages from check nodes to variable nodes goes to infinity.

Table 2.1: Two degree sequences of IRA codes. The first degree sequence is used by the component IRA codes of SC-IRA codes. The second is used by the equivalent rate IRA codes.

a	8	8
λ_3	0.875	0.253
λ_4	0.023	
λ_6	0.102	
λ_{11}		0.081
λ_{12}		0.327
λ_{46}		0.185
λ_{48}		0.154
Rate	$\frac{1}{\sqrt{2}}$	$\frac{1}{2}$

From Equations (2.3) and (2.6), we have

$$\begin{aligned} R &= \frac{a}{a + \sum i f_i} \\ &= \frac{a}{a + \frac{1}{\sum_j \lambda_j / j}}. \end{aligned} \tag{2.7}$$

Thus we have to minimize $\sum i f_i$ or maximize $\sum_j \lambda_j / j$, under the condition that the mean of the LLR messages from check nodes to variable nodes goes to infinity. For more details please see [6].

The rate of the SC-IRA codes we designed for simulation is 1/2. To achieve this rate, we assigned equal rates of $1/\sqrt{2}$ to the two component IRA codes. The degree sequence we designed for the component IRA codes is in the left column in Table 2.1. The degree sequence we used for equivalent IRA codes simulation is in the right column in Table 2.1, which is from [6]. Note that the maximum degree for the rate $1/\sqrt{2}$ degree sequence is 6, while for the rate 1/2 degree sequence it is 48. For higher rate codes, a smaller maximum degree is necessary in order to have a smaller threshold.

The BER performance at higher SNR is related to the Tanner graph of the code. A good Tanner graph generally has a smaller number of short loops. It can be shown

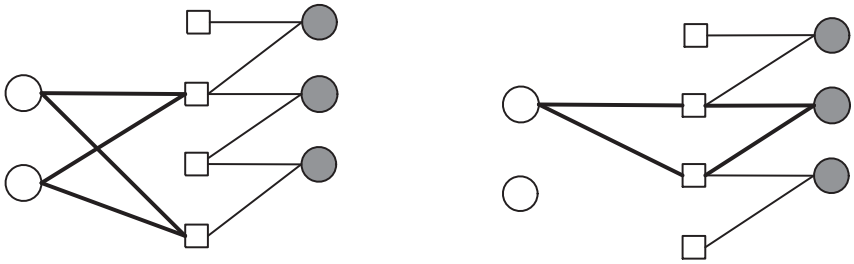


Figure 2.2: The two kinds of loops in IRA codes (bold lines).

that the BER performance of a code at higher SNR is no longer determined by the code's minimum distance, but by the inter-connected short loops in the Tanner graph. Because of the short loops, the variable nodes in the short loops (also known as the stopping sets [17]) are likely to receive positive feedback from themselves, and the messages in the stopping sets become correlated. This violates the assumption of the SPA algorithm that all the messages are independent and identically distributed (i.i.d.). As a consequence, the messages tend to converge to pseudo codewords instead of the true codeword. This is the cause of the well known error floor problem.

Figure 2.2 shows two kinds of possible loops in IRA codes. The bold lines in the left figure form a length 4 loop between the information nodes and the check nodes. To avoid this kind of loop, no two variable nodes should be connected to the same two check nodes. The bold lines in the right part of Figure 2.2 form a length 4 loop between one information node, two check nodes and one parity node. This loop pattern is unique to IRA codes, because in IRA codes, the parity nodes in the accumulator are always connected to their two adjacent check nodes. Therefore, the variable nodes in IRA codes should not connect to two adjacent check nodes.

We try to eliminate as many length 4 loops as possible from the component IRA codes we use in the SC-IRA codes. We design three component IRA codes with code

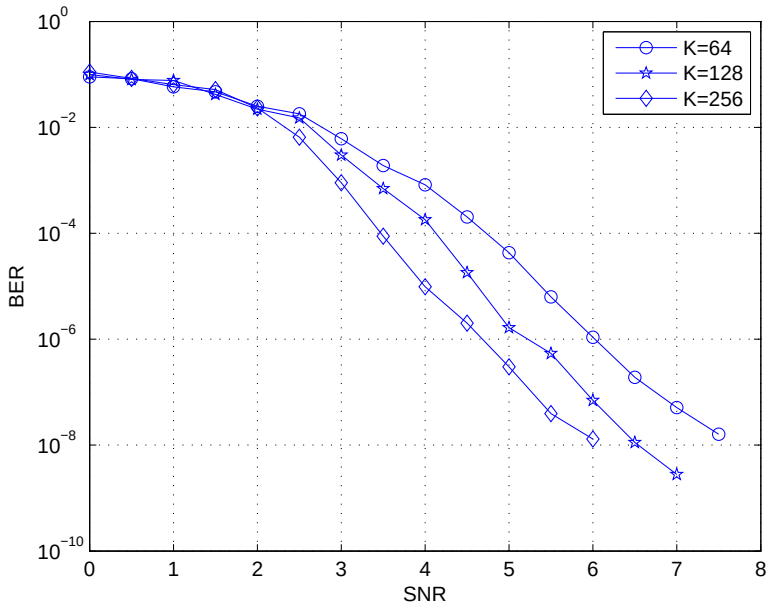


Figure 2.3: The BER performance of three component codes. The code lengths are 64, 128 and 256, respectively. The code rate is $1/\sqrt{2}$.

lengths of 64, 128 and 256, respectively. In the length 256 code, there are no length 4 loops; in the length 128 code, there are 2 remaining length 4 loops; in the length 64 code, there are 6 remaining length 4 loops. Figure 2.3 shows the performance of the three component IRA codes we use for our SC-IRA codes.

2.4 Introduction to SC-IRA Codes

Like LDPC codes, IRA codes have the error floor problem, too. The BER curves of IRA codes obtained by iterative SPA decoding are the so called "waterfall" curves. These curves have a "waterfall" region where the BER drops rapidly. Then the bottom of the "waterfall" is reached, where the BER slope decreases; this region is referred to as the error floor region. In LDPC and IRA codes, the error floor problem

is not caused by low weight codewords as in Turbo codes, but rather by interconnected loops in the Tanner graph.

The iterative SPA decoding algorithm for IRA codes assumes the code graph is tree like and there are no loops in the graph. Under this assumption, the LLR messages passed back and forth on the graph are i.i.d. The messages on such a graph will converge to their true values after a number of iterations, given that the SNR is above the threshold. But any practical code we design is of finite length, so there are inevitably loops in the code's graph. This violates the assumption of the iterative decoding algorithm. After a number of iterations, the nodes in the graph begin to receive information that originates from themselves, which means they are correlated. This phenomenon is also known as "postive feedback". With correlated information on the graph, or "positive feedback", the LLRs are more likely to converge to pseudo codewords.

The pseudo codewords in the higher SNR region in IRA codes are caused by interconnected sets of short loops on the Tanner graph called stopping sets. The size of a stopping set is defined as the number of variable nodes in the set. The larger the number of small stopping sets is in the Tanner graph of an IRA code, the more correlated the information will become, and the more likely they will converge to pseudo codewords. Besides designing IRA codes with fewer small stopping sets, another way to mitigate this problem is the serial concatenation of IRA codes. In SC-IRA codes, two component IRA codes are used to encode a two dimensional information block, first row by row and then column by column. The Tanner graphs of the two component codes cooperate in the iterative decoding algorithm. Messages are updated on the two different graphs successively. If a set of nodes are from short loops of one Tanner graph, the interleaver will make sure that they are not on short

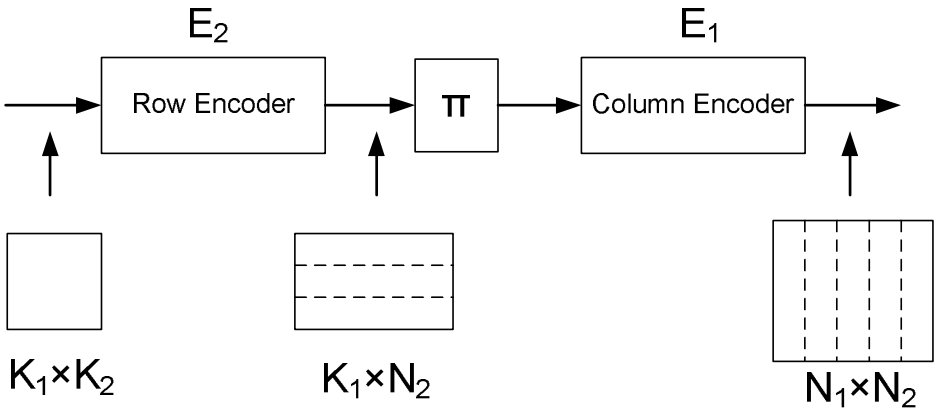


Figure 2.4: The encoder structure of an SC-IRA code.

loops of the other Tanner graph. If the messages from short loops of one graph begin to receive positive feedback, they can be corrected by the message update on the other graph. Thus they are less likely to become correlated. In other words, they become correlated more slowly than on only one Tanner graph.

Figure 2.4 shows the encoder block diagram of an (N_1N_2, K_1K_2) SC-IRA code. The information bits are arranged into a $K_1 \times K_2$ two-dimensional block S. The encoder consists of a column encoder E_1, an interleaver π, and a row encoder E_2. E_1 and E_2 are (N_1, K_1) and (N_2, K_2) systematic IRA codes with code rates R_1 and R_2, respectively. The size of the interleaver π is $K_1 \times N_2$.

E_2 connects directly to the source input S (outer code), and E_1 connects to the channel (inner code). To encode, first apply E_2 to each row of S. This results in a block of $K_1 \times N_2$, and each row is a codeword of E_2. Then this block is interleaved by π. Finally E_1 is applied to each column of the interleaved block. The final codeword is of size $N_1 \times N_2$. The code rate for the serial code is $K_1K_2/N_1N_2 = R_1R_2$.

