Design of Rate-Compatible Punctured Repeat-Accumulate Codes

by

Shiva Kumar Planjery
B.Tech, Indian Institute of Technology - Madras, India, 2005

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of

MASTER OF APPLIED SCIENCE

in the Department of Electrical Engineering

Design of Rate-Compatible Punctured Repeat-Accumulate Codes

by

Shiva Kumar Planjery

M.A.Sc, University of Victoria, 2007

Supervisory Committee

Dr. T. Aaron Gulliver, Supervisor

(Department of Electrical and Computer Engineering)

Dr. Lin Cai, Department Member

(Department of Electrical and Computer Engineering)

Dr. Venkatesh Srinivasan, Outside Member

(Department of Computer Science)

Supervisory committee

Dr. T. Aaron Gulliver, Supervisor (Department of Electrical and Computer Engineering)

Dr. Lin Cai, Department Member (Department of Electrical and Computer Engineering)

Dr. Venkatesh Srinivasan, Outside Member (Department of Computer Science)

ABSTRACT

In present day wireless applications, especially for time-varying channels, we require flexible coding schemes that utilize a minimum of bandwidth and can support different code rates. In addition, we require coding schemes that are simple in terms of complexity but give good performance. Recently a special class of turbo-like codes called repeat accumulate (RA) codes were proposed. These codes are extremely simple in terms of complexity compared to turbo or LDPC codes and have been shown to have decoding thresholds close to the capacity of the AWGN channel. In this thesis, we propose rate-compatible punctured systematic RA codes for the additive white Gaussian noise (AWGN) channel. We first propose a three phase puncturing scheme that provides rate compatibility and show that very high rate code rates can be obtained from a single mother code. We then provide a methodology to design rate-compatible RA codes based on our three phase puncturing scheme. The design involves optimizing the punctured profile of the code such that the resulting high rate codes give good performance. The design is done with the help of existrinsic information transfer (EXIT) charts which are plots used to analyze the constituent decoders. Code rates up to 10/11 are obtained from a single rate 1/3 regular RA code. Performance results show that our design methodology combined with our proposed puncturing scheme can provide significant coding gains at high code rates even with practical blocklengths. Hence rate-compatible punctured RA codes are suitable for many wireless applications.

Table of Contents

List of Figures

List of Abbreviations

AWGN	Additive white Gaussian noise
BCH	Bose-Chaudri-Hocquenghem codes
BCJR	Bahl-Cocke-Jelinek-Raviv
BER	Bit error rate
BPSK	Binary Phase shift keying
BSC	Binary symmetric channel
CSI	Channel state information
EXIT	Extrinsic information transfer
IRA	Irregular repeat accumulate
LDPC	Low-density parity check
LLR	Log-liklihood ratio
PDF	Probability density function
RA	Repeat accumulate
RCPC	Rate-compatible punctured convolutional codes
RCPT	Rate-compatible punctured turbo codes
SNR	Signal to noise ratio
SOVA	Soft output Viterbi algorithm
SPC	Single parity check code
QP	Quadratic programming
UWB	Ultra wideband channel

Terminology

(n, m) block code	Set of codewords of length n obtained by encoding messages of length m
a priori information	Amount of information present in the *a priori* LLR values
blocklength	Number of bits in a message, i.e. message length
BPSK modulation	Bits transmitted across the channel are $\{+1, -1\}$
code rate	Ratio of message length to codeword length given by m/n
decoding threshold	The minimum value of E_b/N_0 for which the decoding converges
error floor	Region in the BER performance curve where the curve flattens out below a certain value of BER
extrinisic information	Amount of infromation present in the extrinsic LLR values
minimum distance	Smallest hamming distance between all codewords in a code
mutual information	Measures information content in random variables

parity check	Binary sum of a group of bits
random interleaver	Randomly permutes a sequence of bits
rate-compatibility	Ability of a code to support different code rates depending on the channel conditions
s-random interleaver	An interleaver whose permutation pattern is such that an integer in the pattern has a distance greater than $\pm S$ compared to any of the previous S integers in the pattern
systematic bits	Message bits that are transmitted across the channel and form part of the codeword
zero column vector	A column vector consisting of only zeros as its elements

Acknowledgement

I would like to express my warmest gratitude to my supervisor Dr. T. Aaron Gulliver who has been more than just a supervisor to me. Ever since being his student, I have relied on him for guidance in almost every matter. Working with him for the last two years has been an enjoyable learning experience, although I still have much to learn from him. His friendly and humble nature along with his concern for students' welfare has truly made him my role model. I am extremely grateful for everything he has done for me.

I would like to extend my gratitude to Dr. Venkatesh Srinivasan and Dr. Lin Cai who served as my committee members and provided useful insights to my research work. I would also like to thank Dr. Jianping Pan for being my external examiner.

I would like to specially thank Dr. Wu-Sheng Lu for his kind assistance and advice during the course of my research work. Although I was his student for just one course, his teaching deeply inspired me and I gained valuable knowledge from him.

I would like to thank all my friends and lab mates who have always been helpful, providing me with advice whenever I needed.

I would finally like to thank my parents who have always stood by me in the toughest of times and without whose support I would not have reached this far.

Dedication

Dedicated to my family

Chapter 1

Introduction

1.1 The channel coding problem: Brief history

The recent emergence of large-scale, high-speed data networks for exchange, processing, and storage of digital information has increased the demand for efficient and reliable digital transmission and storage systems. The control of errors so that reliable reproduction of the data can be obtained is a major concern in the design of communication systems. Error control coding is a technique used to combat the errors introduced by a noisy channel while transmitting information . The technique essentially adds redundancy to the message in the form of extra symbols called parity symbols. This allows us to detect and correct the most likely error patterns.

The concept of using error correcting codes was first investigated by R. W. Hamming at Bell Laboratories in 1947. The motivation for this work started from his increasing frustration with relay computers. These machines could detect errors but could not rectify them and hence the jobs he submitted were abandoned once an error occurred. Hamming guessed that if codes could be devised to detect an error, then codes that could also rectify errors should exist and thus he started searching for such codes. In doing so, he originated the field of coding theory and he finally published his results [1] in 1950.

In 1948, the publication of C. E. Shannon's landmark paper [2] sparked the development of Information theory and essentially laid down the limits of error control coding. Shannon's noisy channel coding theorem implied that arbitrarily low decoding error prob-

abilities can be achieved at any transmission rate r less than the channel capacity C by using randomly constructed error-correcting codes with sufficiently long block lengths. In particular, Shannon showed that randomly chosen codes, along with maximum likelihood decoding, can provide capacity-achieving performance with high probability. Shannon mentioned the (7,4) Hamming code in his paper as an example of an error-correcting code. However, he provided no insight as to how to actually construct good codes that are random but decodable at the same time.

In the ensuing years, much research was conducted into the construction of specific codes with good error-correcting capabilities and the development of efficient decoding algorithms for these codes. The best code designs contained a large amount of structure since it guaranteed good minimum distance properties. Codes such as Bose-Chaudri-Hocquenghem (BCH) and Reed-Solomon codes were constructed based on an algebraic structure and convolutional codes, which are commonly used in communication systems, were based on a topological structure called a trellis. The decoding algorithms that were mainly used for these codes such as the Berlekamp-Massey algorithm and the Viterbi algorithm were based on these structures [3]. It seemed that the more structure a code contained, the easier it was to decode.

However, these codes perform poorly for asymptotically large blocklengths and they lack the random-like properties that were originally envisioned by Shannon. Little attention was focussed on the design of random-like codes because they were thought to be too difficult to decode. However, in recent years, construction of coding schemes that have random-like properties has become the primary objective of research in coding theory. In 1993, the paper of Berrou *et al.* [4] introduced a new coding technique called iterative decoding (also known as turbo decoding) that succeeded in achieving a random-like code design with just enough structure to allow for efficient decoding. The fundamental property of turbo codes that underlied their exceptional performance was the random-like weight spectrum of codewords produced by a pseudorandom interleaver. The randomness of the code induced by the interleaver and the use of iterative decoding together revolutionalised

the field of coding theory. This lead to the emergence of a new class of codes called turbo-like codes which were capacity-achieving codes based on iterative decoding. Turbo codes are able to perform within 1dB of the Shannon limit on the AWGN channel. Another type of turbo-like codes, were the Low Density Parity check (LDPC) codes introduced by Gallager [5], which was based on introducing randomness by constructing a sparse parity check matrix. Both LDPC and turbo codes have today found use in a wide range of applications such as wireless communication systems, deep space communications, etc.

The success of turbo codes and LDPC codes kindled an interest for researchers to theoretically analyse these codes as well as come up with other constructions that have reduced decoding complexity. Turbo codes have a naturally linear time encoding algorithm and the pseudorandom interleaver induces randomness into the code. However, they suffer from a large decoding complexity and a large decoding delay due to the nature of the decoding algorithm (BCJR) used. LDPC codes on the other hand have a complex encoding algorithm due to the construction of a sparse parity check matrix. In 2000, a new class of turbo-like codes called Repeat Accumulate codes was introduced by Jin *et al.* [6] [7] .

These codes are much simpler in complexity compared to LDPC or turbo codes but are quite competitive in performance. They were shown to perform close to Shannon capacity for the AWGN channel. The work in this thesis is concerned with this particular class of codes. Before going into the details of this class of codes, we shall first provide some fundamentals of error-correcting codes in the next section.

1.2 Error correcting code fundamentals

An error correcting block code denoted by (n, m) is a set of codewords of length n that are obtained by encoding a message consisting of m symbols. If the symbols used are binary, an (n, m) block code would contain 2^m codewords of length n that are selected from a set of all 2^n possible words of length n. The encoding specifies the mapping from the m message symbols to the n codeword symbols. There are two types of encoding schemes: systematic

and non-systematic encoding. The encoding in which the message symbols form the first m symbols of a codeword is called systematic encoding. The remaining $n-m$ symbols that are appended to the message to form the rest of the codeword are called parity symbols. If the message symbols do not form part of the codeword, the encoding is called non-systematic encoding. The rate of a code is defined as the ratio

$$R = \frac{m}{n}$$

For all practical purposes we mostly deal with error-correcting block codes that are linear. A (n, m) block code is linear iff it is a subspace of the vector space of dimension n, and so is an additive group. An important consequence of this is that the sum of two codewords in a linear code must also be a codeword. The span of the vector subspace is m.

The Hamming weight of a codeword, **wt**$[x]$, is the number of nonzero elements contained in it. The Hamming distance between two codewords, $\mathbf{d}(x, y)$, is defined as the number of places in which they differ,

$$\mathbf{d}(x, y) = \mathbf{wt}[x - y].$$

The smallest Hamming distance between all codewords in a code is called the minimum distance,

$$d_{min} = \min \ \mathbf{d}(x, y) \ \forall x, y; x \neq y.$$

The minimum distance of a linear code is the weight of the smallest nonzero codeword, since the linear combination of any two codewords is also a codeword,

$$d_{min} = \min \ \mathbf{wt}[x] \ \forall x; x \neq 0.$$

The number of errors a code can correct is

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

where $\lfloor x \rfloor$ is the largest integer less than or equal to x.

The generator matrix, G, of an (n, m) linear block code [3] is a $m \times n$ matrix of linearly independent codewords. All codewords can be formed from a combination of the rows of this matrix. The generator matrix defines the encoding process of the message. Thus a codeword c from the code is given by $c = uG$ where u is the message vector of length m.

The parity check matrix of this code is an $(n - m) \times n$ matrix H such that

$$GH^T = \mathbf{0},$$

where $\mathbf{0}$ is a $m \times (n - m)$ matrix of zeros. The parity check matrix also satisfies the condition $Hc^T = \mathbf{0}$ and is very useful especially during decoding. The parity check matrix can be used to ensure that the estimated codeword obtained during decoding belongs to the set of valid codewords.

The weight enumerator of a code is defined as a polynomial in z ,

$$A(z) = \sum_{i=0}^{n} A_i z^i$$

where A_i is the number of codewords of weight i. From this definition it is evident that for a binary code

$$\sum_{i=0}^{n} A_i = 2^m$$

which is the total number of codewords. For further details on error correcting codes, we refer to [3]. Having provided the fundamentals of coding theory, we shall describe the structure of RA codes in the next section.

1.3 Repeat Accumulate Codes : Structure and encoding

RA codes are serially concatenated codes consisting of a Repetition Code as the outer code and an Accumulator as the inner code with a pseudorandom interleaver in between them. The repetition code is defined as a (n, m) code where each message bit is repeated q times and thus $n = q \cdot m$. The accumulator can be viewed as a truncated rate-1 recursive convolutional encoder with transfer function 1/(1+D). But it is preferable to think of it as a

block code whose input block $[x_1, x_2, \cdots, x_n]$ and output block $[y_1, y_2, \cdots, y_n]$ are related by the formula

$$\begin{aligned} y_1 &= x_1 \\ y_2 &= x_1 + x_2 \\ y_3 &= x_1 + x_2 + x_3 \\ &\vdots \\ y_n &= x_1 + x_2 + x_3 + \cdots + x_n \end{aligned} \tag{1.1}$$

Fig. 1.1 shows the encoder of an RA code. The number of input bits is denoted by m i.e.

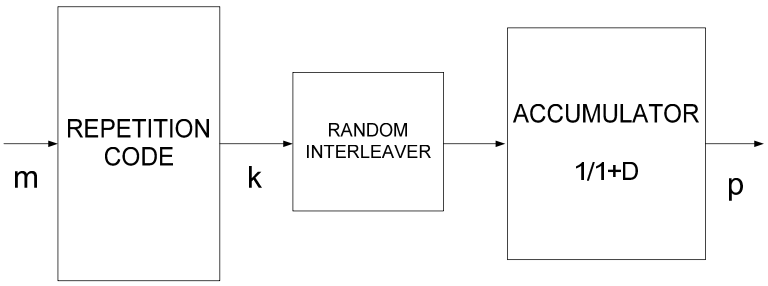

Figure 1.1. *Encoder of an RA code.*

the message length is m. Each message bit is repeated q times for a (k, m) repetition code and we get $k = (q \cdot m)$ bits at the output. These k bits are then randomly interleaved and passed through a rate-1 accumulator. The output bits of the accumulator denote the parity bits of the code and are transmitted across the channel. Non-systematic encoding is used for this encoder which means the message bits do not form part of the codeword and hence are not transmitted. If $(u_1, \ldots, u_m)$ denotes the sequence of information bits and $(b_1, \ldots, b_p)$ denotes the sequence of parity bits, the codeword would be $(b_1, \ldots, b_p)$. The overall rate for this code is $1/q$. It is evident that the RA code structure consists of two very simple codes and the interleaver induces randomness thus providing potential for near-capacity performance. However, the major drawback of these codes is their low code rate. It has been shown in [6] that $q \geq 3$ is required in order to get near-capacity performance. This

implies that the maximum code rate for this code is 1/3 which is extremely low especially for current wireless applications which have strict constraints on bandwidth utilization.

To overcome the low code rate problem, the encoder is slightly modified by performing an additional parity check before the accumulator. Systematic encoding is used in this encoder structure where the message bits are also transmitted across the channel. Non-systematic form cannot be used and the reason shall be provided in the next section. Fig 1.2 shows the modified encoder structure.

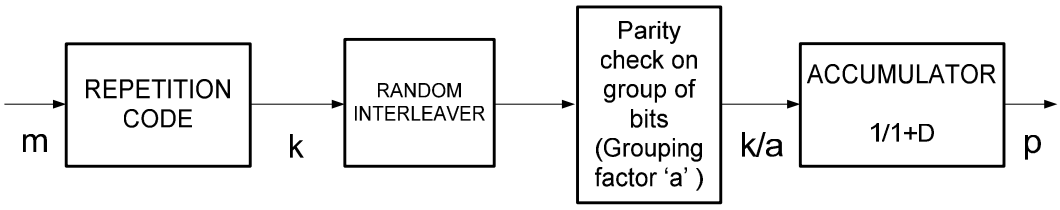

Figure 1.2. *Systematic encoder of an RA code.*

In this encoder structure, the message bits are repeated as usual but a parity check is performed on a group of bits before they are passed through the accumulator. The grouping factor a implies that a parity check is performed on groups of a interleaved bits. Due to the grouping, only k/a bits are passed through the accumulator instead of k bits thus increasing the code rate. Therefore, for a particular code rate, if we use higher values of a, higher values of q can be used leading to larger performance gains.

The RA codes we discussed till now were regular RA codes where each message bit was repeated regularly (q times). However, higher performance gains can be achieved with Irregular Repeat Accumulate (IRA) codes [7]. The difference in the structure of these codes compared to the previous structure is the fact that an irregular repetition code is used. For the purpose of analysis and better visualization of the code structure, IRA codes can be best described by graphical structures called Tanner graphs. Fig 1.3 shows the Tanner graph of an IRA code.