Figure 1.3 is an $N_1 \times N_2$ codeword of an SC-IRA code. For systematic SC-IRA

codes, there are four blocks in a codeword: Block 1 is the information sequence. Block 2 is the check nodes for the row component code. Block 3 is the checks of the column component codes. Block 4 is the check-on-checks nodes. If there is no interleaver between the concatenation, block 4 consists the row check nodes for the column codes and also the column check nodes for the row codes. However, if an interleaver is used, block 4 consists only column check nodes, because after interleaving, the rows of the block are scattered and thus they are not row codewords. After applying the column encoder, they do not satisfy the row check equations any longer.

2.5 SC-IRA Code Design

This section addresses the problem of designing good SC-IRA codes. The performance of SC-IRA codes is affected by the degree distribution, the component codes and the interleaver. We will discuss these three factors one by one.

There are several existing methods to design good degree distributions for LDPC-like codes, such as curving fitting [25], EXIT chart analysis [27], linear programing [6] and density evolution [3]. But these methods design good degree distributions in the sense of small thresholds, while they give no information about the high SNR performance.

From the construction of the one-graph decoding (to be discussed in Section 3.3), the Tanner graphs of the two component codes are incorporated into the single Tanner graph for the SC-IRA code. SPA still updates messages on the basis of the Tanner graphs of the two component codes. The performance of the SC-IRA code at low SNR is related to the two component codes. Therefore, in order to minimize the gap between the SC-IRA codes and the equivalent single IRA codes, we design the degree

distributions for the two component codes using the linear programming method [6], which is described in Section 2.3. This method gives degree sequences with small thresholds.

Because our component codes have $K = 64, 128, 256$, which are very short, their true degree distributions are only rough approximations of the target degree distributions. The granularity will improve as the code length increases. Simulation results in Figure 3.5 show that the waterfall region of SC-IRA codes with the above three component codes and with the one-graph decoding begins at around 1.6 dB.

In SC-IRA codes there is a smaller number of short loops than in their equivalent-length and equivalent-rate single IRA codes. When designing LDPC or IRA codes, we have to watch out not to assign edges that form short loops. For example, we try to avoid double or parallel edges, length 4 loops and length 6 loops, etc. The longer the loop length is, the more difficult it is to eliminate those loops. In our single IRA code design, we eliminate length 4 loops only, and do not attempt to eliminate loops of longer lengths. But in SC-IRA codes, the number of length 4, 6 and 8 loops are more controllable. If the numbers of length 4 loops in component codes E_1 and E_2 are l_1 and l_2, respectively, then the total number of length 4 loops in the SC-IRA code is $l_1 N_2 + l_2 K_1$. This relation holds for length 6 and 8 loops, too. Thus the number of short loops in SC-IRA codes grows linearly with the component code sizes. But in equivalent single IRA codes, it grows linearly with the product of the component code sizes. It grows faster with the code size also because of higher degrees (See Table 2.1, where the maximum degree for the single IRA codes is 48, while that of the component codes for SC-IRA codes is 6). Therefore, the loop length spectrum in SC-IRA codes is shifted to higher length regions than in equivalent single IRA codes. This statement is formulated as the following proposition.

Proposition 2.5.1 *In SC-IRA codes, the number of length 4, 6 and 8 loops grows linearly with the lengths of their component codes.*

Proof Suppose an SC-IRA code has two component codes E_1 and E_2, which are (N_1, K_1) and (N_2, K_2) systematic IRA codes, respectively. Then choose an interleaver such that in the SC-IRA two-dimensional block, variable nodes of the same E_1 codeword are not mapped to variable nodes of the same E_2 codeword. Because E_1 and E_2 are connected in a two-dimensional manner, the loops jointly formed by E_1 and E_2 must have:

1. at least 4 variable nodes and 4 check nodes if there is no interleaver between E_1 and E_2;

2. more than 4 variable nodes and 4 check nodes if there is an interleaver between E_1 and E_2.

Since in our SC-IRA codes, there is always an interleaver between E_1 and E_2, the loops jointly formed by E_1 and E_2 must have more than 4 variable nodes and 4 check nodes. Thus the lengths of these loops are greater than 8.

Now consider the length 4 loops. Suppose there are l_1 length 4 loops in E_1 and l_2 length 4 loops in E_2. There are N_2 of the E_1 codewords and K_1 of the E_2 codewords in an SC-IRA code. Then the total number of length 4 loops in the SC-IRA code is $l_1 N_2 + l_2 K_1$, because the loops jointly formed by E_1 and E_2 have lengths more than 8. Thus the number of length 4 loops in the SC-IRA code grows linearly with the lengths of the component codes.

For length 6 (8) loops, suppose l_1 and l_2 are the number of loops of length 6 (8) in E_1 and E_2. Again because the loops jointly formed by E_1 and E_2 have lengths more

than 8, the relation $l_1 N_2 + l_2 K_1$ also holds for length 6 (8) loops. This completes the proof of the proposition. ∎

2.6 Interleaver Design

This section is divided into two parts: the interleaver design for SC-IRA codes with the serial decoder and the interleaver design for SC-IRA codes with the one-graph decoder. Because the optimal interleavers for these two kinds of decoders are different, separate optimization techniques are applied to the interleaver design.

2.6.1 Interleaver Design for the Serial Decoder

In [14], the stopping sets information is used to design the interleaver. The interleaver is designed to prevent mapping stopping set error events of one component code into stopping set error events of the other component code. With the aid of such an interleaver, the serial concatenation structure helps improve the performance at high SNR.

The starting interleaver in [14] is a good random interleaver picked from a set of random interleavers. In our design, instead of a random interleaver, we start with an S-random interleaver [23]. The S-random interleaver randomly maps the bits over the whole block, while ensuring that bits that are within S positions from each other are mapped at least S positions apart. Then we optimize the S-random interleaver with the stopping sets information. Simulation results show that the optimized S-random interleaver gives better BER performance than the optimized random interleaver. Section 3.2 presents our simulation results.

2.6.2 Interleaver Design for the One-graph Decoder

In the one-graph decoding, the SC-IRA codes are decoded on a single Tanner graph derived from the two component Tanner graphs; thus the degree distribution of the overall SC-IRA Tanner graph also plays a role in determining the threshold. Besides the degree distributions of the component codes, the degree distribution of a SC-IRA code also depends on the interleaver π. The degree of a variable node in an SC-IRA code satisfies $d_{i+j} = d_i^1 + d_j^2$, where d_i^1 and d_j^2 are the degrees of nodes in the two component codes. The interleaver π controls the mapping between d_i^1 and d_j^2.

The stopping sets technique [17] for interleaver optimization [14] does not work in the one graph decoding method. The reason is as follows: when we use the one graph decoding on the single Tanner graph of a SC-IRA code, both the row and column codes are decoded simultaneously. There is no explicit extrinsic information exchange between them in the iterative decoding.

Therefore, instead of the interleaver optimized with stopping sets, we use the S-random interleaver [23] for the SC-IRA codes with the one-graph decoding. Simulation results show that the S-random interleaver gives good BER performance, while the S-random interleaver optimized with stopping sets gives negligible improvement.

An additional criterion of the interleaver design is to avoid introducing additional length 4 loops when concatenating the two component codes. As in Figure 2.5, suppose two nodes A and B are in the same row and they are connected to a common row check node C_1 from the row code E_2. If A and B are mapped into the same column and they again have a common check node C_2 from the column code E_1, then for the SC-IRA code, variable nodes A and B and check nodes C_1 and C_2 form a length 4 loop. The S-random interleaver should not allow such mappings. One way to ensure

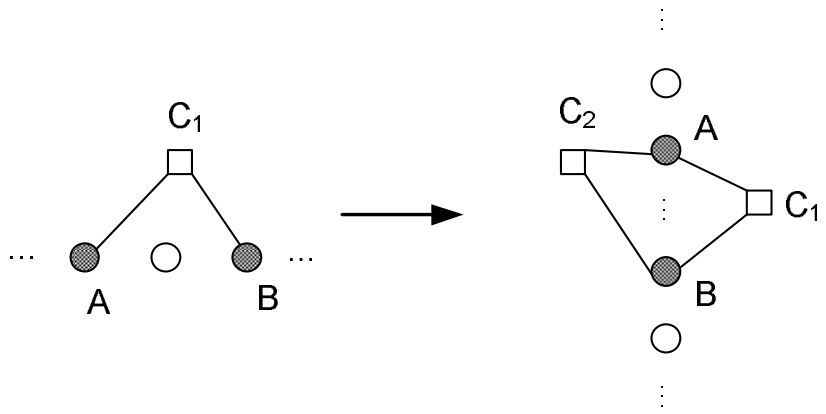


Figure 2.5: Possible loops in SC-IRA codes. These loops are formed in the concatenation of the two component codes.

this is not to map nodes in the same row into the same column. But due to the limited number of valid mappings, a looser condition is not to map variable nodes with the same row check nodes into positions with the same column check nodes. The looser condition is used to design the interleavers for the simulations in Chapter 3.

Since there are three factors affecting the degree distribution of SC-IRA codes simultaneously, to make the analysis a little easier, we choose a good interleaver π with a large S, and then fix the interleaver. When π is fixed, we can obtain the degree distribution of the SC-IRA code by counting the number of nodes with certain degrees. Since the rate of the SC-IRA code is the product of the rates of the two component codes, the rate is already fixed once the component codes are designed. Thus the linear programming method in [6] cannot be applied (this method tries to maximize the code rate). Instead, we try to find the best one-graph degree distribution by varying the degree sequences of the two component codes, such that the BER converges to zero at the largest SNR.

This method generates codes which have smaller thresholds than the codes with separately optimized component degree distributions. Figure 2.6 shows the simulation results of separately optimized component degrees versus four jointly optimized degrees as described above. The same S-random interleaver is used in these simulations. The four jointly optimized component degree sequences are the ones with the largest noise variance at which the BER converges to zero. The threshold gain with this method is about 0.2 dB. Unfortunately, the codes also have worse error floors. Since in this thesis our main purpose is to search for codes with low error floors, we stick to those component codes whose degree distributions are separately optimized using the linear programming method.