The parameters of the Tanner graph are $(f_1, \ldots, f_j\ ; a)$, where $f_i \geq 0, \sum_{\forall i} f_i = 1$ and a is a positive integer. The open circles represent variable nodes and the dark circles represent

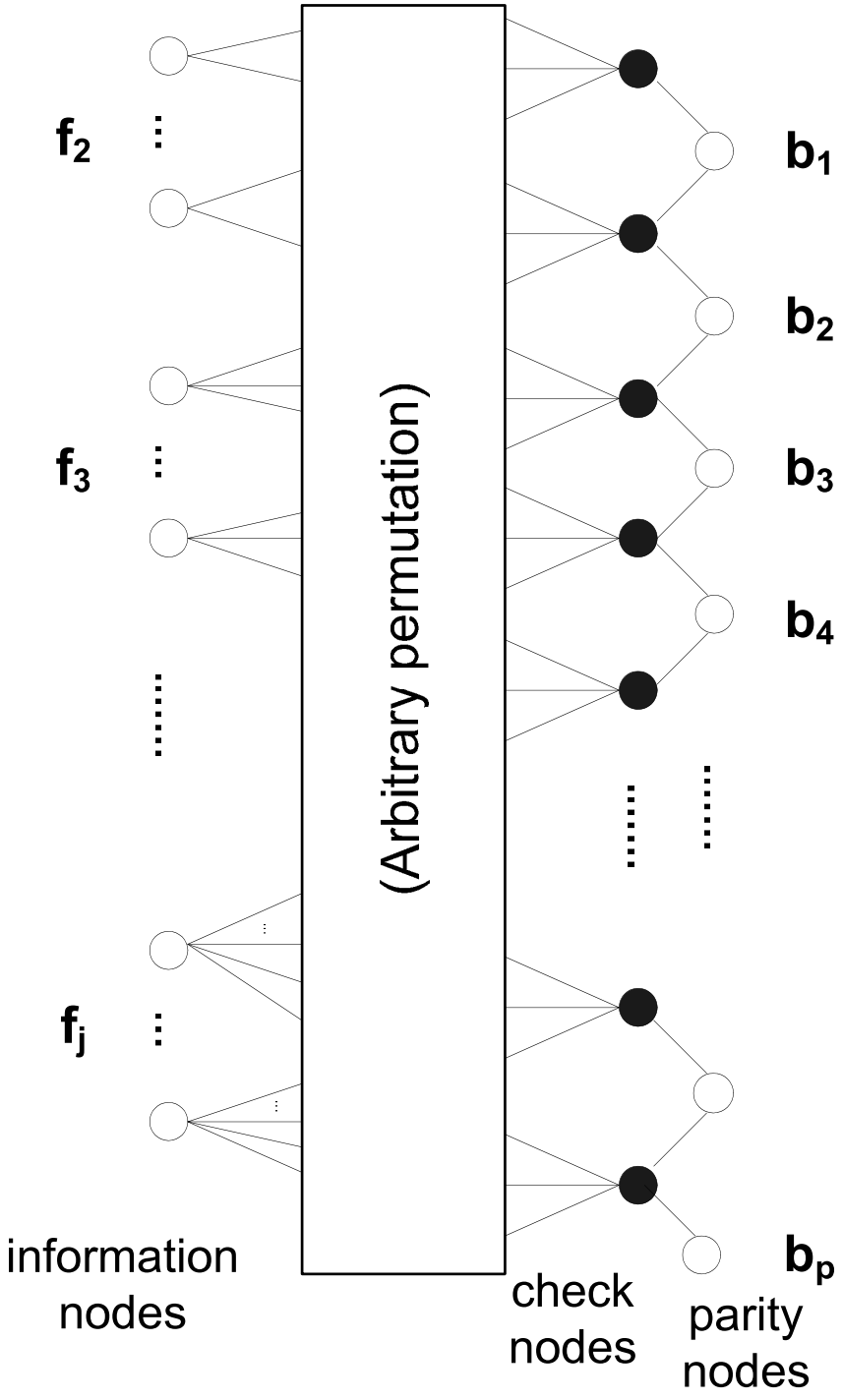

Figure 1.3. *Tanner graph of an RA code.*

check nodes. The variable nodes on the left are called information nodes and represent the message bits to be encoded. The variable nodes on the right are called parity nodes and represent the parity bits. f_i represents the fraction of information nodes with degree i and a represents the degree of the check nodes. In the encoder for the IRA code, a fraction f_i of the information bits $(u_1, \ldots, u_m)$ is repeated i times. The resulting sequence is interleaved and then fed into an accumulator which outputs one bit for every a input symbols resulting in a sequence of parity bits $(b_1, \ldots, b_p)$. The distribution $(f_1, \ldots, f_j\ ; a)$ is referred to as the degree profile of the code and is used in the optimization of the IRA ensemble. The fraction of edges with degree i is denoted by λ_i, and is related to f_i by $f_i = (\lambda_i/i)/(\sum_{\forall j} \lambda_j/j)$. It is evident from the code structure that the degrees of information nodes are not uniform and the code is irregular. Systematic encoding is done due to the presence of parity checks $(a > 1)$ and the codeword bits correspond to $(u_1, \ldots, u_m\ ; b_1, \ldots, b_p)$. The overall rate of IRA codes is given by

$$R = \frac{a}{a + \sum_i i f_i}. \tag{1.2}$$

Representing the IRA code structure as a Tanner graph is important since the decoding algorithm involves updating messages across the edges of the graph. In the next section, we shall describe in detail the decoding of IRA codes.

1.4 Decoding of IRA codes

Decoding of IRA codes involves an iterative algorithm called the sum-product message-passing algorithm [8]. In this algorithm, all messages are assumed to be log-likelihood ratios (LLRs). In other words, if x is a transmitted symbol and y is a received symbol then the message q would be

$$q = \log\left(\frac{p(y \mid x = 0)}{p(y \mid x = 1)}\right) \tag{1.3}$$

These messages are passed iteratively between the information nodes and the parity nodes along the edges of the tanner graph. An important aspect of the algorithm is to ensure that all the incoming messages at a particular node are independent of each other,

so that the outgoing message (called *extrinsic information*) on a particular edge from the node can be determined. Theoretically speaking, due to the nature of the code structure and the iterative algorithm, this is not possible since there will be loops in the graph. But we can conveniently assume independence if we make sure the incoming message of a particular edge is not included in the calculation of the outgoing message for the same edge. Hence while calculating the extrinsic information on a particular edge from a node, all the incoming messages whose edges are incident to that particular node are considered as inputs for the calculation except the edge for which we are calculating. In addition, the assumption becomes more accurate as the size of the blocklength increases. In this thesis, the term blocklength is defined as the number of input or message bits to be encoded.

In order to describe the decoding algorithm, we introduce the following notation:

- $u_{i\to c}$ denotes the extrinsic information from an information node to a check node and let $u_{c\to i}$ denote the converse.
- $u_{c\to p}$ denotes the extrinsic information from a check node to a parity node and $u_{p\to c}$ denote the converse.
- m_o and p_o denote the information recieved from the channel on the message bits and parity bits, respectively.

At the variable node, the outgoing message $u_{i\to c}$ along the k^{th} edge can be obtained by

$$u_k = m_o + \sum_{m\neq k} u_m, \tag{1.4}$$

where m now denotes all edges incident to the variable node except for the one for which $u_{i\to c}$ is being calculated. In the initial iteration u_m is zero.

At the check node, the outgoing message $u_{c\to i}$ along the k^{th} edge is obtained using the $\tanh$ rule [9] as follows

$$\tanh\frac{u_k}{2} = \prod_{m\neq k} \tanh\frac{u_m}{2}, \tag{1.5}$$

where m denotes all the edges that are incident to the check node except the edge for which the outgoing message is being calculated. Although (1.4) and (1.5) specify the update rules

for a particular node, the sequence in which the messages are passed depends on the type of scheduling scheme used. There are two types of scheduling schemes. They are briefly described as follows.

- Turbo-like scheduling: In the initial iteration, channel information m_o is sent to the check nodes. The outgoing message from parity node, $u_{p\to c}$ is calculated using the Bahl-Cocke-Jelinek-Raviv (BCJR) [10] algorithm by utilizing the channel information p_o since the accumulator can be represented by a trellis. At the check node, the outgoing message $u_{c\to i}$ is calculated using (1.5) and then deinterleaved to be passed on to the information nodes. At the information node, the outgoing message $u_{i\to c}$ is calculated using (1.4), interleaved and passed on to the check nodes. The outgoing message $u_{c\to p}$ is then calculated using (1.5) and then fed to the accumulator as *a priori* information. This process is iterated.
- LDPC-like scheduling: In the initial iteration, channel information m_o and m_p are sent to the check nodes. Both $u_{c\to p}$ and $u_{c\to i}$ are calculated using (1.6) and passed to the information and parity nodes, respectively. At the information and parity nodes, both $u_{i\to c}$ and $u_{p\to c}$ are calculated at the same time and passed to the check nodes. This process is iterated. In this case, one iteration consists of the activation of all the variable nodes followed by the activation of all the check nodes.

In the final iteration, at the variable node, all incoming messages are considered as inputs and (1.5) then gives the *a posteriori* LLR values of the information nodes. The message bits are then decoded by observing the sign of the LLR values. If the LLR value for a particular information node is positive, the corresponding message bit is decoded as zero and if the LLR value is negative, the corresponding message bit is decoded as one. Note that in LDPC-like scheduling, all variable nodes (information and parity nodes) are updated at once whereas in the turbo-like scheduling, the information nodes and parity nodes are updated alternatively with a check node update in between.

The performance of these codes is measured in terms of the bit error rate (BER) of the code for different values of E_b/N_0. The bit error rate is defined as the rate of errors

occurring in a decoded message bit stream. E_b/N_0 is defined as the average signal to noise ratio per bit on the AWGN channel and is typically measured in dB. E_b denotes the average energy of a bit and N_0 denotes the noise power spectral density of the AWGN channel. The BER vs E_b/N_0 plot determines the performance of the codes. For a given IRA code of finite blocklength, both schedulings give similar BER performance. However, turbo-like scheduling will have faster convergence because it utilizes the trellis structure of the accumulator. LDPC-like scheduling on the other hand is more advantageous in terms of implementation since parallel processing of the messages is possible. In addition, this scheduling scheme is more accurate to use during analysis of the codes. Hence, our main performance results are obtained using LDPC-like scheduling in the decoding algorithm. However, we do implement turbo-like scheduling for one particular design which was used to obtain preliminary results. Performance comparisons between the two schedulings is not considered in this thesis since it is well known in literature that the two schedulings perform similarly for a given IRA code and both schedulings are instances of the sum-product decoding algorithm.

It is evident from the decoding algorithm that due to the presence of parity checks, channel information on the message bits is required to initiate the decoding algorithm. This is because during the initial iteration when the check nodes receive information from the parity nodes, initial information on the message bits must be passed from the information nodes to the check nodes in order for the check node update to output non-zero messages and this information comes from the channel (by transmitting the message bits across the channel). Note that if any incoming message to the check node is zero, from (1.5), the output will always be zero and thus decoding can never converge. Hence, for any RA or IRA code with $a > 1$, systematic encoding must be used. In the literature, RA codes are typically considered as non-systematic rate $1/q$, $a = 1$ codes just as described in the beginning of Section 1.2, and IRA codes are considered systematic. However, in this thesis, we shall use systematic regular RA codes which are regular RA codes with $a > 1$.

1.5 Need for rate-compatibility

Often, the design of coding schemes such as convolutional codes or turbo codes involves selecting a fixed code rate that is well adapted to the channel characteristics, and determining good codes for that particular rate. However, for time-varying channels, a more flexible coding scheme is required since the data may require different levels of protection at different times. A common error control strategy that is employed to resolve this is to adapt the code rate according to the channel state information (CSI). In addition, with bandwidth becoming increasingly scarce due to the increasing demands of wireless applications, coding schemes are required that utilize a minimum of bandwidth. Hence a single coding scheme that can vary from lower code rates to higher code rates depending on the channel characteristics is required so that the bandwidth can be utilized efficiently. An effective solution in this case is to use a single code and puncture it to achieve rate-compatibility. Hagenauer proposed Rate Compatible Punctured Convolutional Codes (RCPC) in [11], and was able to obtain high code rates by puncturing a single low rate convolutional code.

After the emergence of capacity-achieving codes such as turbo and LDPC codes, efforts were made to introduce rate-compatibility into such codes. Rate-compatible punctured turbo (RCPT) codes were proposed by Barbulescu *et al.* in [12]. In [13], Hagenauer *et al.* constructed several high-rate turbo codes using punctured convolutional component codes and the soft output Viterbi algorithm (SOVA) for decoding. In [14], Acikel and Ryan designed RCPT codes using an exhaustive method that involves a systematic computer search for optimal constituent codes, puncturing patterns and interleavers. Ha *et al.* [15] introduced rate-compatibility in LDPC codes and used density evolution [9] to obtain punctured codes with good decoding thresholds. Recently, Lan *et al.* proposed rate-compatible IRA codes [16] for the BSC channel. They induced rate-compatibility by uniformly puncturing parity bits of a rate 1/3 mother IRA code to obtain different code rates. Their design involved optimizing the degree profile of the mother IRA code using density evolution, such that the lowest and highest code rates give good performance. They were able to

reach a maximum code rate of 5/6, and the codes obtained outperformed turbo codes when applied to image transmission.

In this thesis, we propose rate-compatible punctured systematic RA codes for the AWGN channel. Our main contribution is two-fold: $a)$ a three-phase puncturing scheme that provides superior performance to just puncturing parity bits, and $b)$ a design methodology that employs Extrinsic Information Transfer (EXIT) charts based on our puncturing scheme. We will first describe the three-phase puncturing scheme in detail in the next chapter. We will then describe the design methodology using EXIT charts in the subsequent chapter.

Chapter 2

Rate-Compatible Systematic RA Codes

In this chapter we describe the three phases of puncturing that are used to obtain very high code rates from a single mother RA code. At the end of each phase of the puncturing, the code rate is increased. Unlike the work of Lan *et al.* [16] in which an optimized IRA code is used as the mother code, we start with a rate $1/3$, $a = 4$ systematic regular RA code as the mother code. The motivation for using a systematic RA code as the mother code comes from the fact that regular RA codes have relatively better distance properties than IRA codes, and hence we are better able to avoid introducing error floors which hamper the performance of subsequent codes obtained through puncturing. Error floors are regions in the BER performance curves of the codes where the curves flatten out below a certain value of BER (typically at 10^{-5}) due to poor minimum distance, and they typically occur in IRA codes. The reason for the choice of $a = 4$ shall be explained in the next chapter. The three phases of puncturing are depicted in the block diagram shown in Fig. 2.1.

Note that the symbols used in the block diagram denote the number of bits in the bit sequence and not the sequence themselves. For example, the symbol 'm' denotes that the number of input message bits is 'm'. Depending on what code rate is required, a particular phase of puncturing is carried out. Ultimately Phase 3 puncturing is required to obtain very high code rates. Before describing each puncturing phase in detail, we introduce some basic notations. Let R_1, R_2 and R_3 denote the new code rates at the end of Phases 1, 2 and 3 of the puncturing process, respectively. From the diagram in Fig. 2.1, it can be seen that systematic encoding is used, i.e. the message bits are transmitted across the channel along

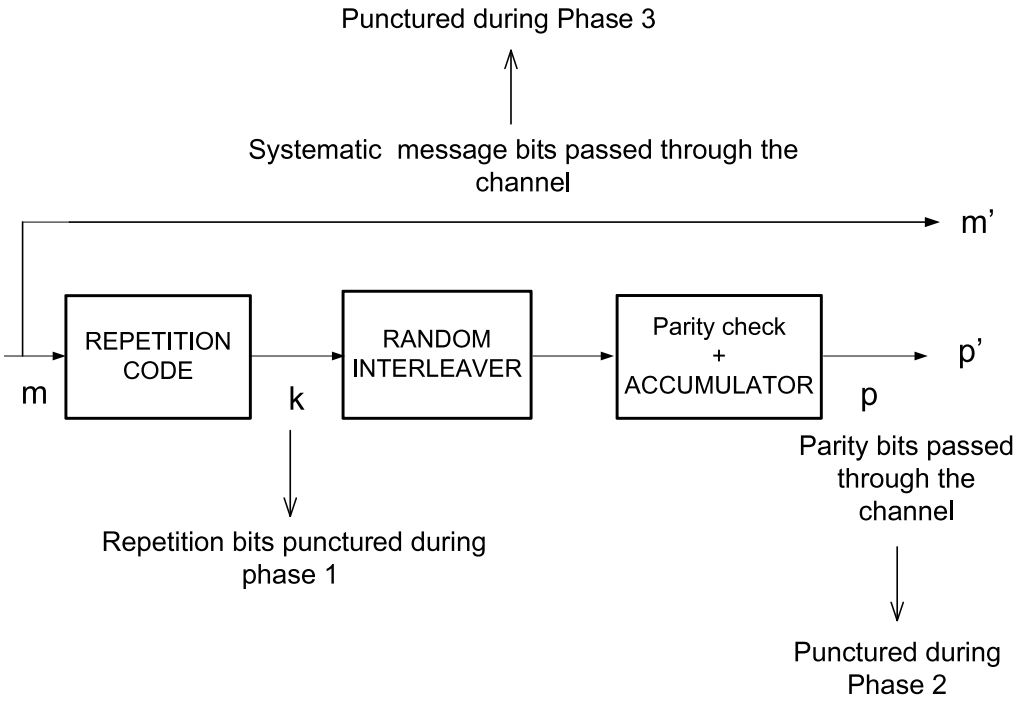

Figure 2.1. *Block diagram of encoder with the three phases of puncturing*

with the parity bits. Note the difference between m and m', and, p and p', respectively. p is the number of parity bits that come out of the accumulator whereas p' is the number of parity bits transmitted across the channel. Similarly m denotes the number of message bits that are input into the encoder whereas m' is the number of message bits transmitted across the channel. In the case of no puncturing, $m = m'$ and $p = p'$. However, at the end of a particular phase of puncturing, they may not be equal. We will use subscripts to indicate the number of bits that are transmitted at the end of a particular phase, i.e. p'_i denotes the number of parity bits p' that are transmitted at the end of phase i of the puncturing. The idea behind puncturing is to reduce the number of bits transmitted thus increasing the code rate.

2.1 Phase 1 - Puncturing repetition bits

In this phase, the output bits of the repetition code are punctured irregularly in order to increase the code rate. The original rate of this code without puncturing is given by $R_0 = m/(m+p)$. As depicted in Fig. 2.1, since the output of the repetition encoder is punctured, the number of repetition bits k is reduced. This results in fewer parity bits p at the output of the accumulator. The new code rate after puncturing is $R_1 = m/(m'_1 + p'_1)$. Compared to the original code, $m'_1 = m$ but $p'_1 < p$, and hence $R_1 > R_0$.