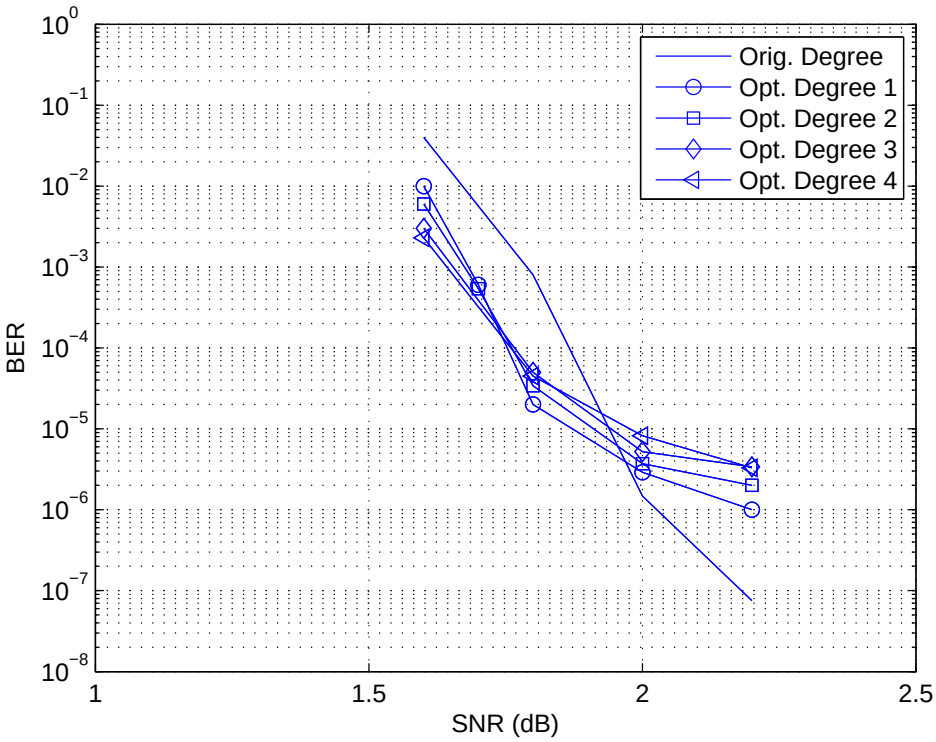


Figure 2.6: Simulation results for separately optimized component degree sequences and 4 best jointly optimized component degree sequences. Note the trade off between the threshold and the error floor.

Chapter 3

Decoding of SC-IRA Codes

In Chapter 2, various aspects of IRA and SC-IRA code design are discussed. In this chapter, we discuss the decoding of SC-IRA codes. For a particular code, there is usually more than one possible decoding scheme. People who use the code can choose the decoding scheme according to their specific needs. For example, in DVB-S2, a serial concatenation of LDPC codes and BCH codes is used. The standard only specifies how the encoding is done, but leaves the decoding procedure open. It is up to the DVB receiver vendors to devise their own decoder and compete with each other.

Decoding of SC-IRA codes is a similar story. It is an open challenge with different possible decoding schemes. In [14], Cheng et. al. propose a serial decoding scheme, and they simulate the serial decoder to compare with equivalent single IRA codes. There is an expected cross over in the BER curves of IRA and SC-IRA codes. Therefore the serial decoding scheme gives a lower error floor compared to the equivalent IRA codes. However, SC-IRA codes also have a severe penalty in the waterfall region: about 2 dB performance loss compared to equivalent IRA codes. The penalty is due to the short component codes according to [14].

3.1 The Serial Decoder of SC-IRA Codes

The decoder of the SC-IRA code in [14] consists of a column decoder and a row decoder connected by an interleaver and a de-interleaver. The two component decoders for the serially concatenated system employ SPA on the Tanner graphs of the two component codes iteratively, and exchange extrinsic information between them.

Figure 3.1 is the block diagram of the serial decoder. The serial decoder consists of a column decoder D_1, a row decoder D_2, an interleaver π and a de-interleaver π^{-1} [14]. Input to D_1 is the received channel values and the interleaved version of the extrinsic information from D_2. Input to D_2 is the de-interleaved version of received channel values and the de-interleaved extrinsic information from D_1. Note that D_2 only decodes the systematic portion of the column codewords, i.e., the first K_1 rows of the two dimensional codeword block, as the last $N_1 - K_1$ rows are not row codewords because of the interleaver.

Before decoding, the extrinsic information from D_2 is initialized to zero. In one iteration, D_1 first decodes the received channel values column by column for i iterations. Next π^{-1} de-interleaves both the received channel values and the extrinsic information from D_1. Then D_2 decodes the de-interleaved channel values row by row for another i iterations. And the last step is that D_2 passes the interleaved extrinsic information back to D_1.

The SC-IRA code system achieves a lower error floor compared to single IRA codes with equivalent block length and code rate, and still has linear encoding complexity. However, due to the short code length effect (the component codes in [14] were of length 64, 128 and 256), the length 16,384, rate 1/2 SC-IRA code suffers about 1.7 dB performance loss in the waterfall region compared to equivalent single IRA codes.

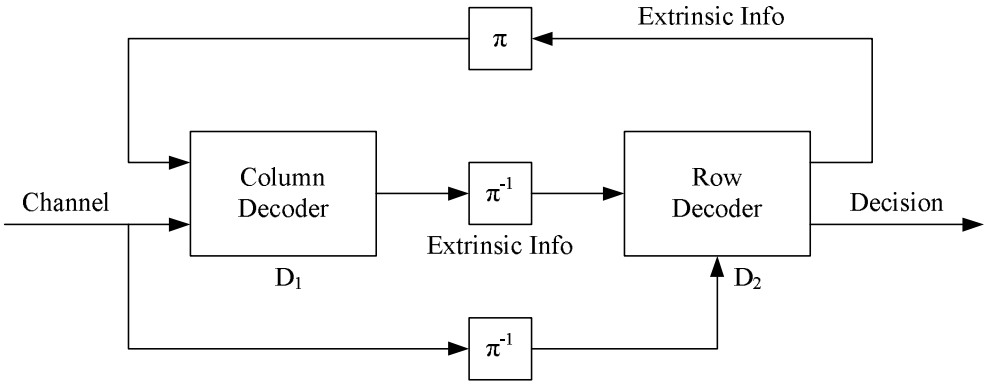


Figure 3.1: The serial decoding of SC-IRA codes.

One factor that affects the performance of the serial decoder is the iteration scheme scheduling for the two component decoders. Simulation results show that different iteration schemes can result in BER difference up to two orders of magnitude in the error floor region. The next section discusses how the Extrinsic Information Transfer (EXIT) chart technique can be used to optimize the iteration scheme.

3.2 EXIT Chart Analysis of the Serial Decoder

Because the row and column decoders use SPA to decode the corresponding component codes, D_1 or D_2 can employ more than one inner iteration on each row or column code before passing extrinsic information on to the other decoder. There are also multiple outer iterations between D_1 and D_2. Thus, there are many possible combinations of inner iteration I_i and outer iteration I_o. The choice of (I_i, I_o) is important to the BER performance of the codes at higher SNR. It will affect the message update rules between the variable nodes and the check nodes. The reason is as follows.

For LDPC-like codes, variable nodes in stopping sets [17] converge to pseudo-codewords with a higher probability than the variable nodes that are not in stopping sets. Due to the pseudo-codewords, once a group of nodes begin to converge to a pseudo-codeword, more decoding iterations will not lead to fewer decoding error bits. By using two component codes and an interleaver between them, nodes in stopping sets of one component code will not be mapped to stopping sets of the other component code, and thus SPA decoding of these nodes on the second component code will stop them from converging to pseudo-codewords. During the inner iterations of the component decoders, at the first stage nodes in stopping sets are exchanging information with other nodes, but after a certain number of iterations, they will receive information originated from themselves (positive feedback) due to the stopping sets. Thus we must choose a proper I_i before severe positive feedback happens and then pass the messages on to the other component decoder.

In [14], a fixed ($I_1 = 10, I_2 = 10$) iteration scheme is used. In this thesis, in order to get the best iteration scheme, the EXIT chart technique proposed in [27] is used to explore the input-output extrinsic information transfer property of the row and column decoders. The output extrinsic information I_e^{out} of a decoder is measured as a function of input extrinsic information I_e^{in}. In our two-decoder iterative decoding structure, I_e^{out} of the outer decoder is I_e^{in} of the other decoder and vice versa. At a given SNR, only if there is a zigzag path between the extrinsic information transfer curves of the two decoders, can the LLR values converge to their true values. I_i should be as small as possible in order to take advantage of extrinsic information, provided that the LLRs can converge. For our IRA decoders, EXIT chart analysis shows that a path between EXIT curves opens after 5 inner iterations (See Figure 3.2). Thus ($I_i = 5, I_o = 10$) is the best iteration scheme. Simulation results show

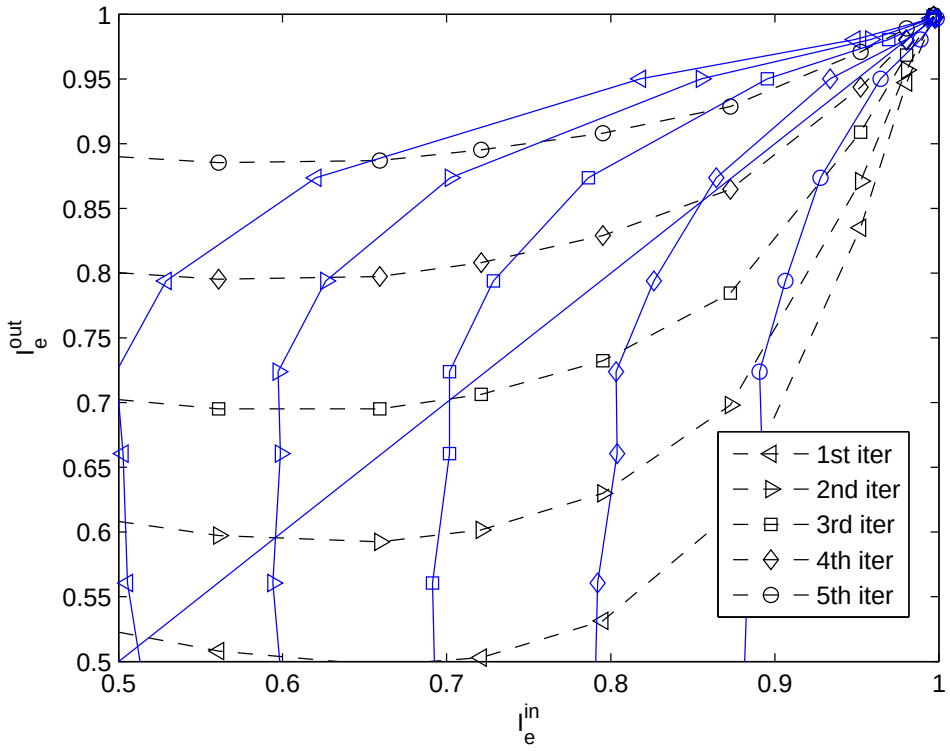


Figure 3.2: The EXIT chart of the two component codes. The solid blue curves are from the 1st component decoder, and the dashed black curves are from the 2nd component decoder.

that the different iteration schemes can result in BER difference as big as two orders of magnitude. The simulation results of an 128×128 serial code using different iteration schemes are given in Figure 3.3. Five iteration schemes are selected such that $2I_i \times I_o = 200$.