Puncturing repetition bits in this manner amounts to deleting links on the repetition side of the Tanner graph. In other words, we are irregularly puncturing some of the links from the information nodes of the Tanner graph. Hence, the degrees of the information nodes are no longer constant and the regular RA mother code is transformed to an IRA code in the process. The pattern for puncturing in this phase defines the degree profile of the resulting IRA code. Depending on the required rate, for different puncturing patterns employed, we will effectively get different IRA codes. However, the maximum possible degree will remain the same as that of the mother code. The puncturing patterns should be chosen such that the repetition degree for some information nodes is higher than for others. In our case, since we are starting with a rate 1/3, $a = 4$ regular RA code, the maximum possible degree of the information nodes is 8. We puncture many links from some nodes and leave some nodes unpunctured. The number of links to be punctured for each node depends on the chosen degree profile. In order to evaluate this puncturing scheme before incorporating it into our design, we choose a degree profile by studying some of the existing optimal degree profiles of IRA codes. Based on the profiles in [6], which were optimized for the AWGN channel, and having the additional constraint that the maximum possible degree of an information node is 8, we chose a profile with $(f_8, f_4, f_3, f_2; a)$. This profile may not be the best possible choice, but it is sufficient to show that this puncturing scheme can increase the code rate and still provide good performance.

From the chosen degree profile, the values of the degrees are fixed but the values of the

fractions allotted to the four degrees (f_8, f_4, f_3, f_2) can vary based on the required code rate. The degree profile along with the chosen fractions denote the irregular puncturing pattern that is employed to obtain an IRA code of required code rate from the mother RA code. Using this degree profile, a simulation of the code was carried out using binary phase shift keying (BPSK) modulation over an AWGN channel. BPSK modulation implies that a bit 0 is transmitted as $+1$ and bit 1 is transmitted as -1 across the channel. Fig. 2.2 shows the BER performance of the code at the end of Phase 1 for three different random puncturing patterns that result in code rates of 3/7, 4/9 and 1/2 by choosing the fractions (f_8, f_4, f_3, f_2) appropriately. For the rate 1/2 code, the fractions chosen were $f_8 = 0.2441$, $f_4 = 0.1953$, $f_3 = 0.1445$, and $f_2 = 0.4161$. The simulation was performed for a blocklength of $m = 1024$ message bits and a random interleaver was used. Turbo-like scheduling was used for decoding and the number of decoding iterations used was 10.

Fig. 2.2 shows that Phase 1 puncturing gives a slight performance gain at low SNRs. This confirms the widely accepted notion that irregularity improves code performance at lower SNRs. However, the distance properties of the code are greatly affected by puncturing so that as the rate increases, the error floors get worse. Hence Phase 1 puncturing is useful when the required code rates are not too high, i.e. less than 1/2. Avoiding high error floors is one of the main reasons why we start with a regular RA code as the mother code. Had we started with an IRA code as the mother code, not only would determining the puncturing pattern be much more difficult, but the propagation of error floors would be much worse since the mother code itself has a higher error floor.

A pseudorandom interleaver was employed for this code and though a better choice of interleaver might improve this error floor, the behavior at higher SNRs due to the puncturing will remain essentially the same. We considered twenty different random interleavers and observed the same phenomena in all cases. Hence it is evident that the worsening of error floors can be mainly attributed to Phase 1 puncturing of the regular RA mother code rather than the choice of a poor interleaver. Therefore, Phase 1 puncturing is not adequate for obtaining code rates greater than 1/2. In order to obtain much higher code rates, phases 2

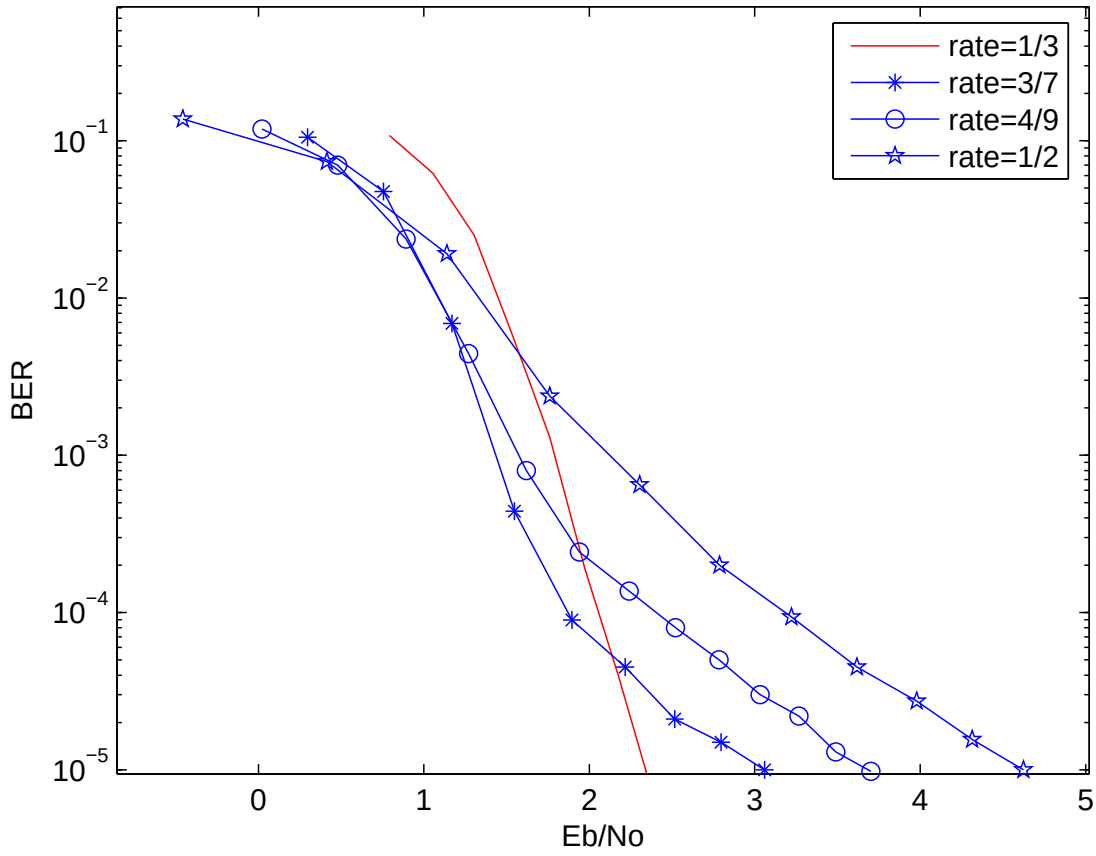

Figure 2.2. *Regular RA code punctured to IRA code using only phase 1 puncturing*

and 3 are required. Note that to simplify implementation, the interleaver used in Phase 1 remains throughout the next two phases of puncturing since the degree profile of the code does not change. This is very important for practical wireless communication systems. In our optimal code design which we shall describe in the next chapter, we use s-random interleavers which reduce the effects of error floors. We will now describe Phase 2 of the puncturing process.

2.2 Phase 2 - Puncturing parity bits

In this phase, the parity bits from the accumulator, p, are punctured (in addition to those punctured in Phase 1). From the block diagram in Fig. 2.1, it is evident that the number of parity bits transmitted across the channel is reduced thus further increasing the code rate compared to that in Phase 1. The new code rate is $R_2 = m/(m'_2 + p'_2)$ and since $p'_2 < p'_1$ and $m'_2 = m$, $R_2 > R_1$. Though the puncturing of parity bits in this phase is similar to that used elsewhere [16], the decoding is done in a slightly different manner. The choice of scheduling for decoding plays a role when parity bits are punctured in the typical manner. Typically when parity bits are punctured, the check node connected to the deleted parity node receives no channel information and the check node outputs zero information. In the case of turbo-like scheduling, the check node and all the links associated with the check node are essentially deleted from the Tanner graph of the code since the outgoing information from those particular check nodes would always be zero. Therefore, the puncturing of even a few parity bits could result in the deletion of many links in the Tanner graph and hence a degradation in performance. In the case of LDPC-like scheduling, the check node connected to the deleted parity node receives no channel information. However in this case, the decoding is still continued and still may be able to converge, but there will still be propagation of zero information through the links, leading to a loss in performance. In order to get good performance and increase the code rate, the puncturing must be done without affecting the structure of the graph, i.e. without deleting any links in the case of

turbo-like scheduling and avoid the propagation of zero-information messages in the case of LDPC-like decoding.

In our scheme, we uniformly puncture the parity bits, but we avoid the propagation of zero information through the links. The decoding process is illustrated in Fig. 2.3.

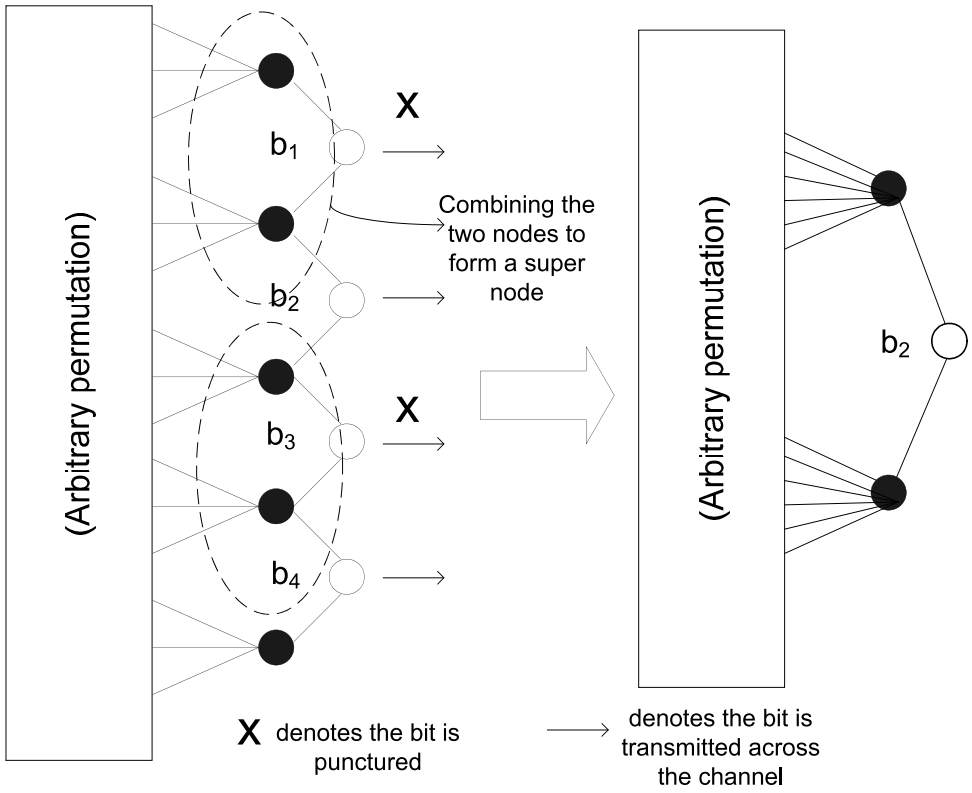

Figure 2.3. *Decoding on the Tanner graph after Phase 2 puncturing.*

From the Tanner graph of the code, it is evident that all the links attached to a check node are also attached to the next check node through a single link (due to the accumulator). Thus although a parity bit is deleted, we can combine the check node associated with the deleted parity bit and the next check node (that receives information from its corresponding undeleted parity bit), to form a single check node called a *super node*. In Fig. 2.3, b_1 is the deleted parity bit and b_2 is undeleted. Thus all links attached to the parity node b_1 are effectively attached to the next check node which is linked to the parity node b_2 to form a super node. Since this newly formed node now receives channel information from the undeleted parity bit, the outgoing messages can be calculated on all the links

connected to the check node. Hence the combining of check nodes allows us to puncture parity bits freely without deleting any link in the graph and avoid the propagation of zero-information messages, thus resulting in performance gain. Using the puncturing scheme in Phase 2, the performance will be less dependent on the scheduling method chosen. The performance results for typical puncturing and our proposed Phase 2 puncturing on the designed rate 3/4 IRA code from chapter 4 with $a = 4$ and $m = 1024$ is shown in Fig. 2.4. LDPC-like scheduling was used in decoding for both cases. The loss in performance due to typical puncturing of parity bits with the use of turbo-like scheduling is significantly higher compared to LDPC-like scheduling and in fact the decoding often fails to converge. Hence comparison results for the LDPC-like scheduling are only shown.

In this phase, half the parity bits are punctured to obtain rate $2/3$ while two thirds of the parity bits are punctured for rate $3/4$. This pattern is important as it is used in the Phase 1 profile design. Although there is a performance gain due to the combining of checknodes, this method effectively increases the density of the parity checks in the code. Puncturing a large number of parity bits can result in an extremely dense parity check matrix and hence the performance at low SNRs may be affected due to the presence of short cycles in the graph. Therefore Phase 2 of the puncturing process is not adequate to obtain very high code rates. In fact, we were only able to construct codes with rates up to 3/4 with good performance using Phase 2 puncturing. Beyond rate 3/4, the performance degraded considerably. Therefore, to achieve higher code rates, Phase 3 puncturing is employed.

2.3 Phase 3 - Puncturing systematic bits

In this phase, we puncture a small number of systematic message bits in addition to puncturing repetition bits in Phase 1 and parity bits in Phase 2. The new code rate is $R_3 = m/(m'_3 + p'_3)$ and since $m'_3 < m$ and $p'_3 = p'_2$, $R_3 > R_2$, we can obtain very high code rates using this puncturing scheme. Typically, puncturing is not applied to the systematic message bits as these bits are critical in initiating the iterative decoding algorithm. There-

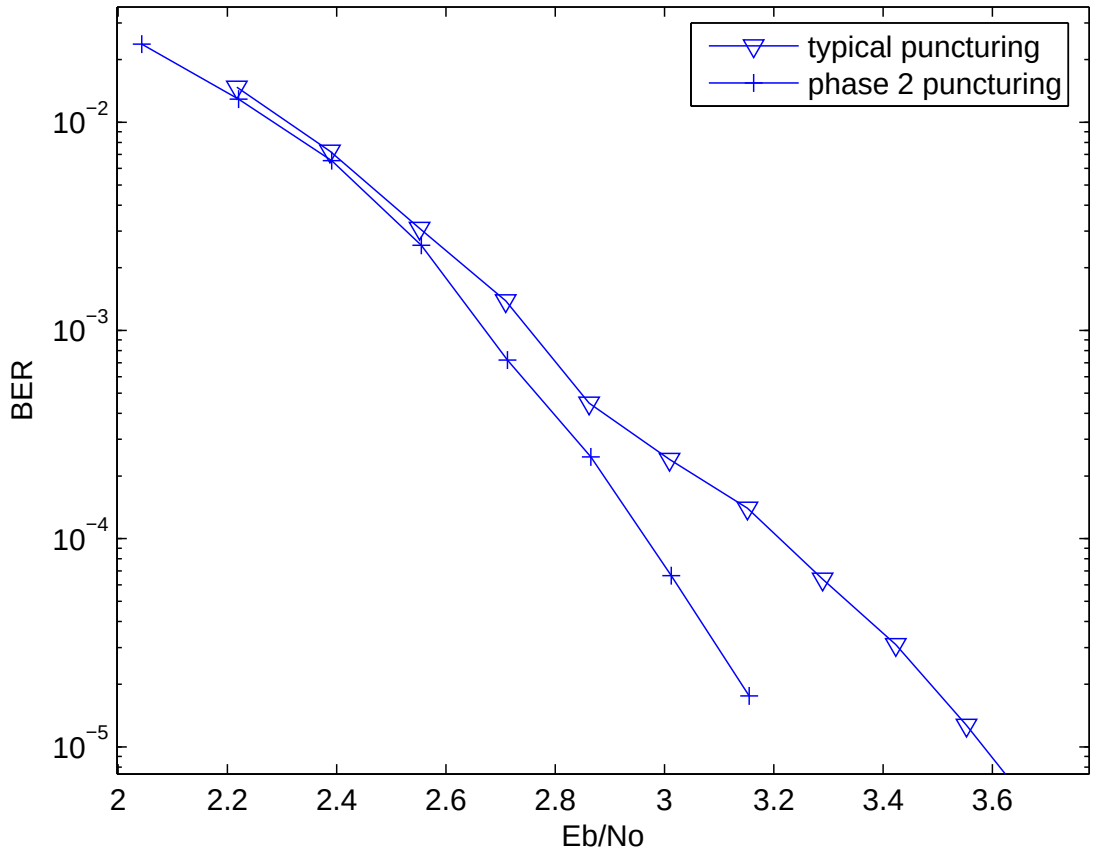

Figure 2.4. *Performance of normal and phase 2 puncturing*

fore only a small number of these bits can be punctured without significantly affecting the performance. As mentioned in Section 1.3, due to the presence of parity checks in IRA codes, we need to use the systematic form. Even though we are puncturing a very small number of systematic message bits, since it is in addition to the puncturing done in Phases 1 and 2, it provides a significant increase in the code rate.

An obvious way to puncture systematic bits would be to delete some of the information nodes and their corresponding links on the Tanner graph, but this would lead to a great loss in performance. In our case, the message bits are punctured but no links or information nodes are deleted from the Tanner graph. Instead, the decoding algorithm is allowed to naturally recover these message bits. This is possible because, since we are puncturing only a small number of bits, an information node whose message bit is punctured still receives information from its other links and thus receives the information about the punctured message bit. We shall briefly explain how this decoding works.

During decoding, if a link incident to a check node carries zero information, all the remaining links will have zero as the outgoing message from the check node. In the initial iteration, channel information is passed from information nodes to the check nodes. Due to the puncturing of systematic bits, some of the links will carry zero information to the check nodes in the initial iteration. As a result, most of the outgoing messages from a check node will become zero and this will continue in subsequent iterations until all the messages incident to a particular check node have non-zero information. Only when this condition occurs will the decoder begin to converge. This is the reason why we cannot puncture a large number of systematic bits as the decoder might not converge if there are initially too many links with zero information. However, Phase 3 puncturing can still provide very high code rates. Uniform puncturing of systematic bits is used although some higher degree nodes are left unpunctured to improve the decoding convergence. We will now present some performance results to show that our three-phase puncturing scheme is effective in achieving rate-compatibility in RA codes.