3.3 The One-graph Decoder of SC-IRA Codes

We find that once the SC-IRA code is constructed with a row code, a column code and an interleaver, the connections between variable nodes and check nodes are fixed. There is an H matrix corresponding to each SC-IRA code. Thus, an SC-IRA code

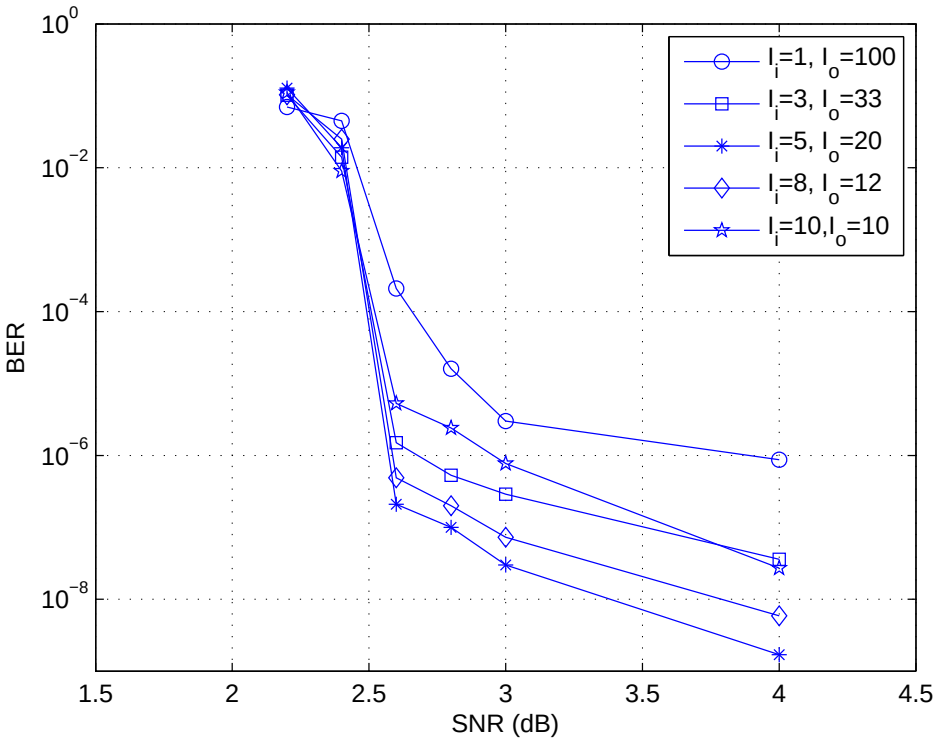


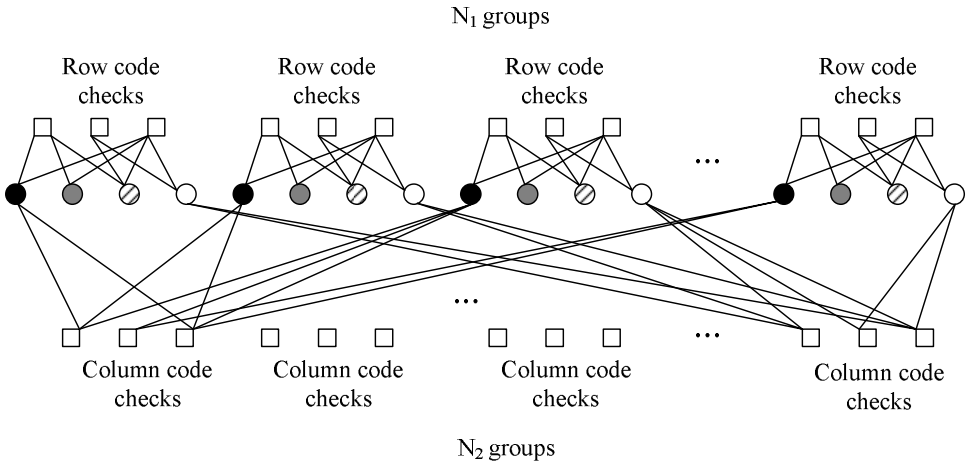
Figure 3.3: Simulation results of SC-IRA codes with the serial decoder. These curves were obtained with different inner and outer iteration schemes. All the schemes have a total of $2I_i \times I_o = 200$ iterations.

has a unique Tanner graph that can be recovered from the two Tanner graphs of the component codes, together with the interleaver.

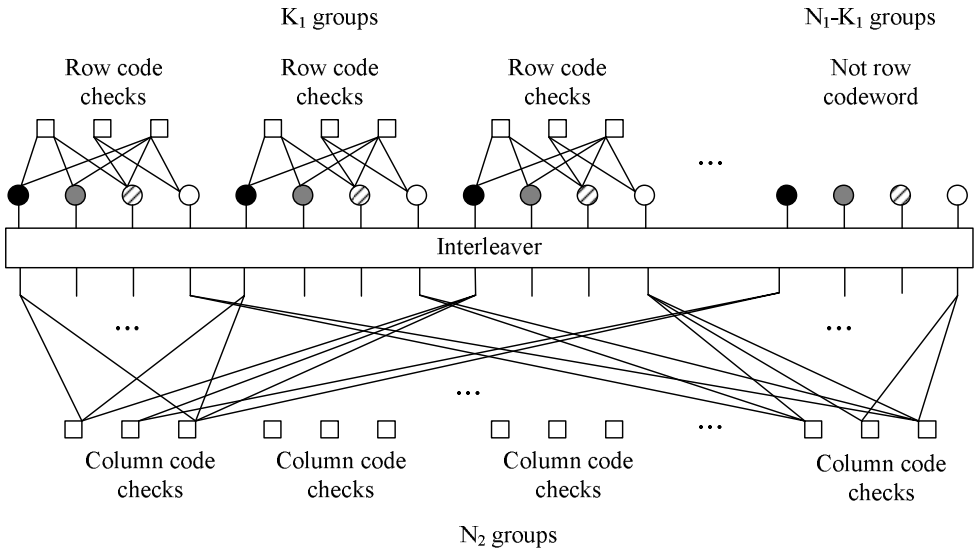
In contrast to regular product codes where each row and each column is a codeword, rows of the SC-IRA code are not codewords after row and column encoding because of the interleaver between the two component codes. Figure 3.4 shows the two cases with and without the interleaver. Part (a) of Figure 3.4 is the SC-IRA code without the interleaver. It is the same as regular product codes. The codeword is divided into N_1 groups. Each group is a row codeword. For $i = 1, 2, ..., N_2$, the ith elements of each group constitute the ith column codeword. Note that the last $N_2 - K_2$ column codewords are composed of row check nodes and checks-on-checks nodes.

Part (b) of Figure 3.4 is the SC-IRA code with the interleaver between the two component codes. Because there is an interleaver gain, in our code design we always incorporate the interleaver. After applying E_2 to the information block, the resulting block is passed through the interleaver π. Thereafter each row of the block is no longer a valid codeword of E_2 because of the randomness introduced by π, although all the row check nodes of the interleaved bits still satisfy the parity checks. Next E_1 is applied column by column to the block, producing a $N_1 \times N_2$ codeword block. Row $K_1 + 1$ to row N_1 are not valid row codewords, and thus they do not satisfy the check equations of E_2. Therefore, with the interleaver π, row $K_1 + 1$ to row N_1 only have check nodes from E_1, but no check nodes from E_2.

For each of the first K_1 rows, there are $N_2 - K_2$ check nodes from E_2; for each of the N_2 columns, there are $N_1 - K_1$ check nodes from E_1. Thus for the overall $N_1 \times N_2$ SC-IRA code, there are $(N_1 - K_1)N_2 + (N_2 - K_2)K_1$ check nodes.



(a)



(b)

Figure 3.4: The one-graph decoding of SC-IRA codes with and without the interleaver. (a) When there is no interleaver, groups of 4 adjacent nodes form N_1 row codewords. The same color nodes form N_2 column codewords. (b) When the interleaver is present, the right-most $N_1 - K_1$ groups of variable nodes are not row codewords. There are two sets of check nodes: the upper row of check nodes are from row codewords, and the lower row of check nodes are from column codewords.

To recover the Tanner graph for the SC-IRA code, we arrange the two dimensional codeword into a single row in a row-by-row manner. For the first K_1 sub-blocks of length N_2 in the single row, there are $N_2 - K_2$ check nodes connected to each of them according to the row code E_2. For example, in Figure 3.4, each group of four circles is a row codeword; there are K_1 groups of them. The last $N_1 - K_1$ groups are not row codewords and thus they have no row check nodes. Then π interleaves the first K_1 sub-blocks of length N_2, i.e., the first $K_1 \times N_2$ bits in the row. When the column encoder E_1 is applied, variable nodes that are N_2 positions apart are in the same column in the two dimensional block, and thus they form a column codeword. There are N_2 groups of such column codewords, each of which has $N_1 - K_1$ checks nodes from E_1. In Figure 3.4 for example, the black circles form a column codeword with 3 check nodes, while the gray circles form another column codeword with 3 check nodes, and so on.

Now that we have the Tanner graph for the SC-IRA code, we can decode the codeword on the single Tanner graph using SPA, as in LDPC codes.

Simulation results of the two decoding methods are given in section 3.4. As seen in Figure 3.5, the serial decoder has a gap of about 1.7 dB compared to equivalent single IRA codes because of the short component code effect. But one graph decoding avoids this problem by considering the whole block as a single codeword. It performs better than the serial decoding in both the waterfall and error floor regions.

The serial concatenation of IRA codes is also a method to design very long codes required in certain applications. Suppose we already have several good (N, K) IRA codes. By serially concatenating two of them, we can get a code of length N^2, while maintaining the linear encoding structure. It can be generalized to the concatenation of n component codes, $n \geq 2$, which produces a serially concatenated code of length

N^n. We did not simulate the case when $n \geq 2$. Our conjecture is that it would have a lower error floor but there would be a greater penalty compared to single IRA codes at low SNR.

3.4 Simulation Results

We designed several rate 0.5 SC-IRA codes as well as their equivalent single IRA codes with the same rate and block size. Both the serial decoder and the one-graph decoder were simulated over the Binary-Input AWGN channel for comparison. For the single IRA codes and the SC-IRA codes with the one-graph decoder, the maximum number of iterations is set to 200, while for the SC-IRA codes with the serial decoder the maximum number of iterations is set to 5 inner iterations and 20 outer iterations. The iteration scheme is optimized with EXIT charts as described in Section 3.2.

The three SC-IRA codes are: the 64×128 code, the 128×128 code, and the 128×256 code, all of which have rate 0.5. The component codes are systematic IRA codes with rate 0.707, a maximum variable degree of 6 and a check degree of $a = 8$. They use the degree sequences in Table 2.1. The performance of the $(90, 64)$ and $(181, 128)$ component codes is shown in Figure 4.4. For comparison, we also simulated two single IRA codes of lengths 8192 and 16384, which have the maximum degree $d_v = 48$ and code rate 0.5. Their degree sequences are optimized using linear programming.

SPA assumes that the Tanner graphs of the codes have no loops. But the actual codes violate this assumption. It is known that the performance of the LDPC codes at high SNR (i.e., in the error floor region) is no longer dominated by the code's minimum distance, but rather by interconnected short loops. Thus, in order to lower

BER in higher SNR region, we try to remove as many length 4 loops as possible. In the codes we designed for simulation, there are six length 4 loops in the length 64 IRA codes, two length 4 loops in the length 128 IRA codes, and no length 4 loops in the length 256 IRA codes. In the equivalent single IRA codes, there are no length 4 loops. As in SC-IRA codes, we do not remove loops of longer lengths.

Figure 3.5 shows simulation results comparing the performance of the serial decoder and the one-graph decoder. It also shows the performance of two equivalent single IRA codes. The SC-IRA codes with the one-graph decoder have lower error floors than their equivalent single IRA codes. The BER curves of SC-IRA codes have steeper slopes, which also indicates that the error floor would be at a lower BER. The one-graph decoding gives about 0.7 dB gain compared to the serial decoding in the waterfall region. But even with the one-graph decoding, the SC-IRA codes still have about 0.8 dB performance loss in the waterfall region compared to single IRA codes. The reason for the gap is that the single IRA codes are optimized to give the smallest threshold, while in SC-IRA codes only the component codes are optimized.

We did not simulate longer codes because as the code length increases, the error floor drops further and it is more difficult to observe and compare error floors using software simulation.

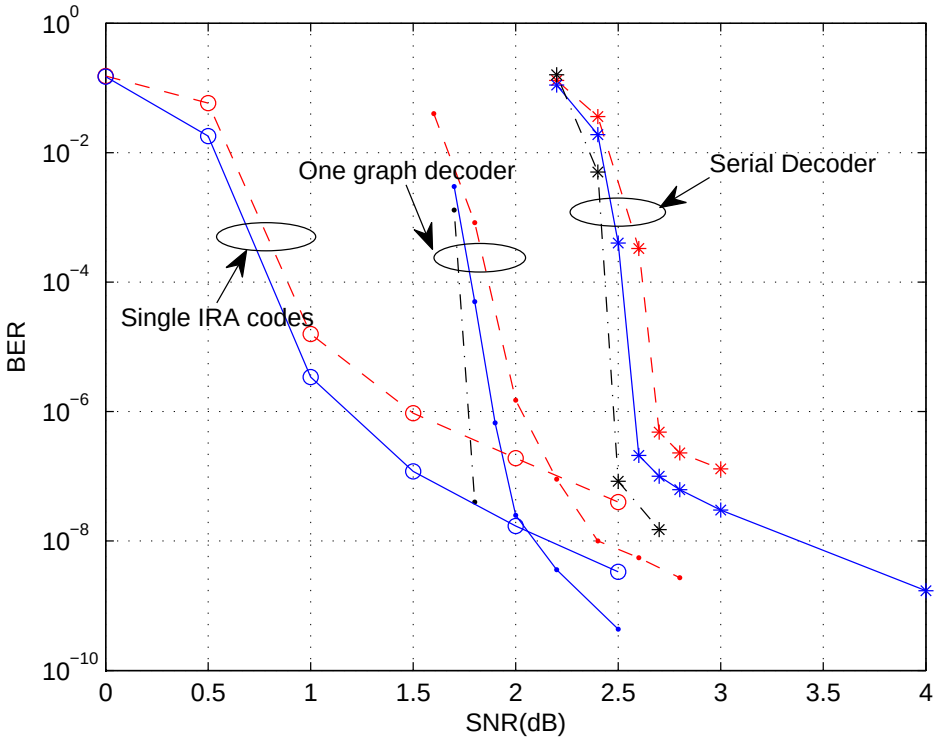


Figure 3.5: Simulation results of IRA codes and SC-IRA codes. All codes are of rate 1/2. The red dashed, blue solid and black dash-dotted curves are the BER performances of codes of length 8192, 16384 and 32768, respectively.

Chapter 4

Performance Analysis of IRA and SC-IRA Codes

Over the last decade, asymptotic results of LDPC codes have been extensively researched, for example, the maximum-likelihood decoding performance analysis [20], the asymptotic codeword weight distribution properties [21], the iterative decoding threshold based on density evolution [3] [22], etc. However, it is also of interest to study the properties of finite length LDPC and IRA codes. It is useful to do the accurate analysis of these finite length codes.

In this section we will analyze the ML bounds of finite length IRA codes and SC-IRA codes. The ML bound is the maximum likelihood decoding upper bound. Over the AWGN channel, the decoding complexity of ML decoding is proportional to 2^K, where K is the information block size. It has been proved that the exact ML decoding of linear block codes is NP-hard [16]. Because of the exponential computational complexity for large block size, the ML decoding can not be used in practical applications. Instead, there are some sub-optimal decoding algorithms, for example, the majority-logic decoding, the bit-flipping decoding and the Sum Product Algorithm (SPA). But compared to ML decoding, their performances are compromised in one

aspect or another.

Although for the AWGN channel, no practical ML decoding algorithm is available yet, the ML bound gives us an estimate of the code performance without hardware or software simulation. This is useful because it provides a criterion to compare different codes. Systematic code design methods can also be proposed based on the ML bounds that lead to good codes.

The ML bound is based on the Input Output Weight Enumerator (IOWE) of a code ensemble. IOWE reflects the average ensemble codeword distance spectrum, i.e., the relationship between the weights of the input information sequences and the weights of the output codewords.

In higher SNR region, most of the ML decoding errors are caused by a set of closest codewords. For linear block codes, the number of closest codewords is equivalent to the number of low weight codewords. Thus analyzing the codeword weight enumerator gives us information about the set of closest codewords, which again influences the code performance. The ML bound is tight at high SNR, and therefore the information obtained from the ML bound becomes more accurate in higher SNR region. In lower SNR region, the decoding errors also heavily depend on the non-closest codewords.

There are several versions of ML bounds, for example, the union bound, the Gallager bound and several variations of the Gallager bound [18], and the simple tight bound [19]. The tight bound gives a simple upper bound on frame-error and bit-error probability.

An IRA code can be represented by a Tanner graph. The Tanner graph is defined with a parameter list of $(f_1, f_2, ..., f_{d_v}, a)$ [6], where f_k denotes the fraction of variable nodes that have degree k; d_v is the maximum variable degree; a denotes the left degree of the check nodes, which in IRA codes is a constant for all check nodes. In

this section, we are going to first derive the IOWE of IRA codes and SC-IRA codes, and then compute the simple tight bound based on the IOWEs.