2.4 Simulation results

We provide performance results of the punctured RA code for different code rates with a blocklength of 1024 message bits and 10 decoding iterations. Turbo-like scheduling was employed while decoding. The degree profile for Phase 1, that was chosen in Section 2.1, was used and Phases 2 and 3 of the puncturing process were carried out. Note that the degree profile in this case was chosen in an ad hoc manner and only serves as a test to evaluate our proposed three-phase puncturing scheme. Performance results for the optimally designed code will be provided in the next chapter. Since the degree profile remains constant for all higher code rates, a single random interleaver is used, which is desirable. Fig. 2.5 shows the performance over an AWGN channel with BPSK modulation.

Performance results show that by using the three-phase puncturing scheme, we are able to reach very high code rates and still get good performance. The highest code rate obtained in our case is 9/10 whereas in the work of Lan *et al.* [16], the highest code rate they were able to reach was only 5/6. This shows that our puncturing scheme is able to provide higher code rates and is better than the typical puncturing of parity bits. Our performance for rate 4/5 is better than the corresponding RCPC code [11] by about 0.5dB at BER = 10^{-5}. For all other rates the performance difference between the codes is very close. These codes also compare favorably with RCPT codes. We simulated our code for rate 5/6 with a blocklength of 10240 to compare with the corresponding RCPT code proposed by Acikel and Ryan in [14]. The performance of our code was about 0.9dB worse at BER = 10^{-5}. However the complexity involved with our code is about one-third that of the RCPT code since it has a memory state of order three whereas ours has a memory state of order one.

In the next chapter, we shall describe the optimization of the degree profile using EXIT charts.

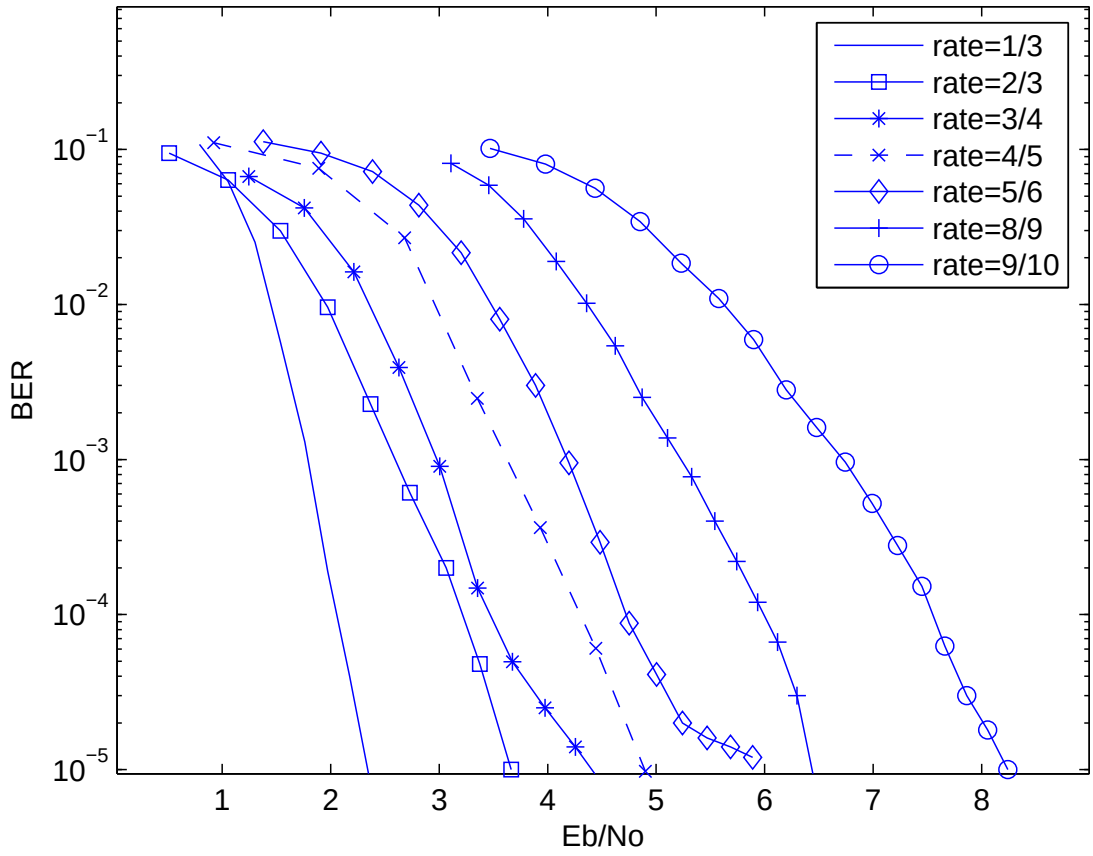

Figure 2.5. *Performance of various code rates for an ad hoc design*

Chapter 3

Design of Rate-Compatible RA Codes

In this chapter we provide a methodology to design rate-compatible punctured systematic RA codes. The design involves optimizing the punctured profile of the mother code such that the resulting high rate codes give good performance. In our case, the optimization is required for obtaining the optimal degree profile in Phase 1 since this profile remains constant throughout the next two Phases of puncturing, i.e. for rates greater than one-half. Note that the Phase 1 degree profile directly implies the irregular puncturing pattern employed in Phase 1. Also the profile should be chosen such that the higher punctured code rates have good performance. In order to design the degree profile, we use Extrinsic Information Transfer (EXIT) chart analysis and incorporate the three-phase puncturing that was previously described in Chapter 2 into the design. We shall first provide some background on EXIT charts and then describe the design methodology to obtain the optimal degree profile.

3.1 Introduction to EXIT charts

With the emergence of iterative decoding and capacity-achieving codes, researchers attempted to theoretically analyze and optimally design these codes. The technique of density evolution proposed by Chung *et al.* [9] was extensively used to analyze iterative codes such as LDPC codes and obtain the decoding thresholds for different code rates on various channels. Jin *et al.* in [7] used density evolution to optimize the code profile of IRA codes

for certain code rates on the AWGN channel. Lan *et al.* [16] used the same technique for optimizing the profile of the mother IRA code on the BSC channel for the highest code rate. However, design using density evolution involves numerical optimization that is quite complex. Recently, Extrinsic information transfer (EXIT) charts were developed by ten Brink [17] in order to analyse iterative codes and predict their performance in the region of low E_b/N_0. These charts are based on the mutual information characteristics of the decoders and provide decoding trajectories that describe the exchange of extrinsic information between the two constituent decoders. Results in [17] suggest that EXIT charts can accurately predict the convergence behaviour of the iterative decoder for large blocklengths and design using EXIT charts reduces the optimization to a simple curve-fitting problem. These charts have been employed by ten Brink *et al.* for the design of LDPC codes [19] as well as IRA codes [18]. Hence, we use EXIT charts to design the required degree profile of the punctured RA code. We shall first review the basic concepts of EXIT charts and then provide the EXIT curves for RA codes as obtained in [18].

Iterative decoding, as described in Chapter 2 for the case of IRA codes, is carried out based on an *a posteriori* probability decoder that converts channel and *a priori* LLR values into *a posteriori* LLR values. The *a posteriori* LLR values minus the *a priori* LLR values are considered as the extrinsic LLR values (which were described as extrinsic information in Chapter 2). These values are passed to the second decoder which interprets them as *a priori* information. An EXIT chart is a plot that describes the transfer of extrinsic information from one constituent decoder to the other given that each decoder utilizes *a priori* information obtained from the other decoder. In other words, the extrinsic information as a function of the *a priori* information needs to be determined for each constituent decoder. The transfer characteristics of each decoder are determined based on mutual information. The mutual information between two random variables X and Y, denoted by $I(X;Y)$, is

given by

$$\begin{aligned} I(X;A) &= H(X) - H(X/A) \\ &= \sum_{x,y} p(x,y) \log_2 p(x,y) \end{aligned} \tag{3.1}$$

where $H(x)$ denotes the entropy of X and $H(X/A)$ denotes the conditional entropy and $p(x,y)$ denotes the joint PDF of X and Y [20]. This is used to measure the information content in the channel, *a priori* and extrinsic LLR values. In our case, the *a priori* information content I_A is defined as the average mutual information between the bits on the graph edges (about which the extrinsic LLR values are passed) and the *a priori* LLR values. The extrinsic information content I_E is the average mutual information between the bits on the graph edges and the extrinsic LLR values. Let us consider that BPSK modulation is applied to the coded bits over an AWGN channel. Then a received signal y from the channel is $y = x + n$ where x is the transmitted bit $\{+1, -1\}$ and n is Gaussian distributed noise with mean zero and variance $\sigma_n^2 = N_o/2$. The conditional probability density function (PDF) is given by

$$p(y/X = x) = \frac{e^{-((y-x)^2/2\sigma_n^2)}}{\sqrt{2\pi}\sigma_n} \tag{3.2}$$

The corresponding LLR values from the channel are denoted as $L_{ch}(y)$ and calculated as

$$L_{ch}(y) = \ln \frac{p(y|x=1)}{p(y|x=-1)} \tag{3.3}$$

which is simplified to

$$L_{ch}(y) = \frac{2}{\sigma_n^2} \cdot y = \frac{2}{\sigma_n^2} \cdot (x+n) \tag{3.4}$$

(3.4) can also be rewritten as

$$L_{ch}(y) = \mu_{ch} \cdot x + n_y \tag{3.5}$$

where

$$\mu_{ch} = 2/\sigma_n^2 \tag{3.6}$$

and n_y is Gaussian distributed with mean zero and variance

$$\sigma_{ch}^2 = 4/\sigma_n^2 \tag{3.7}$$

Thus, the mean and variance of L_{ch} are connected by

$$\mu_{ch} = \frac{\sigma_{ch}^2}{2} \tag{3.8}$$

In order to determine the *a priori* information I_A, EXIT charts make two very important assumptions. They are

1. For large interleavers, the *a priori* LLR values remain fairly uncorrelated from the respective channel observations L_{ch} over many iterations and
2. The PDFs of the extrinsic output values (which become *a priori* values for the other decoder) approach Gaussian-like distributions with increasing number of iterations. This is can be assumed due to the nature of the extrinsic LLR value calculations since sums over many values are involved which leads to Gaussian-like distributions from the central limit theorem. Also in addition, since we are using an AWGN channel, the channel values are Gaussian.

Points 1 and 2 suggest that the *a priori* input values to a constituent decoder (which are extrinsic output values from the other decoder), can be modeled by applying an independent Gaussian random variable n_A with variance σ_A^2 and mean zero in conjunction with the known systematic transmitted bits x

$$A = \mu_A \cdot x + n_A \tag{3.9}$$

Since A is supposed to be an LLR value based on a Gaussian distribution, the conditional PDF belonging to the LLR value A is

$$p_A(\xi|X = x) = \frac{e^{-((\xi-(\sigma_A^2/2)\cdot x)^2/2\sigma_A^2)}}{\sqrt{2\pi}\sigma_A} \tag{3.10}$$

The *a priori* information I_A can be determined as

$$I_A(\sigma_A) = \frac{1}{2}\sum_{x=+1.-1}\int_{-\infty}^{+\infty} p_A(\xi|X = x) \times \log_2 \frac{2 \cdot p_A(\xi|X = x)}{p_A(\xi|X = -1) + p_A(\xi|X = +1)} d\xi \tag{3.11}$$

$$0 \leq I_A \leq 1 \tag{3.12}$$

The function $J(\sigma = \sigma_A)$ is used to denote the function corresponding to the mutual information I_A. This function does not have a closed form and depends on σ_A, however, it is reversible so that

$$\sigma_A = J^{-1}(I_A) \tag{3.13}$$

Similarly, the extrinsic output I_E can be quantified using the mutual information expression of (3.12) with the modification that the conditional probability belonging to the extrinsic output $p_E(\xi|X = x)$ is used in place of $p_A(\xi|X = x)$ and $0 \leq I_E \leq 1$. The mutual information corresponding to the channel LLR values can also be obtained from the $J()$ function. Hence the channel information content is calculated by $I(X; L_{ch}(y)) = J(\sigma_{ch})$. Since the mutual information $I(X; L_{ch}(y))$ is same as $I(X; Y)$ where X and Y are the input and output of the channel, the value $J(\sigma_{ch}) = J(2/\sigma_n)$ is called the capacity of the channel with BPSK modulation at the σ_{ch} being considered. For more details on the interpretation of the quantities I_E and I_A, we refer to [17].

The transfer characteristic of a constituent decoder is defined by viewing I_E as a function of I_A and E_b/N_0 given by

$$I_E = T(I_A, E_b/N_0) \tag{3.14}$$

where $E_b/N_0 = 1/(2R\sigma_n^2)$ and R is the code rate. In order to compute $T(I_A, E_b/N_0)$ for a desired input combination, the distributions of p_E and p_A used in the calculation of mutual information as described in (3.12) are conveniently determined using Monte Carlo simulations. For this, the independent Gaussian random variable of (3.9) is applied as *a priori* input to the constituent decoder ; a certain value of I_A is obtained by appropriately choosing σ_A according to (3.13). In order to implement $J()$ and $J^{-1}()$ in this thesis, we use the computer implementation specified in [19].

The transfer characteristics of the other constituent decoder can be obtained in a similar manner. Once the transfer characteristics of both constituent decoders are determined, the EXIT chart can be plotted. The extrinsic information transfer function of one constituent decoder is plotted with *a priori* information I_A on the abscissa and the extrinsic informa-

tion I_E on the ordinate. The extrinsic information transfer function of the other decoder is plotted on the same chart but with the axes reversed. This is done to depict the nature of iterative decoding in which the extrinsic information of one decoder becomes the *a priori* information for the other decoder. The chart thus visualizes the decoding trajectory of these functions. Based on the trajectory path, we can predict the convergence behaviour of the code. The gap between the transfer functions determines the rate of convergence of decoding and depends on R and E_b/N_0. For a given rate, as E_b/N_0 increases, the gap between the two transfer functions increases and the convergence of decoding is faster. The opposite occurs when E_b/N_0 is reduced, and eventually the gap will close, halting the decoding convergence. This makes sense since, for higher E_b/N_0 values, the information content from the channel is increased which in the process increases the *a priori* information thus resulting in a higher extrinsic information output. Similarly, for lower values of E_b/N_0, the extrinsic information output is lower. The value of E_b/N_0 for which the gap is just about to close is called the decoding threshold.

Having provided sufficient background on EXIT charts, we shall provide the EXIT curves for the case of RA codes and then employ these charts in our design.

3.2 EXIT curves of RA codes

ten Brink *et al.* applied EXIT charts to the design of IRA codes in [18]. We shall describe the computation of the EXIT curves for RA codes. In order to plot the EXIT chart of an RA code, as discussed in the previous section, we must obtain the EXIT functions of the constituent decoders. For the class of serially concatenated codes such as RA codes, an outer decoder and an inner decoder form the two constituent decoders. In our case, the repetition decoder is the outer decoder and the combination of check node and accumulator is the inner decoder. The repetition decoder is also referred as the outer variable node decoder since the decoding is done on the information nodes of the Tanner graph (which are variable nodes). We first determine the EXIT function of the outer variable node decoder

for a regular RA code with degree d_v and then extend it to the irregular case.