Suppose the code rate is R. The IOWE is denoted as $A_{\omega,h}$, with input weight ω and output weight h; the codeword distance spectrum is $A_h = \sum_{\omega=1}^{RN} A_{\omega,h}$. In this thesis, we use the tight bound [19] to get the upper bounds of the frame error rate (FER) and bit error rate (BER). Equations (4.1) through (4.4) are summarized from [19]. Define the normalized distance with respect to the all-zero codeword as $\delta = \frac{h}{N}$. Define the normalized codeword distance spectrum as $r(\delta) = \frac{\ln A_h}{N}$. Then as in [19] the FER is

$$P_e \leq \sum_{d=d_{min}}^{d_{max}} e^{-nE(\delta,\beta,\frac{E_c}{N_o})} \tag{4.1}$$

where

$$E(\delta,\beta,\frac{E_c}{N_o}) = -\frac{1}{2}\ln(1-\beta+\beta e^{-2r(\delta)}) + \frac{\beta\delta}{1-(1-\beta)\delta}\frac{E_c}{N_o} \tag{4.2}$$

and

$$\beta = \frac{1-\delta}{\delta}\left[\sqrt{\frac{E_c}{N_o}\frac{\delta}{1-\delta}\frac{2}{1-e^{-2r(\delta)}} + \left(1+\frac{E_c}{N_o}\right)^2 - 1} - \left(1+\frac{E_c}{N_o}\right)\right]. \tag{4.3}$$

This tight bound is valid only if β is a real number between 0 and 1. Thus, for β to be in the above range, $\frac{E_c}{N_o}$ has to satisfy the constraint:

$$(1-e^{-2\ln r(\delta)})\frac{1-\delta}{2\delta} \leq \frac{E_c}{N_o} \leq \frac{e^{2\ln r(\delta)}-1}{2(1-\delta)\delta}. \tag{4.4}$$

If $\frac{E_c}{N_o}$ does not satisfy Equation (4.4), β is set to 1. When $\beta = 1$, the tight bound recedes to the union bound. If we replace A_h with $\sum_{\omega=1}^{RN}\frac{\omega}{RN}A_{\omega,h}$ in the above FER bound, we get a tight BER bound.

Equations (4.1) and (4.2) need the IOWEs of IRA codes. We will derive the IOWE of IRA codes by decomposing the IRA codes into several components, analyzing one component at a time, and then integrating them. An IRA code is a serial

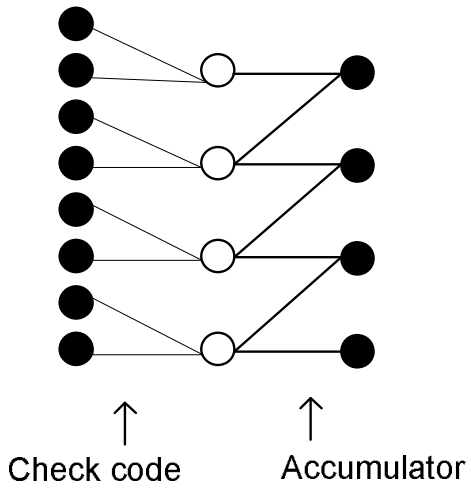


Figure 4.1: An example of the concatenation of a check code with $a = 2$ and an accumulator.

concatenation of three components: an irregular repetition code, a regular check code and an accumulator without puncturing. They are separated by interleavers of corresponding sizes. Figure 4.1 is an example of concatenating a check code with an accumulator. The check code has $a = 2$. So Figure 4.1 can also be viewed as a punctured accumulator with $a = 2$.

We derive the IOWEs for the above three components separately. Then using the uniform interleaver technique [24], we can get the IOWE of an IRA code by serially concatenating them. The uniform interleaver technique ensures the IOWE we derive is the ensemble average over all possible codes.

4.1 IOWE of Irregular Repetition Codes

Consider an (N, K) irregular repetition code. A node with repeat index k means the node is repeated k times. In other words, this node has k edges connecting to it. Let $n_k = K \times f_k$ be the number of variable nodes with repetition k, i.e., these n_k nodes are repeated k times each. Obviously, $\sum_{k=1}^{k=d_v} n_k = K$. It directly follows that for the sub-block n_k with repetition k, if input has weight ω_k, the output can only have weight $h_k = k\omega_k$, and there are $\binom{n_k}{\omega_k}$ possible combinations. Thus its IOWE is

$$A^{(k)}_{\omega_k,h_k} = \binom{n_k}{\omega_k} \delta_{h_k,k\omega_k}, \tag{4.5}$$

where $\delta_{i,j}$ is the Kronecker delta function.

The irregular repetition is a parallel concatenation of all n_k sub-blocks, $k = 1, 2, ..., d_v$, where $f_k \neq 0$. It is easy to show that the IOWE of the irregular repetition code is

$$A^{(IR)}_{\omega,h} = \sum_{\omega_1=0}^{n_1} \cdots \sum_{\omega_{d_v}=0}^{n_{d_v}} \sum_{h_1=0}^{n_1} \cdots \sum_{h_{d_v}=0}^{d_v n_{d_v}} \left(\left(\prod_{l=1}^{d_v} A^{(l)}_{\omega_l,h_l} \right) \delta_{\omega,\sum_{k=1}^{d_v}\omega_k} \delta_{h,\sum_{k=1}^{d_v} h_k} \right). \tag{4.6}$$

4.2 IOWE of Punctured Accumulate Codes

The accumulator has a constant left degree a, with $a = 1$ in RA codes and $a \geq 2$ in IRA codes. When $a \geq 2$ it is also known as the punctured accumulator. The IRA codes we designed have $a = 8$. We view a punctured accumulator as the serial concatenation of a check code with degree a and an accumulator without puncturing:

$$A^{acc(a)}_{\omega,h} = \sum_{k=0}^{N_p} \frac{A^{c(a)}_{\omega,k} \times A^{acc}_{k,h}}{\binom{N_p}{k}}, \tag{4.7}$$

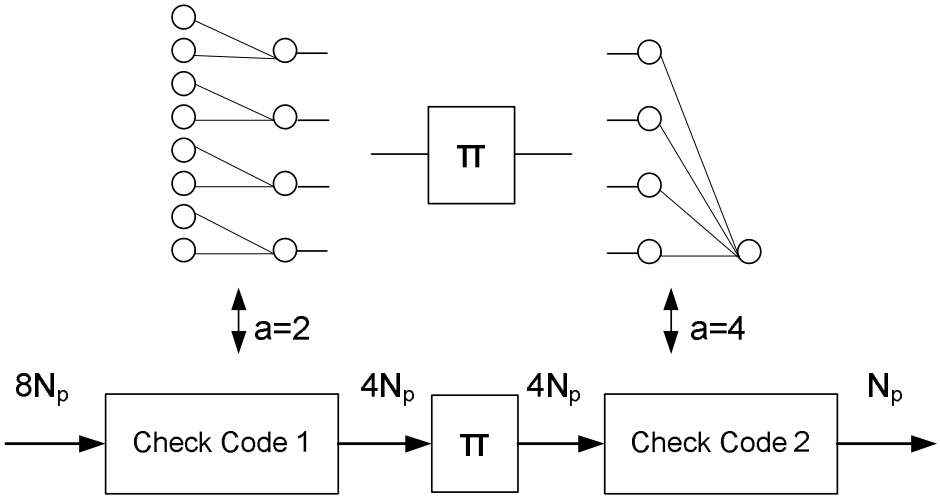


Figure 4.2: The serial concatenation of two check codes. The first check code has $a = 2$, the second check code has $a = 4$.

where $A_{\omega,k}^{c(a)}$ is the IOWE of the check code, $A_{\omega,h}^{acc}$ is the IOWE of an accumulator without puncturing, and $N_p = N - K$ is the number of check nodes. The IOWE of an accumulator without puncturing [26] is:

$$A_{\omega,h}^{acc} = \binom{N_p - h}{\lfloor \frac{\omega}{2} \rfloor} \binom{h-1}{\lceil \frac{\omega}{2} \rceil - 1}. \tag{4.8}$$

To calculate $A_{\omega,k}^{c(a)}$ is more complicated. In [26], two dimensional Z-transform and inverse Z-transform are applied to get the IOWE of checks codes. $A_{\omega,h}^{c(2)}$, $A_{\omega,h}^{c(3)}$ and $A_{\omega,h}^{c(4)}$ are given. $A_{\omega,h}^{c(2)}$ and $A_{\omega,h}^{c(3)}$ can be used as two base IOWEs to calculate the IOWEs with arbitrary a. For example, when $a = 7$, it can be decomposed as $a = 2 + 2 + 3$.

In our example, to get $a = 8$, we concatenate two check codes with $a = 2$ and $a = 4$ respectively, as in Figure 4.2. The two base IOWEs are

$$A_{\omega,h}^{c(2)} = \binom{N_p}{h} \sum_{j=0}^{N_p - h} \binom{N_p - h}{j} 2^h \delta_{\omega,2j+h} \tag{4.9}$$

and

$$A_{\omega,h}^{c(4)} = \binom{N_p}{h} \sum_{j=0}^{N_p-h} \sum_{i=0}^{2N_p-2j-h} \binom{2N_p-2j-h}{i} \binom{N_p-h}{j} \times 2^{2h+2j} \delta_{\omega,2i+2j+h}. \quad (4.10)$$

We add an interleaver of size $4N_p$ between the two constituent codes. It does not affect the IOWE because the input and output of an interleaver have the same weight. It is the averaged IOWE over the ensemble. Thus,

$$A_{\omega,h}^{c(8)} = \sum_{k=0}^{4N_p} \frac{A_{\omega,k}^{c(2)} \times A_{k,h}^{c(4)}}{\binom{4N_p}{k}}. \quad (4.11)$$

Note that in Equation (4.11) $A_{\omega,k}^{c(2)}$ is obtained from Equation (4.9) by substituting $4N_p$ for N_p. The exact expression of $A_{\omega,h}^{c(8)}$ is derived by substituting Equation (4.9) and Equation (4.10) into Equation (4.11):

$$\begin{aligned} A_{\omega,h}^{c(8)} = & \binom{N_p}{h} \sum_{k=0}^{4N_p-2i-2j-h} \sum_{j=0}^{N_p-h} \sum_{i=0}^{2N_p-2j-h} \binom{N_p-h}{j} \\ & \binom{4N_p-2i-2j-h}{k} \binom{2N_p-2j-h}{i} \times 2^{2i+4j+3h} \delta_{\omega,2i+2j+2k+h}. \end{aligned} \quad (4.12)$$

Then we concatenate the check code and the accumulator:

$$A_{\omega,h}^{acc(8)} = \sum_{m=0}^{N_p} \frac{A_{\omega,m}^{c(8)} \times A_{m,h}^{acc}}{\binom{N_p}{m}}. \quad (4.13)$$

Finally, using Equation (4.8) and Equation (4.12), Equation (4.13) is expanded to the following expression:

$$\begin{aligned} A_{\omega,h}^{acc(8)} = & \sum_{m=0}^{N_p} \sum_{k=0}^{4N_p-2i-2j-m} \sum_{j=0}^{N_p-m} \sum_{i=0}^{2N_p-2j-m} \binom{N_p-m}{j} \binom{2N_p-2j-m}{i} \\ & \binom{N_p-h}{\lfloor \frac{m}{2} \rfloor} \binom{h-1}{\lceil \frac{m}{2} \rceil - 1} \binom{4N_p-2i-2j-m}{k} 2^{2i+4j+3m} \delta_{\omega,2i+2j+2k+m}. \end{aligned} \quad (4.14)$$

Equation (4.14) is the averaged IOWE of a check code with $a = 8$ and an accumulator. Next step we need to concatenate Equation (4.14) and Equation (4.5) to get the IOWE of IRA codes.

4.3 IOWE of IRA Codes

Now that we have the IOWEs for both the inner and the outer code, using the uniform random interleaver technique, as in [28], the average IOWE for systematic IRA codes is

$$A_{\omega,h} = \sum_{\sigma=0}^{N_I} \frac{A_{\omega,\sigma}^{(IR)} \times A_{\sigma,h-\omega}^{acc(8)}}{\binom{N_I}{\sigma}}, \tag{4.15}$$

where N_I is the block size of the interleaver between the repetition code and the punctured accumulator. N_I also equals the number of edges emanating from the information nodes.

4.4 IOWE of SC-IRA Codes

Having the IOWE of IRA codes, we next derive the IOWEs of SC-IRA codes using the concatenation system equation again:

$$A_{\omega,h}^{\text{sIRA}} = \sum_{\sigma=0}^{N} \frac{A_{\omega,\sigma}^{(P,2)} \times A_{\sigma,h-\omega}^{(P,1)}}{\binom{N}{\sigma}} \tag{4.16}$$

where $N = K_1 \times N_2$ is the block size of the interleaver π in Figure 2.4. $A_{\omega,h}^{(P,2)}$ is the IOWE of the parallel concatenation of K_1 row codes, while $A_{\omega,h}^{(P,1)}$ is the IOWE of the parallel concatenation of N_2 column codes. Thus $A_{\omega,h}^{(P,2)}$ is the K_1-fold convolution of $A_{\omega,h}^{2}$ with itself, while $A_{\omega,h}^{(P,1)}$ is the N_2-fold convolution of $A_{\omega,h}^{1}$ with itself. $A_{\omega,h}^{2}$ and $A_{\omega,h}^{1}$ are the IOWEs of the row code E_2 and the column code E_1, respectively. Define the IOWE Function [24] as

$$A(\Omega, H) = \sum_{\omega=0}^{K} \sum_{h=0}^{N} A_{\omega,h} \Omega^{\omega} H^{h}. \tag{4.17}$$

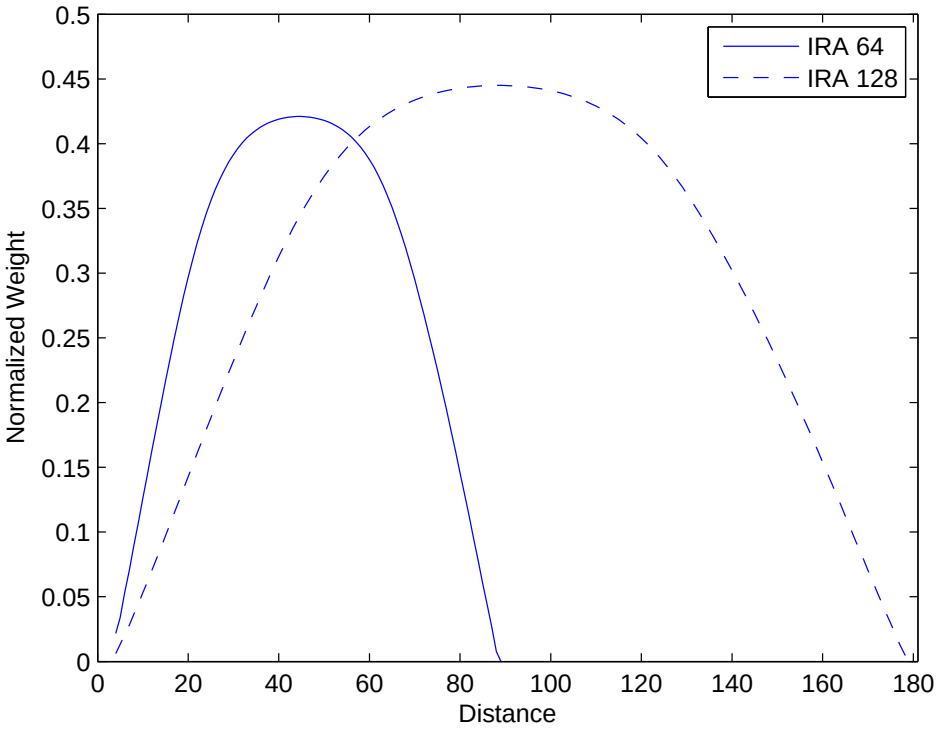


Figure 4.3: The codeword weight spectrum property of two component IRA codes. Their lengths are 64 and 128; the code rate is $1/\sqrt{2}$.

Then the 2D convolution can be expressed as polynomial multiplication: $A_{\Omega,H}^{(P,2)} = [A_{\Omega,H}^{2}]^{K_1}$ and $A_{\Omega,H}^{(P,1)} = [A_{\Omega,H}^{1}]^{N_2}$. Coefficients of $A_{\Omega,H}^{(P,2)}$ and $A_{\Omega,H}^{(P,1)}$ are the IOWEs of the outer and inner codes in Equation (4.16) for the SC-IRA code.

Figure 4.3 is the normalized codeword distance spectrum for the two component IRA codes. The tight bounds of these two codes are computed based upon these two curves.

Figure 4.4 compares the BER simulation results using the SPA algorithm and the tight ML bounds of the two component codes. The two codes are (90, 64) and (181, 128) systematic IRA codes, respectively. Their code rates are 0.7111 and 0.7072,

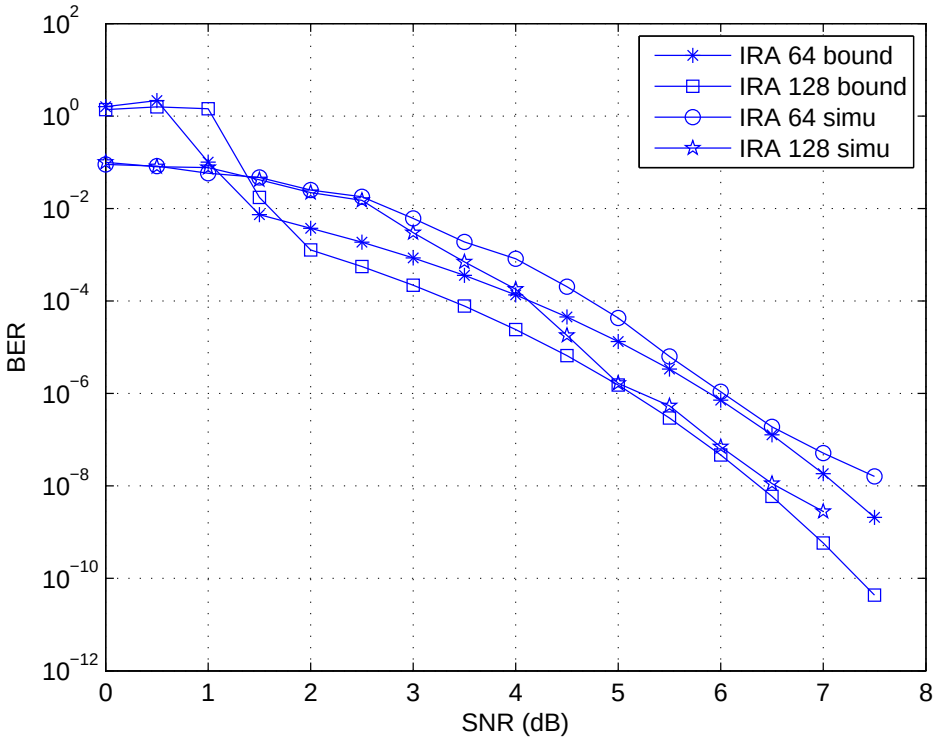


Figure 4.4: The BER simulations and the tight bounds of two component codes. The code rate is $1/\sqrt{2}$.

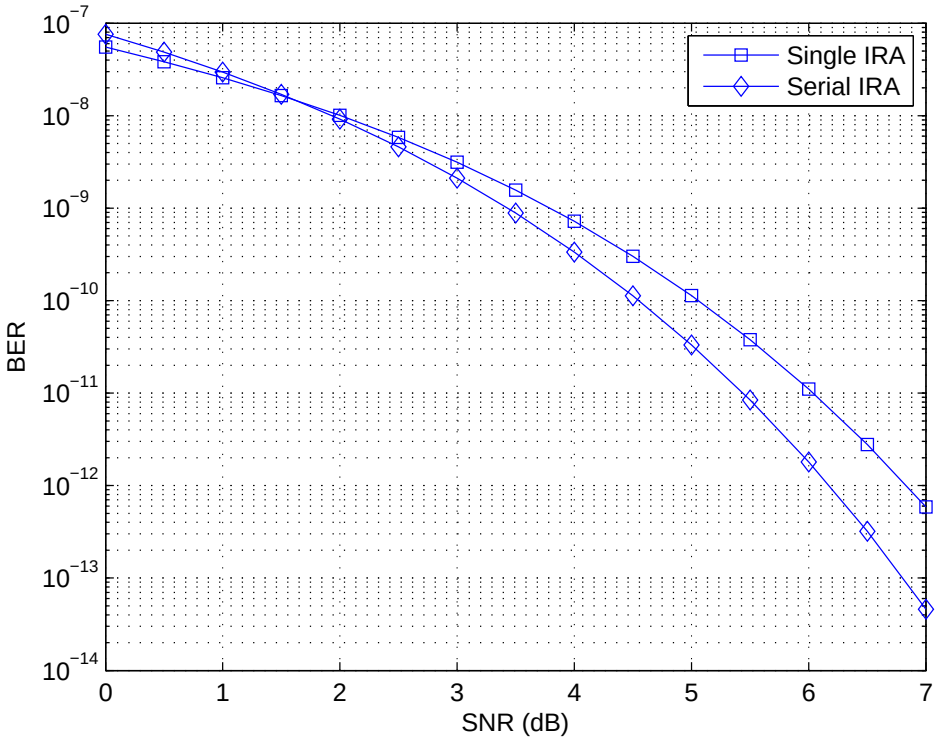


Figure 4.5: The ML bounds of single IRA codes and SC-IRA codes. The code length is 8192, and the code rate is 0.5. Due to the high computational complexity, only the low weight part of the IOWEs is used to compute the ML bounds.

approximating $1/\sqrt{2}$ but subject to granularity of short code sizes. These two codes are designed using the same degree distribution, but again due to granularity their true degree distributions are not exactly the same. From the curves, we can see that the bounds become tight beyond 5 dB. But after 7 dB, the simulation curves begin to diverge from the bounds due to the error floor problem.