From the decoding described in Chapter 1, at the information node, the incoming messages of LLR values are added at the node as in (1.4). For an AWGN channel with BPSK modulation and noise variance σ_n^2, $E_b/N_0 = 1/2(R\sigma_n^2)$. The variance of the LLR value from the channel is given by

$$\sigma_{ch}^2 = \frac{4}{\sigma_n^2} = 8R \cdot \frac{E_b}{N_0} \tag{3.15}$$

From (1.4) at the information node, the decoder outputs are

$$L_{i,out} = L_{ch} + \sum_{j \neq i} L_{j,in} \tag{3.16}$$

where $L_{j,in}$ is the jth *a priori* LLR value going into the outer variable node decoder., $L_{i,out}$ is the ith extrinsic output LLR value coming out of the outer variable node decoder and L_{ch} is the channel LLR value. In order to obtain the EXIT function, the *a priori* LLR value $L_{j,in}$ is modelled as an LLR output from an AWGN channel whose input is the jth interleaved bit transmitted using BPSK. The exit function of a degree d_v variable node is given by

$$I_{E,vnd}\Big(I_{A,vnd}, d_v, E_b/N_0, R\Big) = J\Big(\sqrt{(d_v - 1)[J^{-1}(I_{A,vnd})]^2 + \sigma_{ch}^2}\Big) \tag{3.17}$$

where $I_{E,vnd}$ denotes the extrinsic information of the variable node decoder as a function of the input *a priori* information $I_{A,vnd}$. The functions $J()$ and $J^{-1}()$ are the same functions defined in the previous section that were used to determine the mutual information and are implemented using the description given in [19].

$$I_{E,vnd}\Big(0, d_v, E_b/N_0, R\Big) = J\big(\sigma_{ch}\big) \tag{3.18}$$

is the capacity of the AWGN channel with BPSK modulation at the σ_{ch} that is being considered. In the case of an IRA code where the degrees of the information nodes are not constant, the EXIT function can be obtained by considering the fraction of edges λ_i with degree d_i. Let N_v denote the number of different variable node degrees. It was shown in [22] and [23] that the EXIT curve of a mixture of codes is an average of the component EXIT curves. In our case, the variable node decoder now consists of variable nodes with

different degrees which implies that it consists of several component codes (repetition) with different degrees. Hence, the EXIT function for the outer variable node decoder can be determined by averaging over all the component EXIT curves of variable nodes with each component EXIT curve corresponding to a variable node of degree d_i determined using (3.17). The averaging must be done using the fraction of edges λ_i since the messages are being passed along the edges of the Tanner graph. The EXIT function of the outer variable node for an IRA code is then determined as

$$I_{E,vnd}\Big(I_{A,vnd}, E_b/N_0, R\Big) = \sum_{i=1}^{N_v} \lambda_i \cdot I_{E,vnd}\Big(I_{A,vnd}, d_i, E_b/N_0, R\Big) \tag{3.19}$$

For the inner decoder, the accumulator and check nodes are treated separately and then combined to obtain the EXIT function of the whole inner decoder. We will first consider a check node. The decoder output LLR values from (1.5) can be written as

$$L_{i,out} = \ln \frac{1 - \prod_{j\neq i} \frac{1-e^{L_{j,in}}}{1+e^{L_{j,in}}}}{1 + \prod_{j\neq i} \frac{1-e^{L_{j,in}}}{1+e^{L_{j,in}}}} \tag{3.20}$$

$L_{j,in}$ is modelled as the output LLR value of an AWGN channel whose input is the$j$th check node input bit transmitted using BPSK. For further analysis, the two directions of information flow through the check nodes are considered separately: $1)$ the Check node to accumulator direction and $2)$ the check node to the interleaver direction. Using the duality property that expresses the EXIT curve of a single parity check code (SPC) $I_{E,SPC}$ of length a in terms of the EXIT curve of the repetition code of length a [21] as

$$I_{E,SPC}(I_A, a) \approx 1 - I_{E,REP}(1 - I_A, a) \tag{3.21}$$

we can write the exit curve of the check node to accumulator direction as

$$\begin{aligned} I_{A,acc}(I_{A,cnd}, a) &\approx 1 - I_{E,REP}(1 - I_{A,cnd}, a + 1) \\ &= 1 - J\Big(\sqrt{a} \cdot J^{-1}(1 - I_{A,cnd})\Big) \end{aligned} \tag{3.22}$$

where $I_{A,cnd}$ denotes the *a priori* input to the check node. Using the duality property again,

we can write the EXIT curve of the check node to interleaver direction as

$$I_{E,cnd}(I_{A,cnd}, I_{E,acc}, a) \approx 1 - I_{E,REP}(1 - I_{A,cnd}, 1 - I_{E,acc}, a)$$
$$= 1 - J\left(\sqrt{(a-1)\cdot[J^{-1}(1-I_{A,cnd})]^2 + [J^{-1}(1-I_{E,acc})]^2}\right) \quad (3.23)$$

$I_{E,acc}$ is the EXIT function from the accumulator to the check node which is a function of the *a priori* input $I_{A,acc}$ to the accumulator. Based on [18], we can write the EXIT curve for the accumulator as

$$I_{E,acc}(I_{A,acc}, q) = \left[\frac{1-q}{1-q\cdot I_{A,acc}}\right]^2 \quad (3.24)$$

where $1-q = J(\sigma_{ch})$ which is the capacity of bits per use of the AWGN channel with BPSK modulation for a given SNR E_b/N_0. Using (3.22), (3.23), and (3.24) we can combine the exit curves for the accumulator and check node as one decoder and determine the overall EXIT function as

$$I_{E,acc\ \&\ cnd}(I_{A,cnd}, a, E_b/N_0, R) = I_{E,cnd}\left(I_{A,cnd}, I_{E,acc}, \left(I_{A,acc}, E_b/N_0, R\right), a\right)$$
$$= I_{E,cnd}\left(I_{A,cnd}, I_{E,acc}\left(I_{A,acc}(I_{A,cnd}, a), E_b/N_0, R\right), a\right) \quad (3.25)$$

In this manner, the EXIT functions of the decoders can be determined. We refer to [18] for details regarding the derivation of these functions. Once we have determined the EXIT functions of the constituent decoders, the EXIT chart of the RA code can be obtained. The transfer characteristics of both decoders are plotted on the same chart but with their axes inverted. In our case, $I_{A,vnd}$ versus $I_{E,vnd}$ and $I_{E,acc\ \&\ cnd}$ versus $I_{A,cnd}$ are plotted on the same chart. This depicts the fact that the extrinsic information of one decoder becomes the *a priori* information for the other decoder, and the chart visualizes the decoding trajectory of these functions. We shall now use these EXIT charts and explain the design methodology.

3.3 Design using EXIT charts

For our design, it was mentioned previously that we need to optimize the Phase 1 degree profile of the punctured RA code such that higher code rates obtained via puncturing give

good performance. From the previous discussion, it is evident that the degree profile of the repetition code determines the EXIT function characteristics of the outer variable node decoder. For a given a, the transfer function characteristic of the inner decoder is known. In our case, a is predetermined and fixed before the optimization. Hence we can design the degree profile of the code for given values of a and R by optimizing the degree fractions λ_i of the code such that its transfer function fits with the known transfer function of the inner decoder (curve fitting) for the lowest possible value of E_b/N_0 (decoding threshold). Curve fitting is done since the corresponding value of E_b/N_0 at which the two transfer characteristics meet is the decoding threshold. It must be noted that EXIT charts provide accurate predictions of the convergence behaviour for asymptotically large blocklengths and the optimization carried out is for the lowest possible decoding threshold. However, we consider finite blocklengths while evaluating the performance of the designed codes to ensure practicality. Since analysis can be done only for asymptotic thresholds, the design is carried out asymptotically but their BER performance at finite blocklengths is expected to be good, if not optimal. We will validate this statement with the performance results of the designed code.

In our design, we start with a rate 1/3 regular RA code with $a = 4$. The motivation behind this is to start with a simple regular code that is guaranteed to have good distance properties. The reason for the choice of $a = 4$ comes from our three phase puncturing scheme. For $a > 4$, because of the manner in which we puncture in Phase 2, the parity checks become very dense leading to a loss in performance. At the same time a lower value of a implies lowering the code rate and reducing the maximum possible degree which affects Phase 1 puncturing. Hence $a = 4$ is a good choice. Our choice of a is justified using EXIT charts as well as through performance results.

Next we need to design the Phase 1 degree profile of the code for a particular rate as it will remain constant for the next two phases of puncturing and will play a major role in the performance of the resulting higher code rates. The design is done for $R = 1/2$ since our main focus is on very high code rates, i.e. code rates greater than one-half

and this is the highest rate attainable in Phase 1. However, the profile should be chosen such that higher rate codes obtained via Phase 2 and Phase 3 puncturing are also good. Phase 3 puncturing cannot directly be incorporated into the EXIT chart analysis because the charts assume that the channel provides complete information on the message bits. However, Phase 2 puncturing can be incorporated since the formation of supernodes in Phase 2 directly implies an increase in the value of a, depending on the code rate. In our case, the highest rate in Phase 2 is 3/4, so the code with $a = 4$ at the end of Phase 1 has the same profile but with $a = 12$ due to the puncturing pattern employed. Therefore we can design the profile of the code such that it is optimal for rate 3/4 and expect good codes at the end of Phase 3. This is because Phase 3 puncturing involves puncturing only a small number of systematic bits and hence the performance depends on how good the code is before Phase 3. We summarize the requirements as follows:

1. Optimize the code profile for rate $R = 3/4$.
2. The value of a is effectively 12 at the end of Phase 2 puncturing.
3. The EXIT function of the variable decoder should be matched to the EXIT function of the inner decoder (which is determined for $a = 12$).
4. Since we start with a rate 1/3 regular $a = 4$ RA code, the maximum possible degree d_{max} in the profile is 8. Hence we need to optimize the profile $(\lambda_2 \cdots \lambda_8; a)$ with the following constraints:
   (a) $\sum_{i=2}^{d_{max}} \lambda_i = 1$. This is because λ_i are the fractions of edges and the minimum possible degree of the information nodes is 2.

   (b) $a \sum_{i=2}^{d_{max}} \frac{\lambda_i}{i} = \frac{R}{1-R}$. This constraint is to satisfy the code rate requirements. The fractions of edges and the code rate are related by this expression.

   (c) $\lambda_i \geq 0$

Since curve fitting typically means minimising the sum of squares of the errors between the

two functions, the optimization problem can be formulated as a convex Quadratic Programming (QP) problem. The problem formulation for the optimization shall now be described.

We must first formulate the objective function of the problem that is to be minimised. Let $e = fx - Y$ denote the error vector between the two transfer functions where fx denotes the overall transfer function vector of the variable node decoder and Y denotes the transfer function vector of the inner decoder. Let p denote the number of points in the vector. Y is given by $Y = [I_{A,cnd}(1), \cdots, I_{A,cnd}(p)]^T$. x is the variable vector we must optimize which consists of the fraction of edges λ_i where i is the value of the degree, i.e. $x = [\lambda_2, \cdots, \lambda_8]^T$. The length of x in our case is 7 since $i = 2, 3, 4, 5, 6, 7, 8$. f is a matrix that specifies the component EXIT functions for each degree value. Let $I_{E,vnd,i}$ denote the component EXIT function vector for a variable node of degree i which consists of p points. Then the matrix f is given by

$$f = \begin{bmatrix} I_{E,vnd,2}(1) & I_{E,vnd,3}(1) & \cdots & I_{E,vnd,8}(1) \\ I_{E,vnd,2}(2) & I_{E,vnd,3}(2) & \cdots & I_{E,vnd,8}(2) \\ I_{E,vnd,2}(3) & I_{E,vnd,3}(3) & \cdots & I_{E,vnd,8}(3) \\ \vdots & \vdots & \vdots & \vdots \\ I_{E,vnd,2}(p) & I_{E,vnd,3}(p) & \cdots & I_{E,vnd,8}(p) \end{bmatrix}$$

where $i = 2, 3, \cdots, 8$. Since we are minimising the sum of squares of the errors between the two functions, the objective function that is to be minimised is $e^T e$ which can be simplified to the standard QP form as follows

$$\begin{aligned} e^T e &= (fx - Y)^T (fx - Y) \\ &= x^T f^T fx - 2x^T f^T Y + Y^T Y \\ &= \frac{1}{2} x^T H x + x^T p \end{aligned} \tag{3.26}$$

where $H = (1/2) f^T f$ and $p = -(1/2) f^T Y$. We then formulate the constraints of the problem based on the criteria of (a), (b), (c). (a) and (b) form the equality constraints. They

can be put in the form $Ax = b$ where A and b are given by

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1/2 & 1/3 & 1/4 & 1/5 & 1/6 & 1/7 & 1/8 \end{bmatrix}$$

$$b = [\ 1 \quad R/((1-R)\cdot a)\ ]^T$$

(c) forms the inequality constraint and implies that $x \geq 0$. In addition we must ensure that the error vector e is strictly positive since a small gap between the two EXIT functions guarantees decoding convergence at the threshold value. To consider it in the standard form, we use the constraint $e \geq \delta$ where δ is a small value. Let $\bar{\delta}$ be a vector of length p and given by $\bar{\delta} = [\delta \cdots \delta]^T$. Let $\bar{0}$ denote a zero column vector of length equal to 7. I is an identity matrix of size 7. The inequality constraints are then formulated in standard form as $Cx \geq d$ where

$$C = \begin{bmatrix} f \\ I \end{bmatrix} \quad \text{and} \quad d = \begin{bmatrix} Y + \bar{\delta} \\ \bar{0} \end{bmatrix}$$

The optimization problem can now be summarized as follows:

$$\begin{aligned} \text{minimize } & \frac{1}{2}x^T H x + x^T p \\ \text{subject to: } & Ax = b \\ & Cx \geq d \end{aligned}$$

The optimization is done only for a specific value of E_b/N_0. Hence we start with a particular high value of E_b/N_0 and solve the optimization problem at that value. If the optimization algorithm converges to a solution, we reduce the value of E_b/N_0 and repeat the optimization. In this manner, the optimization is carried out for decreasing values of E_b/N_0 until we reach a point where the optimization algorithm is not able to converge. The minimum value of E_b/N_0 for which the algorithm converges is the decoding threshold and the solution obtained is the optimal degree profile for the given constraints.

On solving the formulated optimization problem for various values of E_b/N_0, the degree profile obtained was $(\lambda_6, \lambda_5, \lambda_3, \lambda_2) = (0.5376, 0.0818, 0.2776, 0.1030)$ for $R = 3/4$ with the remaining degree fractions equal to zero. The corresponding decoding threshold value of the obtained degree profile is $E_b/N_0 = 2$dB.

3.4 Exit charts and performance results

Fig. 3.1 shows the EXIT chart of the optimized code for $R = 3/4$ and a $E_b/N_0 = 2$dB. Note the two EXIT characteristic functions do not cross over but just touch each other, implying that the corresponding E_b/N_0 value is the decoding threshold of the code.

Fig. 3.2 shows the EXIT chart of the optimized code for the same code rate but at a higher value of E_b/N_0. Notice the gap between the two EXIT functions which implies that the decoding will converge faster for that particular value of E_b/N_0.

In order to justify the chosen value of $a = 4$ for our mother RA code, we applied Phase 2 puncturing to the optimally designed rate 1/2 $a = 8$ and $a = 6$ IRA codes of Jin *et al.* [7] and ten Brink *et al.* [18], respectively, and performed EXIT chart analysis on the punctured codes. From Phase 2 puncturing, the effective values of a for these codes would be 24 and 18 respectively and the code rate becomes 3/4. Figs. 3.3 and 3.4 show the EXIT charts of the corresponding IRA codes. Note that unlike our design problem which uses a regular RA code as mother code, the design of these codes do not have any constraint on the maximum variable degree since the codes used are IRA codes and thus can have higher degree distributions leading to better performance. Also note that the IRA codes are optimized for a particular code rate. However, the charts show that for our choice of $a = 4$, the rate 3/4 code has the same decoding threshold of $E_b/N_0 = 2$dB as the punctured $a = 6$ IRA code and is slightly better than the $a = 8$ IRA code. Hence $a = 4$ is a good choice.

Optimization was also carried out for a rate 1/3 regular RA mother code with $a = 8$ instead of $a = 4$. This would have a maximum variable degree d_{max}=16, and at the end of Phase 2, the punctured code would have $a = 24$. The optimal degree profile obtained has

a decoding threshold value of $E_b/N_0 = 2.4$dB. The decoding threshold value for $a = 8$ is considerably larger than that of $a = 4$. This further validates the argument that for $a > 4$, the parity checks become too dense resulting in a performance loss and thus our choice of $a = 4$ is the best possible choice. The EXIT chart of the punctured code for the optimal degree profile obtained is shown in Fig. 3.5.

EXIT chart analysis was also performed on the $a = 4$ rate 1/2 code obtained at the end of Phase 1 puncturing of the mother code, using the degree profile that was optimized for rate 3/4. The EXIT chart is shown in Fig. 3.6 and the corresponding decoding threshold value is $E_b/N_0 = 0.7$dB. This implies that, in spite of the degree profile being optimized for rate 3/4, the code for rate 1/2 still has a good decoding threshold that is comparable to other optimized codes that have more complex degree distributions such as the code designed by Jin *et al.* in [7].

Simulations were performed to evaluate the BER performance of various codes for BPSK modulation over the AWGN channel. In order to maintain uniformity, all performance results shown in this chapter were obtained using LDPC-like scheduling in the decoding algorithm. The number of decoding iterations used in all simulations was 30. S-random interleavers are used instead of random interleavers [24] since they help in reducing the error floors. In order to compare the EXIT chart results to the actual BER performance of the codes, the performance results of our optimized $a = 4$ and $a = 8$ codes for $R = 3/4$, along with the punctured IRA codes of Jin *et al.* and ten Brink *et al.* which were optimally designed for $R = 1/2$, are shown in Fig. 3.7. Note that our method of Phase 2 puncturing was employed on the optimal IRA codes instead of typical puncturing of the parity bits as typical puncturing will result in a loss of at least 0.4dB, as shown in Chapter 2 . The code rate is $R = 3/4$ and the blocklength used is 1024 message bits. The results obtained conform with the EXIT chart results in the sense that the performance of our optimized $a = 4$ code and that of the optimal IRA codes are quite similar as predicted by the EXIT charts although our optimized $a = 4$ code performs slightly better at this particular blocklength. Also notice that the optimized $a = 4$ code performs significantly better

than the optimized $a = 8$ code, as predicted by the EXIT charts.

Figs. 3.8 and 3.9 show results for blocklengths of 512 and 10,000 bits. Notice that for a blocklength of 10,000 bits, the results are closer to the EXIT chart results (as expected).

Fig. 3.10 shows the performance results of the optimized $R = 3/4$ code for different blocklengths. It can be observed from the results that as the blocklength increases, the BER performance approaches the decoding threshold of $E_b/N_0 = 2$dB which was predicted by the EXIT chart.

We also evaluated the performance of the optimized code for $R = 1/2$ and compared it with the unpunctured IRA code of Jin *et al.* and ten Brink *et al.* which were optimally designed for rate $R = 1/2$. The results are shown in Fig. 3.11. The EXIT charts predicted a decoding threshold of 0.5dB for both the optimally designed IRA codes [18], and the threshold for our $R = 1/2$ code is 0.7dB, which implies that all three codes should have similar BER performance for $R = 1/2$. Hence the performance results once again conform with the EXIT chart results showing that our code for $R = 1/2$ performs slightly worse than the other two. The blocklength used in this simulation was $m = 1024$ bits.

We shall now present the performance results of rate-compatible punctured RA codes utilizing the optimal degree profile obtained in the previous section and employing the three phase puncturing scheme to attain a wide range of code rates. Simulations were done for blocklengths of $m = 1024, 512, 256$ bits. Since the code profile remains the same after Phase 1, a single s-random interleaver [24] with s$\epsilon\{25,16,15\}$ respectively, was used for all higher rate codes obtained via puncturing. The results are shown in Figs. 3.12– 3.14.

From the performance results, it is evident that we are able to achieve very high rate code rates by employing our three phase puncturing scheme along with the optimal degree profile. The maximum code rate we are able to achieve is 10/11 for $m = 1024$. Even at such a high code rate, we are able to obtain a coding gain of more than 2dB over uncoded BPSK at a BER of 10^{-5}, which is quite remarkable given the simplicity of these codes. These codes give good performance even at short blocklengths such as 256 bits.