Figure 4.5 shows the ML bounds for a single IRA code and a SC-IRA code. Both codes have length 8192 and rate 0.5. Due to the long code length and the high computational complexity, we only calculate the IOWE of low weight codewords. Thus the bounds in Figure 4.5 are tight only at high SNR, because at low SNR,

decoding errors also depend heavily on the non-closest codewords. But Equation (4.16) is an accurate expression of the IOWE of the SC-IRA codes. Therefore it should be possible to obtain the accurate tight bounds using more powerful computers, or using distributed computing technology. From the bounds we computed in Figure 4.5, it is clear that in higher SNR region, the SC-IRA code has a lower BER bound than the single IRA code, which is consistent with the simulation results presented in Chapter 3.

Chapter 5

Conclusion and Future Work

In this thesis, IRA codes and SC-IRA codes are analyzed. Part of the work in this thesis is composed into a paper and submitted to IEEE Globecom 2009, the Communication Theory Symposium.

First the IRA codes are introduced and the design methods are summarized. Then several aspects on how to design good SC-IRA codes are discussed. In particular, it is proven that the number of length 4, 6 and 8 loops in an SC-IRA code grows linearly with the component code lengths. Thus the loop spectrum is shifted to higher length loop region, which helps explain the improved performance of the SC-IRA codes at high SNR.

Next the serial decoder of SC-IRA codes is presented. Then we use the EXIT chart technique to optimize the serial decoder by finding the best inner and outer iteration scheme. We then propose the one-graph decoding method for SC-IRA codes. The one-graph decoding has better performance than the previously proposed serial decoding structure, lowering BER in both the waterfall region and the error floor region. With one-graph decoding, the serial concatenation of IRA codes is also a good way to design very long LDPC-like codes with low error floors.

Then the performance of IRA and SC-IRA codes are analyzed using the ML bounds. A step by step derivation is shown on how to analyze the IOWE of IRA and SC-IRA codes. A simple tight bound is computed based on the IOWE. The ML bounds of IRA codes and SC-IRA codes show that SC-IRA codes have lower BER at high SNR than equivalent single IRA codes.

The SC-IRA code is an alternative way to construct long LDPC codes that have linear encoding algorithm and low error floors. It is possible to concatenate codes in more than two dimensions. It would be interesting to analyze the behavior of these high dimensional serially concatenated codes. It is also of interest to research the relationship between the error floor problem versus the "waterfall" region penalty for longer SC-IRA codes; exploring this relationship will require either greater computing power or future error floor prediction techniques for LDPC codes with SPA decoding.

Bibliography

[1] R.G. Gallager, "Low density parity check codes," IRE Trans. Inform. Theory, IT-8: 21-28, Jan., 1962.

[2] R. M. Tanner, "A recursive approach to low complexity codes," IEEE Trans. Inform. Theory, IT-27: 533-47, Sep. 1981.

[3] T. J. Richardson, M. A. Shokrollahi, and R. L. Urbanke, "Design of Capacity-Approaching Irregular Low-density Parity-check Codes," IEEE Trans. Inform. Theory, vol. 47, no. 2, pp. 619 -637, Feb. 2001.

[4] F. R. Kschischang, B. J. Frey, and H. A. Loeliger, "Factor Graphs and the Sum-Product Algorithm," IEEE Trans. Inform. Theory, 47: 498-519, Feb. 2001.

[5] J. L. Massey, "Threshold decoding," MIT Press, Cambridge, 1963.

[6] H. Jin, A. Khandekar, and R. McEliece, "Irregular Repeat-Accumulate Codes," Proc. 2nd International Symposium on Turbo Codes, pp. 1-8, 2000.

[7] J. Lu, JMF Moura, U Niesen, "A class of structured LDPC codes with large girth," IEEE International Conference on Communications, 2004.

[8] Y. Zhang, W.E. Ryan, "Structured IRA codes: Performance analysis and construction," IEEE Transactions on Communications, 2007.

[9] Y. Wang, M. Fossorier, "Doubly generalized LDPC codes," IEEE International Symposium on Information Theory, 2006.

[10] ETSI Standard TR 102 376 V1.1.1: Digital Video Broadcasting (DVB) User guidelines for the second generation system for Broadcasting, Interactive Services, News Gathering and other broadband satellite applications (DVB-S2), Feb. 2005.

[11] S. Lin and D. J. Costello, "Error Control Coding," 2nd edition, Prentice Hall, 2004.

[12] P. Elias, "Error-free coding," IRE Trans. on Inform. Theory, vol. IT-4, pp. 29-37, Sept. 1954.

[13] R. M. Pyndiah, "Near-Optimum Decoding of Product Codes: Block Turbo Codes," IEEE Trans. Commun., vol. 46, no. 8, pp. 1003-1010, Aug. 1998.

[14] T. Cheng, K. Sivakumar, and B. J. Belzer, "Serially concatenated IRA codes," in Proc. 45th Allerton Conf. on Comm., Computing, and Control, (CD-ROM), pp. 318-321, U. of Illinois Urbana-Champaign, Sept. 2007.

[15] E. R. Berkekamp, R. J. McEliece, and H.C.A. Van Tilborg, "On the inherent intractability of certain coding problems," IEEE Trans. on Inform. Theory, vol. 24, issue 5, pp. 384-386, May 1978.

[16] W. Xu, B. Hassibi, "On the Complexity of Exact Maximum-Likelihood Decoding for Asymptotically Good Low Density Parity Check Codes," CoRR 2007. Available at http://www.citebase.org/abstract?id=oai:arXiv.org:cs/0702147.

[17] S.H. Lee, K.S. Kim, J.K. Kwon, Y.H. Kim, and J.Y. Ahn, "Design of an LDPC code with low error floor," in Proc. IEEE Int. Symp. Info. Theory (ISIT 2005), Adelaide, Australia, Sept. 2005, pp. 990-994.

[18] S. Shamai, I. Sason, "Variations on the Gallager bounds, connections, and applications," IEEE Trans. Inform. Theory, Vol. 48, Issue 12, pp. 3029 - 3051, Dec 2002.

[19] D. Divsalar, "A simple tight bound on error probability of block codes with application to turbo codes," Jet Propulsion Laboratory, CA, JPL TMO Progress Report 42-139, Nov. 15, 1999, pp. 1-35.

[20] D. Divsalar, H. Jin, R. McEliece, "Coding theorems for 'Turbo-like' codes," in Proc. Allerton Conf. Commun., Control, Comput., Monticello, IL, 2000, pp. 201-210.

[21] S. Litsyn, V. Shevelev, "On Ensembles of Low-Density Parity-Check Codes: Asymptotic Distance Distributions," IEEE Trans. Inform. Theory, vol. 48, pp. 887-908, 2002.

[22] T. Richardson and R. Urbanke, "The capacity of low-density parity check codes under message-passing decoding," IEEE Trans. Inform. Theory, vol. 47, pp. 599-618, 2001.

[23] D. Divsalar, F. Pollara, "On the design of turbo codes," TDA Progress Report 42-123, July-September 1995, pp. 99-121, Nov. 15, 1995.
Available at: http://tmo.jpl.nasa.gov/tmo/progress_report/42-123/title.htm

[24] S. Benedetto, D. Divsalar, G. Montorsi, and F.Pollara, "Serial concatenation of interleaved codes: Performance analysis, design, and iterative decoding," IEEE Trans. Infom. Theory, vol. 44, no. 5, pp. 909-926, May 1998.

[25] Y. Wang and M. Fossorier, "Doubly Generalized LDPC Codes," Inform. Theory, pp. 669-673, 2006.

[26] A. Abbasfar, D. Divsalar, and K. Yao, "Maximum likelihood decoding analysis of accumulate-repeat-accumulate codes," in Proc. IEEE Global Telecommun. Conf., Dallas, TX, 2004, pp. 514-519.

[27] S. T. Brink, "Convergence behavor of iterative decoded parallel concatenated codes," IEEE Transactions on Communications, vol. 49, pp. 1727-1737, Oct. 2001.

[28] X. Wu, H. Xiang, X. You, and S. Li, "On the asymptotic input-output weight distribution of some accumulate-based codes," IEEE Trans on Commun. Vol. 54, no. 7, July 2006, pp. 1154-1159.

[29] A. Abbasfar, D. Divsalar, K. Yao, "Accumulate-Repeat-Accumulate Codes," IEEE Trans. Comm. Vol. 55, no. 4, pp. 692-702, April 2007.

[30] C. Di, T. Richardson, R. Urbanke, "Weight distribution of low-density parity-check codes," IEEE Transactions on Information Theory, vol. 52, Issue 11, pp. 4839-4855, Nov. 2006.

[31] Q. Ge, B. Belzer, and K. Sivakumar, "Decoding and performance analysis of serially concatenated IRA codes", submitted to IEEE Globecom 2009 Communication Theory Symposium.