It was already shown in Chapter 2 that our puncturing scheme provides superior per-

formance compared to just puncturing parity bits as employed in [16] by Lan *et al.*. The codes designed in this chapter provide a further improvement of at least 1dB, thus justifying the advantage of our puncturing scheme as well as our design methodology. In the next chapter, we shall present some conclusions and future work.

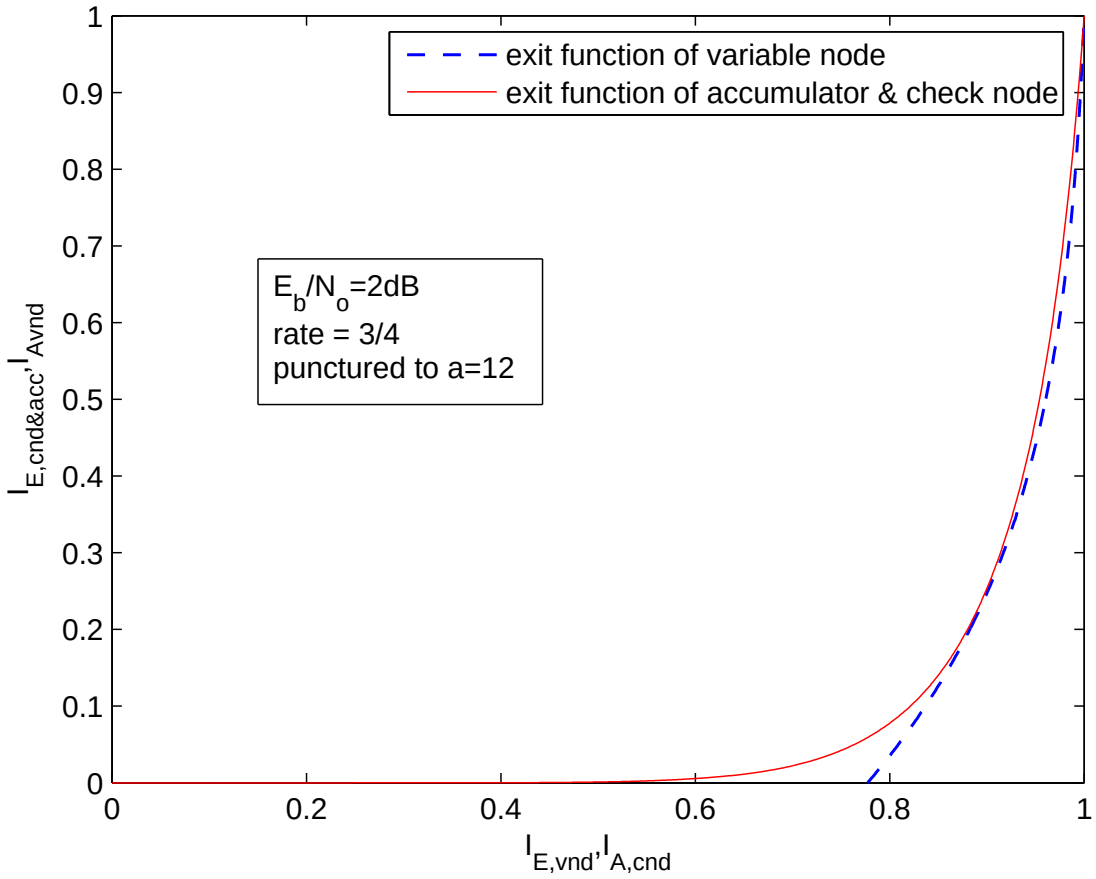

Figure 3.1. *Exit chart of the optimized code for* $R = 3/4$, $E_b/N_0 = 2dB$

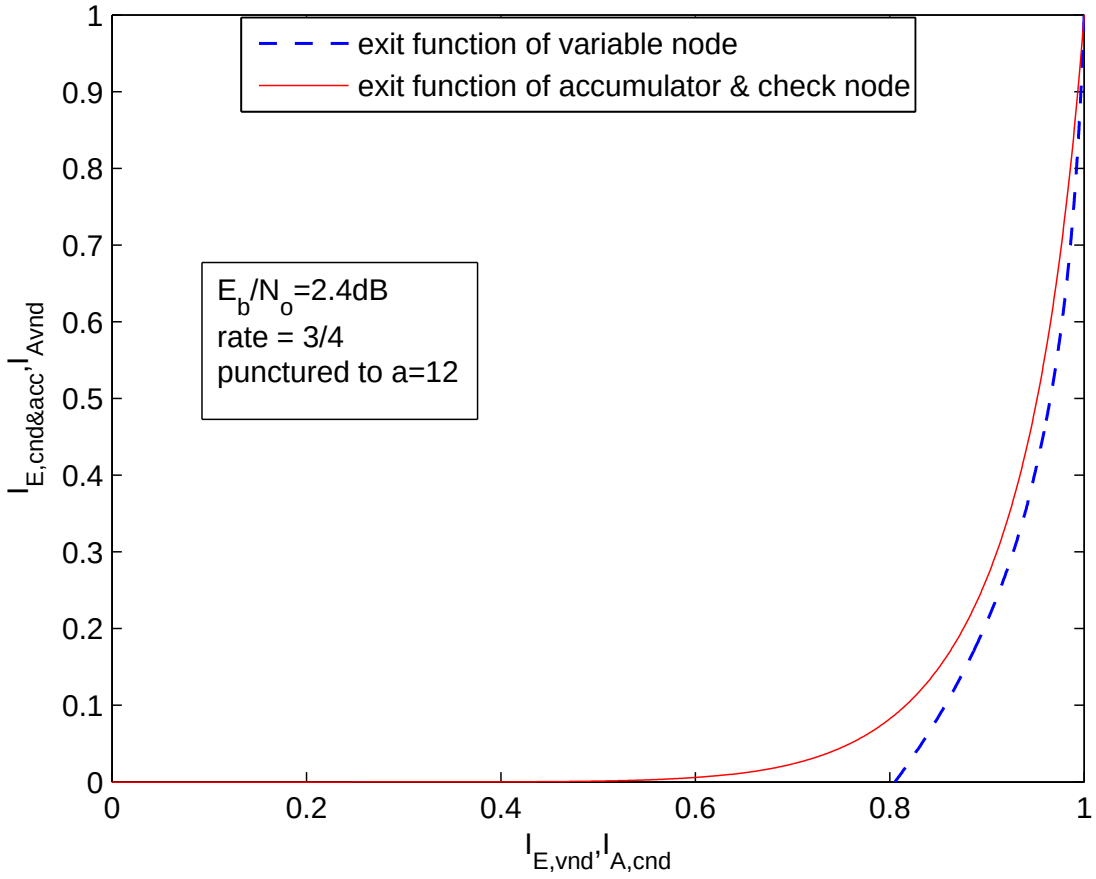

Figure 3.2. *Exit chart of the optimized code for* $R = 3/4$, $E_b/N_0 = 2.4dB$

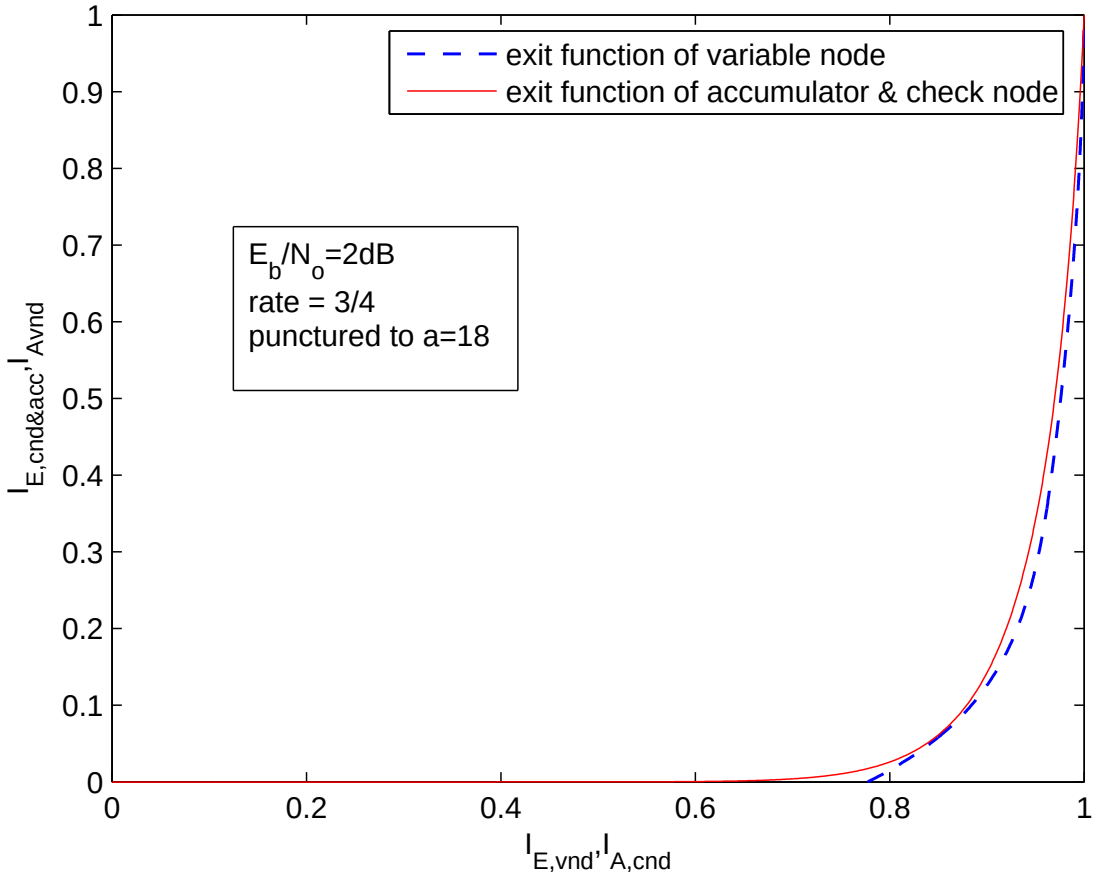

Figure 3.3. *Exit chart of the punctured IRA code of ten Brink et al. for* $R = 3/4$

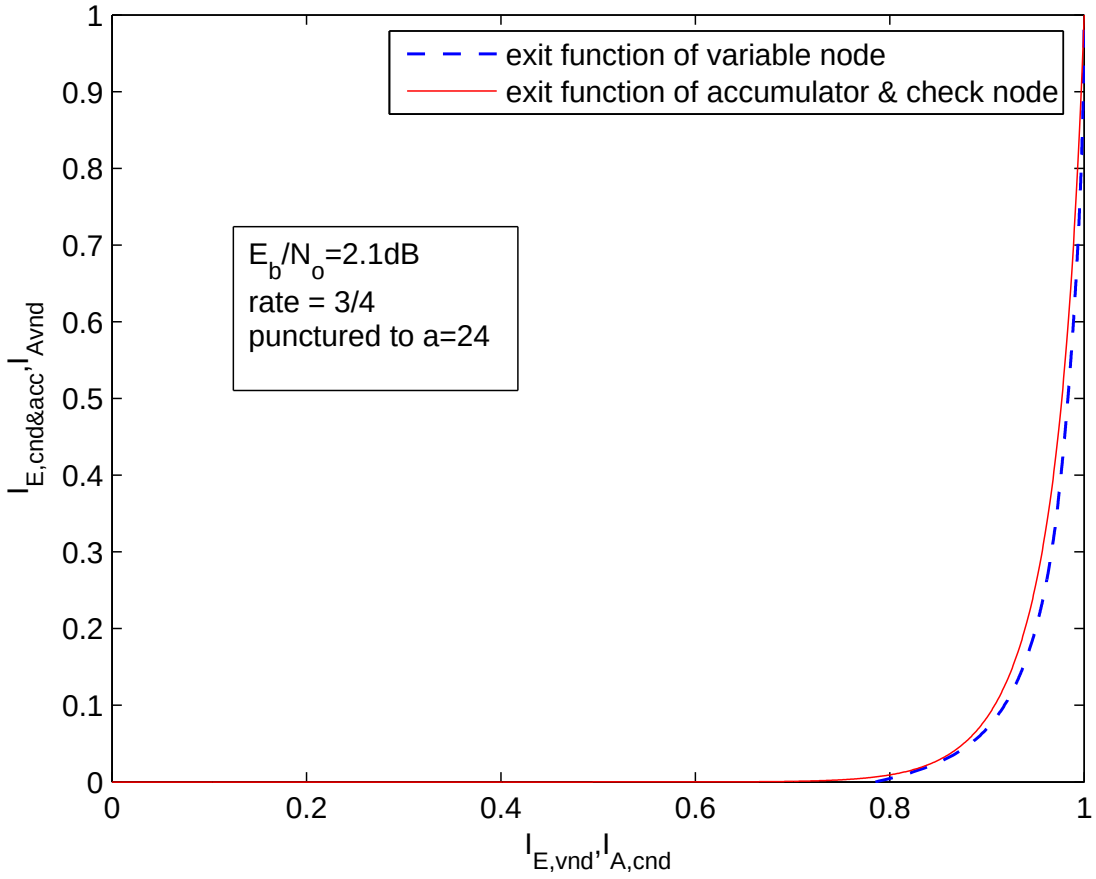

Figure 3.4. *Exit chart of the punctured IRA code of Jin et al. for* $R = 3/4$

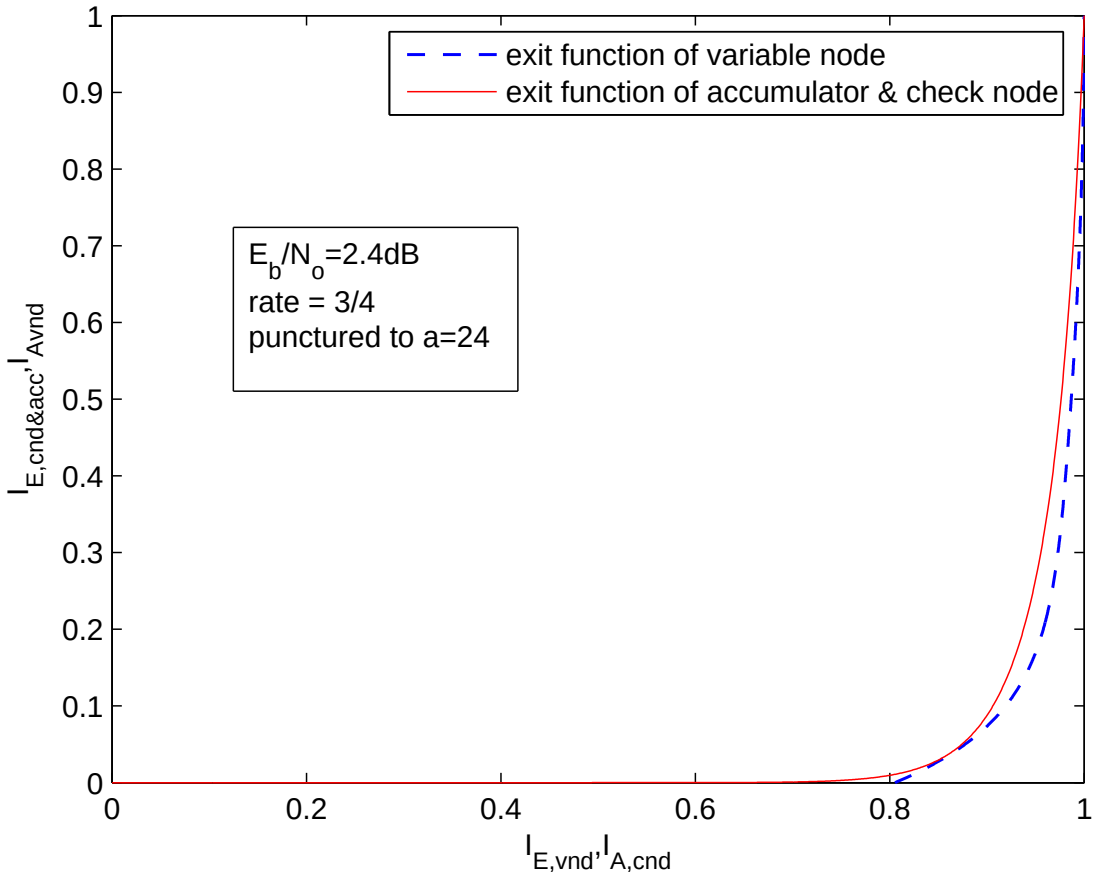

Figure 3.5. *Exit chart of the code optimized for the* $a = 8$ *mother code*

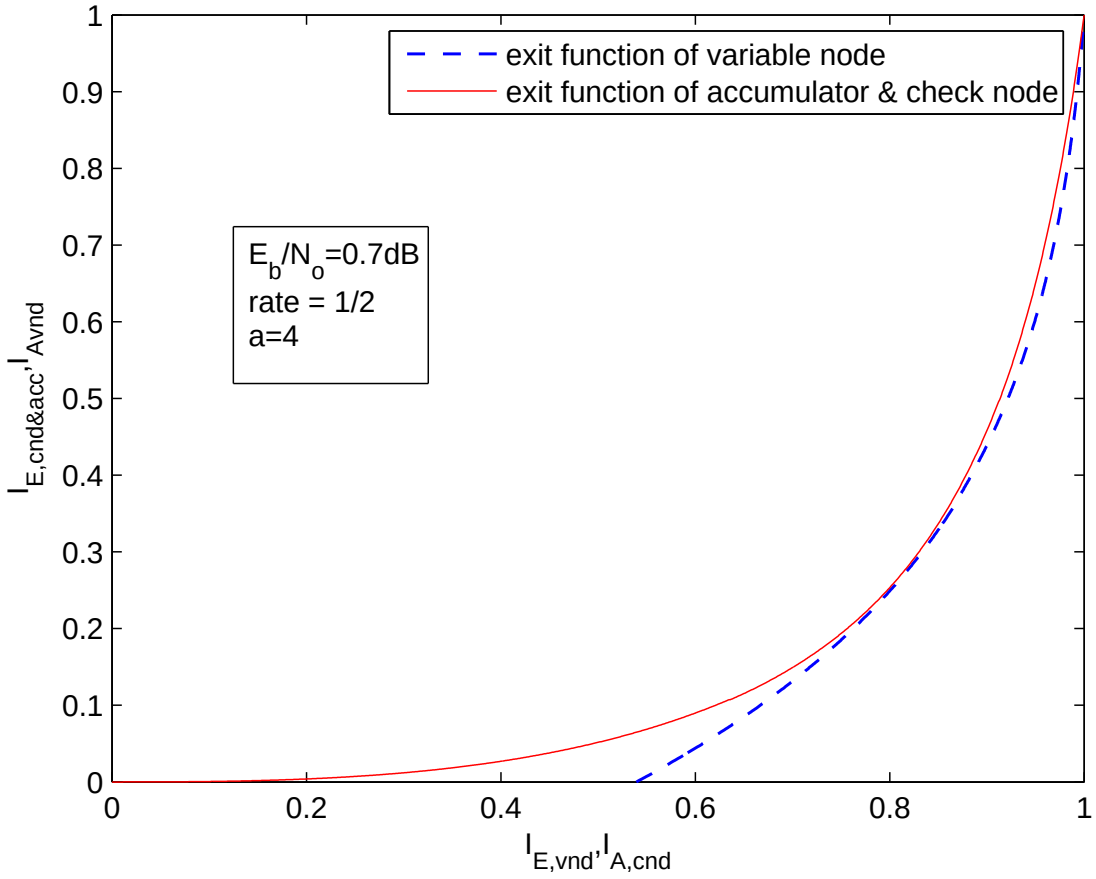

Figure 3.6. *Exit chart of the optimized code for* $R = 1/2$

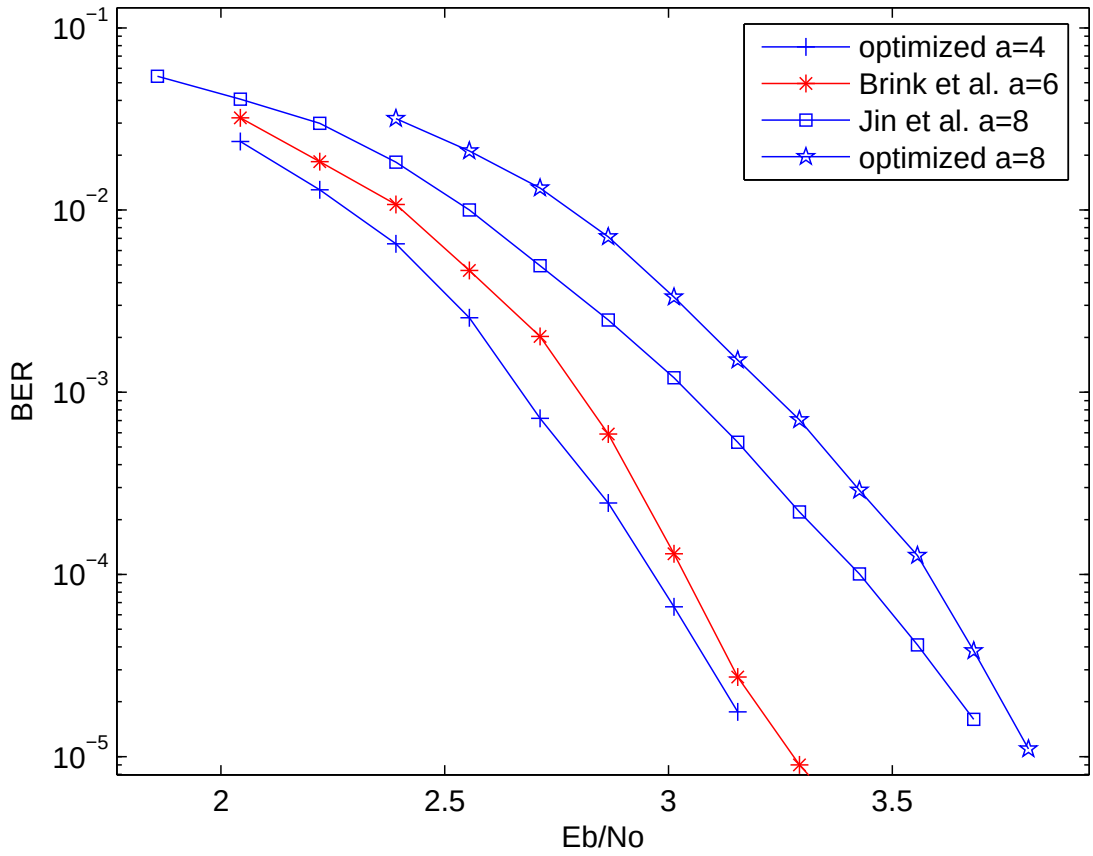

Figure 3.7. *BER performance of various codes for* $R = 3/4$, $m = 1024$ *bits*

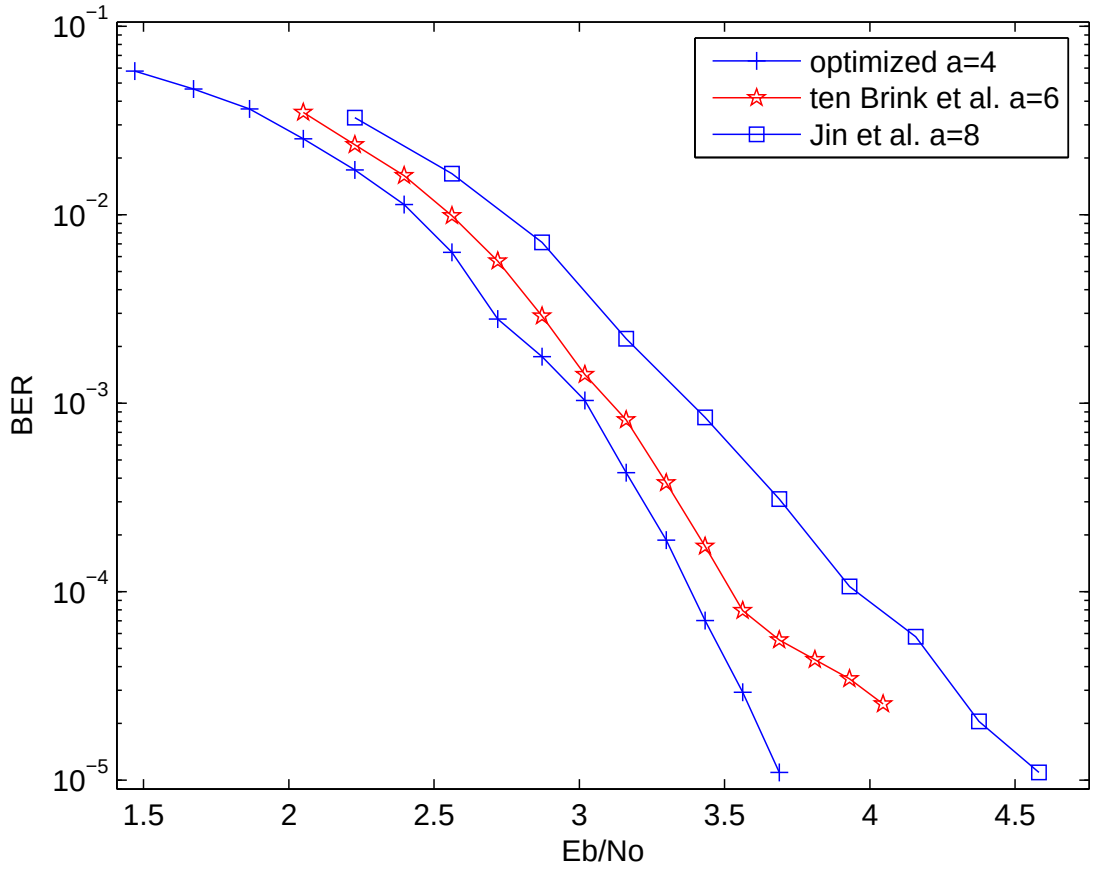

Figure 3.8. *BER performance of various codes for* $R = 3/4$, $m = 512$ *bits*

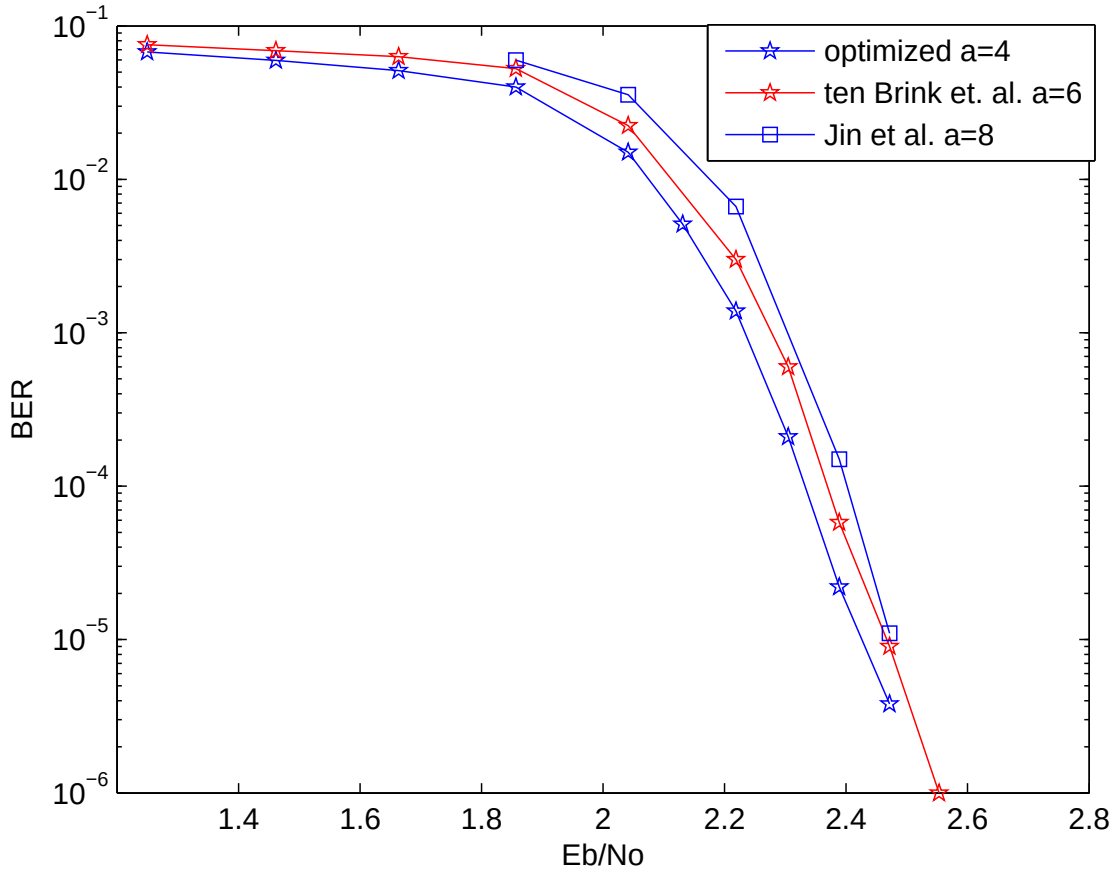

Figure 3.9. *BER performance of various codes for* $R = 3/4$, $m = 10,000$ *bits*

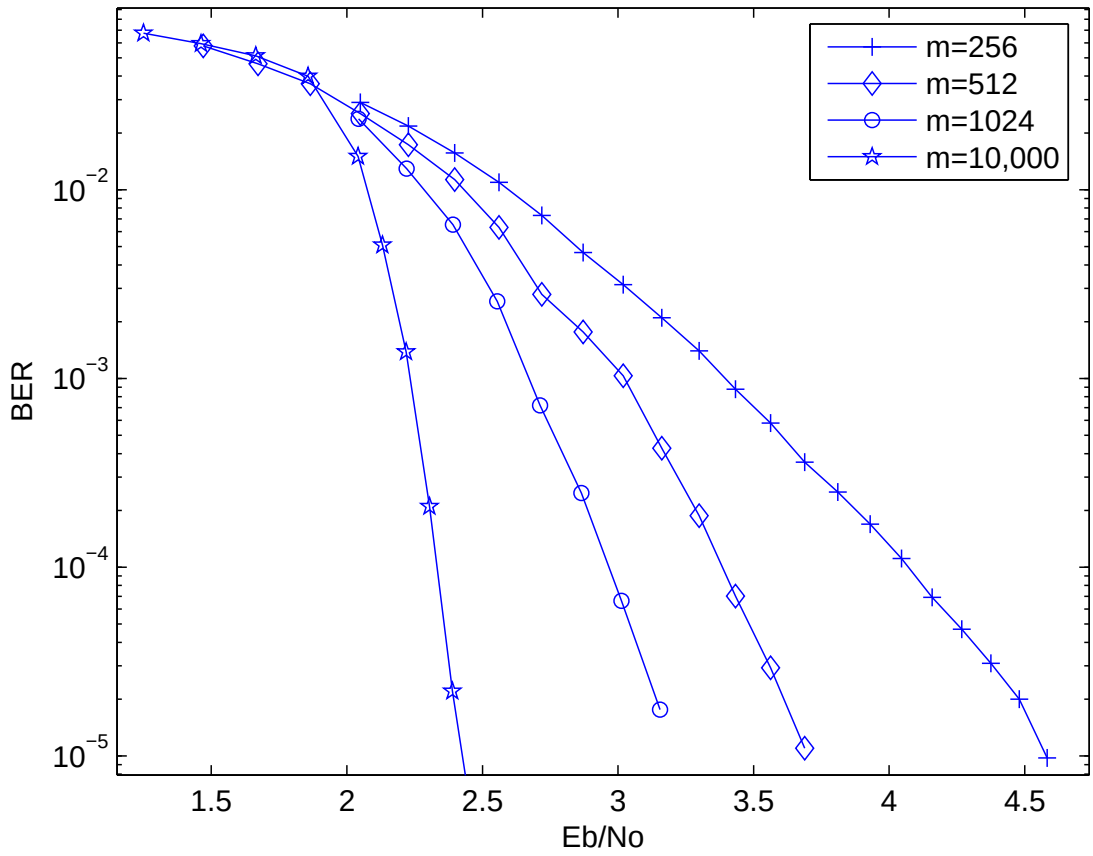

Figure 3.10. *BER performance of optimized codes of* $R = 3/4$ *for different blocklengths.*

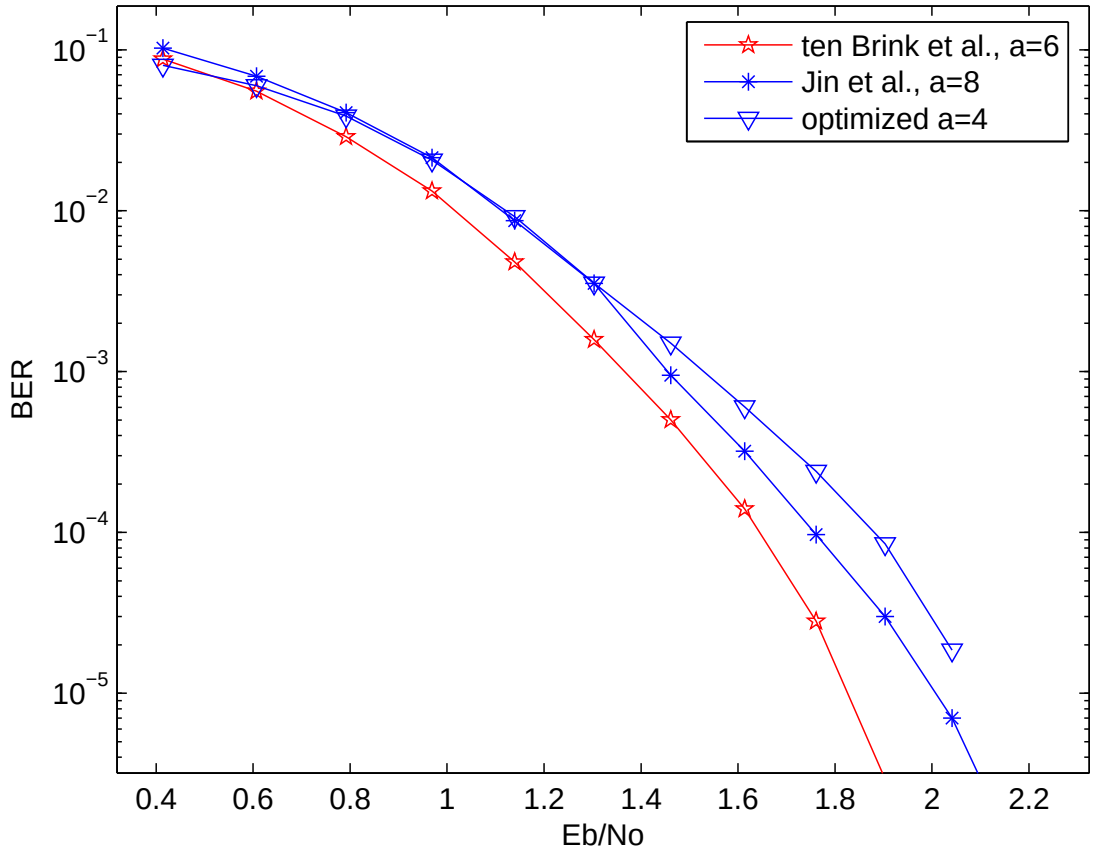

Figure 3.11. *BER performance of various codes for* $R = 1/2$, $m = 1024$ *bits.*

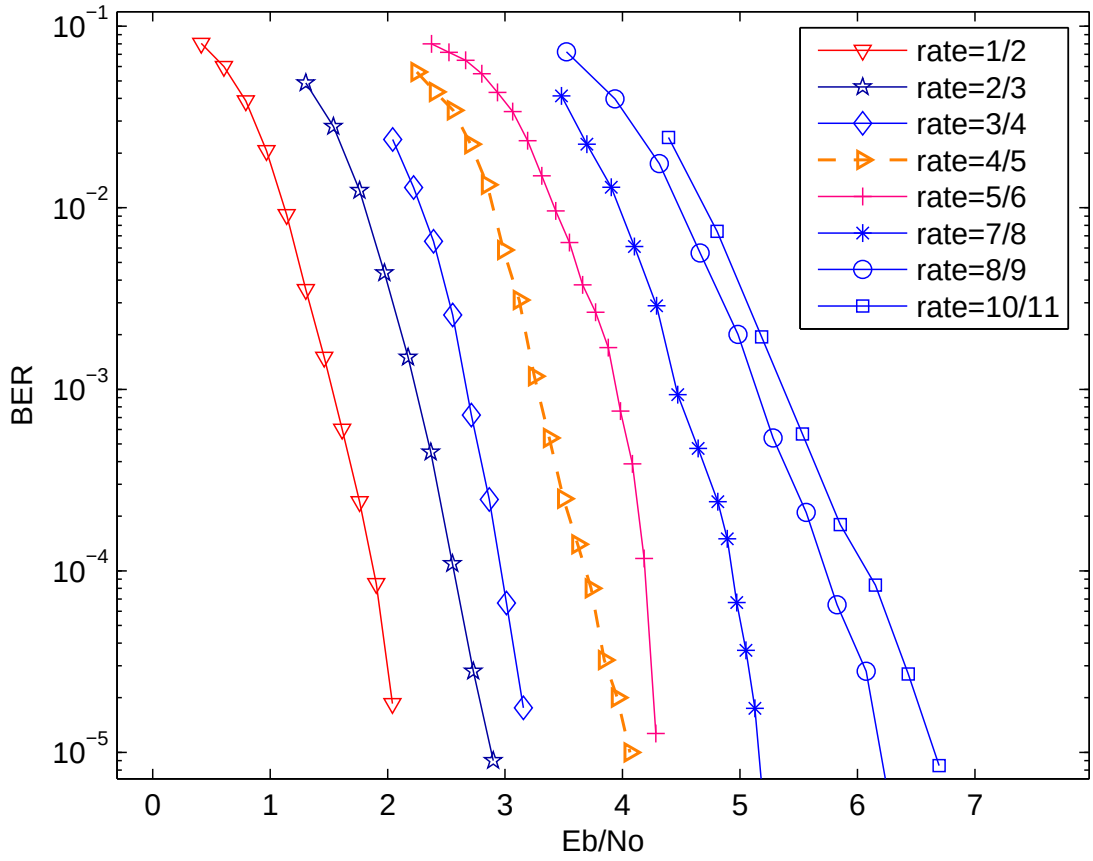

Figure 3.12. *BER performance for various code rates,* $m = 1024$ *bits.*

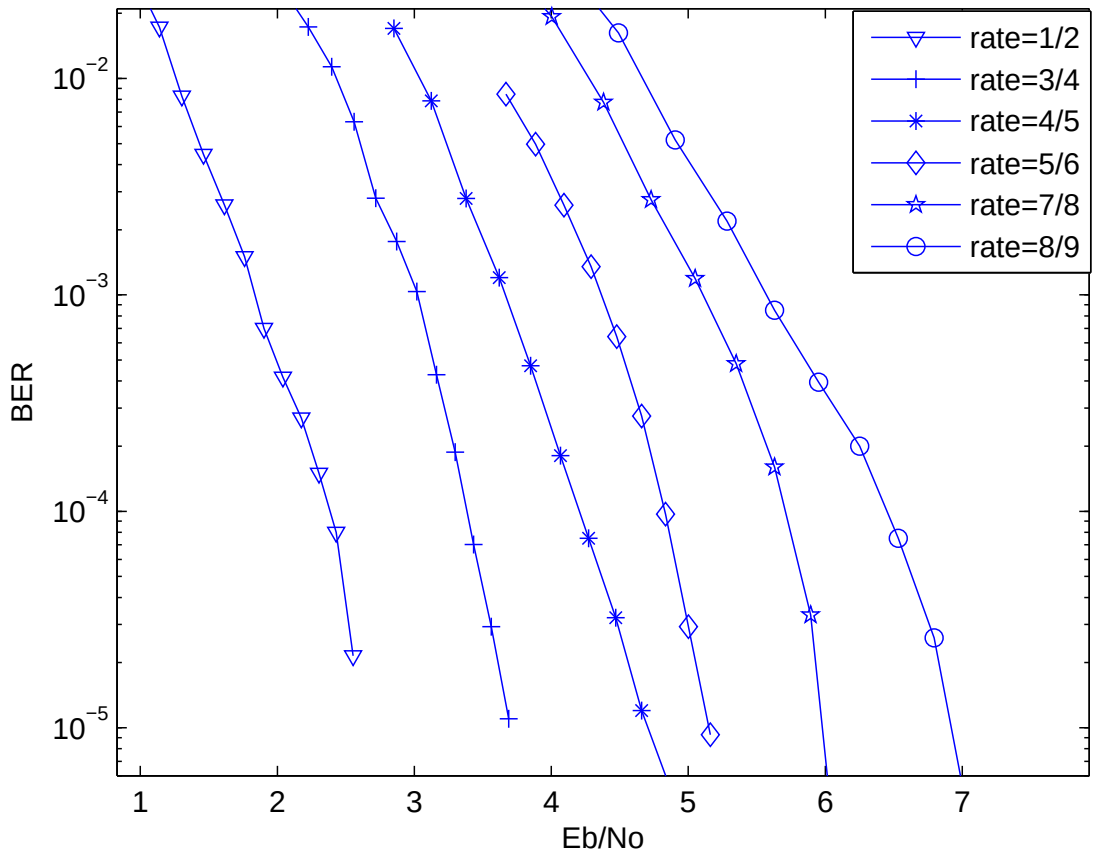

Figure 3.13. *BER performance for various code rates,* $m = 512$ *bits.*

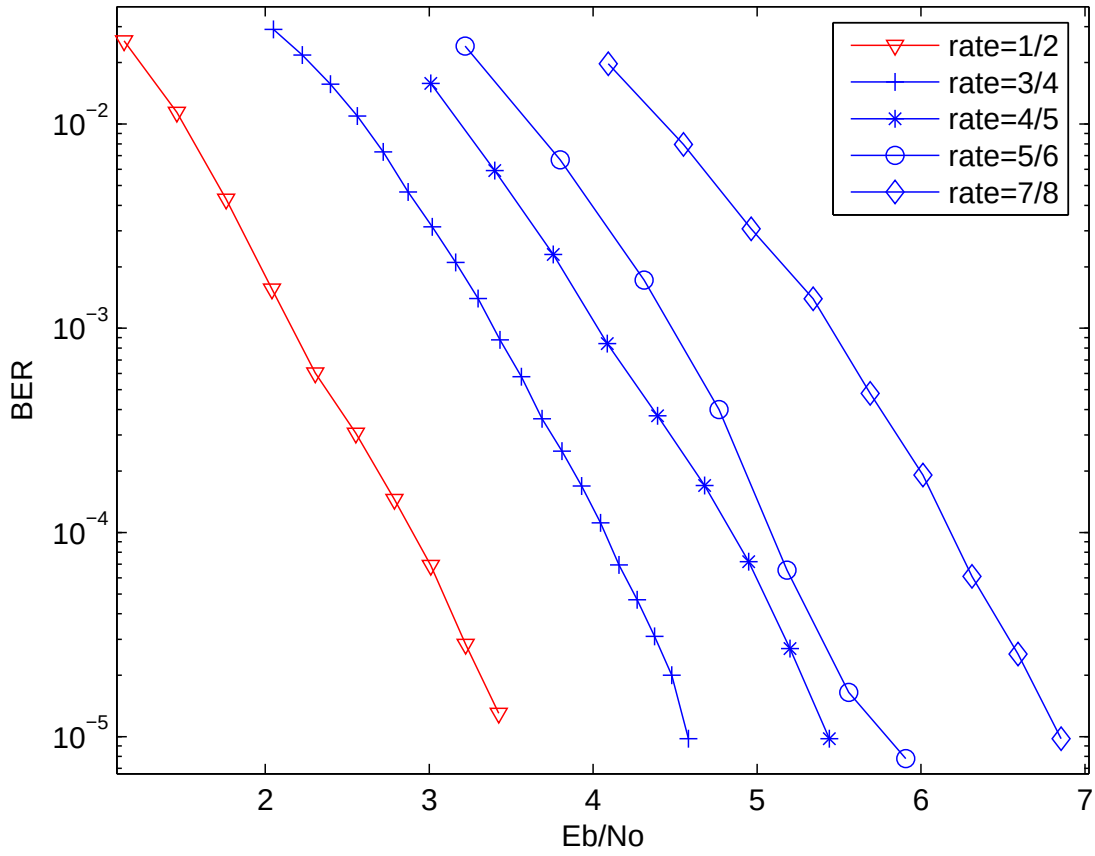

Figure 3.14. *BER performance for various code rates,* $m = 256$ *bits.*

Chapter 4

Conclusions and Future Work

In this thesis, we presented rate-compatible punctured systematic RA codes for the AWGN channel. A three phase puncturing scheme was proposed in order to introduce rate-compatibility into the codes and a design methodology was given that employs EXIT charts. Based on the results obtained, we present some conclusions and scope for future work.

1. RA codes are extremely simple compared to turbo or LDPC codes and hence are easier to implement. However, the performance of rate-compatible RA codes show that they are very competitive to other rate-compatible codes such as RCPT and RLDPC codes.
2. Our proposed three phase puncturing scheme was shown to provide performance superior to just puncturing parity bits. Hence the three phase puncturing can be used to induce rate-compatibility for any class of regular RA codes based on the design requirements. The method of puncturing in Phase 2 enabled us to incorporate it into the design of the degree profile that employs EXIT charts.
3. Our design methodology using EXIT charts enabled us to obtain very high code rates from a single rate 1/3, $a = 4$ mother RA code. The advantage of starting with a regular RA code as a mother code was highlighted and the performance results showed that the minimum distance properties of the subsequent codes obtained via puncturing were not significantly affected. The choice of a was validated using EXIT charts and BER performance results. The maximum code rate we were able to attain was 10/11 for a blocklength of 1024 bits.

4. Although EXIT chart analysis predicts the behaviour of decoding convergence only for asymptotically large blocklengths, the performance results show that the designed codes are good even at finite blocklengths. Hence EXIT charts are useful and valid tools for the design of rate-compatible RA codes.

5. The results showed that our design methodology combined with the three phase puncturing scheme provided high rate codes with good performance even at shorter blocklengths such as 512 and 256 bits. The number of decoding iterations used in the implementation, which was 30, is relatively small and practical. In addition, for higher code rates, only a single s-random interleaver is required as opposed to having different interleavers for different code rates. Hence the results show that these codes are extremely practical in their performance and support simple VLSI implementations.

Future work includes the following:

1. incorporating the systematic puncturing into the optimization problem by making certain assumptions and thus optimizing for the highest possible code rate, which in our case is 10/11, instead of just optimizing for $R = 3/4$.

2. optimizing the degree profile for a finite blocklength instead of designing for asymptotic thresholds. The finite blocklength factor needs to be incorporated into the EXIT charts in order to obtain codes that optimally perform at finite shorter blocklengths. Whether there would be any significant gain in performance by carrying out optimization for finite blocklengths remains an open question.

3. a more rigorous approach to choosing the value of a and incorporating it into the optimization problem. The choice of a may play a greater role when designing codes for shorter blocklengths. Indeed this problem is the next step after the previous problem. One possible way would be to treat the whole optimization problem as a mixed integer programming problem with a being an integer variable and the degree fractions as continuous variables. However, determining the solution to such an optimization

problem is extremely difficult.

4. considering the design of rate-compatible RA codes for a Rayleigh fading channel.
5. considering the design of rate-compatible RA codes for the ultra wide band (UWB) channel.

Based on the above conclusions, rate-compatible systematic RA codes are extremely simple codes that give good performance over the AWGN channel even at practical block-lengths. Hence these codes can be extremely useful in a wide range of applications such as wireless communications.

Bibliography

[1] R. W. Hamming, "Error detecting and error correcting codes, *Bell Sys. Tech. Journal*, vol. 29, pp. 147–160, April, 1950.

[2] C.E. Shannon, "A Mathematical Theory of Communication," *Bell Sys. Tech. Journal*, vol. 27, pp. 379–423, 623–656, July, October, 1948.

[3] S. Lin and D. J. Costello, "*Error Control Coding: Fundamentals and Applications*," second edition, Prentice Hall: Englewood Cliffs, NJ, 2005.

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

[5] R. G. Gallager "*Low-density parity-check codes*," Cambridge, MA, MIT press, 1963.

[6] D. Divsalar, H. Jin and R.J. McEliece, "Coding theorems for 'turbo-like' codes," Proc. *Allerton Conf. Commun., Control and Computing*, pp. 201–210, Sept. 1998.

[7] H. Jin, A. Khandekar and R.J. McEliece, "Irregular repeat-accumulate codes," Proc. *Int. Symp. Turbo Codes*, pp. 1–8, Brest, France, Sept. 2000.

[8] F. R. Kschischang, B. J. Frey, H. A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Inform. Theory*, vol. 47, no. 2, pp. 498–519, Feb. 2001.

[9] S. Chung, T.J. Richardson and R.L. Urbanke, "Analysis of sum-product decoding of low-density parity-check codes using a Gaussian approximation," *IEEE Trans. Inform. Theory*, vol. 47, no. 2, pp. 657–670, Feb. 2001.

[10] L.R. Bahl, J. Cocke, F. Jelinek and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inform. Theory*, vol. 20, pp. 284–287, Mar. 1974.

[11] J. Hagenauer, "Rate-compatible punctured convolutional codes (RCPC codes) and their applications," *IEEE Trans. Commun.*, vol. 36, no. 4, pp. 389–400, Apr. 1988.

[12] A.S. Barbulescu and S.S. Pietrobon, "Rate-compatible turbo codes," *Elect. Letts.*, vol. 31, no. 7, pp. 535–536, Mar. 1995.

[13] J. Hagenauer, E. Offer and L. Papke, "Iterative decoding of binary block and convolutional codes," *IEEE Trans. Inform. Theory*, vol. 42, no. 2, pp. 429–445, Mar. 1996.

[14] O.F. Acikel and W.E. Ryan, "Punctured turbo codes for BPSK/QPSK," *IEEE Trans. Commun.*, vol. 47, no. 9, pp. 1315–1323, Sept. 1998.

[15] J. Ha, J. Kim and S.W. McLaughlin, "Rate-compatible puncturing of low-density parity-check codes," *IEEE Trans. Inform. Theory*, vol. 50, no. 11, pp. 2824–2836, Nov. 2004.

[16] C. Lan, T. Chu, K. R. Narayanan and Z. Xiong, "Scalable image and video transmission using irregular repeat-accumulate codes with fast algorithm for optimal unequal error protection," *IEEE Trans. Commun.*, vol. 52, no. 7, pp. 1092–1101, July 2004.

[17] S. ten Brink, "Convergence behaviour of iteratively decoded parallel concatenated codes," *IEEE Trans. Commun.*, vol. 49, no. 10, pp. 1727–1737, Oct. 2001.

[18] S. ten Brink and G. Kramer, "Design of repeat-accumulate codes for iterative detection and decoding," *IEEE Trans. Signal Process.*, vol. 51, no. 11, pp. 2764–2772, Nov. 2003.

[19] S. ten Brink and G. Kramer, "Design of low-density parity-check codes for modulation and detection,"*IEEE Trans. Commun.*, vol. 52, no. 4, pp. 670–677, April 2004.

[20] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. New York: Wiley, 1991.

[21] E. Sharon, A. Ashikhmin and S. Litsyn, " Exit functions for the Gaussian channel," in *Proc. Allerton Conf. Communication, Control, Computers*, pp. 972–981, Oct. 2003.

[22] A. Ashikhmin, G. Kramer and S. ten Brink, "Extrinsic information transfer functions: A model and two properties," in *Proc. 36th Ann. Conf. Inform. Sciences Syst.*, Princeton, Mar. 2002.

[23] M. Tuechler and J. Hagenauer, "Exit charts and irregular codes," in *Proc. Conf. Inform. Sciences Syst.*, Princeton, Mar. 2002.

[24] D. Divsalar and F. Pollara, "Turbo codes for PCS applications," Proc. *IEEE Int. Conf. Commun.*, pp. 54–59, June 1995.

VITA

Surname: Planjery ***Given Names:*** Shiva Kumar

Place of Birth: Chicago, IL, U.S.A. ***Date of Birth:*** June 6th 1983

Degrees Awarded

B.Tech	Indian Institute of Technology- Madras, Chennai, India	2005

Publications

1. S. Planjery, T. A. Gulliver and A. Thangaraj, "Rate-compatible punctured systematic repeat-accumulate codes," Proc. *IEEE Wireless Commun. and Networking Conf.*, Mar. 2007.

Title of Thesis:

Design of Rate-Compatible Punctured Repeat-Accumulate Codes

Author: _____

SHIVA KUMAR PLANJERY

June 22, 2007

THESIS WITHHOLDING FORM

At our request, the commencement of the period for which the partial licence shall operate shall be delayed from June 22, 2007 for a period of at least six months.

_____

(Supervisor)

_____

(Department Chairman)

_____

(Dean of Graduate Studies)

_____

(Signature of Author)

_____

(Date)

Date Submitted to the Dean's Office: _____